Submodular Trajectory Optimization for Aerial 3D Scanning

Mike Roberts[1,2] Debadeepta Dey[2] Anh Truong[3] Sudipta Sinha[2]
Shital Shah[2] Ashish Kapoor[2] Pat Hanrahan[1] Neel Joshi[2]

[1]Stanford University [2]Microsoft Research [3]Adobe Research

Abstract

Drones equipped with cameras have become a powerful tool for large-scale aerial 3D scanning, but existing automatic flight planners do not exploit all available information about the scene, and can therefore produce inaccurate and incomplete 3D models. We present an automatic method to generate drone trajectories, such that the imagery acquired during the flight will later produce a high-fidelity 3D model. Our method uses a coarse estimate of the scene geometry to plan camera trajectories that: (1) cover the scene as thoroughly as possible; (2) encourage observations of scene geometry from a diverse set of viewing angles; (3) avoid obstacles; and (4) respect a user-specified flight time budget. Our method relies on a mathematical model of scene coverage that exhibits an intuitive diminishing returns property known as submodularity. We leverage this property extensively to design a trajectory planning algorithm that reasons globally about the non-additive coverage reward obtained across a trajectory, jointly with the cost of traveling between views. We evaluate our method by using it to scan three large outdoor scenes, and we perform a quantitative evaluation using a photorealistic video game simulator.

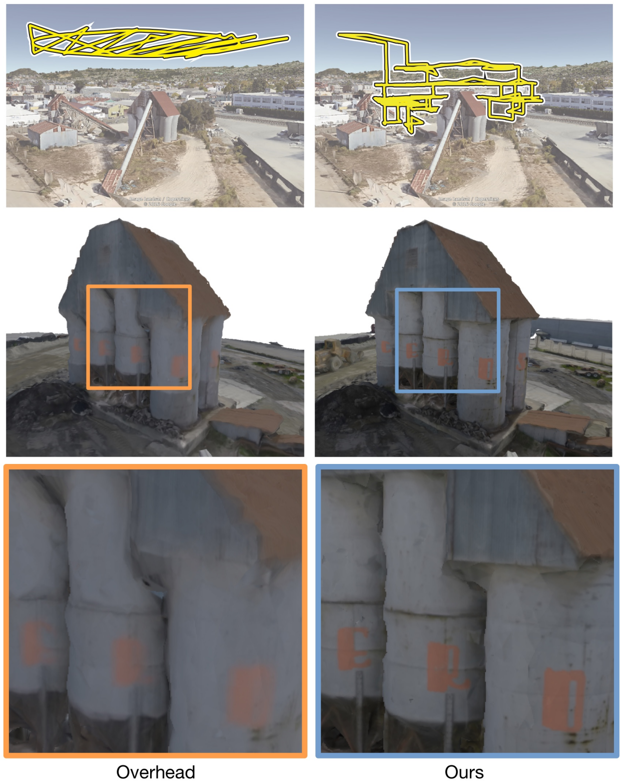


Figure 1. 3D reconstruction results produced using our algorithm for generating aerial 3D scanning trajectories, as compared to an overhead trajectory. Top row: A Google Earth visualization of the trajectories. Middle row: The 3D reconstructions obtained by flying a drone along each trajectory, capturing images, and feeding the images to multi-view stereo software. Bottom row: A close-up view of the 3D reconstructions. Our trajectories lead to noticeably more detailed 3D reconstructions than overhead trajectories. In all our experiments, we control for the flight time, battery consumption, number of images, and quality settings used in the 3D reconstruction.

1. Introduction

Small consumer drones equipped with high-resolution cameras, coupled with recent progress in 3D reconstruction algorithms for structure-from-motion and dense multi-view stereo, have spurred interest in large-scale 3D reconstruction from aerial imagery. In order to obtain high-quality 3D reconstructions, a drone must capture images that densely cover the scene. Additionally, 3D reconstruction methods typically require surfaces to be viewed from multiple viewpoints, at an appropriate distance, and with sufficient angular separation (i.e., baseline) between views. Existing autonomous flight planners do not always satisfy these requirements, which can be difficult to reason about, even for a skilled human pilot manually controlling the drone. Furthermore, the limited battery life of consumer drones provides only 10–15 minutes of flight time, making it even more challenging to obtain high-quality 3D reconstructions.

In lieu of manual piloting, commercial flight planning tools generate conservative trajectories (e.g., a lawnmower or orbit pattern at a safe height above the scene) that cover the scene while respecting flight time budgets [1, 42]. However, because these trajectories are generated with no awareness of the scene geometry, they tend to over-sample some regions (e.g., rooftops), while under-sampling others (e.g., facades, overhangs, and fine details), and therefore sacrifice quality.

We propose a method to automate aerial 3D scanning, by planning good camera trajectories for reconstructing large 3D scenes (see Figure 1). Our method relies on a mathematical model that evaluates the usefulness of a camera trajectory for the purpose of 3D reconstruction. Given a coarse estimate of the scene geometry as input, our model quantifies how well a trajectory covers the scene, and also quantifies the diversity and appropriateness of views along the trajectory. Using this model for scene coverage, our method generates trajectories that maximize coverage, subject to a travel budget. We bootstrap our method using coarse scene geometry, which we obtain using the imagery acquired from a simple initial flight over the scene.

We formulate our trajectory planning task as a reward-collecting graph optimization problem known as *orienteering*, that combines aspects of the traveling salesman and knapsack problems, and is known to be NP-hard [22, 54]. However, unlike the additive rewards in the standard orienteering problem, our rewards are non-additive, and globally coupled through our coverage model. We make the observation that our coverage model exhibits an intuitive diminishing returns property known as *submodularity* [33], and therefore we must solve a *submodular orienteering* problem. Although submodular orienteering is strictly harder than additive orienteering, it exhibits useful structure that can be exploited. We propose a novel transformation of our submodular orienteering problem into an additive orienteering problem, and we solve the additive problem as an integer linear program. We leverage submodularity extensively throughout the derivation of our method, to obtain approximate solutions with strong theoretical guarantees, and dramatically reduce computation times.

We demonstrate the utility of our method by using it to scan three large outdoor scenes: a barn, an office building, and an industrial site. We also quantitatively evaluate our algorithm in a photorealistic video game simulator. In all our experiments, we obtain significantly higher-quality 3D reconstructions than a strong baseline method.

2. Related Work

Aerial 3D Scanning and Mapping High-quality 3D reconstructions of very large scenes can be obtained using offline multi-view stereo algorithms [19] to process images acquired by drones [41]. Real-time mapping algorithms for drones have also been proposed, that take as input either RGBD [26, 35, 39, 50] or RGB [58] images, and produce as output a 3D reconstruction of the scene. These methods are solving a reconstruction problem, and do not, themselves, generate drone trajectories. Several commercially available flight planning tools have been developed to assist with 3D scanning [1, 42]. However, these tools only generate conservative lawnmower and orbit trajectories above the scene. In contrast, our algorithm generates trajectories that cover the scene as thoroughly as possible, ultimately leading to higher-quality 3D reconstructions.

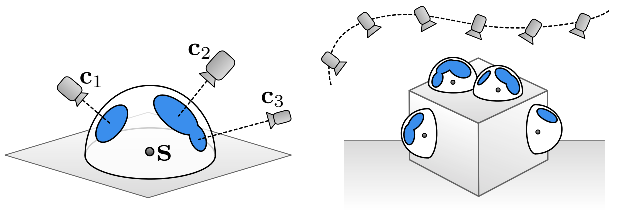
Generating trajectories that explore an unknown environment, while building a map of it, is a classical problem in robotics [52]. Exploration algorithms have been proposed for drones based on local search heuristics [55], identifying the frontiers between known and unknown parts of the scene [25, 47], maximizing newly visible parts of the scene [5], maximizing information gain [6, 7], and imitation learning [11]. A closely related problem in robotics is generating trajectories that cover a known environment [20]. Several coverage path planning algorithms have been proposed for drones [3, 4, 24]. In an especially similar spirit to our work, Heng et al. propose to reconstruct an unknown environment by executing alternating exploration and coverage trajectories [24]. However, existing strategies for exploration and coverage do not explicitly account for the domain-specific requirements of multi-view stereo algorithms (e.g., observing the scene geometry from a diverse set of viewing angles). Moreover, existing exploration and coverage strategies have not been shown to produce visually pleasing multi-view stereo reconstructions, and are generally not evaluated on multi-view stereo reconstruction tasks. In contrast, our trajectories cover the scene in a way that explicitly accounts for the requirements of multi-view stereo algorithms, and we evaluate the multi-view stereo reconstruction performance of our algorithm directly.

Several path planning algorithms have been proposed for drones, that explicitly attempt to maximize multi-view stereo reconstruction performance [16, 27, 40, 45]. These algorithms are similar in spirit to ours, but adopt a two phase strategy for generating trajectories. In the first phase, these algorithms select a sequence of *next-best-views* to visit, ignoring travel costs. In the second phase, they find an efficient path that connects the previously selected views. In contrast, our algorithm reasons about these two problems – selecting views and routing between them – jointly in a unified global optimization problem, enabling us to generate more rewarding trajectories.

View Selection and Path Planning The problem of optimizing the placement (and motion) of sensors to improve performance on a perception task is a classical problem in computer vision and robotics, where it generally goes by the name of *active vision* (see the comprehensive surveys [10, 46, 51]). We discuss directly related work not included in these surveys here. A variety of active algorithms for 3D scanning with ground-based range scanners have been proposed, that select a sequence of next-best-views [32], and then find an efficient path to connect the views [17, 64]. In a similar spirit to our work, Wang et al. propose a unified optimization problem that selects rewarding views, while softly penalizing travel costs [57]. We adapt these ideas to account for the domain-specific requirements of multi-view stereo algorithms, and we impose a hard constraint on travel budget, which is an important safety requirement when designing drone trajectories.

Several algorithms have been proposed to select an appropriate subset of views for multi-view stereo reconstruction [15, 28, 36, 37], and to optimize coverage of a scene [21, 38]. However, these methods do not model travel costs between views. In contrast, we impose a hard constraint on the travel cost of the path formed by the views we select.

Submodular Path Planning Submodularity [33] has been considered in path planning scenarios before, first in the theory community [8, 9], and more recently in the artificial intelligence [48, 49, 65] and robotics [24] communities. The coverage path planning formulation of Heng et al. [24] is similar to ours, in the sense that both formulations use the same technique for approximating coverage [29, 30]. We extend this formulation to account for the domain-specific requirements of multi-view stereo algorithms, and we evaluate the multi-view stereo reconstruction performance of our algorithm directly.



(a) Evaluating coverage for three cameras and a single surface point

(b) Evaluating coverage for a camera trajectory and multiple surface points

Figure 2. Our coverage model for quantifying the usefulness of camera trajectories for multi-view stereo reconstruction. More useful trajectories cover more of the hemisphere of viewing angles around surface points. (a) An illustrative example showing coverage of a single surface point with three cameras. Each camera covers a circular disk on a hemisphere around the surface point $\mathbf{s}$, and the total solid angle covered by all the disks determines the combined usefulness. Note that the angular separation (i.e., baseline) between cameras $\mathbf{c}_2$ and $\mathbf{c}_3$ is small and creates diminishing returns in their combined usefulness. (b) The usefulness of a camera trajectory over multiple surface points is determined by summing the total covered solid angle for each of the individual surface points. Our model naturally encourages diverse observations of the scene geometry, and encodes the eventual diminishing returns of additional observations.

3. Technical Overview

In order to generate scanning trajectories, our algorithm leverages a coarse estimate of the scene geometry. Initially, we do not have any estimate of the scene geometry, so we adopt an *explore-then-exploit* approach.

In the *explore* phase, we fly our drone (i.e., we command our drone to fly autonomously) along a default trajectory at a safe distance above the scene, acquiring a sequence of images as we are flying. We land our drone, and subsequently feed the acquired images to an open-source multi-view stereo pipeline, thereby obtaining a coarse estimate of the scene geometry, and a strictly conservative estimate of the scene's free space. We include a more detailed discussion of our *explore* phase in the supplementary material.

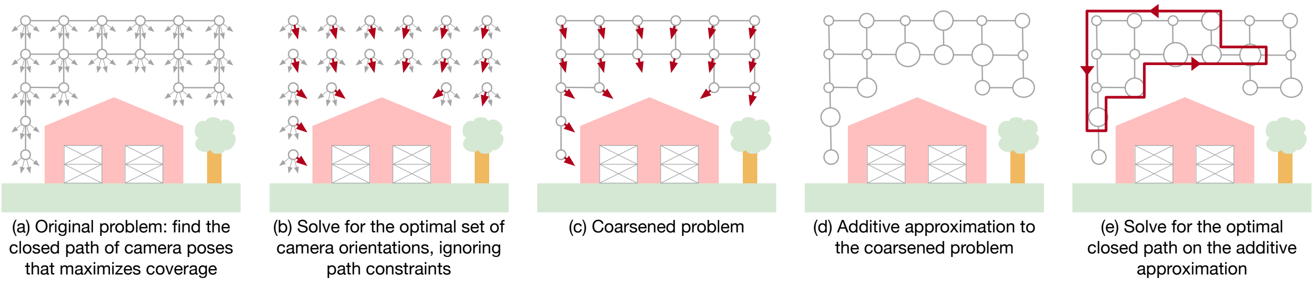
In the *exploit* phase, we use the additional information about the scene to plan a scanning trajectory that attempts to maximize the fidelity of the resulting 3D reconstruction. At the core of our planning algorithm, is a coverage model that accounts for the domain-specific requirements of multi-view stereo reconstruction (Section 4). Using this model, we generate a scanning trajectory that maximizes scene coverage, while respecting the drone's limited flight time (Section 5). We fly the drone along our scanning trajectory, acquiring another sequence of images. Finally, we land our drone again, and we feed all the images we have acquired to our multi-view stereo pipeline to obtain a detailed 3D reconstruction of the scene.

4. Coverage Model for Camera Trajectories

In this section, we model the usefulness of a camera trajectory for multi-view stereo reconstruction, in terms of how well it covers the scene geometry. We provide an overview of our coverage model in Figure 2.

In reality, the most useful camera trajectory is the one that yields the highest-quality 3D reconstruction of the scene. However, it is not clear how we would search for such a camera trajectory directly, without resorting to flying candidate trajectories and performing expensive 3D reconstructions for each of them. In contrast, our coverage model only roughly approximates the true usefulness of a camera trajectory. However, as we will see in the following section, our coverage model: (1) is motivated by established best practices for multi-view stereo image acquisition; (2) is easy to evaluate; (3) only requires a coarse estimate of the scene geometry as input; and (4) exhibits submodular structure, which will enable us to efficiently maximize it.

Best Practices for Multi-View Stereo Image Acquisition As a rule of thumb, it is recommended to capture an image every 5–15 degrees around an object, and it is generally accepted that capturing images more densely will eventually lead to diminishing returns in the fidelity of the 3D reconstruction [19]. Similarly, close-up and fronto-parallel views can help to resolve fine geometric details, because


(a) Original problem: find the closed path of camera poses that maximizes coverage

(b) Solve for the optimal set of camera orientations, ignoring path constraints

(c) Coarsened problem

(d) Additive approximation to the coarsened problem

(e) Solve for the optimal closed path on the additive approximation

Figure 3. Overview of our algorithm for generating camera trajectories that maximize coverage. (a) Our goal is to find the optimal closed path of camera poses through a discrete graph. (b) We begin by solving for the optimal camera orientation at every node in our graph, ignoring path constraints. (c) In doing so, we remove the choice of camera orientation from our problem, coarsening our problem into a more standard form. (d) The solution to the problem in (b) defines an approximation to our coarsened problem, where there is an additive reward for visiting each node. (e) Finally, we solve for the optimal closed path on the additive approximation defined in (d).

these views increase the effective resolution of estimated depth images, and contribute more reliable texture information to the reconstruction [56]. We explicitly encode these best practices for multi-view stereo image acquisition into our coverage model.

Formal Definition Given a candidate camera trajectory and approximate scene geometry as a triangle mesh, our goal is to quantify how well the trajectory covers the scene geometry. We first uniformly sample the camera trajectory to generate a discrete set C, consisting of individual camera poses $\mathbf{c}_{0:I}$. Similarly, we uniformly sample oriented surface points $\mathbf{s}_{0:J}$ from the scene geometry. For each oriented surface point $\mathbf{s}_j$, we define an oriented hemisphere H_j around it. For each surface point $\mathbf{s}_j$ and camera $\mathbf{c}_i$, we define a circular disk $\mathbf{d}_i^j$ that covers an angular region of the hemisphere H_j, centered at the location where $\mathbf{c}_i$ projects onto H_j (see Figure 2). When the surface point $\mathbf{s}_j$ is not visible from the camera $\mathbf{c}_i$, we define the disk $\mathbf{d}_i^j$ to have zero radius, and we truncate the extent of each disk so that it does not extend past the equator of H_j. We define the total covered region of the hemisphere H_j as the union of all the disks that partially cover H_j (see Figure 2), referring to this total covered region as $V_j = \bigcup_{i=0}^{I} \mathbf{d}_i^j$. We define our coverage model as follows,

$$f(C) = \sum_{j=0}^{J} \int_{V_j} w_j(\mathbf{h}) d\mathbf{h} \tag{1}$$

where the outer summation is over all hemispheres; $\int_{V_j} d\mathbf{h}$ refers to the surface integral over the covered region V_j; and $w_j(\mathbf{h})$ is a non-negative weight that assigns different reward values for covering different parts of H_j. Our model can be interpreted as quantifying how well a set of cameras covers the scene's *surface light field* [13, 59]. We include a method for efficiently evaluating our coverage model in the supplementary material.

To encourage close-up views, we set the radius of $\mathbf{d}_i^j$ to decay exponentially as the camera $\mathbf{c}_i$ moves away from the surface point $\mathbf{s}_j$. To encourage fronto-parallel views, we design each function $w_j(\mathbf{h})$ to decay in a cosine-weighted fashion, as the hemisphere location $\mathbf{h}$ moves away from the hemisphere pole. We include our exact formulation for $\mathbf{d}_i^j$ and $w_j(\mathbf{h})$ in the supplementary material.

Submodularity Roughly speaking, a set function is submodular if the marginal reward for adding an element to the input set always decreases, as more elements are added to the input set [33]. Our coverage model is submodular, because all coverage functions with non-negative weights are submodular [33]. Submodularity is a useful property to identify when attempting to optimize a set function, and is often referred to as the discrete analogue of convexity. We will leverage submodularity extensively in the following section, as we derive our algorithm generating camera trajectories that maximizing coverage.

5. Generating Optimal Camera Trajectories

We provide an overview of our algorithm in Figure 3. Our approach is to formulate a reward-collecting optimization problem on a graph. The nodes in the graph represent camera positions, the edges represent Euclidean distances between camera positions, and the rewards are collected by visiting new nodes. The goal is to find a path that collects as much reward as possible, subject to a budget constraint on the total path length. This general problem is known as the *orienteering problem* [22, 54].

A variety of approaches have been proposed to approximately solve the orienteering problem, which is NP-hard. However, these methods are not directly applicable to our problem, because they assume that the rewards on nodes are additive. But the total reward we collect in our problem is determined by our coverage model, which does not exhibit additive structure. In other words, the marginal reward we collect at a node might be very large, or very small, depending on the entire set of other nodes we visit.

The marginal reward we collect at each node also depends strongly on the orientation of our camera. In other words, our orienteering problem involves extra choices – how to orient the camera at each visited node – and these choices are globally coupled through our submodular coverage function. Therefore, even existing algorithms for *submodular orienteering* [8, 9, 24, 48, 49, 65] are not directly applicable to our problem, because these algorithms assume there are no extra choices to make at each visited node.

Our strategy will be to apply two successive problem transformations. First, we leverage submodularity to solve for the approximately optimal camera orientation at every node in our graph, ignoring path constraints (Fig. 3b, Section 5.1). In doing so, we remove the choice of camera orientation from our orienteering problem, thereby coarsening it into a more standard form (Fig. 3c). Second, we leverage submodularity to construct a tight additive approximation of our coverage function (Fig. 3d, Section 5.2). In doing so, we relax our coarsened submodular orienteering problem into a standard additive orienteering problem. We formulate this additive orienteering problem as a compact integer linear program, and solve it approximately using a commercially available solver (Fig. 3e, Section 5.3).

Preprocessing We begin by constructing a discrete set of all the possible camera poses we might include in our path. We refer to this set as our *ground set* of camera poses, C. We construct this set by uniformly sampling a user-defined bounding box that spans the scene, then uniformly sampling a downward-facing unit hemisphere to produce a set of look-at vectors that our drone camera can achieve. We define our ground set as the Cartesian product of these positions and look-at vectors.

We construct the graph for our orienteering problem as follows. First, we construct the king's graph (i.e., an undirected weighted grid graph that includes diagonal edges) of all the unique camera positions in C. Second, we prune the graph so it is entirely restricted to the known free space in the scene (see Section 3).

Our Submodular Orienteering Problem Let $\mathbf{P} = (\mathbf{p}_0, \mathbf{v}_0), (\mathbf{p}_1, \mathbf{v}_1), \ldots, (\mathbf{p}_q, \mathbf{v}_q)$ be a camera path through our graph, represented as a sequence of camera poses taken from our ground set. We represent each camera pose as a position $\mathbf{p}_i$ and a look-at vector $\mathbf{v}_i$. Let $C_\mathbf{P} \subseteq C$ be the set of all the unique camera poses in the path $\mathbf{P}$. We would like to find the optimal path as follows,

$$\begin{aligned} \mathbf{P}^\star &= \arg\max_{\mathbf{P}} f(C_\mathbf{P}) \\ \text{subject to} \quad & l(\mathbf{P}) \leq B \quad \mathbf{p}_0 = \mathbf{p}_q = \mathbf{p}_\text{root} \end{aligned} \tag{2}$$

where $l(\mathbf{P})$ is the length of the path; B is a user-defined maximum budget on path length; and $\mathbf{p}_\text{root}$ is the position where our path must start and end. For safety reasons, we would also like to design trajectories that consume close to, but no more than, some fixed fraction of our drone's battery (e.g., 80% or so). However, constraining battery consumption directly is difficult to express in our orienteering formulation, so we model this constraint indirectly by imposing a budget constraint on path length.

We make the observation that our problem is intractable in its current form, because it requires searching over an exponential number of paths through our graph. This observation motivates the following two problem transformations.

5.1. Solving for Optimal Camera Orientations

Our goal in this subsection is to solve for the optimal camera orientation at every node in our graph, ignoring path constraints. We achieve this goal with the following relaxation of the problem in equation (2). Let $C_S \subseteq C$ be a subset of camera poses from our ground set. We would like to find the optimal subset of camera poses as follows,

$$\begin{aligned} C_S^\star &= \arg\max_{C_S} f(C_S) \\ \text{subject to} \quad & |C_S| = N \quad C_S \in \mathcal{M} \end{aligned} \tag{3}$$

where $|C_S|$ is the cardinality of C_S; N is the total number of unique positions in our graph; and the constraint $C_S \in \mathcal{M}$ enforces mutual exclusion, where we are allowed to select at most one camera orientation at each node in our graph.

Intuitively, in this relaxed problem, we are attempting to maximize coverage by selecting exactly one camera orientation at each node in our graph. We can interpret such a solution as a coarsened ground set for the problem in equation (2), thereby transforming it into a standard submodular orienteering problem.

Because our coverage function is submodular, the problem in equation (3) can be solved very efficiently, and to within 50% of global optimality, with a very simple greedy algorithm [33]. Roughly speaking, the greedy algorithm selects camera poses from our ground set in order of marginal reward, taking care to respect the mutual exclusion constraint, until no more elements can be selected. Submodularity can also be exploited to significantly reduce the computation time required by the greedy algorithm (e.g., from multiple hours to less than a minute, for the problems we consider in this paper) [33]. The approximation guarantee in this subsection relies on the fact that selecting more camera poses never reduces coverage, i.e., our coverage function exhibits a property known as *monotonicity* [33]. We include a more detailed discussion of the greedy algorithm, and provide pseudocode, in the supplementary material.

5.2. Additive Approximation of Coverage

Our goal in this subsection is to construct an additive approximation of coverage. In other words, we would like to define an additive reward at each node in our graph, that closely approximates our coverage function.

To construct our additive approximation, we draw inspiration from the approach of Iyer et al. [29, 30]. First, we choose a permutation of elements in our coarsened ground set. Second, we define the additive reward for each element in our permutation, as the true marginal reward we would get by adding it to an input set, assuming we add elements to the input set in permuted order. Due to submodularity, such an additive approximation is guaranteed either to be exact, or to underestimate our coverage function. This guarantee is useful for our purposes, because any solution we get from optimizing our additive approximation, will yield an equal or greater reward on our true coverage function.

When choosing a permutation, we make the observation that subsets taken from the front of the permutation are approximated most accurately. So, for our purposes, it is advantageous to place the most valuable camera poses at the front of our permutation. With this intuition in mind, we form our permutation by greedily ordering the camera poses in our coarsened ground set according to their marginal reward. Fortunately, we have already computed this ordering in Section 5.1 using the greedy algorithm. So, we simply reuse this ordering, and the corresponding additive reward at each node, to construct our additive approximation.

5.3. Orienteering as an Integer Linear Program

After constructing our additive approximation of coverage, we obtain the following additive orienteering problem,

$$\begin{aligned} \mathbf{P}^{\star} &= \arg\max_{\mathbf{P}} \sum_{C_{\mathbf{P}}} \tilde{f}_i \\ \text{subject to} \quad & l(\mathbf{P}) \leq B \quad \mathbf{p}_0 = \mathbf{p}_q = \mathbf{p}_{\text{root}} \end{aligned} \tag{4}$$

where $\tilde{f}_i$ is the additive reward for each unique node along the path $\mathbf{P}$. In its current form, it is still not clear how to solve this problem efficiently, because we must still search over an exponential number of paths through our graph. Fortunately, we can express this problem as a compact integer linear program, using a formulation suggested by Letchford et al. [34]. The main insight in this approach is to transform our undirected graph into a directed graph. Then, we define integer variables to represent if nodes are visited and directed edges are traversed. Remarkably, we can constrain the configuration of these integer variables to form only valid paths through our graph, with a compact set of linear constraints. We include a more detailed derivation of this formulation in the supplementary material.

Leveraging the formulation suggested by Letchford et al., we convert the problem in equation (4) into a standard form that can be given directly to an off-the-shelf solver. We use the modeling language CVXPY [14] to specify our problem, and we use the commercially available Gurobi Optimizer [23] as the back end solver. Solving integer programming problems to global optimality is NP-hard, and can take a very long time, so we specify a solver time limit of 5 minutes. Gurobi returns the best feasible solution it finds within the time limit, along with a worst-case optimality gap. In our experience, Gurobi consistently converges to a close-to-optimal solution in the allotted time (i.e., typically within 70% of global optimality). At this point, the resulting orienteering trajectory can be safely and autonomously executed on our drone.

6. Evaluation

In all the experiments described in this section, we execute all drone flights at 2 meters per second, with a total travel budget of 960 meters (i.e., an 8 minute flight) unless otherwise noted. All flights generate 1 image every 3.5 meters. Each method has the same travel budget, and generates roughly 275 images. Small variations in the number of generated images are possible, due to differences in how close each method gets to the travel budget. We describe our drone hardware and data acquisition pipeline in more detail in the supplementary material.

Real-World Reconstruction Performance We scanned three large outdoor scenes to evaluate our algorithm: a barn, an office building, and an industrial site. We compared our results to two baseline methods: OVERHEAD and RANDOM. We show results from these experiments in Figure 4, and as well as in supplementary material.

OVERHEAD. We designed OVERHEAD to generate trajectories that are representative of those produced by existing commercial flight planning software [1, 42]. OVERHEAD generates a single flight at at a safe height above the scene; consisting of an orbit path that always points the camera at the center of the scene; followed by a lawnmower path that always points the camera straight down.

RANDOM. We designed RANDOM to have roughly the same level of scene understanding as our algorithm, except that RANDOM does not optimize our coverage function. We gave RANDOM access to the graph of camera positions generated by our algorithm, which has been pruned according to the free space in the scene. RANDOM generates trajectories by randomly selecting graph nodes, and finds an efficient path to connect them using the Approx-TSP algorithm [12]. RANDOM continues selecting nodes until no more nodes can be added, due to the travel budget. RANDOM always points the camera towards the center of the scene, which is a reasonable strategy for the scenes we consider in this paper.

We configured our algorithm as follows. During our *explore* phase, we generate an orbit trajectory exactly as we do for OVERHEAD. For the scenes we consider in this paper, this initial orbit trajectory is always less than 200 meters. During our *exploit* phase, we generate trajectories using the

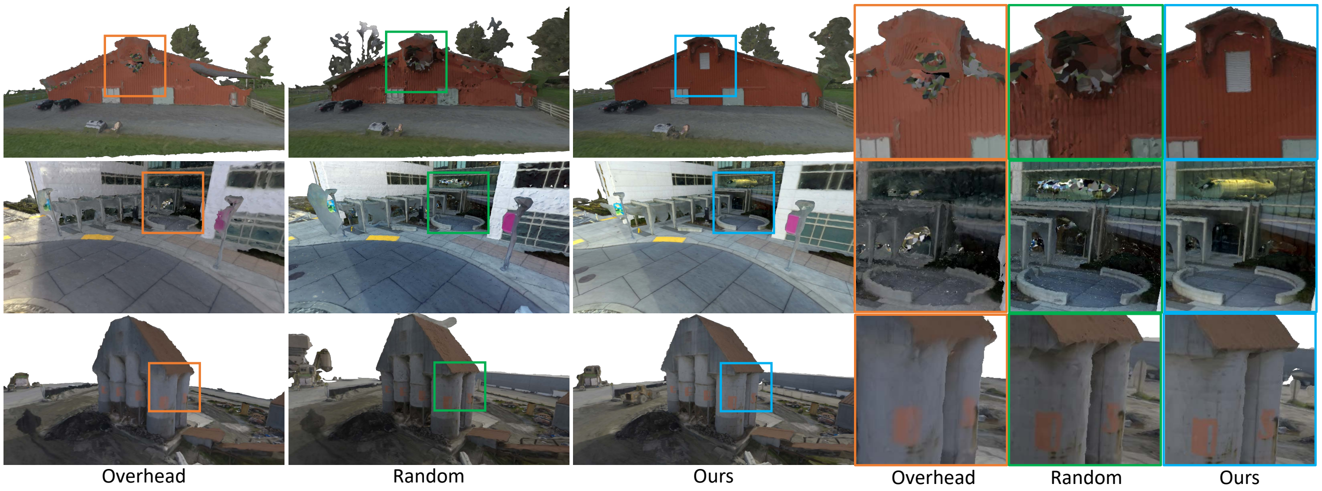


Figure 4. Qualitative comparison of the 3D reconstructions produced by an overhead (left), a random (middle), and our (right) trajectories for our real-world scenes. Our results are noticeably higher-quality than would be possible to obtain otherwise. Please view our supplemental materials for high-resolution renderings. For all experiments, we control for the flight time, battery consumption, number of images, and quality settings used in the 3D reconstruction.

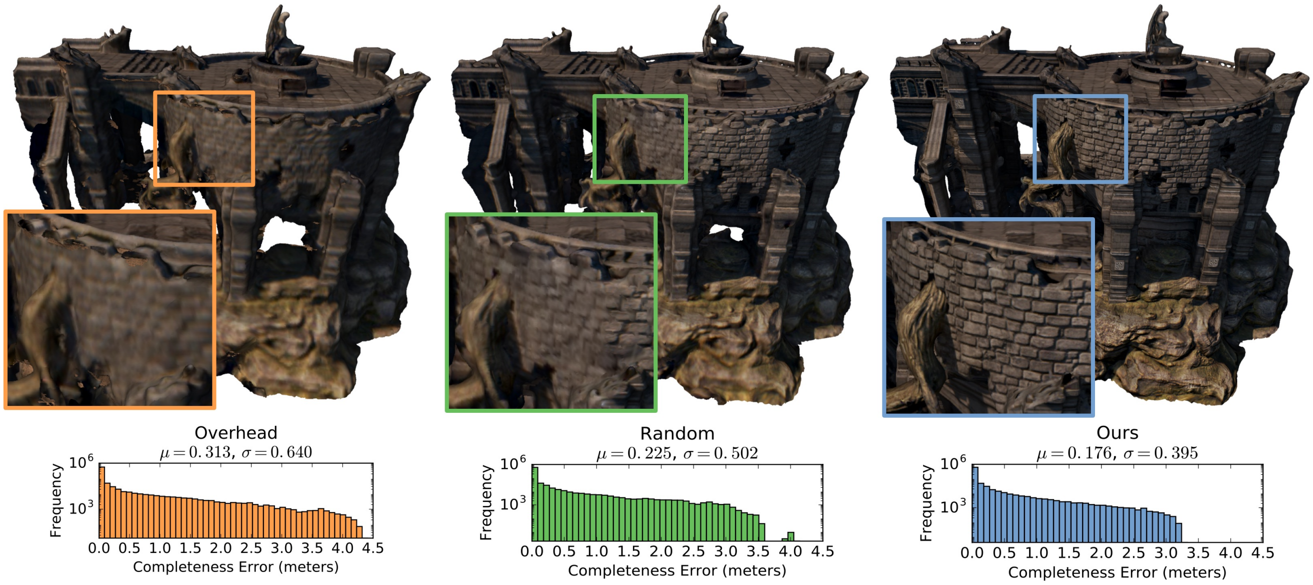


Figure 5. Quantitative comparison of the 3D reconstructions produced by an overhead (left), a random (middle), and our (right) trajectories for our synthetic GRASSLANDS scene. We show the histograms of completeness errors (i.e., the distances from each ground truth point to the closest reconstructed point), as well as the mean μ and standard deviation σ of these error histograms. The y-axis of each histogram is on a log scale, more histogram mass to the left is better, and a smaller mean and standard deviation is better. Our results are quantitatively and qualitatively higher-quality than would be possible to obtain otherwise.

approach described in Section 5.

When generating 3D reconstructions, our algorithm and RANDOM have access to the images we collect during our *explore* phase, but OVERHEAD does not. The images in our *explore* phase are nearly identical to the orbit images from OVERHEAD, and would therefore provide OVERHEAD with negligible additional information, so all three methods are directly comparable. We generate 3D reconstructions with the commercially available Pix4Dmapper Pro software [43], configured with maximum quality settings.

Reconstruction Performance on a Synthetic Scene As shown in Figure 5, we also evaluated our algorithm using a photorealistic video game simulator, which enabled us

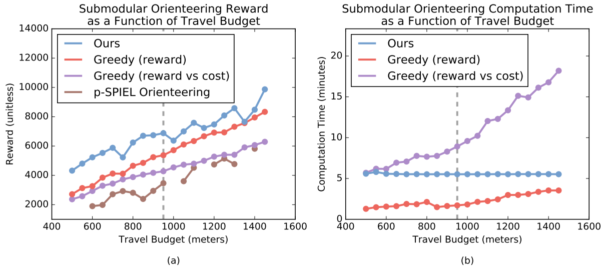


Figure 6. Quantitative comparison of submodular orienteering algorithms on our synthetic GRASSLANDS scene. (a) Submodular reward as a function of travel budget. Our algorithm consistently obtains more reward than other algorithms. All results in this paper were produced with a budget of 960 meters (i.e., 8 minutes at 2 meters per second) shown with a grey dotted line. For this budget, we obtain 27% more reward than the next best algorithm. The p-SPIEL Orienteering algorithm [49] failed to consistently find a solution. (b) Computation time as a function of travel budget. On this plot, lower is better. Unlike the greedy algorithms, the computation time of our algorithm is stable across a range of travel budgets. We do not show the p-SPIEL Orienteering algorithm because it took over an hour in all cases where it was successful.

to measure reconstruction performance relative to known ground truth geometry. Our experimental design here is exactly as described previously, except we acquired images by programmatically maneuvering a virtual camera in the Unreal Engine [53], using the UnrealCV Python library [44].

We chose the GRASSLANDS environment as our Unreal test scene, because it is freely available, has photorealistic lighting and very detailed geometry, and depicts a large outdoor scene that is well-suited for 3D scanning with a drone.

We quantitatively evaluated performance by measuring *accuracy* and *completeness*, as described by Aanæs et al. [2], relative to a ground truth point cloud obtained by rendering reference depth images arranged on an inward-looking sphere around the scene, taking care to manually remove any depth images that were inside objects. We obtained the point clouds for our approach by running VisualSFM [60, 61, 62, 63], followed by the Multi-View Environment [18] software configured with maximum quality settings, on the images generated by each method. We generated the textured 3D models in Figure 5 by running the Screened Poisson Surface Reconstruction algorithm [31], followed by the texturing algorithm of Waechter et al. [56].

The mean accuracy error (lower is better) of each algorithm was: OVERHEAD 0.179 meters; RANDOM 0.157 meters; our algorithm 0.197 meters. The mean completeness error (lower is better) of each algorithm was: OVERHEAD 0.313 meters; RANDOM 0.225 meters; our algorithm 0.176 meters. We observe that our algorithm performs slightly worse than the other algorithms in terms of accuracy, but significantly better in terms of completeness. This improvement in completeness results in noticeably higher visual quality, as shown in Figure 5.

Submodular Orienteering Performance We evaluated the submodular orienteering performance of our algorithm on our synthetic GRASSLANDS scene. We show results from this experiment in Figure 6.

We perform this experiment after we have solved for the optimal camera orientation at every node in our graph, to facilitate the comparison of our algorithm to other submodular orienteering algorithms [49, 65]. The Greedy (reward) algorithm selects nodes according to their marginal submodular reward, and finds an efficient path to connect them using the Approx-TSP algorithm [12] until no more nodes can be added, due to the travel budget. This algorithm is intended to be representative of the *next-best-view* planning strategies that occur frequently in the literature [17, 32, 64], including those that have been applied to aerial 3D scanning [16, 27, 40, 45]. The Greedy (reward versus cost) algorithm behaves similarly, except this algorithm selects nodes according to the ratio of marginal reward to marginal cost [65]. We implemented all algorithms in Python, except for the p-SPIEL Orienteering algorithm [49], where we used the MATLAB implementation provided by the authors. We performed this experiment on a Late 2013 Macbook Pro with a 2.6 GHz Intel Core i7 processor and 16GB of RAM.

7. Discussion and Future Work

We proposed an intuitive and computationally efficient method for aerial 3D scanning, that reasons jointly about coverage rewards and travel costs, and results in significantly higher-quality 3D reconstructions than baseline methods. Yet, our work has some limitations, and suggests several exciting opportunities for future work.

Two optimality gaps are introduced in our method, where we first solve for the approximately optimal set of camera orientations, and subsequently solve for the approximately optimal path to an additive orienteering problem. Although we show strong empirical results against baseline methods, there is an opportunity to better understand the effects of these approximations in each stage of our method.

There is also an opportunity to investigate other ways to model the usefulness of camera trajectories, that more faithfully capture the true 3D reconstruction process, while still being computationally tractable to optimize.

In future work, we are interested in considering confidence and uncertainty in our 3D reconstructions more explicitly, such that our planned trajectories could prioritize gathering observations from low-confidence and unobserved areas. Ultimately, we expect to run our method in an iterative fashion, where each pass could discover more of an initially unknown scene. We are optimistic that this iterative approach would enable our method to scale up to very large scenes (e.g., an entire university campus).

Acknowledgements

We thank Jim Piavis and Ross Robinson for their expertise as our safety pilots, and for their assistance in collecting data; Don Gillett for granting us permission to scan the barn scene; 3D Robotics for granting us permission to scan the industrial scene; Weichao Qiu for his assistance with the UnrealCV Python library; and Jane E and Abe Davis for proofreading the paper.

References

[1] 3D Robotics. Site Scan. `http://3dr.com`, 2017.

[2] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl. Large-scale data for multiple-view stereopsis. *IJCV*, 120(2), 2016.

[3] K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes. Uniform coverage structural inspection path-planning for micro aerial vehicles. In *IEEE International Symposium on Intelligent Control 2015*.

[4] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart. Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *International Conference on Robotics and Automation (ICRA) 2015*.

[5] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon "next-best-view" planner for 3D exploration. In *International Conference on Robotics and Automation (ICRA) 2016*.

[6] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar. Information-theoretic planning with trajectory optimization for dense 3D mapping. In *Robotics: Science and Systems (RSS) 2015*.

[7] B. Charrow, S. Liu, V. Kumar, and N. Michael. Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information. In *International Conference on Robotics and Automation (ICRA) 2015*.

[8] C. Chekuri, N. Korula, and M. Pal. Improved algorithms for orienteering and related problems. *Transactions on Algorithms*, 8(3), 2012.

[9] C. Chekuri and M. Pal. A recursive greedy algorithm for walks in directed graphs. In *Foundations of Computer Science (FOCS) 2005*.

[10] S. Chen, Y. Li, and N. M. Kwok. Active vision in robotic systems: A survey of recent developments. *International Journal of Robotics Research*, 30(11), 2011.

[11] S. Choudhury, A. Kapoor, G. Ranade, and D. Dey. Learning to gather information via imitation. *International Conference on Robotics and Automation (ICRA) 2017*.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. MIT Press, 2009.

[13] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. *Computer Graphics Forum (Proc. Eurographics 2012)*, 31(2, Part 1), 2012.

[14] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83), 2016.

[15] E. Dunn and J.-M. Frahm. Next best view planning for active model improvement. In *Intelligent Robots and Systems (IROS) 2009*.

[16] E. Dunn, J. van den Berg, and J.-M. Frahm. Developing visual sensing strategies through next best view planning. In *Intelligent Robots and Systems (IROS) 2009*.

[17] X. Fan, L. Zhang, B. Brown, and S. Rusinkiewicz. Automated view and path planning for scalable multi-object 3D scanning. *Transactions on Graphics (Proc. SIGGRAPH Asia 2016)*, 35(6), 2016.

[18] S. Fuhrmann, F. Langguth, N. Moehrle, M. Waechter, and M. Goesele. MVE-An image-based reconstruction environment. *Computer Graphics Forum*, 53(Part A), 2015.

[19] Y. Furukawa and C. Hernandez. *Multi-View Stereo: A Tutorial*. Now Publishers, 2015.

[20] E. Galceran and M. Carreras. A survey of coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12), 2013.

[21] B. Ghanem, Y. Cao, and P. Wonka. Designing camera networks by convex quadratic programming. *Computer Graphics Forum (Proc. Eurographics 2015)*, 34(2), 2015.

[22] A. Gunawan, H. C. Laua, and P. Vansteenwegenb. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2), 2016.

[23] Gurobi. Gurobi Optimizer. `http://www.gurobi.com`, 2017.

[24] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys. Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *International Conference on Robotics and Automation (ICRA) 2015*.

[25] L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Autonomous visual mapping and exploration with a micro aerial vehicle. *Journal of Field Robotics*, 31(4), 2014.

[26] L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. Real-time photo-realistic 3D mapping for micro aerial vehicles. In *Intelligent Robots and Systems (IROS) 2011*.

[27] C. Hoppe, A. Wendel, S. Zollmann, K. Pirker, A. Irschara, H. Bischof, and S. Kluckner. Photogrammetric camera network design for micro aerial vehicles. In *Computer Vision Winter Workshop 2012*.

[28] A. Hornung, B. Zeng, and L. Kobbelt. Image selection for improved multi-view stereo. In *CVPR 2008*.

[29] R. Iyer and J. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS 2013*.

[30] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML 2013*.

[31] M. Kazhdan and H. Hoppe. Screened Poisson surface reconstruction. *Transactions on Graphics*, 32(3), 2013.

[32] M. Krainin, B. Curless, and D. Fox. Autonomous generation of complete 3D object models using next best view manipulation planning. In *International Conference on Robotics and Automation (ICRA) 2011*.

[33] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014.

[34] A. N. Letchford, S. D. Nasirib, and D. O. Theis. Compact formulations of the Steiner traveling salesman problem and related problems. *European Journal of Operational Research*, 228(1), 2013.

[35] G. Loianno, J. Thomas, and V. Kumar. Cooperative localization and mapping of MAVs using RGB-D sensors. In *International Conference on Robotics and Automation (ICRA) 2015*.

[36] M. Mauro, H. Riemenschneider, L. V. Gool, A. Signoroni, and R. Leonardi. A unified framework for content-aware view selection and planning through view importance. In *BMVC 2014*.

[37] M. Mauro, H. Riemenschneider, A. Signoroni, R. Leonardi, and L. V. Gool. An integer linear programming model for view selection on overlapping camera clusters. In *3DV 2014*.

[38] A. Mavrinac and X. Chen. Modeling coverage in camera networks: A survey. *IJCV*, 101(1), 2013.

[39] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5), 2012.

[40] C. Mostegel, M. Rumpler, F. Fraundorfer, and H. Bischof. UAV-based autonomous image acquisition with multi-view stereo quality assurance by confidence prediction. In *CVPR Workshop on Computer Vision in Vehicle Technology 2016*.

[41] Pix4D. Projeto redentor white paper, 2015.

[42] Pix4D. Pix4Dcapture. `http://pix4d.com/product/pix4dcapture`, 2017.

[43] Pix4D. Pix4Dmapper Pro. `http://pix4d.com/product/pix4dmapper-pro`, 2017.

[44] W. Qiu and A. Yuille. UnrealCV: Connecting computer vision to Unreal Engine. arXiv, 2016.

[45] K. Schmid, H. Hirschmuller, A. Domel, I. Grixa, M. Suppa, and G. Hirzinger. View planning for multi-view stereo 3D reconstruction using an autonomous multicopter. *Journal of Intelligent & Robotic Systems*, 65(1), 2012.

[46] W. R. Scott, G. Roth, and J.-F. Rivest. View planning for automated three-dimensional object reconstruction and inspection. *Computing Surveys*, 35(1), 2003.

[47] S. Shen, N. Michael, and V. Kumar. Autonomous indoor 3D exploration with a micro-aerial vehicle. In *International Conference on Robotics and Automation (ICRA) 2012*.

[48] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34(1), 2009.

[49] A. Singh, A. Krause, and W. J. Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *International Joint Conference on Artifical Intelligence (IJCAI) 2009*.

[50] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. Dense tracking and mapping with a quadrocopter. In *Unmanned Aeriel Vehicles in Geomatics 2013*.

[51] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *Transactions on Robotics and Automation*, 11(1), 1995.

[52] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[53] Unreal. Unreal Engine. `http://www.unrealengine.com`, 2017.

[54] P. Vansteenwegena, W. Souffriaua, and D. V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1), 2011.

[55] L. von Stumberg, V. Usenko, J. Engel, J. Stuckler, and D. Cremers. Autonomous exploration with a low-cost quadrocopter using semi-dense monocular SLAM. arXiv, 2016.

[56] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! Large-scale texturing of 3D reconstructions. In *ECCV 2014*.

[57] P. Wang, R. Krishnamurti, and K. Gupta. View planning problem with combined view and traveling cost. In *International Conference on Robotics and Automation (ICRA) 2007*.

[58] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *CVPR 2012*.

[59] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface light fields for 3D photography. *Transactions on Graphics (Proc. SIGGRAPH 2000)*, 35(1), 2000.

[60] C. Wu. Towards linear-time incremental structure from motion. In *3DV 2013*.

[61] C. Wu. SiftGPU: A GPU implementation of scale invaraint feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[62] C. Wu. VisualSFM: A visual structure from motion system. `http://ccwu.me/vsfm`, 2011.

[63] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *CVPR 2011*.

[64] S. Wu, W. Sun, P. Long, H. Huang, D. Cohen-Or, M. Gong, O. Deussen, and B. Chen. Quality-driven Poisson-guided autoscanning. *Transactions on Graphics (Proc. SIGGRAPH Asia 2014)*, 33(6), 2014.

[65] H. Zhang and Y. Vorobeychik. Submodular optimization with routing constraints. In *Conference on Artificial Intelligence (AAAI) 2016*.