AeroTraj: Trajectory Planning for Fast, and Accurate 3D Reconstruction using a Drone-based LiDAR

FAWAD AHMAD, Rochester Institute of Technology, USA
CHRISTINA SUYONG SHIN, University of Southern California, USA
RAJRUP GHOSH, University of Southern California, USA
JOHN D'AMBROSIO, University of Southern California, USA
EUGENE CHAI, Nokia Bell Labs, USA
KARTHIKEYAN SUNDARESAN, Georgia Institute of Technology, USA
RAMESH GOVINDAN, University of Southern California, USA

This paper presents AeroTraj, a system that enables fast, accurate, and automated reconstruction of 3D models of large buildings using a drone-mounted LiDAR. LiDAR point clouds can be used directly to assemble 3D models if their positions are accurately determined. AeroTraj uses SLAM for this, but must ensure complete and accurate reconstruction while minimizing drone battery usage. Doing this requires balancing competing constraints: drone speed, height, and orientation. AeroTraj exploits building geometry in designing an optimal trajectory that incorporates these constraints. Even with an optimal trajectory, SLAM's position error can drift over time, so AeroTraj tracks drift in-flight by offloading computations to the cloud and invokes a re-calibration procedure to minimize error. AeroTraj can reconstruct large structures with centimeter-level accuracy and with an average end-to-end latency below 250 ms, significantly outperforming the state of the art.

CCS Concepts: • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools.*

Additional Key Words and Phrases: 3D Reconstruction, Mapping, Trajectory Planning, Localization

ACM Reference Format:
Fawad Ahmad, Christina Suyong Shin, Rajrup Ghosh, John D'Ambrosio, Eugene Chai, Karthikeyan Sundaresan, and Ramesh Govindan. 2023. AeroTraj: Trajectory Planning for Fast, and Accurate 3D Reconstruction using a Drone-based LiDAR. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 3, Article 83 (September 2023), 27 pages. https://doi.org/10.1145/3610911

1 INTRODUCTION

Drones have developed to the point where they can be equipped with on-board compute, cellular radios (LTE) and sophisticated sensors like stereo cameras, and LiDARs, *etc.* This has spurred interest in drone-based *3D reconstruction* of buildings and other large structures for construction site monitoring [4], damage assessment [1,

Authors' addresses: Fawad Ahmad, fawad@cs.rit.edu, Rochester Institute of Technology, USA; Christina Suyong Shin, cshin956@usc.edu, University of Southern California, USA; Rajrup Ghosh, rajrupgh@usc.edu, University of Southern California, USA; John D'Ambrosio, jfdambro@usc.edu, University of Southern California, USA; Eugene Chai, eugene.chai@nokia-bell-labs.com, Nokia Bell Labs, USA; Karthikeyan Sundaresan, karthik@ece.gatech.edu, Georgia Institute of Technology, USA; Ramesh Govindan, ramesh@usc.edu, University of Southern California, USA.

2474-9567/2023/9-ART83 $15.00
https://doi.org/10.1145/3610911

3, 7, 57], and documenting repair and retrofit tasks [5]. In these applications, a drone captures imagery outside a building, and uses it to generate a *3D model* of the building.

The term *3D model* covers a range of geometric representations of the surfaces of objects, from coarse-grained approximations (cylinders, cubes, intersection of planes), to more fine-grained representations such as *meshes* (small-scale surface tessellations that capture structural variations). In this paper, we seek to extract an even finer-grained *point-cloud* of a large structure (*e.g.,* a building) which consists of dense *points* on the surface of the structure. Each point has an associated 3D position, along with other attributes (depending on the sensor used to generate the point-cloud). A point-cloud based 3D model can generate all other representations.

Today, to build a 3D model of a building, one can fly a drone around the building in a given trajectory to collect 2D images then use *photogrammetry* which infers a 3D model from a sequence of 2D images [10, 26, 70]. Photogrammetry is *computationally intensive* and can require multiple iterations, each requiring human intervention (§2).

Unlike photogrammetry, LiDAR-based reconstruction can more directly infer 3D models, because LiDARs directly provide depth information (unlike cameras). A LiDAR sensor measures distances to surfaces using lasers mounted on a mechanical rotating platform. The number of lasers determines the resolution of the LiDAR. With each revolution of the lasers, the LiDAR returns a point cloud, or a *LiDAR frame*. The point cloud is a set of 3D data points, each corresponding to a distance measurement of a particular position of the surrounding environment from the LiDAR.

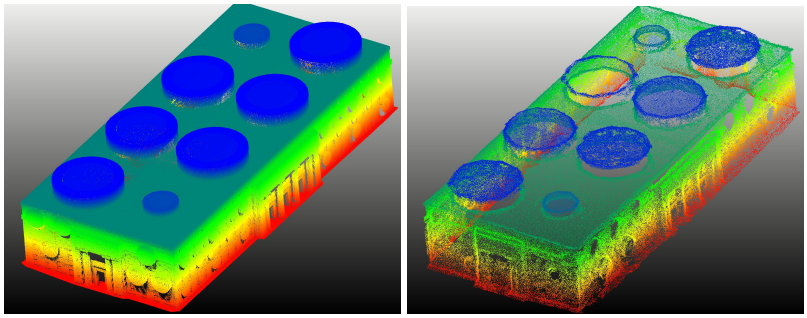
To obtain a high quality reconstruction of a structure such as a building, one can mount a LiDAR on a drone, scan the building by flying the drone around it, and *merge* points clouds captured from different locations around the building. Consider point clouds p and p'. A point x on the surface of the building may appear both in p and p'. However, since the drone has moved, this point x appears at different positions (relative to the LiDAR) in the two point clouds. If we can position the drone accurately at each instant, we can precisely transform both point clouds to the same coordinate frame of reference. Then, the union of points in p and p' constitutes part of the 3D model of the building. To obtain the complete model, the drone must scan the entire building using a *trajectory* of minimal flight duration to minimize drone battery usage.

GPS is inadequate to position the drone accurately (§2), but LiDAR SLAM (Simultaneous Localization and Mapping [19, 28]) is a potential candidate. LiDAR SLAM can provide centimeter-level positioning for autonomous vehicles. However, in our setting, off-the-shelf LiDAR SLAM does not work well (§2), because we use the LiDAR for positioning *and* reconstruction *and* must respect drone battery constraints. These goals are mutually in conflict (§3) because SLAM determines pose transformations between successive point clouds, and accurate transformations need a significant number of *overlapped* points between point clouds. To ensure overlap:

- The drone can fly slowly, but this results in increased flight duration.
- It can fly further from the building surface. This way, the LiDAR can capture more of the surface, and the drone can fly faster. However, LiDAR point resolution decreases with distance, resulting in a poor reconstruction.
- Even if we can address these competing constraints, SLAM's position estimates can drift over time. If not corrected, drift can adversely impact reconstruction quality.

In this paper, we describe the design, implementation and evaluation of AeroTraj, which addresses these challenges using two insights: it (a) exploits building geometry to design a scan trajectory that enables high quality reconstruction while minimizing flight time, and (b) detects and corrects SLAM drift in-flight. The paper makes the following contributions (§3):

- It presents a novel optimization formulation that generates minimum-flight-time *model collection* trajectories for a large class of buildings while respecting speed, height, and LiDAR orientation and resolution constraints.
- It proposes a cloud-offload architecture for automated in-flight drift estimation, so drone flights can be *re-calibrated* when drift becomes too high.



(a) Ground truth 3D model. (b) MVS/photogrammetry-dervied 3D model.

Fig. 1. Compared to a ground truth 3D model (a), an MVS/photogrammetry-based 3D reconstruction (b) of a building can be *incomplete i.e.,* it contains holes and missing regions.

Building	Accuracy (m)	Completeness (m)	Time (s)
Small	0.11	0.80	23084
Large	0.16	0.75	31600

Table 1. Photogrammetry can be slow, inaccurate and incomplete. The small building is 50m x 50m x 20m (length x breadth x height), the large building is 100m x 50m x 20m.

- It uses a fast and efficient *reconnaissance* flight design to estimate building geometry parameters for trajectory generation.

As a by-product of this design, AeroTraj can construct models on-the-fly: the 3D model is available within hundreds of milliseconds of flight completion. Experiments (§4) with a complete AeroTraj implementation on real-world flights and a photorealistic drone simulator (AirSim [66]) demonstrate that AeroTraj can reconstruct 3D models to within 10 cm accuracy, for a variety of building shapes and sizes. Moreover, AeroTraj is *faster*, and *more accurate* than existing state-of-the-art approaches, photogrammetry [64], and LiDAR-based 3D reconstruction [38, 77].

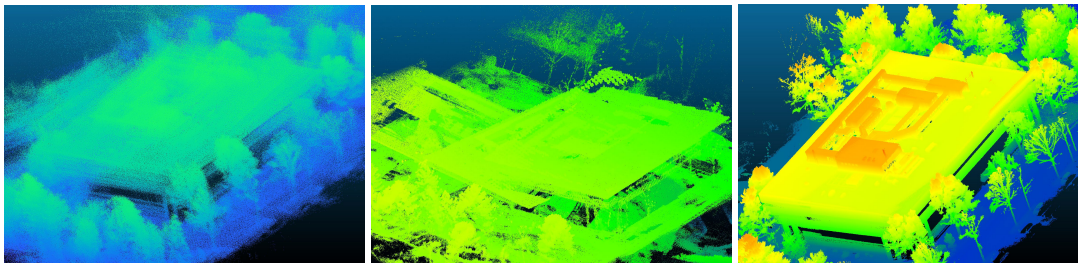
2 BACKGROUND AND MOTIVATION

In this section, we describe metrics that capture the quality of 3D reconstruction, and results from experiments that motivate AeroTraj.

2.1 Measures of 3D model Quality

Prior work on 3D reconstruction [65] has proposed two metrics for reconstruction quality: *accuracy* and *completeness*. Consider a 3D model M and a corresponding ground-truth M_g. Accuracy describes how *closely* the 3D model M resembles the ground truth M_g. Accuracy is the root mean square error (RMSE) of the distance from each point in M to the nearest point in M_g. Completeness describes *what extent* of the ground truth M_g the 3D model M captures. Completeness is the RMSE of the distance from each point in M_g to the nearest point in M.

A 10 cm accuracy means that every point on the 3D model M is displaced by 10 cm, on average, from its ground truth position in M_g. A 50 cm completeness means that, on average, for every point on the ground truth M_g, the nearest point on the 3D model M is 50 cm away. If both accuracy and completeness are zero, M perfectly matches M_g. So, for both metrics, *lower is better*. These two metrics are roughly analogous to precision (accuracy) and recall (completeness). Fig. 1b shows an *accurate but incomplete* 3D reconstruction.



(a) GPS. (b) SLAM. (c) AeroTraj.

Fig. 2. Three models of a large complex on our campus: GPS (a), SLAM (b), and AeroTraj (c). All models use the same height ramp function to color-code the z-axis, with blue representing the lowest points and orange representing the highest. The AeroTraj model has distinct features and trees, with clear and crisp coloring, indicating a good reconstruction. In contrast, the GPS and SLAM models have diffused colors and lack visible features, suggesting poor and noisy reconstruction.

2.2 Photogrammetry

Photogrammetry, a technique to construct 3D models, uses multi-view stereo [32] (MVS). MVS infers the scene's 3D structure, and the depth of every pixel in all the images. Then, it fuses these into a dense 3D reconstruction. MVS 3D reconstructions are generally accurate, but may contain missing regions or holes (Fig. 1b) when it cannot confidently estimate pixel depths. Moreover, MVS is also compute-intensive.

In our setting, to build a model using MVS, an operator would fly a drone to collect 2D images. Then, after landing the drone, she would upload the images to a cloud service to run MVS. If the model *contains missing regions* (detected by visual inspection), the operator must adjust the trajectory and fly the drone again [9, 15]. It can take multiple iterations to get the desired 3D model.

To quantify these shortcomings, we conducted an experiment (methodology described in §4) where we used an open source implementation of MVS (ColMap [64]) to reconstruct 3D models of two buildings. For each building, we tried several trajectories, and selected the one that provided the highest quality model. MVS accuracy is slightly higher than 10 cm (Table 1); this is desirable. Completeness, however, is 75 cm or more. Fig. 1b illustrates incomplete regions in the MVS reconstruction. AeroTraj achieves 5 to 9 cm completeness (§4), almost an order of magnitude better. Finally, MVS requires nearly 7–9 *hours* after flight completion to reconstruct the model. AeroTraj assembles the model on-the-fly so the model is ready immediately after flight completion.

In theory, photogrammetry could detect, in-flight, when the model is incomplete and adapt the trajectory accordingly. Detecting model incompleteness is challenging without running MVS on the entire set of images [58, 72]. To reduce the number of iterations, companies hire drone pilots or photogrammetrists [9, 15] who design trajectories based on their experience and rules of thumb [6, 64]. Even so, it is very challenging to ensure complete and high-quality reconstructions without iteration [58].

2.3 GPS

On-board GPS receivers can be used to position point clouds. Unfortunately, GPS errors can range up-to several meters [49]. Fig. 2a shows a 3D model assembled using a drone flight over a commercial building that uses GPS for positioning; the building's outline is fuzzy, as are the contours of the trees surrounding the building. Table 2 shows that the accuracy and completeness for the GPS reconstructed model are 1.6 m and 0.53 m, respectively, both of which are undesirable. Fig. 2c shows a 3D reconstruction using techniques proposed in this paper, which does not use GPS.

Approach	Accuracy (m)	Completeness (m)	Flight time (s)
GPS	1.60	0.53	944
SLAM	2.30	1.30	944
AeroTraj	**0.13**	**0.09**	**750**

Table 2. Relative to AeroTraj, alternative reconstruction approaches for drone-based LiDAR reconstruction give poor quality 3D models.

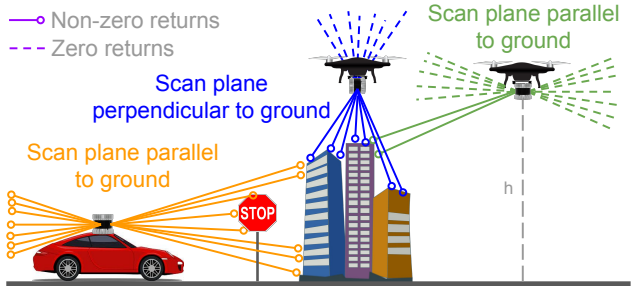


Fig. 3. LiDAR returns when mounted on a vehicle (orange lines) and a drone platform (blue, and green lines). Solid lines represent the laser beams which hit other objects (non-zero returns), whereas the dotted lines represent ones that did not (zero returns).

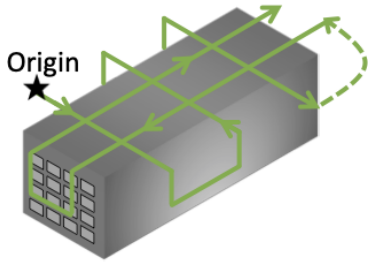


Fig. 4. Rectilinear trajectory for SLAM-based reconstruction.

High-precision GNSS/RTK receivers can provide more accurate positioning but require additional infrastructure, are costly, and can perform poorly (tens of meters of error [40]) in urban environments. Prior work [47, 74] has used such receivers in *remote sensing* for *offline* reconstruction, using specialized unmanned aerial vehicles (UAVs) with long-range LiDAR sensors. In contrast, we consider solutions that employ off-the-shelf technologies: drones, and commodity GPS and LiDAR for online reconstruction.

2.4 SLAM

In this paper, we use SLAM to ensure accurate drone positioning. LiDAR-based SLAM algorithms continuously align 3D point clouds using scan [38], and feature matching [67, 77] techniques to precisely estimate the pose (position, and orientation) of the LiDAR. These algorithms can obtain centimeter-level accuracy for LiDARs mounted on autonomous vehicles.

On a drone, however, un-modified SLAM cannot obtain similar accuracy. To understand why, consider Fig. 3. On a vehicle, the LiDAR scan plane (the plane formed by the laser beams in the center) is parallel to the ground. This way, the LiDAR receives numerous returns (solid orange lines) from objects in the surrounding *e.g.*, the road, traffic signs, and buildings, *etc.* This results in dense point clouds, better alignment and hence accurate SLAM positioning. On a drone at height h, a similar orientation (green lines in Fig. 3) results in significantly fewer returns (if any). When the scan plane is perpendicular to the ground (blue lines in Fig. 3), the LiDAR receives more returns but only about 9% of the returns from a vehicle-mounted LiDAR. With sparse returns, SLAM cannot precisely align consecutive point clouds and hence results in imprecise and inaccurate positioning. Thus, LiDAR orientation with respect to the building surface is an important factor in determining accuracy; we discuss other factors in §3.

To quantify this, we used a rectilinear flight (Fig. 4) to build a 3D model of a large building using the Google Cartographer LiDAR SLAM [38] implementation. The resulting 3D model (Fig. 2b) has 2.3 m accuracy, 1.3 m completeness (Table 2), and is visually worse than AeroTraj-based reconstruction (Fig. 2c). SLAM cannot align successive point clouds and hence results in imprecise pose estimates.

Metric	Cartographer [38]	LOAM [77]
Frame Rate (fps)	2	5
Accuracy (m)	0.21	5.75
Completeness (m)	0.09	2.74

Table 3. LiDAR SLAM on Jetson TX2. For both accuracy and completeness, lower is better.

Roof type	AeroTraj support	Witten	Ann Arbor	Manhattan	Farmingham
Polygonal	✓	709	1686	75	5
Gabled	✓	572	7	96	4047
Hipped	✓	1027	3	23	538
Pyramidal	✓	110	0	0	0
Skillion	✓	189	0	2	0
Gambrel	✓	0	0	0	143
Others	✓	130	772	14	0
Dome/antenna	×	3	20	0	1
Percentage of buildings AeroTraj can reconstruct		99.8%	99.1%	100.0%	99.9%

Table 4. AeroTraj can reconstruct more than 99% commercial and residential buildings in four representative cities.

AeroTraj is significantly more accurate and complete than un-modified SLAM. It uses SLAM as a starting point in its design. However, for AeroTraj to detect SLAM drift (§1) in-flight, it must run SLAM as the drone flies. Because of payload restrictions, on-board compute on drones may be insufficient to run SLAM. A DJI M600Pro (which we use in this paper) hexacopter has a maximum payload weight of 5 kg. On this, we mounted an LTE radio, an Ouster OS1-64 LiDAR and a Jetson TX2 board. This compute capability is far from sufficient to run LiDAR SLAM at full-frame rate[1]. On the TX2, we ran two popular, representative LiDAR-SLAM algorithms, Cartographer [38] and LOAM [77]. Cartographer, a scan matching algorithm, can only process 2 frames per second (LiDARs generate 10–20 frames per second) and LOAM, a feature-matching algorithm, can process 5 frames per second (Fig. 3). LOAM's reconstruction quality is unacceptable. Cartographer produces good reconstructions, but its frame rate is too slow to detect and correct drift in-flight using on-board compute (Table 3 shows results for a short flight in which drift is not a factor).

These illustrative results motivate the design of AeroTraj, which layers trajectory generation and drift correction on top of SLAM to achieve centimeter-level accuracy for LiDAR-on-drone based 3D reconstruction.

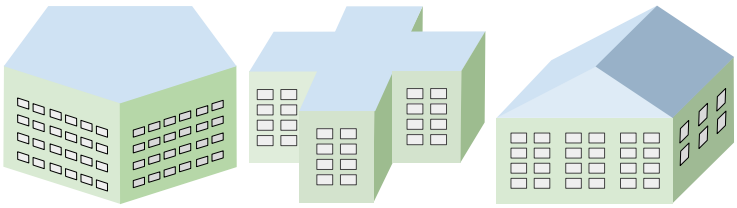
3 AEROTRAJ DESIGN

In this section, we describe the design of AeroTraj, beginning with an overview.

3.1 Overview

To use AeroTraj (Fig. 6), the user must specify: a) an *area of interest*, b) the drone-mounted LiDAR specifications, and c) a minimum *target point density*. Point density, the number of points per unit area on the surface of a point cloud determines the level of detail of a 3D model. By setting a minimum point density, AeroTraj ensures that all parts of the building surface are captured with at least the same minimum level of detail. The user can adjust this parameter to balance model fidelity and battery usage, since more detailed models require longer flights. The LiDAR specifications include the number of beams, horizontal and vertical field-of-view (FOV), and maximum range, which are all provided on a LiDAR's datasheet.

[1]With a 64-beam LiDAR, SLAM processes up to 480 Mbps of 3D data.



(a) Pentagon. (b) Plus. (c) Gabled.

Fig. 5. AeroTraj is designed for buildings with vertical sides and either polygonal or gabled roofs and their variations (hipped, mansard, pyramidal, skillion, *etc.*).

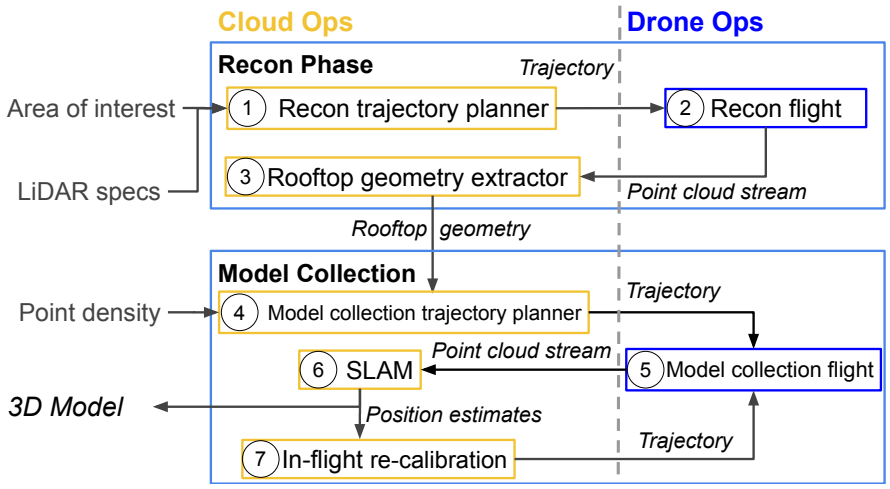


Fig. 6. AeroTraj operational model.

3.1.1 Building Types. AeroTraj exploits building *geometry* (shape and size) to obtain high quality reconstructions. AeroTraj can generate 3D models for buildings with (a) vertical sides and (b) (convex or non-convex) polygonal flat roofs or gabled roofs and their variations (Fig. 5). Variations of the gabled roof include hipped, pyramidal, skillion, and mansard roofs [2]. AeroTraj can also reconstruct buildings where walls and roofs have outcroppings such as chimneys, HVAC systems, or windows, *etc.* (our evaluations include such structures in Fig. 1, Fig. 2, and Fig. 16).

This covers most commercial buildings as well as many residential structures. To quantify this, we used open source building geometry datasets [24, 43] of four cities (Witten, Ann Arbor, Manhattan, and Farmingham), to determine the percentage of buildings that AeroTraj can reconstruct. Table 4 shows that AeroTraj can reconstruct more than 99% of the buildings found in these four cities. We leave to future work (§7) extensions of AeroTraj to reconstruct rare building shapes (*e.g.*, buildings with dome roofs or large antennas) and other large physical structures (*e.g.*, aircraft and blimps).

3.1.2 Components. Given these inputs, in the area of interest, AeroTraj guides a drone to automatically discover buildings, and constructs a 3D model of the buildings on-the-fly while *minimizing flight duration* at that given minimum point density. To a first approximation, drone battery usage increases with flight duration; we have left it to future work to incorporate drone battery models (§7).

Property	Photogrammetry	SLAM	AeroTraj
Complexity	High	Low	Low
Reconstruction	Offline	Offline	Online
Automated trajectory planning	×	×	✓
Quality feedback	×	×	✓
Density control	×	×	✓

Table 5. AeroTraj comparison with Photogrammetry and SLAM.

AeroTraj splits its functionality across two components: (a) a lightweight subsystem that runs on the drone (Drone Ops in Fig. 6), and (b) a cloud-based component (Cloud Ops in Fig. 6) that discovers buildings, generates drone trajectories, and reconstructs the 3D models on-the-fly.

To automatically discover buildings, AeroTraj's cloud component (Fig. 6) generates an efficient *reconnaissance (recon) trajectory* [Fig. 6 (1)] over the area of interest to discover the *rooftop geometry* of buildings. The drone follows this trajectory [Fig. 6 (2)], *streams compressed point clouds* to the cloud, extracting the geometry of the roof [Fig. 6 (3) and §3.4].

Using the rooftop geometry and the user-defined *point density*, the cloud service prepares a more careful *model collection* trajectory [Fig. 6 (4) and §3.2] that designs a minimal duration flight to ensure high 3D model accuracy at the given point density. As the drone flies along this trajectory [Fig. 6 (5)], AeroTraj *streams compressed point clouds* to the cloud component, which continuously runs SLAM [Fig. 6 (6)] and estimates whether SLAM drift is sufficient to warrant *recalibration* [Fig. 6 (7) and §3.3]. Soon after the drone completes the trajectory, the 3D model is available at the cloud service.

Below, we first describe model collection (§3.2), since that is the most challenging of AeroTraj's components. We then describe how AeroTraj extracts the rooftop geometry (§3.4), and conclude by describing point-cloud compression (§3.5).

3.2 Model Collection

To build a high quality 3D model, AeroTraj must simultaneously (a) ensure high accuracy and completeness, (b) satisfy the minimum target point density, and (c) use short flights. To achieve these, AeroTraj uses an optimization formulation that generates a minimum length trajectory that covers the building sides and roof whilst satisfying two constraints imposed by: a) SLAM, and b) target point density.

3.2.1 SLAM-imposed constraints. The trajectory of the drone flight impacts 3D model completeness and accuracy because a poorly designed trajectory can increase SLAM error. For example, if a drone flies too fast, two successive point clouds may have little overlap (Fig. 8), increasing SLAM error. Besides *speed*, other parameters that affect SLAM error include *distance from the building surface* and the *orientation* of the LiDAR with respect to the ground.

SLAM algorithms are complex, so it is difficult to derive analytical models that can predict the error resulting from different choices of these parameters. Instead, we resort to a parameter sweep. For this parameter sweep, we run SLAM in the AirSim simulator [66] and on real-world flight traces for different flight parameters, and find the best ones. We present the results of the sweep in §4, but summarize the main findings below. These findings point out two important factors that determine SLAM error: *point density*, and *degree of overlap* between successive point clouds[2].

Orientation impacts accuracy. As demonstrated in Fig. 3, on a drone, a LiDAR receives maximum returns when its scan plane is perpendicular to the ground surface. In this position, there are multiple ways to orient the LiDAR with respect to the motion of the drone. For instance, in a *parallel* orientation (Fig. 7), the scan lines of the LiDAR are parallel to the drone's direction of motion. On the other hand, the scan lines are perpendicular to the drone's

[2]SLAM estimates pose by matching successive point clouds (§1).

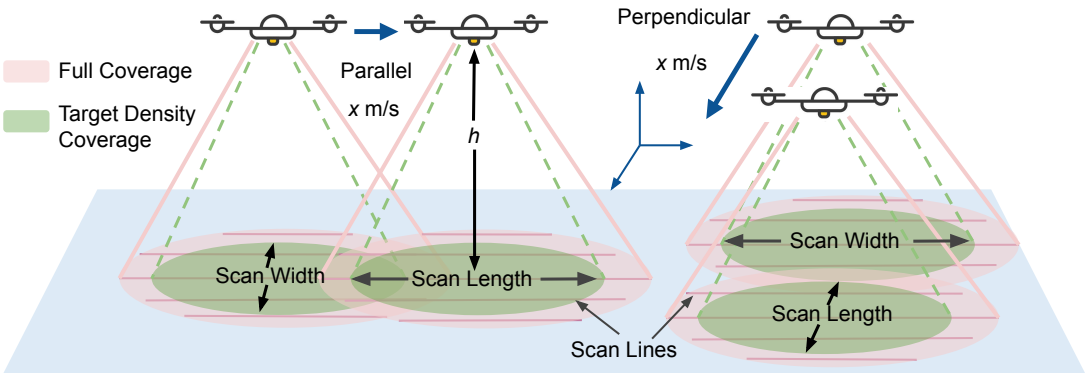


Fig. 7. Parallel and perpendicular LiDAR orientation.

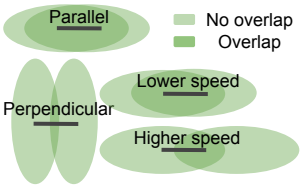


Fig. 8. Overlap at various orientation and speeds.

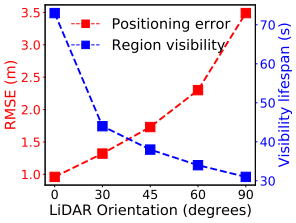


Fig. 9. LiDAR's orientation Vs. SLAM positioning error.

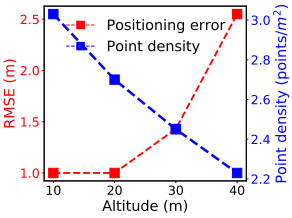


Fig. 10. Point density Vs. SLAM positioning error.

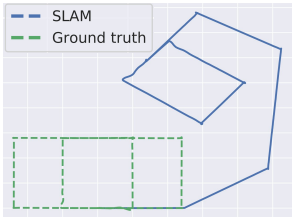


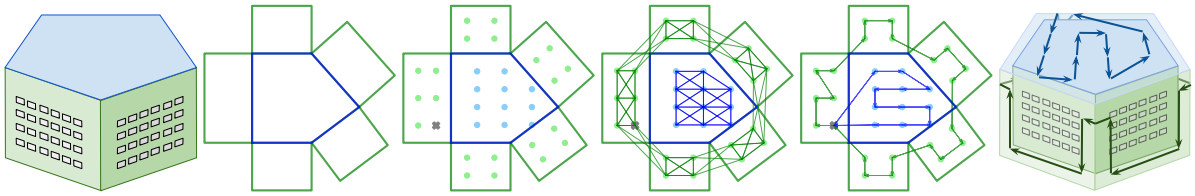
Fig. 11. Rotation throws SLAM off.

motion in a *perpendicular* orientation. As shown in Fig. 7, at a fixed height and speed, a *parallel* orientation results in a higher overlap between successive point clouds as compared to a *perpendicular* orientation (Fig. 8). A higher overlap results in better point cloud alignment, lower positioning error and hence higher accuracy. Fig. 9, obtained using the methodology described in §4, quantifies this intuition: different orientations have different degrees of overlap, and as overlap decreases, SLAM's positioning error increases. A parallel orientation (0°) has the lowest SLAM error because it has the highest visibility lifespan. (*Visibility lifespan*, the time for which a point on the building's surface is visible during flight, is a proxy for overlap; a longer lifespan indicates greater overlap).

Speed impacts model accuracy. If the drone flies fast, two successive point clouds will have fewer overlapping points, resulting in errors in the SLAM's pose transformations and (therefore) pose estimates, which leads to poor 3D model accuracy. For high accuracy, AeroTraj must fly the drone slowly (*i.e.,* at 1 m/s as we demonstrate in §4.4).

Height impacts both accuracy and completeness. Because LiDAR beams are radial, the higher a drone flies, the less dense the points on the surface of the building. Lower density results in worse completeness. Accuracy is also worse, because the likelihood of matching the same point on the surface between two scans decreases with point density. Fig. 10 obtained using methodology described in §4 illustrates this. It plots positioning error as a function of point density by altering the altitude of the drone. The positioning errors for point densities of 2.2 points per m^2 and 3.0 points per m^2 are 2.5 m and 1.0 m respectively. So, for high accuracy, AeroTraj must fly the drone as low as possible (*i.e.,* no higher than 20 m above the surface, as we demonstrate in §4.4).

Finally, the drone must ***never rotate*** the LiDAR. This maneuver can increase SLAM error. Fig. 11 shows an example in which the green dashed line is the ground truth drone trajectory, and the blue line SLAM's estimated pose. In the bottom right corner, when the drone rotates, SLAM is completely thrown off.



(a) Pentagon building. (b) Unfolded pentagon. (c) Meshed pentagon. (d) Graph of pentagon. (e) Derived trajectory. (f) Folded trajectory.

Fig. 12. Illustrating steps in optimized trajectory generation.

Two of these constraints are universal: that the parallel orientation is best, and that the drone must never rotate the LiDAR. Two others (height and speed) depend on the LiDAR characteristics and the SLAM algorithm. For a different model of LiDAR than the one we have used in this paper (a 64-beam Ouster) or SLAM implementation (Cartographer), we may need to re-run the simulations to obtain these flight parameters. We envision a practical implementation of AeroTraj will include pre-computed parameters for different LiDAR models, so users will not have to determine these.

3.2.2 Point density-imposed constraints. As the drone flies, its LiDAR obtains points on the surface of the building at successive instants. Let the term *scan* denote the portion of the surface for which the LiDAR's points have a density no less than the target point density (the green area in Fig. 7). For a given orientation, scan length is measured along and scan width perpendicular to the drone's motion (Fig. 7). Given a distance h at which the drone flies from the surface, the target point density requirement imposes constraints on *scan width*. Two successive scans of the surface (*e.g.,* legs of a U-shaped trajectory) cannot be separated by more than the *scan width* without violating the density constraint.

AeroTraj's key observation is that these constraints can be derived from an analytical model of the LiDAR, given its configuration. For instance, for an Ouster LiDAR with 64 beams, a vertical field of view of 45°, two consecutive beams are separated by 0.7° ($\frac{45}{64}$). During one full 360° rotation, the laser emits 1024 *pulses*; successive pulses are 0.35° ($\frac{360}{1024}$) apart. Each pulse generates a point in the point cloud[3]. Then, the 3D coordinates of a point from the n-th pulse of the b-th beam from a LiDAR mounted on a drone at height h pointing downwards is:

$$(x, y, z) = (r \sin\theta_L, r \cos\theta_L \sin\theta_B, r \cos\theta_L \cos\theta_B) \tag{1}$$

where ($\theta_L = \frac{\theta_F}{n}$) is the angle of the n pulse of the b-th beam whose beam angle is $\theta_B = \frac{b}{360}$, r_{max} is the maximum range, θ_F is the vertical field of view, and r is the distance between the LiDAR and where the pulse hits the surface:

$$r = \frac{h}{\cos(\theta_L + \theta_B)}, \qquad \forall r <= r_{max} \tag{2}$$

From this, AeroTraj derives the coordinates of every point on the surface. From this point cloud, it can derive the fine-grained point density distribution. Thus, given a minimum point density distribution and a distance h (obtained from the SLAM-derived constraints), AeroTraj calculates the scan width. Using the same procedure, AeroTraj derives the *scan length*, the length on the surface along the direction of the flight containing points from a single frame satisfying the minimum density constraint (Fig. 7). Scan length helps discretize the search space for trajectory optimization.

3.2.3 Optimized trajectory generation. AeroTraj uses a novel optimization formulation to generate the shortest flight path around the building. Inspired by prior work [18, 56], it uses an Integer Linear Programming (ILP)

[3]This is idealized. In practice, LiDARs may drop some reflections if they are noisy [53]. So, our density guarantee is *nominal*. Future work can model this noise for better equi-dense trajectory designs.

based optimization to generate the shortest flight path that respects point density and SLAM-imposed constraints for the class of buildings discussed above. Fig. 12 illustrates the steps, described below, in trajectory generation for a building with a flat pentagonal roof (Fig. 12a).

Unfolding. Unfolding a 3D building on a 2D plane reduces trajectory planning to coverage path planning (CPP) [34]. AeroTraj unfolds the sides of the building, so that the resulting object is planar. In Fig. 12b, this results in a pentagon surrounded by rectangles. (AeroTraj applies a similar unfolding trick to the gabled roof and its variations, details omitted for brevity.)

Meshing. This imposes a rectangular grid on the roof and all sides to discretize the unfolded structure into grid elements with dimensions *scan length*/2 by *scan width*/2 (the points in Fig. 12c depict the centers of the rectangles). Discretization reduces the search space [18, 33], and using the scan width ensures target point density. The longer side of the grid element parallels the longer side or dominant orientation of the corresponding polygon; AeroTraj aligns the LiDAR parallel to the longer side to maximize flight segments with a parallel orientation. Finally, a traversal that visits all mesh centers guarantees coverage of the structure. For a non-convex flat roof, AeroTraj generates the mesh on the convex hull of the roof.

Graph generation. AeroTraj now embeds a graph on the mesh; the vertices are the centers of rectangles, and each center is connected to every neighbor with an edge (Fig. 12d). AeroTraj actually embeds two sub-graphs: one on the roof mesh, and one on the side meshes taken together. It generates *tours* separately for each sub-graph, because these two require different LiDAR orientations. For each sub-graph, the tours start and terminate at a fixed origin (denoted by $\times$ in Fig. 12c), which ensures a SLAM loop closure before the LiDAR needs to be rotated for the other sub-graph.

Optimization formulation. For each sub-graph, AeroTraj models the trajectory as a Travelling Salesman Problem (TSP) tour, starting and ending at the origin. It formulates the TSP traversal as an ILP. This step takes a subgraph $G = (V, E)$ as input, where V and E are the set of vertices and edges, respectively and vertices indexed $i = 1, 2, ..., N$, with $i = 1$ the origin. Each edge $(i, j) \in E$ has a weight w_{ij} which is the 2D Euclidean distance between the position of the vertices i and j. If the drone flies from vertex i to vertex j, given $(i, j) \in E$, the binary decision variable x_{ij} set to 1, 0 otherwise. The formulation below finds a minimum length trajectory:

$$\min \quad \sum_i \sum_j \left(w_{ij}x_{ij} + \lambda p_{ij}\right) \tag{3a}$$

$$\text{s.t.} \quad \sum_{\forall i} x_{ij} = 1, \qquad \forall j \neq 1, \tag{3b}$$

$$\sum_{\forall i} x_{ij} \geq 1, \qquad j = 1, \tag{3c}$$

$$\sum_{\forall i \neq j} x_{ij} = \sum_{\forall k \neq j} x_{jk}, \qquad \forall j, \tag{3d}$$

$$u_1 = 1 \quad \text{and} \quad 2 \leq u_i \leq N, \qquad \forall i \neq 1, \tag{3e}$$

$$u_i - u_j + 1 \leq (N-1)(1 - x_{ij}), \quad 2 \leq i \neq j \leq N \tag{3f}$$

The first term in the objective is the tour length. To force tours to traverse the structure in a parallel orientation as much as possible, the second term penalizes edges selected in non-parallel direction by a factor $\lambda \geq 0$, where λ is a hyperparameter. If a selected edge $x_{ij} = 1$ is in non-parallel direction p_{ij} is set to 1, 0 otherwise.

The optimization is subject to several constraints: (**3b**, **3c**) the drone visits each vertex of the graph exactly once, except the origin; (**3d**) if a drone visits a vertex j, it has to leave the same vertex; (**3e**, **3f**) no subtours of size

$< N$ are allowed. We use Miller-Tucker-Zemlin based subtour elimination constraint [59] for this optimization. It requires auxiliary decision variables u_i for each vertex.

Refolding. Once it obtains the 2D trajectories (Fig. 12e), the trajectory planner projects each 2D point to 3D coordinates. Since the drone has to fly at a distance h from the building, AeroTraj obtains the 3D coordinates at a distance of h from the scanned face by projecting each point on the plane parallel to the face at a distance h away from it (Fig. 12f).

3.3 Drift estimation and re-calibration

Although AeroTraj generates a careful model collection trajectory designed to minimize positioning error, SLAM accumulates error on long flights. To minimize drift error, SLAM uses *loop closure* — this return to a previously-visited spot allows SLAM to re-calibrate itself. However, it is hard to predict *when* SLAM incurs significant drift, so AeroTraj continuously estimates drift and triggers a mid-flight return to *origin*, a designated spot at which to close the loop, and restart a new SLAM session. Mid-flight re-calibration is the primary reason for AeroTraj's cloud offload of SLAM, because, to detect drift while the drone is flying, it must obtain SLAM's position estimates to assess if there is a drift. Detecting excessive drift is a non-trivial problem since AeroTraj has no way of knowing when SLAM's position estimates are wrong because it does not have accurate ground truth.

Input : SLAM poses S and GPS tags G
Output: Imperfect regions I

```
1  S', G' ← TimeSynchronization(S, G)
2  G'_a ← GPSToMercator( G' )
3  t_cw ← UmeyamaAlignment( G'_a, S' )
4  S'_a ← TransformTrajectory( S', t_cw)
5  foreach s'_{a-i} in S'_a, g'_{a-i} in G'_a do
6      r_i ← RMSE( s'_{a-i}, g_{a-i})
7      if IsExcessive(r_i) then
8          I.Append(g_{a-i})
9      else
10         pass
11     end
12 end
```

Algorithm 1: Detecting excessive drift.

AeroTraj's key insight is to detect drift by comparing the *shape* of SLAM's estimated trajectory (see Fig. 11 for an example) with the *shape* of the GPS trajectory. GPS does not suffer from drift error, and this approach is robust to GPS errors, since it matches larger segments of the two trajectories, not their precise positions. Specifically, AeroTraj continuously executes 3D SLAM on the stream of compressed LiDAR frames from the drone, and estimates the pose of each frame. Then, it runs the algorithm described in Algorithm 1. It transforms GPS readings using the Mercator projection (line 2). Then, it aligns the GPS trajectory and the SLAM-generated trajectory using the Umeyama algorithm [73] (line 3) to determine the rigid transformation matrices (*i.e.,* translation, and rotation) that best align SLAM and GPS poses (line 4). AeroTraj partitions trajectories into fixed length segments and after alignment, computes the RMSE between the two trajectories in each segment, and uses these as an indicator of excessive drift (lines 5-11): if the RMSE is greater than a threshold ρ, AeroTraj invokes return-to-origin (Fig. 13).

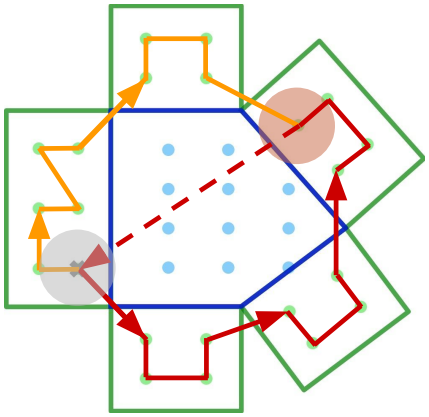


Fig. 13. AeroTraj re-calibration flight of a pentagon-shaped building (central blue region is the roof, and green regions are the unfolded sides). The initial model collection flight for the sides of the building (red line) detects drift (red circle), and then initiates a return-to-origin maneuver (dotted red line). Then, a re-calibration flight (orange line) from the origin (grey circle) collects 3D point clouds from the remaining unscanned regions.

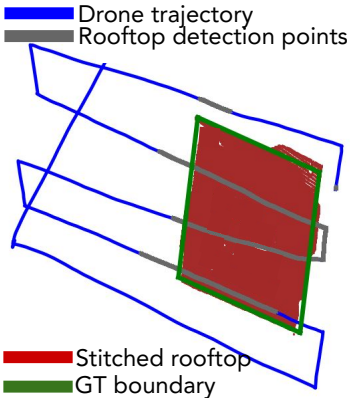


Fig. 14. AeroTraj's building detector on a real 40m x 70m x 20m building.

3.4 Discovering Rooftop Geometry

AeroTraj's model collection requires the rooftop geometry as input. Depending on the type of structure, it may be difficult to get this information. To estimate this information, AeroTraj uses a reconnaissance (recon) flight over the area of interest. This recon flight must be as short as possible, to minimize battery usage. In addition, recon (a) must not assume prior existence of a 3D model of the building (prior work in the area makes this assumption [44, 61]); (b) must be robust to nearby objects like trees that can introduce error; and (c) must generalize to the class of buildings AeroTraj targets (§3).

3.4.1 The recon trajectory. Recon uses a rectilinear scan (blue line in Fig. 14), but unlike model collection, during recon the drone flies *fast* (4 m/s) and *high* (60 m above the building's roof[4]), with the LiDAR mounted in a *perpendicular* orientation in order to have the shortest duration flight possible (we justify these flight configurations in §4.5). During this flight, AeroTraj streams point clouds to its cloud component, which runs the *boundary detection* algorithms described briefly below.

3.4.2 Surface extraction. The cloud component receives GPS-tagged compressed point clouds from the drone. It first uncompresses them, then computes the *surface normal* of every point in the point cloud. A surface normal to a point determines the direction normal to the surface formed by points within a fixed radius of the point. Calculating surface normals is compute-intensive, so AeroTraj offloads this operation to the GPU. Then, AeroTraj uses RANSAC [30] (a plane-fitting algorithm) to segment the LiDAR points into groups of points that fall onto planes. It removes the plane furthest from the drone (likely to be the ground plane) and eliminates planes with irregular surface normals (*e.g.,* from nearby trees). The remaining planes correspond to rooftops. For additional robustness, it uses majority voting across multiple frames to identify the rooftop.

3.4.3 Estimating the boundary. AeroTraj uses the drone's GPS location to transform each surface to the same coordinate frame of reference, then combines all surfaces into a single point cloud that represents the extracted rooftop of the building (red rectangle in Fig. 14). To extract the boundary of the building, it extracts the *alpha*

[4]We assume the nominal building heights in an area are known, for example, from zoning restrictions.

shape [17] (a sequence of piece-wise linear curves in 2D) of the stitched point cloud (green outline in Fig. 14). This allows AeroTraj to generalize to non-convex shapes as well. Finally, to detect the boundary of multiple buildings, AeroTraj clusters the rooftop point clouds.

3.5 Point-Cloud Compression

LiDARs generate voluminous 3D data. For instance, the Ouster OS1-64, with 360° horizontal and 45° vertical field-of-view (FoV), generates 20 point clouds per second requiring 480 Mbps, well beyond the capabilities of today's cellular standards. AeroTraj compresses these point clouds to a few Mbps (1.2 to 4.0), using two techniques: viewpoint filtering, and octree compression.

Viewpoint filtering removes returns from beams directed towards the sky, from objects beyond the LiDAR range (because they generate *zero returns*) and also from the drone's body. AeroTraj compresses the retained data using an octree compression algorithm [63] designed for point clouds (so better than data-agnostic compression techniques like Gzip). It uses different configurations to control octree and point resolution that govern compressibility, to achieve point-cloud transmission rates of 1.2, 2.5, and 3.8 Mbps (§4) corresponding to high, medium and low compression, respectively (all within achievable LTE speeds).

4 EVALUATION

We have implemented AeroTraj using the Point Cloud Library (PCL [63]), the Cartographer [38] LiDAR SLAM implementation, the Boost C++ libraries [62], and the Robotic Operating System (ROS [69]). For the recon phase described in §3.4, we used functions from the Point Cloud Library (PCL [63]) for plane-fitting, outlier removal and clustering. Our compression and extraction modules are ROS nodes that use PCL. The drift detection module uses a Python package for the Umeyama alignment [35]. The trajectory optimization uses Gurobi [51]. Not counting libraries and packages it uses, AeroTraj is 16,500 lines of code.

We use a photorealistic simulator, AirSim [66] that models realistic physical environments using a game engine, then simulates drone flights over these environments and records sensor readings taken from the perspective of the drone. AirSim has a parametrizable model for a LiDAR; we used the parameters for the Ouster OS1-64 in our simulation experiments. AeroTraj generates trajectories for the AirSim drone, then records the data generated by the LiDAR, and processes it to obtain the 3D model. For computing the metrics above, we obtain ground truth from AirSim. To build the ground truth 3D model, we flew a drone equipped with a LiDAR several times over the region of interest in AirSim (using exhaustive flights) and then stitched all the resulting point clouds using ground truth positioning from AirSim.

In addition, we have collected data from nearly 30 flights (each of about 25 minutes) on an M600Pro drone with an Ouster OS1-64 LiDAR on a commercial complex. For almost all experiments, we evaluated AeroTraj on both *real-world* and simulation-driven traces. Simulation-driven traces give us the flexibility to explore the parameter space more (as we show below). However, we use real-world traces to validate all these parameter choices and estimate reconstruction accuracy in practice. For real-world experiments, we offload to an AWS VM with 16 cores, 64 GB RAM and an Nvidia T4 GPU. The AWS VM was located approximately 3,900 km from the drone.

We quantify end-to-end latency, 3D model accuracy and completeness (§3.2), and positioning error. We also quantify AeroTraj's energy-efficiency (using flight duration as a proxy for drone battery usage) and the computational capabilities of its processing pipeline. Flight duration includes recon and model collection.

4.1 3D Model Reconstruction

4.1.1 Comparison: Photogrammetry. We compare AeroTraj against ColMap [64], a state-of-the-art photogrammetry-based tool that uses multi-view stereo (MVS [32]). ColMap is extensively used in the vision

Scheme	Acc. (m)	Compl. (m)	Flight (s)	Processing (s)	End-to-end reconstruction (s)
Large building (100 m x 50 m x 20 m)					
ColMap	0.16	0.75	**1320**	31600	32900
AeroTraj	**0.09**	**0.05**	1430	*in-flight*	**1430**
Small building (50 m x 50 m x 20 m)					
ColMap	**0.11**	0.80	**760**	23084	23844
AeroTraj	0.12	**0.09**	864	*in-flight*	**864**

Table 6. Reconstruction accuracy (acc.), completeness (compl.), flight time, processing time and end-to-end reconstruction time for two buildings using: a) photogrammetry reconstruction with an optimized trajectory (ColMap), and b) AeroTraj.

community as the gold standard for 3D reconstruction [48, 54]. For this, we use two rectangular buildings: a) a *large* 100m x 50m x 20m (L x W x H) and, b) a *small* 50m x 50m x 20m building in AirSim. We compare both approaches using three metrics: a) accuracy, b) completeness, and c) end-to-end reconstruction time (proxy for latency). We calculated the accuracy and completeness of the models generated by these approaches by comparing them against ground truth models generated from AirSim. Lower is better for accuracy and completeness. In addition to these, we measured the end-to-end reconstruction time, the total time taken to construct the 3D model end-to-end. End-to-end reconstruction time is a function of flight time, and processing time. Flight time reports the model collection flight duration. Processing time indicates the time to construct a 3D model with the gathered data.

For these experiments, AeroTraj uses *compressed point clouds* with bandwidth requirements that are compatible with LTE speeds today (*i.e.,* upload bandwidth of 3.8 Mbps); we study the effect of compression on AeroTraj model reconstruction more in §4.3. For ColMap, we flew the drone in various trajectories recommended for high-quality reconstruction [14] and collected 2D images for *offline* reconstruction; we report the result for the best performing trajectory. For a more than fair comparison, in ColMap, we assume the building location and roof geometry is known a priori. Without this assumption, ColMap's reconstruction quality is poor and its reconstruction and flight times are much longer.

For both buildings, AeroTraj achieves: a) comparable, if not better, accuracy, b) an order of magnitude higher completeness, and c) 25x faster reconstruction as compared to ColMap. AeroTraj achieves cm-level accuracy and completeness for both buildings (Table 6). ColMap (like AeroTraj) depends on trajectory design. For ColMap, we tried multiple trajectories, including ones suggested for drone photogrammetry [14]. From these, we report the trajectory with the best reconstruction quality. Without proper trajectory design, ColMap reconstruction fails altogether. With proper trajectory planning, ColMap reconstructs both buildings within 0.16 m accuracy and 0.80 m completeness. AeroTraj outperforms ColMap because it has real-time insight into the drone's tracking accuracy and can re-calibrate on-the-fly.

ColMap reconstructions are almost as accurate as AeroTraj but highly incomplete. This is because of the inherent nature of photogrammetry-based reconstructions; *if ColMap cannot converge to a good solution for a given region, it will leave that region empty, resulting in a model with many incomplete regions (as shown in Fig. 1b).* As such, AeroTraj constructs 3D models with comparable if not better accuracy, and an order of magnitude higher completeness than those generated by ColMap.

AeroTraj creates 3D models 25x faster than ColMap. It builds 3D models for the small and large buildings in approximately 15 mins and 20 mins (inclusive of flight times), respectively. Because it uses online reconstruction with cloud offload, AeroTraj simultaneously builds a 3D model at the cloud as it receives streamed point clouds from the drone. This approach makes available the entire 3D model as soon as the model collection flight is complete (details in §4.2) (as indicated in Table 6). For ColMap the end-to-end reconstruction time is the sum of the flight time and the processing time. ColMap takes significantly more time to process the collected data because it uses compute-intensive photogrammetry-based reconstruction. Even with a powerful GPU-equipped

Scheme	BW (Mbps)	Accuracy (m)	Completeness (m)
Cartographer [38]	480	2.30	1.30
LOAM [77]	480	5.75	2.75
GPS-based reconstruction	3.80	1.60	0.53
AeroTraj	**3.80**	**0.13**	**0.09**

Table 7. Reconstruction quality of a *real-world* 70 m x 40 m x 20 m building for AeroTraj, two LiDAR SLAM implementations and GPS-based reconstruction relative to an uncompressed trace.

Structure type	Flight duration (s)	Accuracy (m)	Completeness (m)
Tall rectangle	1430	0.08	0.06
Large rectangle	1145	0.09	0.05
Pentagonal	1204	0.11	0.08
Small rectangle	864	0.12	0.09
Gabled roof	862	0.13	0.07
Plus-shaped	1063	0.16	0.07

Table 8. AeroTraj 3D reconstruction times (recon (§3.4) and model collection) and quality for different structures at low compression.

machine, ColMap takes approximately 6.5 hours and 9.2 hours to build 3D models of the small and large buildings, respectively[5].

The flight times for ColMap are relatively shorter (107 seconds shorter) than AeroTraj. The planned trajectory length for AeroTraj's model collection was actually 185 seconds shorter than ColMap. However, it incurred re-calibration flights, resulting in a longer flight time. Moreover, its flight time also includes 150 seconds for reconnaissance. Despite incurring only 107 seconds of additional flight time, AeroTraj still reconstructs 25x faster than ColMap, and improves accuracy and completeness by 7 cm and 70 cm respectively.

4.1.2 Comparison: LiDAR SLAM-based reconstruction and GPS. Results from our real-world drone flights validate the AeroTraj outperforms LiDAR SLAM implementations [38, 77] (Table 7). For this experiment, we compared AeroTraj against both scan and feature matching based SLAM implementations. We used Google Cartographer [38] and LOAM [77] as representatives for scan, and feature matching approaches, respectively. We reconstructed the 3D model of a real-world 70 m x 40 m x 20 m rectangular building (Fig. 2). Because we lack a reference ground truth for real-world data, we use the 3D model generated from raw, uncompressed traces. Table 7 summarize the results. 3D reconstruction with Cartographer and LOAM fail completely for the same reasons mentioned above for ColMap (*i.e.,* no trajectory planning, and no re-calibration). With GPS, it is possible to do in-flight reconstruction, however, the accuracy and completeness being 1.60 m and 0.53 m, make such 3D models unusable. With AeroTraj, for this building, we can build accurate, and complete 3D models whose completeness and accuracy are 9 cm and 13 cm, respectively. These experiment also demonstrates that AeroTraj performs equally well on real-world traces captured from our drone prototype.

4.1.3 Generalization to different building shapes. AeroTraj is designed to reconstruct a variety of building types (§3). It reconstructs all these buildings within a single drone battery cycle (25 to 30 mins) at cm-level accuracy and completeness (Table 8). Flight durations for fairly large buildings (100 m x 50 m x 20 m large rectangle, pentagon and 50 m x 50 m x 40 m tall building) are relatively longer because they require multiple re-calibrations. Even for these, AeroTraj preserves cm-level accuracy and completeness.

[5]We do not evaluate AeroTraj against smartphone-based LiDAR reconstruction approaches because they are both inaccurate and highly inefficient for outdoor reconstruction (§2). Due to the smartphone's limited sensing range, model collection can take several hours and optimal trajectory planning takes well over 10 hours due to the large search space.

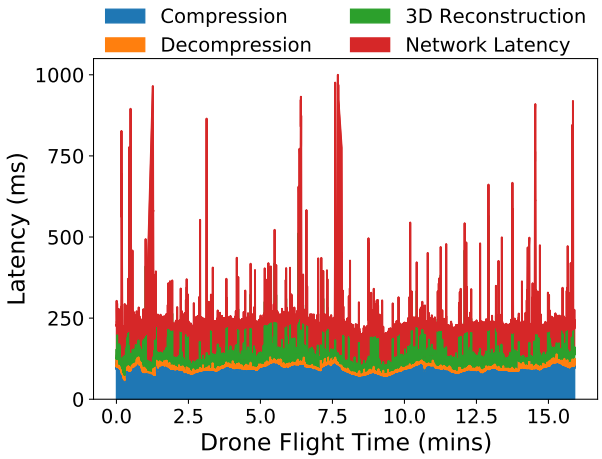


Fig. 15. End-to-end latency from a real-world drone flight.

Component	Location	Average (ms)	99th percentile (ms)
Compression	Drone	88.2	109.45
Network latency	-	104.0	409.9
Decompression	Cloud	8.9	11.7
Reconstruction	Cloud	33.3	87.8
End-to-end latency	-	234.0	552.1

Table 9. AeroTraj enables fast 3D reconstruction over LTE. Each row shows where the component in located (*i.e.,* on the drone or the cloud), its *per frame* average latency, and *per frame* 99th percentile latency.

4.2 Performance

4.2.1 Feasibility of Reconstruction over Cellular. To validate that AeroTraj can collect a 3D model in the real-world, we used our end-to-end implementation of AeroTraj to reconstruct a 70 m x 40 m x 20 m building by streaming compressed point clouds over LTE whilst the drone was in-flight (at 10 Hz) our AWS VM that ran 3D reconstruction. The experiment ran for about 16 minutes of drone flight (Fig. 15 and Table 9). We are interested in two aspects of performance: end-to-end latency and the ability to process at frame rate.

AeroTraj has a per-frame end-to-end 99th percentile latency of approximately half a second. This means that the cloud can detect excessive drift and initiate mid-flight recalibration within a second. An interesting side effect of this design is that the 3D model is ready within half a second of flight completion. The average end-to-end processing latency *per frame* is about 234 ms. Of this, network latency accounts for nearly 104 ms. This network latency is an artifact of our experimental setup: the drone, flying at a location on the west coast of the US streamed point clouds to a VM on the east coast. In practical settings, we expect offloads to nearby cloud regions, resulting in much lower end-to-end latency. To put these numbers in perspective, AeroTraj's latency is comparable to real-time 2D video conferencing systems (*e.g.,* Zoom, Skype), with a latency of 200-300 ms [31, 42], and is twice as fast as 360-degree video streaming services [75].

AeroTraj can maintain LiDAR frame rate. On-board point cloud compression takes on average 88 ms per frame, sufficient to sustain a 10 Hz LiDAR's frame rate. Decompression and 3D reconstruction are fairly fast on the cloud. Trajectory generation runs once and takes on average 14.6 ms per building. Drift estimation runs periodically and takes on average 11.2 ms.

Stage	Sub-component	Time (ms)
Surface extraction	Point cloud decompression	3.0 ±0.3
	Surface normal estimation	76.0 ±82
	RANSAC plane-fitting	5.0 ±9.0
	Outlier removal	0.2 ±0.3
	Rooftop extraction	6.0 ±5.0
Boundary estimation	Rooftop stitching	3.0 ±2.0
Total time		93 ±90.0

Table 10. Per-frame processing times for AeroTraj's building geometry estimation.

Scheme	Acc. (m)	Compl. (m)	Flight (s)	Processing (s)	End-to-end reconstruction (s)
Large building (100m x 50m x 20m)					
Offline-AeroTraj	0.39	0.27	**990**	5330	6320
AeroTraj	**0.09**	**0.05**	1430	*in-flight*	**1430**
Small building (50m x 50m x 20m)					
Offline-AeroTraj	0.29	0.15	**720**	4395	5115
AeroTraj	**0.12**	**0.09**	864	*in-flight*	**864**

Table 11. Reconstruction accuracy, completeness, flight time, processing time, and end-to-end reconstruction times for two buildings using: a) AeroTraj without cloud offload (Offline-AeroTraj), and b) AeroTraj.

4.2.2 Rooftop Geometry Extraction Performance. We profiled the execution time of each component in AeroTraj's building geometry extraction on a 15-minute *real-world* trace. Point cloud compression executes on the drone, and other components run on our AWS VM. Extracting the building geometry requires 93 ms per frame (Table 10); with these numbers, we can sustain 10 fps. At this frame rate, our building detector is quite accurate (§3.4). The most expensive component is surface normal extraction (76 ms), even though we offload it to a GPU. Thus, a moderately provisioned, cloud VM suffices to run AeroTraj at full frame rate with an end-to-end compute latency of about 100 ms for reconnaissance, and 33 ms for model collection.

4.3 Ablation and Sensitivity

4.3.1 Cloud Offload. To show the importance of cloud offload, we compared AeroTraj to an offline approach that does not use cloud offload (Offline-AeroTraj in Table 11). Offline-AeroTraj uses AeroTraj's trajectory planning, then collects and stores *raw* 3D point clouds on the drone, and processes them offline after the drone has landed. Its accuracy is 3x worse, and completeness 3.5x worse (on average) as compared to a complete AeroTraj implementation with cloud offload. Without cloud offload, offline-AeroTraj cannot track and mitigate drift errors in real-time. The flight times for offline-AeroTraj are relatively shorter because there are no re-calibration flights. However, AeroTraj's end-to-end reconstruction time is 6x smaller than offline-AeroTraj because it pipelines model collection and 3D reconstruction, whereas the latter first collects and then processes point clouds into a 3D model.

4.3.2 Trajectory Planning. We evaluated AeroTraj without its optimized trajectory generation. Instead, we use a rectilinear trajectory (Fig. 4) that satisfies the minimum point density requirement (AeroTraj-st, Table 12). AeroTraj-st has poorer accuracy and completeness relative to AeroTraj, even though it has a longer duration flight (which should, in theory, give it a better chance to capture more detail of the building). However, a larger proportion of AeroTraj-st's trajectory is not in a parallel orientation; this illustrates the importance of explicitly optimizing for parallel flights in the objective function (Equation 3, in §3.2).

Scheme	Accuracy (m)	Completeness (m)	Flight time (s)
Large building (100 m x 50 m x 20 m)			
AeroTraj-st	0.21	0.24	1520
AeroTraj	**0.09**	**0.05**	**1430**
Small building (50 m x 50 m x 20 m)			
AeroTraj-st	0.25	0.14	900
AeroTraj	**0.12**	**0.09**	**864**

Table 12. Reconstruction accuracy, completeness, and flight time for two buildings using: a) AeroTraj with a simple rectilinear trajectory (AeroTraj-st), and b) AeroTraj.

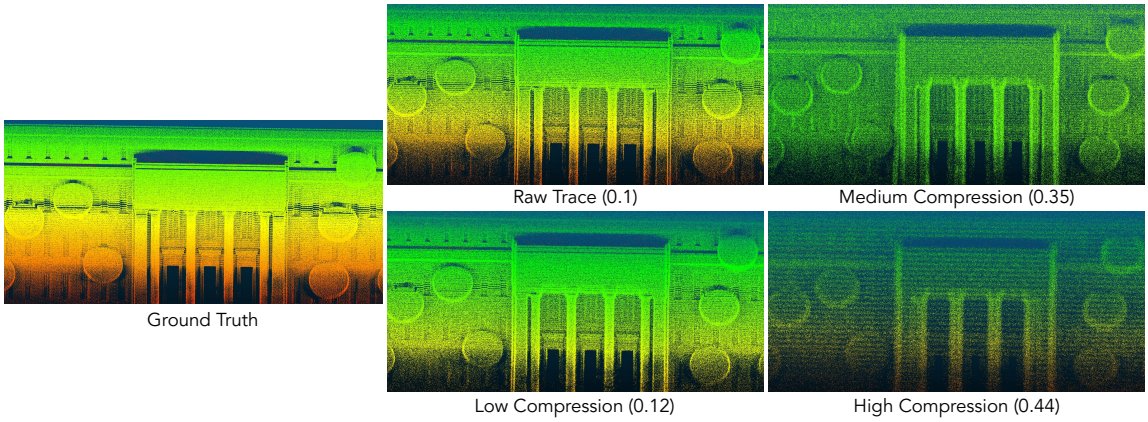


Fig. 16. 3D models accuracy at different compression levels.

4.3.3 Re-calibration. To show the effect of in-flight re-calibration, we compare reconstruction quality with (w) and without (w/o) recalibration in AirSim (Table 13a). Across our six buildings, on average, at the expense of 35% (285 seconds) longer flights, AeroTraj improves accuracy by 70% (28 cm) and completeness by 64% (13 cm) with re-calibration flights. Larger buildings (tall rectangle, large rectangle, and pentagonal) require longer aerial flights which accumulate higher drift. This results in relatively more re-calibration flights and hence higher flight duration. Apart from these large buildings which required two re-calibration flights, the remaining buildings require a single re-calibration flight. Even so, AeroTraj is able to reconstruct these buildings accurately (within cm-level accuracy and completeness), demonstrating the importance of re-calibration whilst in-flight.

4.3.4 Sensitivity: Compression Levels. We explore the impact of compression on accuracy and completeness using (a) a synthetic building in AirSim and (b) *real-world* traces. In addition to the three compression schemes discussed earlier (§3.5), we compute accuracy and completeness for (a) viewpoint compression and (b) lossless compression. The first contextualizes our results, while the second alternative explores reconstruction performance under higher bandwidth as would be available, *e.g.,* in 5G deployments.

Viewpoint filtering is the same as a raw point cloud, but with zero points removed. With this, it achieves a 10× compression throughout. As Table 13b shows, low compression is an order of magnitude more efficient beyond this. Despite this, AeroTraj can achieve high quality reconstruction. For the AirSim building, consider accuracy: the viewpoint filtered point cloud has an accuracy and completeness of 0.10 m and 0.08 m respectively, which is attributable entirely to SLAM error. Low compression, with a bandwidth of 3.8 Mbps (easily achievable over LTE and over 100× more compact than the raw LiDAR output) only adds 2 cm and 1 cm to accuracy and completeness, respectively. This also shows that there is little room for improvement with 5G bandwidths. Medium and high

Structure type	Flight duration (s)		Accuracy (m)		Completeness (m)	
	w/o	*w*	*w/o*	*w*	*w/o*	*w*
Tall rect.	1147	1430	0.31	**0.08**	0.15	**0.06**
Large rect.	855	1145	0.87	**0.09**	0.35	**0.05**
Pentagonal	891	1204	0.61	**0.11**	0.22	**0.08**
Small rect.	678	864	0.53	**0.12**	0.13	**0.09**
Gabled roof	667	862	0.32	**0.13**	0.11	**0.07**
Plus-shaped	620	1063	0.48	**0.16**	0.51	**0.07**

(a) Flight duration and reconstruction quality for buildings at low compression with (w) and without (w/o) re-calibration.

Compression profile	BW (Mbps)	Accuracy (m)	Completeness (m)
Real-world *70 m x 40 m x 20 m large building*			
View-point	42.7	0.00	0.00
Lossless	7.86	0.06	0.07
Low	3.80	0.13	0.09
Medium	2.50	0.23	0.16
High	1.27	0.28	0.29
AirSim *50 m x 50 m x 20 m small building*			
View-point	42.7	0.10	0.08
Lossless	7.86	0.10	0.10
Low	3.80	0.12	0.09
Medium	2.50	0.35	0.09
High	1.27	0.44	0.10

(b) The impact of compression on accuracy and/or completeness.

Table 13. Ablation study results.

compression have significantly poorer accuracy and completeness. Results for other buildings are similar, so we omit them for brevity.

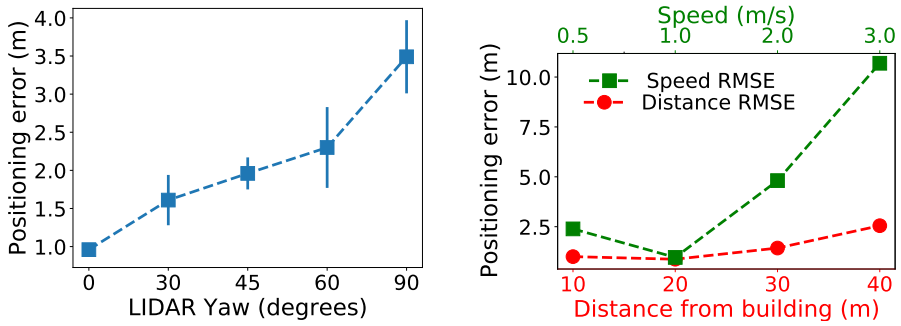
Results from our drone flights validate that *real-world* data of a large building (dimensions in Table 13b) results in comparable performance (Table 13b). Since we lack a reference ground truth for real-world data, we use the 3D model generated from raw traces. With real-world traces, we can build accurate, and complete 3D models that are within 9-13 cm completeness and accuracy for low compression, and about 16-23 cm for medium compression, relative to the raw traces. This suggests that highly compressed point clouds do not significantly impact accuracy and completeness.

To get a visual feel for the degradation resulting from lower accuracy, Fig. 16 shows the ground-truth model, together with the AeroTraj reconstructions at different compression levels. With an accuracy of 0.12 m (with 3.8 Mbps upload bandwidth), the model closely matches the ground truth. As accuracy worsens at higher compression levels, the textures on the building increasingly become less distinct and start to show some small artifacts, arising not because of compression but because of SLAM imperfections (§4.3).

4.3.5 Sensitivity: Target density. We evaluated AeroTraj at two different densities: 7.5 points per m^2 and 1 point per m^2. The lower density flight took only 31% of the higher density flight time. However, as expected, reconstruction is worse at lower target densities (0.18 m accuracy, 0.24 m completeness vs. 0.08 m and 0.06 m for the higher density). For applications that can tolerate this degradation, the smaller flight time might result in cost savings.

Speed (m/s)	Accuracy (m)	Completeness (m)	Flight time (s)
0.5	0.43	0.34	2045
1	**0.09**	**0.05**	1145
2	0.91	0.45	**768**

Table 14. Reconstruction accuracy, completeness, and flight time for AeroTraj with different flight speeds.



(a) SLAM errors for various LiDAR orientations.

(b) SLAM error for speeds and building distances.

Fig. 17. Parameter study for optimal flight parameters in model collection.

4.4 Data Collection Parameter Choices

AeroTraj determines model collection flight parameters with a parameter sweep in simulation and on *real-world traces* (§3.2). In simulations, we evaluated SLAM error for every combination of drone speed (0.5 m/s to 3 m/s), distance from building (10 m to 40 m), and LiDAR orientation (ranging from parallel to perpendicular).

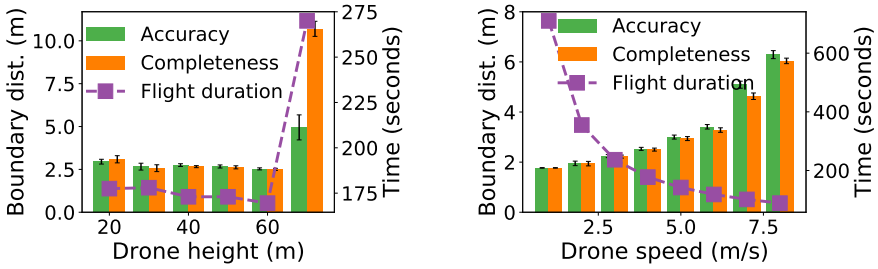
4.4.1 Orientation. Fig. 17a plots SLAM error as a function of LiDAR orientation (Fig. 7) with respect to the direction of motion. A parallel orientation has the lowest SLAM error (in Fig. 17a, yaw 0° is parallel and yaw 90° is perpendicular), because it has the highest overlap between successive frames; as yaw increases, overlap decreases, resulting in higher SLAM error (§3.2).

4.4.2 Distance. Fig. 17b plots the SLAM error as a function of the drone's distance from the building surface for the *parallel* orientation of the LiDAR. Error increases slowly with height; beyond a 20 m distance from the building, the error is more than 1 m. Point densities decrease with height and affect SLAM's ability to track features/points across frames (§3.2). Flying close to the building surface reduces scan width and hence increases flight duration. Rather than fly lower, AeroTraj operates at a 20 m distance from the building to reduce flight duration.

4.4.3 Speed. Speed impacts SLAM positioning significantly (Fig. 17b). Beyond 1 m/s, SLAM cannot track frames accurately due to lower overlap between frames (§3.2). Below 1 m/s *i.e.,* at 0.5 m/s, the flight duration (in seconds) is twice that of 1 m/s resulting in drift accumulation. For accurate reconstruction, AeroTraj flies the drone at 1 m/s. Table 14 explains the impact of drone speed on reconstruction quality. Although flying faster *i.e.,* at 2 m/s decreases the flight time by 33%, this comes at the cost of 80 cm in accuracy and 40 cm in completeness relative to flying at 1 m/s. This is because at higher speeds, SLAM cannot accurately track frames because of lower overlap. On the other hand, flying slower *i.e.,* at 0.5 m/s doubles the flight duration which causes SLAM to accumulate more drift, hence adding 34 cm in accuracy and 29 cm in completeness. AeroTraj flies the drone at 1 m/s which minimizes accuracy, completeness, and flight time.

LiDAR Orientation	Speed (m/s)		Distance (m)	
	1.5	*3.0*	*20*	*40*
Parallel	1.3	3.3	1.2	5.4
Perpendicular	3.1	7.6	2.1	∞

Table 15. Positioning errors for parallel and perpendicular orientations at different speeds (at distance 20 m) and errors for different distances (at a speed of 1 m/s) for *real-world traces.*



(a) A recon flight height of 60 m above the building surface minimizes accuracy, completeness (left y-axis) and flight duration (right y-axis).

(b) A recon flight speed 4-5 m/s ensures low accuracy, completeness (left y-axis) and flight duration (right y-axis).

Fig. 18. Rooftop geometry extraction accuracy, completeness, and flight duration as function of drone height, and speed.

4.4.4 Real-world validation. To validate these observations, we performed a parameter sensitivity study on real-world flights to determine the optimum parameters for SLAM positioning. For the lack of accurate ground truth in the real-world, we compare SLAM positions against a GPS trace. Because GPS is erroneous, we only draw qualitative conclusions.

As Table 15, taken from our drone traces, shows, slower flights have lower SLAM error than faster ones, and parallel orientations have lower SLAM error than perpendicular. Similarly, SLAM error increases with height and, in real-world traces, the parallel orientation seems to be significantly better than the perpendicular orientation (Table 15). At a distance of 20 m from the surface of the building, the parallel orientation has the minimum positioning error *i.e.,* 1.25 m. Beyond 40 m for parallel and 20 m for perpendicular, SLAM loses track completely because of lower point density.

4.5 Rooftop Geometry

We use two metrics for rooftop geometry estimation: accuracy, and completeness. Accuracy is the average (2D) distance between each point (quantized to 0.1 m) on the predicted rooftop boundary and the nearest point on the actual building boundary. Completeness, is the average distance between each point on the actual boundary and the nearest point on AeroTraj's predicted boundary. Lower values of accuracy and completeness are better. We use real-world traces collected from the AeroTraj prototype and synthetic traces from AirSim. For real-world traces, we pinpointed the building's boundary on Google Maps [8] for ground truth. For AirSim, we collected the ground truth from the Unreal Engine.

Table 10 shows the execution time for rooftop geometry estimation, on *real-world* traces, with the cloud VM. The average processing time per frame is 93 ms, dominated by GPU-accelerated surface normal estimation (76 ms). This can sustain 10 fps.

AeroTraj also extracts rooftop geometry accurately. Across 3 *real-world traces* collected over a 70m x 60m x 20m building, its average accuracy is 1.42 m and completeness is 1.25 m, even at the highest compression and when it samples every other frame. These results justify our choice of a fast, high, reconnaissance flight (§3.4).

We extensively evaluated robustness to different building shapes (Table 8), point cloud compression (Table 13b), subsampling, and flight parameters (Fig. 18a and Fig. 18b justify the choice of the recon speed and height). We omit the details for brevity but report the following: (a) recon flights can be short (boundary detection is insensitive to point density and overlap), so it can use perpendicular orientation to reduce flight duration, fly at 60 m from the building's surface (Fig. 18a) at 4 m/s (Fig. 18b); (b) it tolerates subsampling up to 1 Hz; (c) it generalizes to different building geometry described in Table 4.

5 DISCUSSION AND FUTURE WORK

Generalization to other buildings and structures. In this paper, we have focused on reconstructing large outdoor regular-shaped buildings, which make up more than 99% of residential and commercial buildings in large cities (Table 4). AeroTraj cannot reconstruct the remaining 24 buildings in Table 4 because its rooftop geometry extractor and 3D unfolding algorithms were not designed for buildings with hemispherical (domes like cathedrals and mosques), and ellipsoidal roofs (the Empire State Building). We have left extending AeroTraj to support these buildings and other large structures like bridges and stadiums to future work. As an aside, for large buildings with model collection flights that exceed the drone's battery time, AeroTraj can return and land the drone at the origin (similar to a re-calibration maneuver), swap the battery, and then resume the remaining model collection flight.

AeroTraj's model collection techniques are applicable to other large physical structures such as blimps and aircraft. We encapsulated a blimp in a 3D bounding box, and reconstructed it using AeroTraj with an accuracy of 20 cm and completeness of 3 cm. Further examination and extension of AeroTraj to such structures is left for future work.

Reducing end-to-end latency. AeroTraj's average end-to-end latency is 234 ms, with compression and network latency contributing 88 ms and 104 ms, respectively. Recent work [25] has demonstrated 1-2 orders of magnitude improvements in octree search, which can lead to faster compression times. Offload to edge compute instead of a remote cloud instance can reduce network latency to 20ms or less [27, 79].

Cloud offload. In the future, mobile compute capabilities will improve. However, at the same time, LiDAR technology continues to evolve and 128 beam LIDARs are already available, and these will require significantly more compute. So it is unclear if, or when, accurate LiDAR SLAM can run entirely on the drone. Even if that becomes feasible, AeroTraj's trajectory optimization and re-calibration algorithms will be relevant for accurate 3D modeling.

Mobile device LiDAR-based reconstruction. AeroTraj could have used LiDARs now available on mobile devices (*e.g.,* iPhones) to perform 3D reconstruction. However, LiDARs on mobile devices work well only in indoor environments because: a) the mobile device LiDAR has a limited sensing range of about 3 m [52], and b) has high depth estimation errors beyond this range *e.g.,* up to 30 cm at 4 m [13]. In outdoor spaces, these approaches can have tracking errors in meters [12] which leads to undesirable results.

Infrastructure support to improve AeroTraj. Loop closure is a critical component in SLAM and AeroTraj to reduce drift error. AeroTraj invokes loop closure when it detects excess drift; this leads to longer flight times. For future work, AeroTraj can plant April Tags [55] at well-defined locations on the buildings to recalibrate SLAM (hence invoking loop closure) without the drone having to fly back to the origin. Finally, AeroTraj uses

flight distance as a proxy for drone battery. Future work can explore incorporating more sophisticated battery models into the trajectory optimization.

6 RELATED WORK

Networked 3D sensing. Recent work has explored, in the context of cooperative perception [37, 60, 68, 78] and real-time 3D map updates [16], transmitting 3D sensor information over wireless networks. Compared to AeroTraj, they use different techniques to overcome wireless capacity constraints. Of these, VI-Eye [37] and VIPS [68] propose registration techniques to align a pair of point clouds (one captured from a vehicle, and another from an infrastructure-mounted LiDAR). To do this, they use additional scene information (*e.g.,* road landmarks, and vehicle bounding boxes). As with ICP, progressively using VI-Eye and VIPS to align large number of point clouds can result in significant drift. In contrast, AeroTraj uses drift detection, and mitigation techniques (in addition to SLAM's bundle adjustment), making it more suitable for aligning thousands of raw point clouds accumulated over large drone flights.

Drone positioning. The robotics literature has studied efficient coverage path-planning for single [71], and multiple drones [56]. AeroTraj's trajectory design is influenced by more intricate constraints like SLAM accuracy and equi-density goals. Accurately inferring drone motion is important for SLAM-based positioning [22]. Cartographer [38], which AeroTraj uses for positioning, utilizes motion models and on-board IMUs for estimating motion. A future version of AeroTraj can leverage drone orchestration [36] and SLAM edge-offloading [20] for larger scale reconstruction.

Offline reconstruction using images. UAV photogrammetry [29] reconstructs 3D models offline from 2D images. Several pieces of work [26, 41, 45, 70] study the use of RGB, and RGB-D cameras on UAVs for 3D reconstruction. Prior work [26] has proposed a real-time, interactive interface into the reconstruction process for a human guide. The most relevant of these [58, 72] predicts the completeness of 3D reconstruction in-flight, using a quality confidence predictor trained offline, for a better offline 3D reconstruction. However, unlike AeroTraj, this work requires human intervention, computes the 3D model offline, requires close-up flights, cannot ensure equi-dense reconstructions, cannot dynamically re-calibrate for drift and is not an end-to-end system. A body of work has explored factors affecting reconstruction accuracy: sensor error [39], tracking drift, and the degree of image overlap [26, 50]. Other work [23, 45, 46] has explored techniques to reduce errors by fusing with depth information, or using image manipulations such as upscaling. Unlike AeroTraj, almost all of this work reconstructs the 3-D model offline.

Offline reconstruction using LiDAR or radar. 3D model reconstruction using LiDAR [47, 74] relies on additional positioning infrastructure such as base stations for real-time kinematic (RTK) positioning, and long-range specialized LiDAR to achieve tens of centimeters model accuracy. AeroTraj explores a different part of the design space: online reconstruction with sub-meter accuracy using commodity drones, GPS and LiDAR. Recent work has explored drone-mounted LiDAR based offline reconstruction of tunnels and mines, but require specialized LiDARs and a human-in-the-loop [11, 21] for drone guidance (either manually or by defining a set of waypoints). Finally, mmMesh [76] explores cloud-offload for human mesh reconstructions using millimeter-wave radar.

7 CONCLUSIONS

In this paper, we have taken a step towards accurate, near-real time 3D reconstruction using drones. Our system, AeroTraj, uses novel techniques for navigating the tension between cellular bandwidths, SLAM positioning errors, and compute constraints on the drone. It contains algorithms for optimized trajectory generation, and for determining excessive SLAM drift. It can achieve reconstruction accuracy to within 10 cm in near real-time, even after compressing LiDAR data enough to fit within achievable LTE speeds.

REFERENCES

[1] 5 Ways Drones are Being Used for Disaster Relief. https://safetymanagement.eku.edu/blog/5-ways-drones-are-being-used-for-disaster-relief/.

[2] Basic Roof Types. https://www.livehome3d.com/useful-articles/12-basic-roof-types.

[3] Best Drones for Natural Disaster Response. https://www.dronefly.com/blogs/news/drones-flooding-sar-disaster/.

[4] Construction Site Monitoring using Unmanned Aerial Vehicle. https://www.equinoxsdrones.com/blog/construction-site-monitoring-using-unmanned-aerial-vehicle.

[5] DroneDeploy. https://www.dronedeploy.com/solutions/roofing/.

[6] DroneDeploy: Making Successful Maps. https://www.support.dronedeploy.com/docs/making-successful-maps/.

[7] Expediting Disaster Relief Services With Drone Technology. https://www.dronedeploy.com/blog/expediting-disaster-relief-services-with-drone-technology/.

[8] Google Maps. https://www.google.com/maps.

[9] Hard-Learned Lessons in Drone Photogrammetry. https://www.pobonline.com/articles/101581-hard-learned-lessons-in-drone-photogrammetry.

[10] Hover. https://hover.to/.

[11] Hovermap. https://www.emesent.io/hovermap/.

[12] iPhone and iPad LiDAR Spatial Tracking Capabilities: Second Test. https://www.vgis.io/2020/12/02/lidar-in-iphone-and-ipad-spatial-tracking-capabilities-test-take-2/.

[13] iPhone's 12 Pro LiDAR: How to Get and Interpret Data. https://www.it-jim.com/Blog/Iphones-12-Pro-Lidar-How-To-Get-And-Interpret-Data/.

[14] The DroneDeploy App. https://www.dronedeploy.com/.

[15] Troubleshooting Photogrammetry Issues. https://www.aerotas.com/troubleshooting-photogrammetry-issues.

[16] Fawad Ahmad, Hang Qiu, Ray Eells, Fan Bai, and Ramesh Govindan. CarMap: Fast 3D Feature Map Updates for Automobiles. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, Santa Clara, CA, 2020. USENIX Association.

[17] Nataraj Akkiraju, Herbert Edelsbrunner, Michael Facello, Ping Fu, EP Mucke, and Carlos Varela. Alpha Shapes: Definition and Software. In *Proceedings of the 1st International Computational Geometry Software Workshop*, volume 63, page 66, 1995.

[18] Esther M. Arkin, Sándor P. Fekete, and Joseph S.B. Mitchell. Approximation Algorithms for Lawn Mowing and Milling. *Computational Geometry*, 17(1):25–50, 2000.

[19] T. Bailey and H. Durrant-Whyte. Simultaneous Localization and Mapping (SLAM): Part II. *IEEE Robotics Automation Magazine*, 13(3):108–117, 2006.

[20] Ali J Ben Ali, Zakieh Sadat Hashemifar, and Karthik Dantu. Edge-SLAM: Edge-assisted Visual Simultaneous Localization and Mapping. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 325–337, 2020.

[21] Liam Brown, Robert Clarke, Ali Akbari, Ujjar Bhandari, Sara Bernardini, Puneet Chhabra, Ognjen Marjanovic, Thomas Richardson, and Simon Watson. The Design of Prometheus: A Reconfigurable UAV for Subterranean Mine Inspection. *Robotics*, 9(4):95, 2020.

[22] Mitch Bryson and Salah Sukkarieh. Observability Analysis and Active Control for Airborne SLAM. *IEEE Transactions on Aerospace and Electronic Systems*, 44(1):261–280, 2008.

[23] E. Bylow, R. Maier, F. Kahl, and C. Olsson. Combining Depth Fusion and Photometric Stereo for Fine-Detailed 3D Models. In *Scandinavian Conference on Image Analysis (SCIA)*, Norrköping, Sweden, June 2019.

[24] Jeremy Castagno and Ella Atkins. Roof Shape Classification from LiDAR and Satellite Image Data Fusion Using Supervised Learning. *Sensors*, 18(11):3960, 2018.

[25] Faquan Chen, Rendong Ying, Jianwei Xue, Fei Wen, and Peilin Liu. ParallelNN: A Parallel Octree-based Nearest Neighbor Search Accelerator for 3D Point Clouds. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 403–414, 2023.

[26] S. Daftry, C. Hoppe, and H. Bischof. Building with Drones: Accurate 3D Facade Reconstruction using MAVs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3487–3494, 2015.

[27] Kaikai Deng, Dong Zhao, Qiaoyue Han, Shuyue Wang, Zihan Zhang, Anfu Zhou, and Huadong Ma. Geryon: Edge assisted real-time and robust object detection on drones via mmwave radar and camera fusion. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–27, 2022.

[28] H. Durrant-Whyte and T. Bailey. Simultaneous Localization and Mapping (SLAM): Part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[29] A Federman, M Santana Quintero, S Kretz, J Gregg, M Lengies, C Ouimet, and J Laliberte. UAV Photogrammetric Workflows: A Best Practice Guideline. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42, 2017.

[30] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[31] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. Salsify: Low-Latency Network Video Through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 267–282, 2018.
[32] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view Stereo: A Tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.
[33] Yoav Gabriely and Elon Rimon. Spanning-tree based Coverage of Continuous Areas by a Mobile Robot. *Annals of Mathematics and Artificial Intelligence*, 31:77–98, 2001.
[34] Enric Galceran and Marc Carreras. A Survey on Coverage Path Planning for Robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.
[35] Michael Grupp. Evo: Python Package for the Evaluation of Odometry and SLAM. https://github.com/MichaelGrupp/evo.
[36] Songtao He, Favyen Bastani, Arjun Balasingam, Karthik Gopalakrishna, Ziwen Jiang, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. BeeCluster: Drone Orchestration via Predictive Optimization. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 299–311, 2020.
[37] Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. VI-Eye: Semantic-based 3D Point Cloud Registration for Infrastructure-assisted Autonomous Driving. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 573–586, 2021.
[38] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time Loop Closure in 2D LiDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.
[39] Sin-Yi Jiang, Nelson Yen-Chung Chang, Chin-Chia Wu, Cheng-Hei Wu, and Kai-Tai Song. Error Analysis and Experiments of 3D Reconstruction using a RGB-D Sensor. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1020–1025. IEEE, 2014.
[40] Yurong Jiang, Hang Qiu, Matthew McCartney, Gaurav Sukhatme, Marco Gruteser, Fan Bai, Donald Grimm, and Ramesh Govindan. CARLOC: Precise Positioning of Automobiles. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 253–265, New York, NY, USA, 2015. Association for Computing Machinery.
[41] D. Lapandic, J. Velagic, and H. Balta. Framework for Automated Reconstruction of 3D Model from Multiple 2D Aerial Images. In *2017 International Symposium ELMAR*, pages 173–176, 2017.
[42] Insoo Lee, Jinsung Lee, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Demystifying Commercial Video Conferencing Applications. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3583–3591, 2021.
[43] Stephen Lee, Srinivasan Iyengar, Menghong Feng, Prashant Shenoy, and Subhransu Maji. DeepRoof: A Data-driven Approach for Solar Potential Estimation using Rooftop Imagery. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2105–2113, 2019.
[44] Hui Li, Cheng Zhong, Xiaoguang Hu, Long Xiao, and Xianfeng Huang. New Methodologies for Precise Building Boundary Extraction from LiDAR Data and High Resolution Image. *Sensor Review*, 2013.
[45] Jianwei Li, Wei Gao, and Yihong Wu. High-quality 3D Reconstruction with Depth Super-resolution and Completion. *IEEE Access*, 7:19370–19381, 2019.
[46] Z. Li, P. C. Gogia, and M. Kaess. Dense Surface Reconstruction from Monocular Vision and LiDAR. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6905–6911, 2019.
[47] Yi-Chun Lin, Yi-Ting Cheng, Tian Zhou, Radhika Ravi, Seyyed Meghdad Hasheminasab, John Evan Flatt, Cary Troy, and Ayman Habib. Evaluation of UAV LiDAR for Mapping Coastal Environments. *Remote Sensing*, 11(24):2893, 2019.
[48] Sheng Liu, Xiaohan Nie, and Raffay Hamid. Depth-guided Sparse Structure-from-motion for Movies And TV Shows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15980–15989, 2022.
[49] Xiaochen Liu, Suman Nath, and Ramesh Govindan. Gnome: A Practical Approach to NLOS Mitigation for GPS Positioning in Smartphones. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 163–177, 2018.
[50] Jean Liénard, Andre Vogs, Demetrios Gatziolis, and Nikolay Strigul. Embedded, Real-time UAV Control for Improved, Image-based 3D Scene Reconstruction. *Measurement*, 81:264 – 269, 2016.
[51] Gurobi Optimization LLC. Gurobi Optimizer Reference Manual. http://www.gurobi.com, 2020.
[52] Gregor Luetzenburg, Aart Kroon, and Anders A Bjørk. Evaluation of the Apple iPhone 12 Pro LiDAR for an Application in Geosciences. *Scientific reports*, 11(1):1–9, 2021.
[53] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
[54] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[55] Luis A Mateos. Apriltags 3D: Dynamic Fiducial Markers for Robust Pose Estimation in Highly Reflective Environments and Indirect Communication in Swarm Robotics. *arXiv preprint arXiv:2001.08622*, 2020.
[56] Jalil Modares, Farshad Ghanei, Nicholas Mastronarde, and Karthik Dantu. Ub-Anc Planner: Energy Efficient Coverage Path Planning with Multiple Drones. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6182–6189, 2017.
[57] Trevor Mogg. Austrian Airlines is Flying a Drone around its Planes for a Good Reason. https://www.digitaltrends.com/cool-tech/austrian-airlines-is-flying-a-drone-around-its-planes-for-a-good-reason/.
[58] Christian Mostegel, Markus Rumpler, Friedrich Fraundorfer, and Horst Bischof. UAV-based Autonomous Image Acquisition with Multi-view Stereo Quality Assurance by Confidence Prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2016.
[59] Gábor Pataki. Teaching Integer Programming Formulations Using The Traveling Salesman Problem. *SIAM review*, 45(1):116–123, 2003.
[60] Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. AVR: Augmented Vehicular Reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (Mobisys)*, MobiSys '18, pages 81–95, Munich, Germany, 2018. ACM.
[61] Anandakumar M Ramiya, Rama Rao Nidamanuri, and Ramakrishan Krishnan. Segmentation based Building Detection Approach from LiDAR Point Cloud. *The Egyptian Journal of Remote Sensing and Space Science*, 20(1):71–77, 2017.
[62] Boris Schling. *The Boost C++ Libraries*. XML Press, 2011.
[63] Ruwen Schnabel and Reinhard Klein. Octree-based Point-Cloud Compression. In *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'06, page 111–121, Goslar, DEU, 2006. Eurographics Association.
[64] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-view Stereo. In *European Conference on Computer Vision (ECCV)*, 2016.
[65] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A Comparison and Evaluation of Multi-view Stereo Reconstruction Algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006.
[66] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics: Results of the 11th International Conference*, pages 621–635. Springer, 2018.
[67] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-Sam: Tightly-Coupled LiDAR Inertial Odometry via Smoothing and Mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020.
[68] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. VIPS: Real-time Perception Fusion for Infrastructure-assisted Autonomous Driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 133–146, 2022.
[69] Stanford Artificial Intelligence Laboratory Et Al. Robotic Operating System.
[70] L. Teixeira and M. Chli. Real-time Local 3D Reconstruction for Aerial Inspection using Superpixel Expansion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4560–4567, 2017.
[71] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5321–5326, 2013.
[72] Trong Hai Trinh, Manh Ha Tran, et al. Hole Boundary Detection of a Surface of 3D Point Clouds. In *2015 International Conference on Advanced Computing and Applications (ACOMP)*, pages 124–129. IEEE, 2015.
[73] Shinji Umeyama. Least-Squares Estimation of Transformation Parameters between two Point Patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):376–380, 1991.
[74] Bin Wu, Bailang Yu, Qiusheng Wu, Shenjun Yao, Feng Zhao, Weiqing Mao, and Jianping Wu. A Graph-Based Approach for 3D Building Model Reconstruction from Airborne LiDAR Point Clouds. *Remote Sensing*, 9:92, 01 2017.
[75] Xiufeng Xie and Xinyu Zhang. Poi360: Panoramic Mobile Video Telephony over LTE Cellular Networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 336–349, 2017.
[76] Hongfei Xue, Yan Ju, Chenglin Miao, Yijiang Wang, Shiyang Wang, Aidong Zhang, and Lu Su. mmMesh: Towards 3D Real-time Dynamic Human Mesh Construction using Millimeter-wave. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 269–282, 2021.
[77] Ji Zhang and Sanjiv Singh. LOAM: LiDAR Odometry and Mapping in Real-time. In *Robotics: Science and Systems*, volume 2, 2014.
[78] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. EMP: Edge-assisted Multi-vehicle Perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 545–558, 2021.
[79] Yunfan Zhang, Tim Scargill, Ashutosh Vaishnav, Gopika Premsankar, Mario Di Francesco, and Maria Gorlatova. Indepth: Real-time depth inpainting for mobile augmented reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–25, 2022.