geographies

*Article*

# Towards Real-Time 3D Terrain Reconstruction from Aerial Imagery

**Qiaosong Wang**

Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA; qiaosong@udel.edu

**Abstract:** We present a near real-time solution for 3D reconstruction from aerial images captured by consumer UAVs. Our core idea is to simplify the multi-view stereo problem into a series of two-view stereo matching problems. Our method applies to UAVs equipped with only one camera and does not require special stereo-capturing setups. We found that the neighboring two video frames taken by UAVs flying at a mid-to-high cruising altitude can be approximated as left and right views from a virtual stereo camera. By leveraging GPU-accelerated real-time stereo estimation, efficient PnP correspondence solving algorithms, and an extended Kalman filter, our system simultaneously predicts scene geometry and camera position/orientation from the virtual stereo cameras. Also, this method allows for the user selection of varying baseline lengths, which provides more flexibility given the trade-off between camera resolution, effective measuring distance, flight altitude, and mapping accuracy. Our method outputs dense point clouds at a constant speed of 25 frames per second and is validated on a variety of real-world datasets with satisfactory results.

**Keywords:** 3D reconstruction; unmanned aerial vehicles; real-time systems

check for updates

**Citation:** Wang, Q. Towards Real-Time 3D Terrain Reconstruction from Aerial Imagery. *Geographies* **2024**, *4*, 66–82. https://doi.org/10.3390/geographies4010005

Academic Editor: Eliseo Clementini

Received: 13 August 2023
Revised: 15 January 2024
Accepted: 23 January 2024
Published: 29 January 2024

## 1. Introduction

Recent advances in chipmaking and imaging sensor technologies enabled micro unmanned aerial vehicles (UAV) to reduce cost and quickly progress into consumer markets. Micro UAVs are widely used in agriculture, construction, transportation, and film-making because of their exceptional stability, mobility, and flexibility. Specifically, aerial remote sensing provides a rapid, inexpensive, and highly automated approach to producing 3D digital assets for easy measurement, inspection, planning, and management. Presently, a significant portion of terrestrial scans are acquired using either ground-based or aerial-based LiDAR systems. However, such systems are bulky, expensive, power-intensive, and often unable to provide dense textured mesh. It is more desirable to enable real-time high-resolution mapping capabilities for consumer-level camera drones. Compared to terrestrial LiDARs, consumer UAVs capture high-quality data in less time and at a lower overall cost. Also, the operational complexity is much less for drones compared to terrestrial LiDAR workflows. The redundancy of overlapping images, together with georeferencing and auto-alignment enforced by photogrammetry algorithms, translates to high accuracy outputs even in non-ideal operational or imaging conditions.

However, current photogrammetry pipelines usually take hours to days to fully process captured imagery data, which greatly limits their capability for real-time data acquisition. Such an ability is essential in scenarios such as onsite scan and inspection, emergency response, and planning. An ideal scanning pipeline should be able to map the scene in real time as the drone travels along the flight path and gives inspectors the ability to examine the partial scan even before the flight mission is finished. Currently, it is difficult to achieve such goals using photogrammetry approaches, for a few reasons. Firstly, photogrammetry approaches typically require global optimization for camera pose estimation, dense multi-view stereo matching, meshing, and texture mapping. So, at the framework level it prevents online real-time mapping in an incremental manner. Some algorithms first load all feature points into memory, begin from the initial image pairs, and then incrementally extend the scene by registering new images and new 3D points. Our method is along the lines of performing incremental 3D reconstructions. However, our method still has subtle differences compared to the aforementioned approaches. Our method performs feature extraction, verification, camera pose estimation/refinement, and dense stereo all at the same time on a frame-by-frame basis, whereas other approaches still separate the whole process into stages. For example, in most competitor approaches the dense multi-view stereo matching step cannot begin without the camera pose estimation (or SfM/sparse reconstruction) process completed for all images. In such cases, although the camera pose estimation or sparse point cloud reconstruction step is performed incrementally, the whole pipeline still requires the entire dataset as input and is thus not applicable for online or real-time reconstruction. Secondly, for most of the prior work the structure from motion (SfM) and multi-view stereo (MVS) components used in their pipelines are typically computationally intensive and time-consuming. SfM algorithms typically densely scan the input images and extract feature points in order to find correspondences between these images in different views. Thirdly, a few failures or mismatches of images may break the entire photogrammetry pipeline. Imaging issues such as reflection, lack of texture, strong lighting, and moving objects may also result in poorly reconstructed scenes.

To tackle this problem, we propose an alternative approach. At the core of our design, we use high-efficiency algorithms with the lowest computational cost at each and every step of the pipeline. Therefore, we choose to use Harris corner [1] instead of SIFT [2] or SURF descriptors [3] for feature extraction. We adopt real-time Belief Propagation (BP) stereo matching on downsampled images instead of running large-scale full-resolution MVS matching. We also use the very fast EPnP [4] algorithm instead of performing Bundle Adjustment (BA) [5,6] as in many SfM pipelines. Finally, we employ the EKF filter [7] to ensure reconstruction accuracy. The resulting system outputs an accurate dense point cloud in near real-time at a constant speed of 25 frames per second. The overall pipeline is shown in Figure 1.

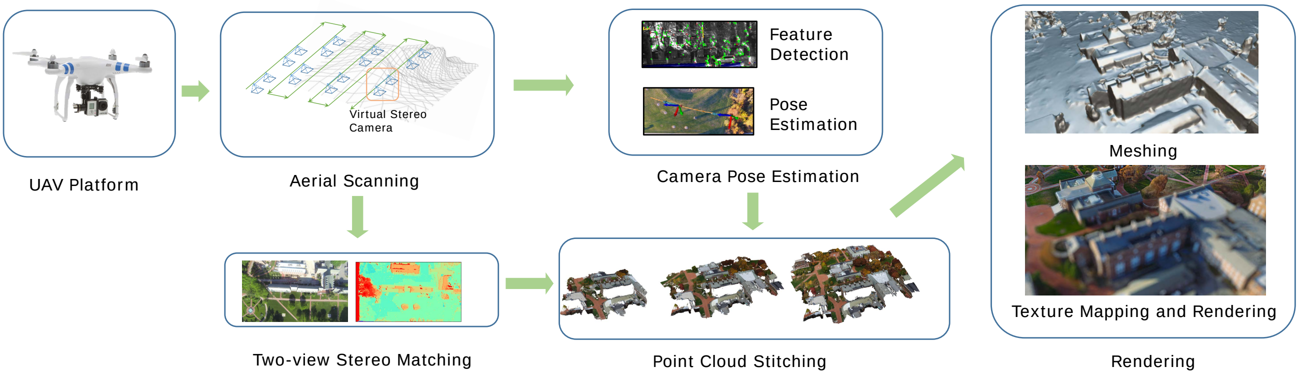


**Figure 1.** The processing pipeline of our system. We decompose the multi-view reconstruction problem into pair-wise two-view stereo-matching problems. We perform fast camera pose estimation to stitch point clouds together and achieve near real-time speed at producing dense point clouds. Finally, we perform a meshing and texture mapping routine to obtain the final scanned model and visualize it in a web environment.

## 2. Related Work

**Global vs. Local Stereo Matching** Because of computing efficiency and accuracy constraints, for this work, we are only investigating stereo algorithms before the deep learning era. Deep learning algorithms have made tremendous progress in terms of both speed and accuracy for this task. However, such algorithms require special hardware and training data. Collecting the training data is very time-consuming and labor-intensive. In contrast, traditional stereo matching algorithms are bottom-up and unsupervised and do not require any training data. Such algorithms are generally composed of four stages: cost computation, cost aggregation, cost optimization, and disparity refinement. The main difference between local/semi-local [8–11] and global [12–17] algorithms is whether one can construct and optimize a global energy function. Typically, local algorithms are based on computing windows and are sensitive to occlusion and homogeneous regions. Global methods are time-consuming but are much more accurate. This is carried out by incorporating smoothness terms and ensuring an optimal/sub-optimal solution via multiple iterations to reduce the global energy. In recent years, due to the rapid development of high-performance parallel computing hardware, it is possible to issue several thousands of threads at the same time to speed up the disparity calculation. By leveraging such massively parallel computing frameworks, it is possible to perform global stereo matching in real-time.

**vSLAM vs. SfM** Visual SLAM [18–22] and SfM approaches [6,23–25] are both based on multiview geometry and both aim at solving the camera motion/scene geometry estimation problem. The main difference between the two methods is in time complexity requirements and practical applications. Typically, SLAM is used in robotic applications focusing on using a single visual sensor with limited computing resources. Algorithms along this line generally use a very fast feature detector (LSD [22], ORB [19], FAST [26] etc.) and incorporate predictive/closed loop estimation frameworks to adjust the trajectory. Scene geometry estimation is a nice to-have feature but is not mandatory. This is because most vSLAM applications are only aiming at obtaining the trajectory/pose graphs. In contrast, SfM algorithms are more geared towards reconstructing accurate 3D representations of the scene/objects. Therefore, SfM algorithms typically trade speed for accuracy, process data offline, and have a high demand for computing power [24]. SfM is often combined with multiview-stereo, surface reconstruction, and texture mapping steps to output the final mesh model. Such pipelines typically have very high requirements for imaging quality and are not very robust to real-world data (lens glare, motion/defocus blur, reflection/transparency, thin structures, etc.).

**Aerial vs. Terrestrial Data Collection** Using consumer drones for 3D reconstruction has many benefits compared to traditional ground-based survey practices. Firstly, drones can automatically fly in a pre-programmed pattern, collecting well-lit and occlusion-free images, while covering inaccessible or dangerous places with much lower risk (e.g., inspecting highway bridges without stopping traffic). Secondly, this only takes a fraction of the time and cost of ground-based surveys. Moreover, the data collection process is reproducible, meaning that the drone could fly in the exact same pattern and collect data for comparison in the future. This is extremely helpful for tasks like inspecting structural cracks on highways/bridges/critical infrastructure, surveying coastline/sea floor/wetlands, or tracking of construction projects [27,28]. While consumer drones are great for aerial photogrammetry, there are a few drawbacks. Firstly, current onboard sensors and local obstacle avoidance systems are still not good enough for close-proximity flights, meaning that the drone could not collect enough parallax on vertical structures, limiting the accuracy of the final mesh when there is heavy occlusion. Secondly, LiDAR works on a variety of objects including specular/metallic surfaces and textureless regions with certain levels of see-through capability on transparent objects (e.g., cloud, rain, and snow). On the contrary, photogrammetry algorithms operating on camera sensors will start to fail when reflective/textureless/transparent regions increase. Our focus revolves primarily around applications such as large-scale city/terrain visualization, flood/construction simulation, surveying, and similar scenarios. We do not target use cases demanding exceptionally high model accuracy, such as the construction of HD maps for autonomous driving.

## 3. Methodology

### *3.1. Camera Calibration*

Our objective is to produce precise scans of buildings and terrain, ensuring accurate metric measurements of model dimensions. Therefore, it is very important to map camera measurements to real-world coordinates. Since the GoPro camera we are using is wide-angle with severe distortion, we need to perform calibration to obtain the intrinsic parameters of the camera. Our method supposes that nearby frames captured by UAVs flying at high altitudes can be approximated as left and right views from a generic stereo camera system. Therefore, for camera calibration, we only recover the intrinsic parameters and ignore the calibration of extrinsic parameters. We also assume that the camera has only radial distortion and does not have any tangential distortion. More formally,

$$\begin{aligned} x_{\text{corrected}} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y_{\text{corrected}} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{1}$$

We use a planar checkerboard to calibrate the stereo camera [29]. In OpenCV [30], this process can be carried out by calling *cvCalibrateCamera2()*. Once we pre-calibrated the camera, we could recover all intrinsic parameters such as focal length $(f_x, f_y)$, principle point $(c_x, c_y)$, skew $s$, and distortion parameters $(k_1, k_2, p_1, p_2, k_3)$. Note that there is only one physical camera, and all intrinsic parameters remain constant throughout the flight.

### *3.2. Depthmap Processing*

Once the images are rectified, we seek to obtain depth maps representing 3D coordinates of real-world points in the camera coordination system. Our GoPro camera device is with 2.7K resolution recording at 60 frames per second. It is almost impossible to run traditional global stereo-matching methods on this resolution in real time. To address this problem, we choose to perform stereo matching on low-resolution image pairs first and subsequently upsample the results with guidance from raw color images [31]. We use the GPU-based Belief Propagation [17] and Joint Bilateral Upsampling [32] in our implementation. We downsample the stereo image pairs to the resolution of 320 × 180 and seek to minimize the following energy function [17]:

$$E_{\mathbf{X}}(d) = E_{D,\mathbf{X}}(d) + E_{S,\mathbf{X}}(d) = E_{D,\mathbf{X}}(d) + \sum_{\mathbf{Y} \in N(\mathbf{X})} M_{\mathbf{Y},\mathbf{X}}(d) \tag{2}$$

where $M_{\mathbf{Y},\mathbf{X}}(d)$ is the message vector passed from a pixel to its neighbor. $E_{D,\mathbf{X}}$ is the data term, and $E_{S,\mathbf{X}}$ is the smoothness term. After a given number of optimization iterations, the label $d$ that minimizes $E_{\mathbf{X}}(d)$ is assigned to each pixel to form the disparity map. Next, we upsample the disparity map to obtain a high-resolution copy $D_p$ [32]:

$$D_p = \frac{1}{K_p} \sum_{q_\downarrow \in W} D'_{q_\downarrow} s\left(\|p_\downarrow - q_\downarrow\|\right) g\left(\|I_p - I_q\|\right) \tag{3}$$

The formula involves iterating over neighboring pixels in the downsampled depth map, represented by $q_\downarrow$, within a filter window $W$ centered around the pixel $p$ in the high-resolution image. The upsampled depth value $D_p$ at position $p$ is obtained by combining the downsampled depth values $D'_{q_\downarrow}$ from neighboring pixels with spatial weight $s\left(\|p_\downarrow - q_\downarrow\|\right)$ and range weight $g\left(\|I_p - I_q\|\right)$. The spatial weight measures the spatial distance between $p_\downarrow$ and $q_\downarrow$, while the range weight captures the color similarity between the pixel values $I_p$ and $I_q$ at positions $p$ and $q$ in the high-resolution and input images, respectively. By combining spatial and range weights, the technique preserves edges and details during upsampling, reducing artifacts and noise in the output. This also combines appearance information with fine levels of details from the full-resolution image and spatial information from the downsampled disparity maps (see Figure 2 for details).

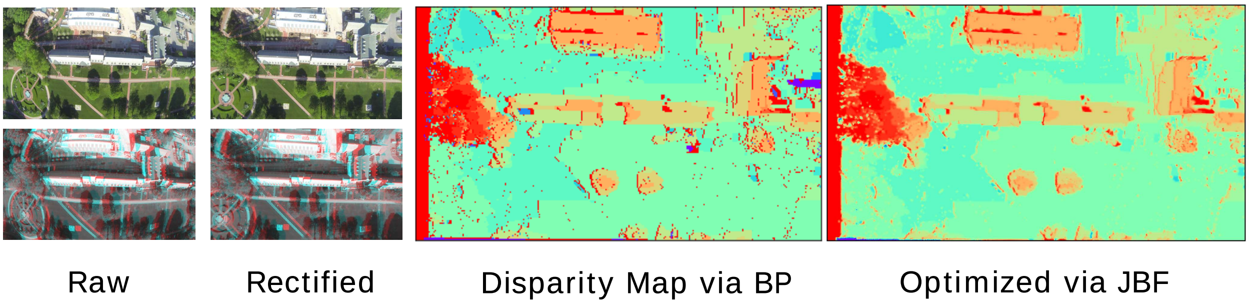


**Figure 2.** Example output from our stereo matching module. The first and second columns show raw images and rectified images and corresponding analygraphs. The third column shows the disparity map generated by belief propagation (BP) stereo matching. The final row shows the optimized disparity map using joint bilateral upsampling (JBF). Note that small errors caused by BP are eliminated via JBF, while all important edge details are preserved.

Once we have obtained the full resolution depth-map, suppose the baseline of the camera is $B$; $d_u$ and $d_v$ are lengths of the pixels on the horizontal and vertical axis; $(f_x, f_y)$ are the focal lengths of a camera and represent the scaling factors for the horizontal and vertical axes, respectively; $(x, y, z)$ are the 3D point coordinate in the camera coordinate system; $(u, v, d)$ are 2D image coordinates and depth value of a point in the image plane; and $(u_0, v_0)$ are principal point coordinates, representing the offset of the image center from the top-left corner. We could obtain:

$$\begin{aligned}
[u, v, d] &= \frac{1}{z}\left[\frac{fx}{\mathrm{d}u}, \frac{fy}{\mathrm{d}v}, \frac{fB}{\mathrm{d}u}\right] + [u_0, v_0, 0] \\
[x, y, z] &= \frac{B}{d}\left[u - u_0, \frac{(v - v_0)\mathrm{dv}}{\mathrm{du}}, \frac{fB}{\mathrm{d}u}\right]
\end{aligned} \tag{4}$$

Therefore, we could obtain point clouds for all virtual stereo image pairs with respect to their own camera coordinate systems.

### 3.3. Pose Estimation and Refinement

Our approach supposes nearby frames taken from the aerial video with a fixed interval can be treated as a set of left–right image pairs taken by virtual stereo cameras with identical baselines. However, it is worth noting that this assumption only holds locally. If the time difference between the two frames is too much, then we cannot assume an ideal stereo camera setup (exact same camera pose) and must re-estimate the camera pose and orientation. This is because if the distance between two captured shots is excessive, there will not be adequate overlap for calculation. Another scenario arises when the drone undergoes roll/pitch/yaw motions, making it impossible to calculate a depth map because corresponding pixels do not align on the scanline (violating the epipolar geometry assumption). In this section, we discuss how to recover orientations and poses for all stereo image pairs. Firstly, we need to introduce a feature descriptor to build correspondence between two camera frames. Instead of the commonly used SIFT [2]/SURF [3] feature descriptor, we choose to use the very fast Harris corner detector for our application. Suppose the image is $I_{u,v}$ and we choose a patch with a small shift $[x, y]$; the change of intensity could be formulated as [1]:

$$E_k = \sum_{\mathrm{u},v} w_{u,v}\left[\mathbf{I}_{x+u,y+v} - \mathbf{I}_{u,v}\right]^2 = \sum_{\mathrm{u},v} w_{u,v}\left[Ax + By + C\left(x^2, y^2\right)\right]^2 = (x, y)\mathbf{M}(x, y)^{\mathrm{T}} \tag{5}$$

where

$$A = X^2 \otimes w, B = Y^2 \otimes w, C = XY \otimes w$$
$$M(x,y) = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} \sum_{u,v} w_{u,v} I_x^2(x+u, y+v) & \sum_{u,v} w_{u,v} I_x I_y(x+u, y+v) \\ \sum_{u,v} w_{u,v} I_x I_y(x+u, y+v) & \sum_{u,v} w_{u,v} I_y^2(x+u, y+v) \end{bmatrix} \quad (6)$$

$A$ and $B$ are first-order partial derivatives of the image, and $w$ is the Gaussian window function to improve robustness to noises. $\mathbf{M}$ is a $2 \times 2$ matrix computed from the image derivatives. The Harris corner is detected when the response $R$ reaches the local maximum:

$$R = \mathrm{Det}(\mathbf{M}) - k\,\mathrm{Tr}^2(\mathbf{M}) \quad (7)$$

where $k$ is a constant multiplier that helps adjust the balance, and Det and Tr are the determinant and trace of the the structure tensor $M$. The Harris corner is related to the response from two directions and is thus invariant to rotation and translation. Also, the computation is relatively simple and could be performed in real-time. For a given threshold $R$, we choose a patch centered at every Harris corner and compute the correspondence between frame $I_{t1}$ and $I_{t2}$, where $t$ is the timestamp. The computation of correspondence will only be performed within the given search range. We use Squared Difference (ZSSD) to calculate the cost. Once the correspondences are obtained, we seek to obtain the rotation and translation matrices $[\mathbf{R}, \mathbf{T}]$, which describe the relative camera pose between time $t_1$ and $t_2$. We adopt the very fast algorithm described in [4] for determining the position and orientation of the cameras given their calibrated intrinsics and a set of correspondences. The core of this algorithm expresses 3D points as a weighted sum of 4 control points, reducing the computational complexity to $O(n)$. Including additional control points beyond the minimum 4 required for pose estimation does not necessarily improve the accuracy of the solution, and it may introduce numerical instability and increased computational complexity. Suppose $\mathbf{A}$ is the intrinsic matrix obtained from the previous section, and $\mathbf{u}_i$ are a set of Harris corners representing 3D points $\{\mathbf{p}_i^c\}_{i=1,\dots,n}$; we now have:

$$\forall i, w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = \mathbf{A}\mathbf{p}_i^c = \mathbf{K} \sum_{j=1}^{4} \alpha_{ij} \cdot \mathbf{c}_j^c \quad (8)$$

where $w_i$ are projective parameters, and $\left\{\mathbf{c}_j^c\right\}_{j=1,2,3,4}$ are control points in the camera coordinate system. By solving this linear system, we could obtain the translation and rotation matrices which map control point coordinates from real-world coordinate system to camera coordinate system:

$$\mathbf{c}_j^c = [\mathbf{R}, \mathbf{T}]\mathbf{c}_j^w \quad (9)$$

Since the proposed system is completely based on visual information, it is possible that bad imaging conditions (e.g., textureless regions, reflective surface glares) would lead to inconsistency in feature matching between the stereo frames. Additionally, the presence of noise interference introduces considerable fluctuations in the computed results. Consequently, the utilization of filtering techniques becomes essential to obtain more reliable and smoother outcomes. Given the relatively smooth nature of the camera's motion, without abrupt movements, we employ the Kalman filter based on a constant velocity model [7]. This choice is justified by its suitability for handling such motion characteristics and its potential to enhance the accuracy and robustness of the pose estimation results.

We begin by defining a 13-dimensional state vector as follows:

$$\mathbf{X}_k = \begin{pmatrix} r_{C_k}^W \\ q_{C_k}^W \\ v_{C_k}^W \\ \omega_{C_k}^C \end{pmatrix}$$

where $r^W_{C_k}$ represents the 3D coordinates of the camera in the world coordinate system at time step $k$. $q^W_{c_k}$ denotes the quaternion representation of the camera's pose in the world coordinate system at time step $k$. $v^W_{C_k}$ is the linear velocity vector of the camera with respect to the world coordinate system at time step $k$. Finally, $\omega^c_{c_k}$ represents the angular velocity vector of the camera relative to the camera coordinate system at time step $k$. These state variables constitute the essential elements for describing the camera's position, orientation, linear velocity, and angular velocity at each time step in a 3D environment. The constant velocity model assumes that the camera motion is relatively smooth, without abrupt changes; thus, it is suitable for accurately estimating the camera's trajectory and poses in a continuous manner. Incorporating this model into our pose estimation framework will enable more reliable and accurate camera tracking results as our drone is programmed to move and capture images in constant velocity and altitude.

Let us denote $\hat{\mathbf{X}}_{k+1|k}$ as the predicted state estimate at time step $k+1$ given the measurements up to time step $k$. It is computed using the function $\mathrm{f}_{k+1}$ as follows:

$$\hat{\mathbf{X}}_{k+1|k} = \mathrm{f}_{k+1}\begin{pmatrix} r^W_{C_k} \\ q^W_{C_k} \\ v^W_{C_k} \\ \omega^C_{C_k} \end{pmatrix} = \begin{pmatrix} r^W_{C_k} + \left(v^W_{C_k} + V^W_{k+1}\right)\Delta t \\ q^W_{C_k} \times q\left(\left(\omega^C_{C_k} + \Omega^C_{k+1}\right)\Delta t\right) \\ v^W_{C_k} + V^W_{k+1} \\ \omega^C_{C_k} + \Omega^C_{k+1} \end{pmatrix}$$

The function $q(\bullet)$ transforms a rotation vector into a quaternion. $V^W_k$ and $\Omega^C_k$ represent the noise sequences associated with linear velocity and angular velocity, respectively. They are considered to be zero-mean Gaussian random sequences with a covariance matrix $\mathbf{Q}_k$. The measurements $\mathbf{Z}_k$ of $\mathbf{X}_k$ satisfy a linear relationship, given by the measurement equation:

$$\mathbf{Z}_k = H\mathbf{X}_k + S_k$$

where $H$ is the measurement matrix

$$H = \begin{pmatrix} 0_{3\times3} & 0_{4\times4} & I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{4\times4} & 0_{3\times3} & I_{3\times3} \end{pmatrix}$$

Here $S_k$ represents the measurement noise at time step $k$, given by $\begin{pmatrix} V^W_k \\ \Omega^C_k \end{pmatrix}$. This linear measurement equation enables the fusion of measurements with the predicted state to update and refine the state estimates using the Kalman filter. Therefore, we could derive the discrete-time predict and update equations on the camera poses. In the prediction stage, the forecasted state and covariance estimate can be expressed as follows:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}_k\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k\right) \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^\top + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^\top \end{aligned} \tag{10}$$

Here, $\hat{\mathbf{x}}_{k|k-1}$ denotes the anticipated state estimate at time step $k$ based on all preceding measurements up to time step $k-1$ and is obtained using the motion model $\mathbf{f}_k$. $\mathbf{P}_{k|k-1}$ denotes the predicted state covariance matrix at time step $k$ incorporating the uncertainty in the state estimate due to the motion model and control input $\mathbf{u}_k$. $\mathbf{F}_k$ is the state transition matrix or Jacobian matrix of the motion model, while $\mathbf{G}_k$ is the control matrix or Jacobian matrix of the motion model with respect to the control input $\mathbf{u}_k$. Finally, $\mathbf{Q}_k$ represents the e zero-mean noise added to the motion model at time step $k+1$. In the update stage, we denote the near-optimal Kalman gain as $\mathbf{K}_k$, the updated state estimate as $\hat{\mathbf{x}}_{k|k-1}$, and the updated covariance estimate as $\mathbf{P}_{k|k-1}$. The three variables can be computed as follows:

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}^{\mathrm{T}}\left(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^{\mathbf{T}} + \mathbf{T}_k\mathbf{R}_k\mathbf{T}_k^{\mathbf{T}}\right)^{-1} \\
\hat{\mathbf{x}}_{k|k-1} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}\right) \\
\mathbf{P}_{k|k-1} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}
\end{aligned} \tag{11}$$

Here, $\mathbf{K}_k$ represents the Kalman gain at time step $k$. The Kalman gain is a matrix used to determine the weight given to the current measurement and the predicted state estimate in the update process. It is calculated based on the predicted state covariance measurement matrix $\mathbf{H}$, the measurement noise covariance matrix $\mathbf{R}_k$, and a Jacobian matrix of the measurement function $\mathbf{T}_k$ to map the measurement noise covariance $\mathbf{R}_k$ from the measurement space to the state space. $\mathbf{P}_{k|k-1}$ denotes the predicted state covariance matrix at time step $k$ given all past measurements up to time step $k-1$. It represents the uncertainty in the state estimate. $\mathbf{H}$ is the measurement matrix, which maps the state space into the measurement space. It defines how the state variables are related to the measurements obtained from the sensor. It is a constant matrix. $\mathbf{z}_k$ represents the actual measurement obtained from the sensor at time step $k$. It is a vector that contains the measurements. $\hat{\mathbf{x}}_{k|k-1}$ denotes the predicted state estimate at time step $k$ based on all past measurements up to time step $k-1$. It is the expected value of the state variables given the previous measurements and the motion model. $\mathbf{T}_k$ is a transformation matrix that is used to adjust the units or dimensions of the measurement noise covariance $\mathbf{R}_k$. It helps incorporate the measurement noise into the correct space. $\mathbf{R}_k$ represents the measurement noise covariance matrix at time step $k$. It describes the uncertainty or noise present in the measurements obtained from the sensor. $\mathbf{I}$ is the identity matrix, which has ones on the main diagonal and zeros elsewhere.

To this end, we could obtain camera poses for every virtually captured stereo pair obtained by the UAV. This also enables us to calculate the real-time speed, traveled distances, and turn angles of the UAV in real-world metrics without using any GPS information. Combined with the point clouds we obtained from Section 3.2, we could stitch the point cloud together into a combined point cloud of the captured scene along the flight trajectory. We use the EKF-filtered poses as initialization and use the CUDA accelerated Iterative Closest Point (ICP) [33] algorithm to fine-register all point clouds together. We run ICP on the point clouds calculated from low-resolution disparity maps using Equation (4). The final module, which includes orientation/pose estimation and fine-registration, is running at a constant speed of more than 100 Hz on a desktop equipped with a standalone GPU.

### 3.4. Meshing and Texture Mapping

To this end, we have shown an efficient approach to generate dense point cloud data. We also leverage Screened Poisson Surface Reconstruction [34] for mesh reconstruction, and Large-Scale Texturing [35] for texture mapping.

Screened Poisson Surface Reconstruction aims to reconstruct a smooth surface, denoted as $S$, from an unorganized point cloud, denoted as $P$, with $N$ points $\mathbf{p}_i$ in $\mathbb{R}^3$. The goal is to find a scalar field $u(\mathbf{x})$ over 3D space $\mathbb{R}^3$ such that it satisfies the Poisson equation with a guidance function $g(\mathbf{x})$:

$$\nabla^2 u(\mathbf{x}) = g(\mathbf{x}) \tag{12}$$

where $\nabla^2$ is the Laplacian operator. The guidance function $g(\mathbf{x})$ is derived from the point cloud $P$ and helps to direct the reconstruction process. To ensure a stable and smooth reconstruction, Screened Poisson Surface Reconstruction introduces a screening function $s(\mathbf{x})$ that acts as a spatially varying weight. The screened Poisson equation is given by:

$$\nabla^2 u(\mathbf{x}) = s(\mathbf{x}) \cdot g(\mathbf{x}) \tag{13}$$

The screening function $s(\mathbf{x})$ assigns higher weights to points near the surface and lower weights to points further away. This effectively suppresses the influence of noisy or distant points, resulting in a more robust and accurate reconstruction. The reconstructed surface $S$ can be obtained by applying the marching cubes algorithm to the level set of the scalar field $u(\mathbf{x})$, where the level set $u(\mathbf{x}) = 0$ defines the boundary of the surface. The marching cubes algorithm generates a triangular mesh that represents the reconstructed surface $S$. By incorporating the screening function into the Poisson equation, Screened Poisson Surface Reconstruction achieves a high-quality and smooth surface representation that respects the input point cloud while mitigating artifacts and irregularities.

Once we obtain the mesh, raw images, and projection matrices, we can proceed with texture mapping. We can first start with image Projection. For each vertex $\mathbf{v}_i$ in the 3D reconstruction, the corresponding 2D UV coordinates $\mathbf{u}_i$ are computed by projecting $\mathbf{v}_i$ onto the camera images. This projection is given by:

$$\mathbf{u}_i = \mathbf{K} \cdot \mathbf{P} \cdot \mathbf{v}_i \tag{14}$$

where $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{P}$ is the camera projection matrix, and $\mathbf{v}_i$ is the 3D vertex. Next, we create the UV map. Using the computed UV coordinates, a UV map is created for each camera image. The UV map is a 2D representation of the camera image, where each pixel corresponds to a 3D point on the reconstructed surface. To assign color information to the 3D reconstruction, we need to perform texture sampling. For each vertex $\mathbf{v}_i$, the corresponding color value $\mathbf{c}_i$ is obtained by sampling the UV map at the UV coordinates $\mathbf{u}_i$. This process is given by:

$$\mathbf{c}_i = \text{Sample}(\text{UV Map}, \mathbf{u}_i) \tag{15}$$

where Sample represents the sampling function. Finally, to handle occlusions and seams, we need to perform texture fusion. The color information from multiple camera images is fused together to obtain the final texture value $\mathbf{c}(\mathbf{v})$ at each vertex $\mathbf{v}$ in the 3D reconstruction. This is achieved using a weighted averaging scheme:

$$\mathbf{c}(\mathbf{v}) = \frac{\sum_{i=1}^{N} w_i \cdot \mathbf{c}_i}{\sum_{i=1}^{N} w_i} \tag{16}$$

where $N$ is the number of cameras, $\mathbf{c}_i$ is the color value at vertex $\mathbf{v}$ from camera $i$, and $w_i$ is the weight associated with camera $i$. The weights are computed based on the distance between the 3D vertex and the camera center.

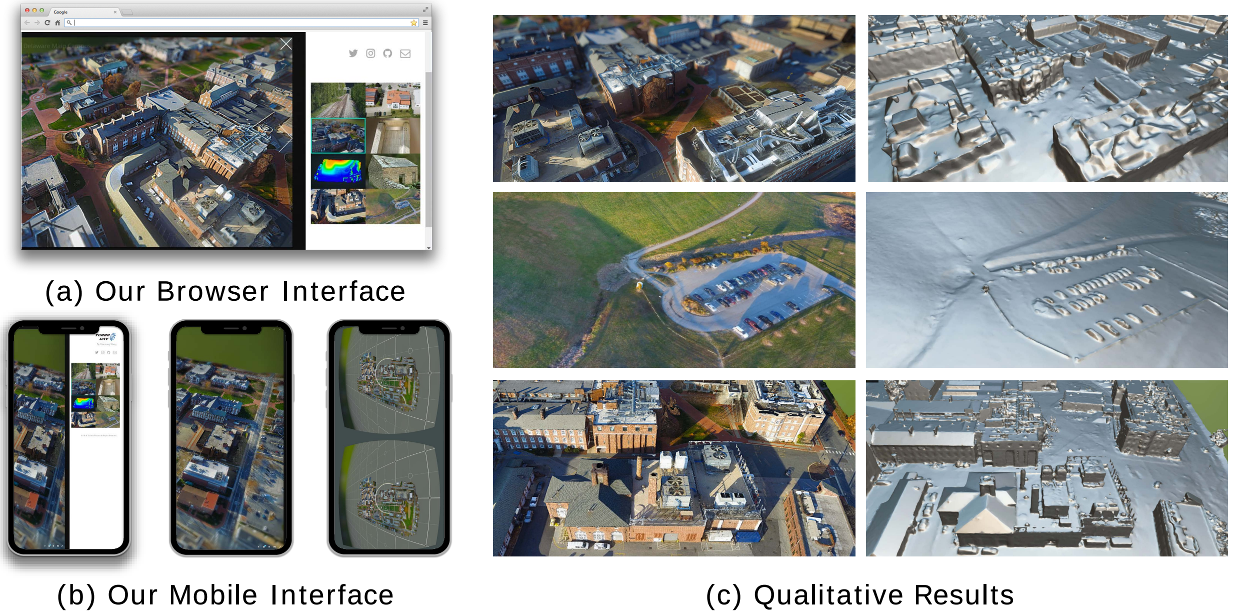
Note that these two steps are used for generating visually pleasing results and are not included in our real-time workflow. In the future, we will investigate different fast meshing and texture mapping approaches and work towards a more complete end-to-end 3D reconstruction pipeline.

## 4. Experiments

### 4.1. Hardware and Software Setup

A DJI-Phantom Vision 2 Plus quadcopter with 2.4G datalink and iPad ground station is used in this project. It weighs about 1.2 kg, with a diameter of 350 mm. The maximum forward, ascent, and descent speed are 15 m/s, 6 m/s, and 3 m/s, respectively. Automatic flights can be set up with a maximum altitude of 122 m (per US Federal Aviation Administration (FAA) General Operating and Flight Rules (CFR) part 107) and a maximum distance of 5 km. A GoPro Hero 3+ Black Edition camera is attached to a Zenmuse H3-3D gimbal to capture stabilized imagery off the UAV. The video signal is transmitted to the ground station via built-in Wi-Fi, and an AVL58 FPV system is applied to increase the video transmission range to about 1 km in open space. An iOSD Mark II system is connected to the onboard controller to transmit real-time flight data such as power voltage, velocity, height, distance from the home point, horizontal attitude, and GPS satellite number. For all our experiments, we collect the videos, extract all the frames, and test on a desktop with NVIDIA Geforce Titan V 12GB graphics card. Although we process the frames in an offline fashion, there is the potential to run the same approach online with real-time speed. To achieve this, either a reliable high-speed datalink or an embedded onboard computing platform (e.g., NVIDIA Jetson) is required. Building such a system is outside of the scope of this work; thus, we only evaluate the offline data.

We tested the CUDA accelerated BP algorithm with the maximum iteration set to 5 and the maximum disparity value set to 16. We set both the spatial and range filter size of the JBF to 15 and the diameter of the filtering neighborhood to 10 pixels. We divide the imaging plane to a 30 × 30 grid and calculate Harris corners inside each grid individually to prevent too many points detected at the same region given a hard threshold. We fly at 100 m with a speed of 3 m/s and maintain altitude using the DJI built-in NAZA flight controller. We choose two frames out of every 10 frames to form a virtual stereo camera. The baseline is approximately 0.5 m, and the overlap between adjacent images is around 90%. We conducted experiments in various environments and uploaded the results to Sketchfab [36] (see Figure 3). We set the field of view (FOV) to 60 degrees; physically based rendering (PBR) to “shadeless”; and enabled additional depth of field (DoF), bloom, and Reinhard tone mapping effects. We also constructed a dynamic web portal for the navigation, preview, and management of the uploaded 3D models.



(a) Our Browser Interface

(b) Our Mobile Interface

(c) Qualitative Results

**Figure 3.** Qualitative results on the reconstructed 3D scenes. Our web interface enabling browsing of different scenes is shown in (**a**). The VR view shown in (**b**) and the mesh rendered in (**c**) are provided by Sketchfab [36].

### 4.2. High-Resolution Reconstruction Experiments

In order to assess the upper bound of our reconstruction quality, we have also carried out experiments using a more recent consumer drone, namely, the DJI Mavic 3 equipped with the Hasselblad L2D-20c 4/3 CMOS 20-megapixel aerial camera. This particular camera features a 24 mm equivalent focal length lens and an adjustable aperture ranging from f/2.8 to f/11, providing a native dynamic range of 12.8 stops. For flight path planning and image capture, we utilized the publicly available Map Pilot Pro application. Our experimentation took place at the Montalvo Arts Center in Saratoga, California.

The results obtained from our proposed algorithm exhibit superiority in both texture and geometric accuracy when compared to the Google Maps 3D reconstruction. Despite the inherent advantage of Google Maps employing satellite imagery for large-scale 3D reconstruction, our utilization of UAV drones presents a noteworthy advantage due to the close proximity of data capture. The reduced distance from the target scene allows for enhanced resolution and finer detail acquisition. Moreover, this advantage is complemented by the cost-effectiveness of our approach. The deployment of UAV drones is considerably more economical than satellite-based methods, making our algorithm not only superior in terms of texture and geometric accuracy but also economically viable for widespread implementation. This cost-effectiveness renders our 3D reconstruction algorithm an attractive option for various applications, particularly those with limited financial resources, and further underscores its significance in the domain of geographical mapping and remote sensing technologies. However, it is important to highlight that our method may not be appropriate for applications that demand high geometric accuracy. This is because Poisson surface reconstruction, while enhancing visual quality by over-smoothing points, can lead to a reduction in accuracy.

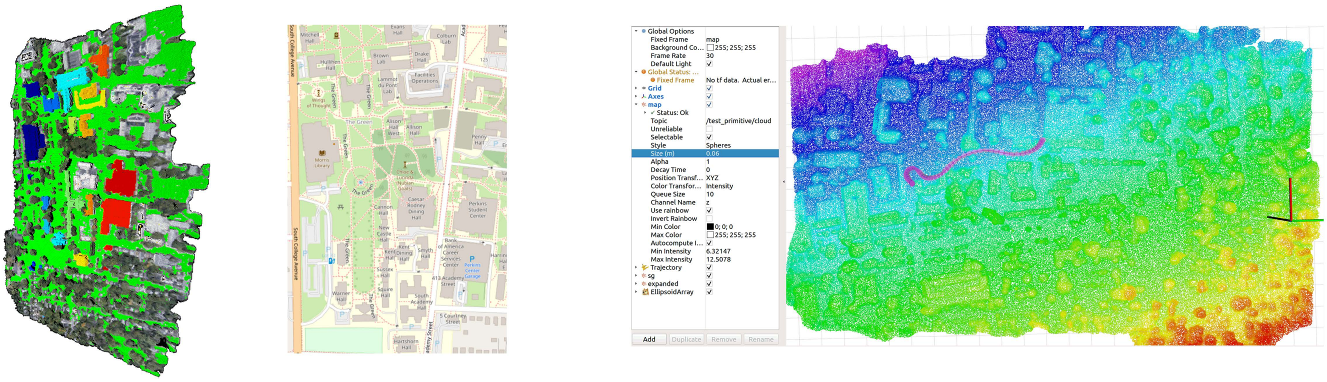
### 4.3. Auto Labeling and Measurements

We seek to automatically label the 3D point cloud, using open-source OpenStreetMap (OSM) [37] geographical mapping tool. For the OSM users, given a GPS boundary request, the OSM API server returns an XML file containing tags represented by various data structures (nodes, ways, and relations). Each tag describes a geographic attribute, including roads, buildings, water, and vegetation. The boundaries are represented by a polygon based on GPS coordinates. We consider generating an orthographic projection from the 3D point cloud and align with the OSM map file to achieve the automatic labeling of the 3D scene object segments. We first perform a RANSAC-based plane detection method to extract the ground plane and define the normal direction of the plane to be the Z axis. We translate the 3D point cloud to Z = 0 and place a virtual orthographic camera at the center +Z direction such that all points are covered in the camera view. Next, we render the scene to obtain an orthophoto as well as a depth map stored in the Z-buffer. Once the orthophoto is obtained, we send GPS coordinates recorded by the original image to request a RGB satellite image from the Google Maps API. SIFT descriptors are extracted from both images and matched against each other. Again, we apply RANSAC to solve for scale, rotation, and translation to align the orthophoto with the satellite image. This fully automated process gives us (pixel accurate) high-precision coordinates of our reconstructed mesh. Next, we filter all objects less than 5 m in height to obtain building segmentation masks, as shown in Figure 4. To validate the accuracy of our approach, we manually label 12 main buildings from the University of Delaware using Google Earth Pro, obtaining the groundtruth areas and perimeters of each building. The measurements obtained from our automatic pipeline closely match the groundtruth data, as presented in Table 1, establishing the precision of our 3D reconstruction pipeline. The final reconstruction results are presented in Figure 5. Note that precise 3D models of the University of Delaware's buildings on Google Earth are not available for quantitative comparison because of copyright constraits from Google. While we acknowledge the importance of quantitative verification using Chamfer distance or F1 scores for surface evaluation, there is difficulty in obtaining highly accurate reference data from Google.

In conclusion, our method demonstrates remarkable accuracy in automatically labeling 3D point clouds, incorporating orthophotos aligned with satellite imagery. The close agreement between our automated measurements and groundtruth data underscores the effectiveness of our pipeline, offering great potential for various applications in geographic mapping and 3D scene analysis.

### 4.4. Flight Path Planning

Automatic flight path planning plays a pivotal role in ensuring the efficient and safe operation of UAVs in various applications. We import the point clouds data obtained from Section 3.3 into the Robot Operating System (ROS) [38] and leverage the state-of-the-art algorithm [39] for trajectory planning. Given any target 3D point in space, the algorithm can generate a trajectory that allows the UAV to autonomously navigate through complex environments with enhanced precision (see Figure 4b). This paper underscores the necessity of automated flight path planning by elucidating how our proposed methodology harnesses the power of real-time point cloud data and advanced trajectory planning techniques to enable UAVs to effectively navigate through intricate spaces, thereby bolstering the reliability and versatility of UAV operations across a spectrum of real-world scenarios. Furthermore, the integration of automatic flight path planning with point cloud data can facilitate task-level UAV command and streamline fleet operations, paving the way for coordinated and collaborative missions with multiple UAVs operating seamlessly in real-world environments. Notably, this approach enables a unique strategy where one drone can be designated to fly at a higher altitude for 3D mapping purposes, subsequently providing other drones with essential 3D map information. These secondary drones can then leverage this map data for low-altitude optimal trajectory planning and obstacle avoidance, enhancing overall operational efficiency and safety during complex missions.



Global Options
Fixed Frame: map
Background Co...: 255; 255; 255
Frame Rate: 30
Default Light: ✓
Global Status: ...
Fixed Frame: No tf data. Actual er...
Grid: ✓
Axes: ✓
map: ✓
Status: Ok
Topic: /test_primitive/cloud
Unreliable: □
Selectable: ✓
Style: Spheres
Size (m): 0.06
Alpha: 1
Decay Time: 0
Position Transf...: XYZ
Color Transfor...: Intensity
Queue Size: 10
Channel Name: z
Use rainbow: ✓
Invert Rainbow: □
Min Color: ■ 0; 0; 0
Max Color: □ 255; 255; 255
Autocompute I...: ✓
Min Intensity: 6.32147
Max Intensity: 12.5078
Trajectory: ✓
sg: ✓
expanded: ✓
EllipsoidArray: ✓

(a) Automatic Building Segmentation

(b) Path Planning on 3D Pointcloud

**Figure 4.** Demonstration of UAV applications. (**a**) Left: Automatic building segmentation results. The ground plane is marked in green and the segmented 3D buildings are coded in random color (see Section 4.3). Right: OpenStreetMap view of the same area including building labels. (**b**) Automatic flight path planning on the point cloud on the same area of (**a**). The purple 3D path shows the planned flight trajectory between the current location and the target. (see Section 4.4).

**Table 1.** Automatic measurement results on buildings from the University of Delaware. GT_Area and GT_Perim denote the ground truth area and perimeter. The camera settings column shows the shutter speed (in seconds) and ISO value. The detailed measurement process is outlined in Section 4.3.

| Name | Area ($m^2$) | GT_Area ($m^2$) | Perim (m) | GT_Perim (m) | Points (M) | Shutter (s), ISO |
|---|---|---|---|---|---|---|
| Morris Library | 7047.85 | 7000.00 | 389.69 | 390.00 | 1.12 | 1/300, 100 |
| Memorial Hall | 2458.78 | 2440.00 | 281.84 | 280.00 | 0.63 | 1/300, 100 |
| Hullihen Hall | 2368.23 | 2320.00 | 195.80 | 196.00 | 0.63 | 1/300, 100 |
| Mitchell Hall | 1091.59 | 1080.00 | 169.54 | 168.00 | 0.55 | 1/300, 100 |
| Gore Hall | 2781.18 | 2760.00 | 224.43 | 224.00 | 0.65 | 1/300, 100 |
| Sharp Laboratory | 2652.83 | 2620.00 | 294.71 | 292.00 | 0.62 | 1/300, 100 |
| Wolf Hall | 2599.18 | 2560.00 | 300.32 | 296.00 | 0.68 | 1/150, 200 |
| Du Pont Hall | 3886.30 | 3840.00 | 341.62 | 338.00 | 0.73 | 1/150, 200 |
| Evans Hall | 2638.77 | 2600.00 | 206.14 | 204.00 | 0.69 | 1/150, 200 |
| Brown Lab | 4690.47 | 4640.00 | 298.94 | 296.00 | 0.82 | 1/150, 200 |
| Lammot du Pont Lab | 1533.21 | 1520.00 | 186.15 | 184.00 | 0.64 | 1/60, 400 |
| Robinson Hall | 768.33 | 764.00 | 119.84 | 118.00 | 0.45 | 1/60, 400 |

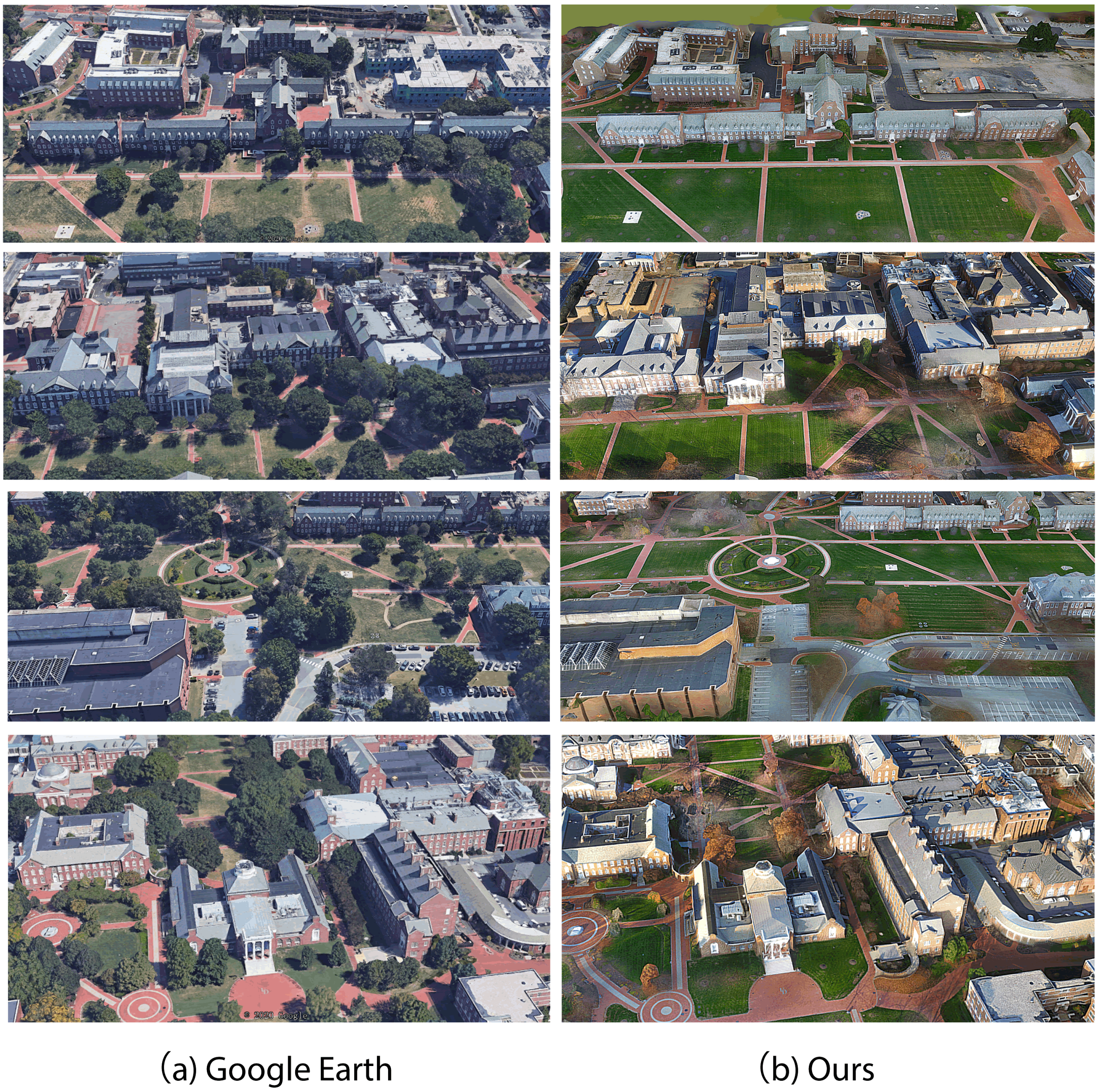


**Figure 5.** Qualitative results between our 3D model and Google Earth. Note that the viewpoints are manually adjusted to roughly align with each other. Our method is highly efficient while performing favorably against Google Earth in terms of final reconstruction quality.

## 5. Discussion

We show quantitative results in Table 1 and compare them with ground-truth measurement data. As can be seen from Table 1, our method achieves highly precise results across different building structures. The qualitative results from Figures 5 and 6 show superiority in both texture and geometric quality when compared to the Google Maps 3D reconstruction. We also demonstrate useful applications such as the fully automated labeling of buildings, building measurements, and flight path planning.

A crucial challenge in the current photogrammetry pipelines is the significant processing time required to fully analyze the captured imagery data, thereby constraining the real-time data acquisition capability. Such rapid data acquisition is indispensable in scenarios like onsite scanning, emergency response, and planning. An optimal scanning pipeline should ideally enable real-time scene mapping as the drone follows its flight path, allowing inspectors to assess partial scans even before the flight mission concludes. However, the high time complexity of traditional photogrammetry approaches makes it inapplicable to achieve these objectives.

Our proposed approach emphasizes the utilization of high-efficiency algorithms with minimal computational burden at each pipeline stage. Notably, we employ the Harris corner detector for feature extraction, as opposed to more resource-intensive options like SIFT or SURF descriptors. For stereo matching, we adopt real-time Belief Propagation (BP) on downsampled images, diverging from the resource-demanding full-resolution multi-view stereo matching. Furthermore, we opt for the expedited EPnP algorithm for camera pose estimation/refinement instead of the conventional Bundle Adjustment (BA) technique found in many SfM pipelines. Finally, we derive a constant speed model using the Kalman filter to improve the pose estimation accuracy. The culmination of these choices results in our system's remarkable capability to generate accurate dense point clouds in near real-time, achieving a consistent speed of 25 frames per second.

## 6. Conclusions

We have presented an efficient, near-real-time solution for 3D reconstruction using off-the-shelf consumer UAVs. Our proposed alternative approach addresses the limitations of current photogrammetry pipelines for real-time mapping using consumer-level camera drones. By employing high-efficiency algorithms with minimal computational cost at each step of the pipeline, we have achieved significant advancements in real-time 3D reconstructions. Our method conducts feature extraction, verification, camera pose estimation/refinement, and dense stereo simultaneously on a frame-by-frame basis, allowing for incremental and online mapping. This is in contrast to conventional approaches that rely on global optimization and require the entire dataset as input, hindering real-time capabilities.

Our approach holds great promise for various applications, including onsite scans and inspections, emergency response, and planning. The ability to examine partial scans even before completing the flight mission is a significant advantage, enhancing decision-making and operational efficiency. Moreover, the real-time data acquisition achieved by our method opens up new possibilities for micro UAVs in diverse industries, such as agriculture, construction, transportation, and film-making, where rapid and cost-effective mapping solutions are increasingly in demand.

While our method has demonstrated remarkable performance, further research and development are necessary to continue refining and expanding its capabilities. Future work could focus on optimizing algorithms even further, exploring different feature extraction and matching techniques and investigating ways to handle challenging imaging conditions more effectively. Also, we plan to compare the proposed approach with other methods on 3D reconstruction benchmark datasets and add more global/loop-closure constraints to minimize trajectory errors. Additionally, we would like to validate the method's accuracy and robustness in various real-world scenarios to speed up its engineering adoption and deployment.

In conclusion, our work represents a significant step towards unlocking the full potential of consumer-level camera drones for real-time high-resolution mapping, offering promising prospects for enhanced efficiency and productivity across numerous industries.

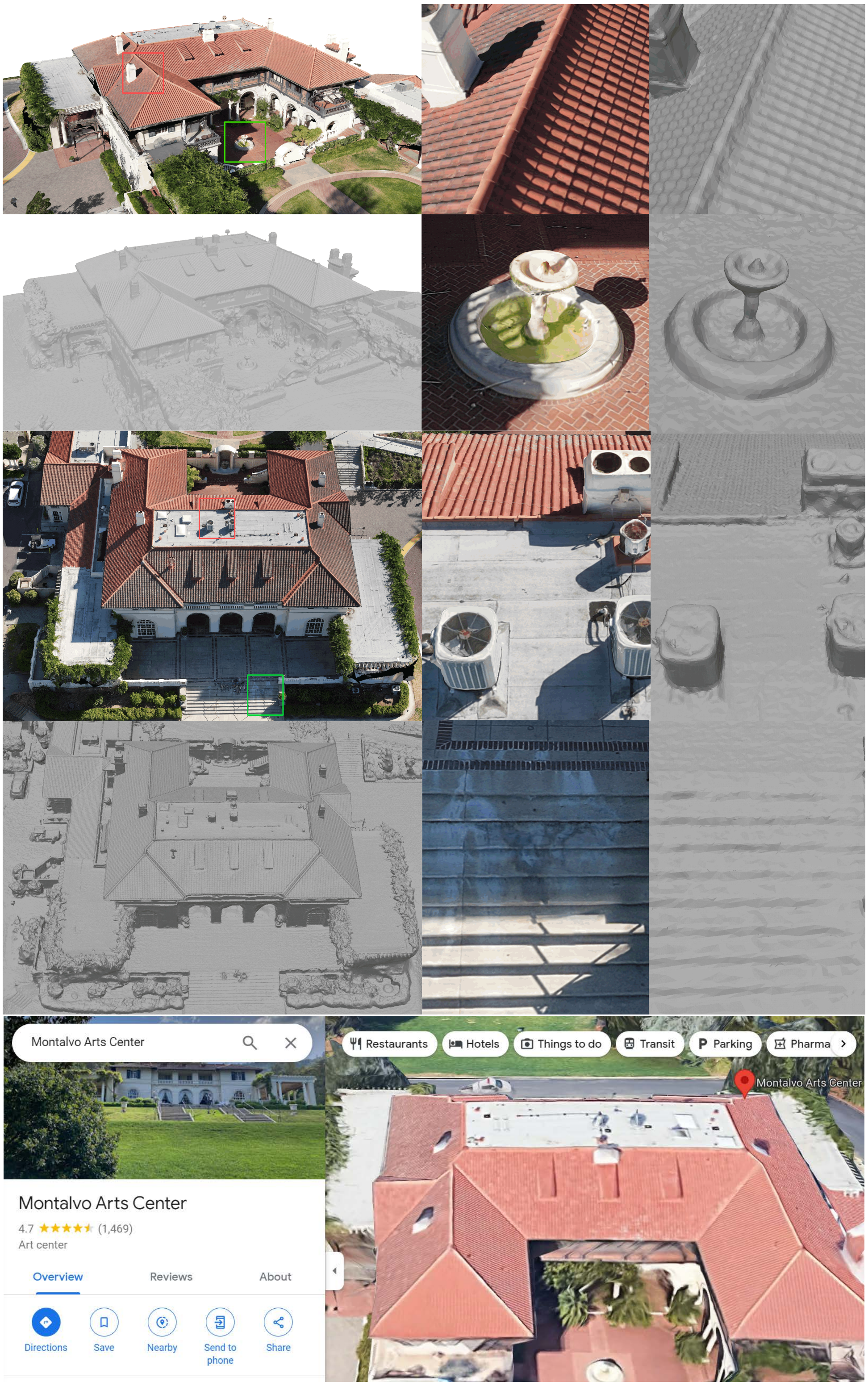


**Figure 6.** High-Resolution Reconstruction Experiments. We reconstructed the Montalvo Arts Center (Saratoga, CA, USA) using a more recent consumer drone equipped with a 20-megapixel aerial camera (Section 4.2). Rows 1 to 4 show the reconstructed mesh and 3D model with zoom-in views. Note that the zoom level is more than 10×. Row 5 shows the same 3D building visualization in Google Maps. The upper and lower enlarged areas are indicated by red and green boxes, respectively. Our results exhibit much-improved quality in both texture and geometry details.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data presented in this study are available at the author's personal website at https://qiaosongwang.com. Under the Publications section please check instructions on how to download data used in this paper.

**Acknowledgments:** The author conducted this research during his PhD study at the University of Delaware. Special thanks are extended to Bobby Healy of Manna Drone for email discussions that sparked the inspiration for this project. Christopher Rasmussen, from the Department of Computer and Information Sciences at the University of Delaware, deserves acknowledgment for offering valuable suggestions and advice throughout the project. Thanks to Kenneth Barner from the Department of Electrical and Computer Engineering at the University of Delaware for reviewing the project proposals. Gratitude is also expressed to Hank Chen and Matija Radovic from the University of Delaware for their efforts in collecting aerial data. Further appreciation goes to Julie Yan and Jennifer Zhao for their contributions in gathering data from the Montalvo Arts Center.

**Conflicts of Interest:** The author declares no conflicts of interest.

## References

1. Harris, C.G.; Stephens, M. A combined corner and edge detector. In Proceedings of the Alvey Vision Conference, Manchester, UK, 31 August–2 September 1988; Volume 15, pp. 10–5244.
2. Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157.
3. Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In Proceedings of the Computer Vision—ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, 7–13 May 2006; Proceedings, Part I 9; Springer: Berlin/Heidelberg, Germany, 2006; pp. 404–417.
4. Lepetit, V.; Moreno-Noguer, F.; Fua, P. Epnp: An accurate o (n) solution to the pnp problem. *Int. J. Comput. Vis.* **2009**, *81*, 155. [CrossRef]
5. Triggs, B.; McLauchlan, P.F.; Hartley, R.I.; Fitzgibbon, A.W. Bundle adjustment—A modern synthesis. In Proceedings of the Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, 21–22 September 1999; Springer: Berlin/Heidelberg, Germany, 2000; pp. 298–372.
6. Agarwal, S.; Furukawa, Y.; Snavely, N.; Simon, I.; Curless, B.; Seitz, S.M.; Szeliski, R. Building rome in a day. *Commun. ACM* **2011**, *54*, 105–112. [CrossRef]
7. Civera, J.; Grasa, O.G.; Davison, A.J.; Montiel, J.M. 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *J. Field Robot.* **2010**, *27*, 609–631. [CrossRef]
8. Yoon, K.J.; Kweon, I.S. Locally adaptive support-weight approach for visual correspondence search. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 2, pp. 924–931.
9. Hirschmuller, H. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *30*, 328–341. [CrossRef]
10. Geiger, A.; Roser, M.; Urtasun, R. Efficient large-scale stereo matching. In *Computer Vision—ACCV 2010*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 25–38.
11. Hosni, A.; Rhemann, C.; Bleyer, M.; Rother, C.; Gelautz, M. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *35*, 504–511. [CrossRef] [PubMed]
12. Boykov, Y.; Veksler, O.; Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 1222–1239. [CrossRef]
13. Sun, J.; Zheng, N.N.; Shum, H.Y. Stereo matching using belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2003**, *25*, 787–800.
14. Yang, Q.; Wang, L.; Yang, R.; Stewénius, H.; Nistér, D. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *31*, 492–504. [CrossRef] [PubMed]
15. Kolmogorov, V.; Zabin, R. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 147–159. [CrossRef]
16. Felzenszwalb, P.F.; Huttenlocher, D.P. Efficient belief propagation for early vision. *Int. J. Comput. Vis.* **2006**, *70*, 41–54. [CrossRef]
17. Yang, Q.; Wang, L.; Ahuja, N. A constant-space belief propagation algorithm for stereo matching. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 1458–1465.
18. Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234.
19. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]

20. Newcombe, R.A.; Lovegrove, S.J.; Davison, A.J. DTAM: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2320–2327.
21. Engel, J.; Koltun, V.; Cremers, D. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 611–625. [CrossRef]
22. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 834–849.
23. Gherardi, R.; Farenzena, M.; Fusiello, A. Improving the efficiency of hierarchical structure-and-motion. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 1594–1600.
24. Schonberger, J.L.; Frahm, J.M. Structure-from-motion revisited. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4104–4113.
25. Wilson, K.; Snavely, N. Robust global translations with 1dsfm. In Proceedings of the Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part III 13; Springer: Berlin/Heidelberg, Germany, 2014; pp. 61–75.
26. Rosten, E.; Porter, R.; Drummond, T. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *32*, 105–119. [CrossRef]
27. Bhardwaj, A.; Sam, L.; Bhardwaj, A.; Martín-Torres, F.J. LiDAR remote sensing of the cryosphere: Present applications and future prospects. *Remote Sens. Environ.* **2016**, *177*, 125–143. [CrossRef]
28. Bolourian, N.; Hammad, A. LiDAR-equipped UAV path planning considering potential locations of defects for bridge inspection. *Autom. Constr.* **2020**, *117*, 103250. [CrossRef]
29. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
30. Bradski, G. The openCV library. *Dr. Dobbs J. Softw. Tools Prof. Program.* **2000**, *25*, 120–123.
31. Wang, Q.; Yu, Z.; Rasmussen, C.; Yu, J. Stereo vision–based depth of field rendering on a mobile device. *J. Electron. Imaging* **2014**, *23*, 023009. [CrossRef]
32. Kopf, J.; Cohen, M.F.; Lischinski, D.; Uyttendaele, M. Joint bilateral upsampling. *ACM Trans. Graph. ToG* **2007**, *26*, 96. [CrossRef]
33. ICPCUDA Open Source Utility Library. Available online: https://github.com/mp3guy/ICPCUDA (accessed on 5 May 2023).
34. Kazhdan, M.; Hoppe, H. Screened poisson surface reconstruction. *ACM Trans. Graph. ToG* **2013**, *32*, 1–13. [CrossRef]
35. Waechter, M.; Moehrle, N.; Goesele, M. Let there be color! Large-scale texturing of 3D reconstructions. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 836–850.
36. Sketchfab. Available online: https://sketchfab.com (accessed on 5 May 2023).
37. Haklay, M.; Weber, P. Openstreetmap: User-generated street maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18. [CrossRef]
38. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
39. Zhou, B.; Gao, F.; Pan, J.; Shen, S. Robust real-time uav replanning using guided gradient-based optimization and topological paths. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 1208–1214.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.