Mesh Reconstruction from Aerial Images for Outdoor Terrain Mapping Using Joint 2D-3D Learning

Qiaojun Feng Nikolay Atanasov

***Abstract*— This paper addresses outdoor terrain mapping using overhead images obtained from an unmanned aerial vehicle. Dense depth estimation from aerial images during flight is challenging. While feature-based localization and mapping techniques can deliver real-time odometry and sparse points reconstruction, a dense environment model is generally recovered offline with significant computation and storage. This paper develops a joint 2D-3D learning approach to reconstruct local meshes at each camera keyframe, which can be assembled into a global environment model. Each local mesh is initialized from sparse depth measurements. We associate image features with the mesh vertices through camera projection and apply graph convolution to refine the mesh vertices based on joint 2-D reprojected depth and 3-D mesh supervision. Quantitative and qualitative evaluations using real aerial images show the potential of our method to support environmental monitoring and surveillance applications.**

I. Introduction

Recent advances in sensors, processors, storage and communication devices have set the stage for mobile robot systems to significantly contribute in environmental monitoring, security and surveillance, agriculture, and many other applications. Constructing terrain maps onboard an unmanned aerial vehicle (UAV) using online sensory data would be very beneficial in these applications. With inertial measurement unit (IMU), GPS, and camera sensors, a UAV can localize itself and incrementally reconstruct the geometric structure of the traversed terrain. Near infrared cameras can additionally provide normalized difference vegetation index measurements for vegetation assessment and semantic segmentation can enrich the map.

This paper considers the problem of building a terrain model in the form of a mesh of an outdoor environment using a sequence of overhead RGB images obtained onboard a UAV. We assume that the UAV is running a localization algorithm, based on visual-inertial odometry (VIO) [1] or simultaneous localization and mapping (SLAM) [2] , which estimates its camera pose and the depths of a sparse set of tracked image keypoints. One approach for outdoor terrain mapping is to recover depth images at each camera view using dense stereo matching. The depth images can be fused to generate a point cloud and triangulate a mesh surface. While specialized sensors and algorithms exist for real-time dense stereo matching, they are restricted to a limited depth range, much smaller than commonly seen in aerial images. Moreover, due to the limited depth variation, the recovered point cloud might not be sufficiently dense for accurate mesh reconstruction. Recently, depth completion methods [3], [4] using deep learning have shown promising performance on indoor [5] and outdoor datasets [6]. However, aerial images are different from the ground-level RGBD datasets used to train these algorithm. Due to the limited availability of aerial image datasets for supervision, learning-based methods have not yet been widely adopted for outdoor terrain mapping.

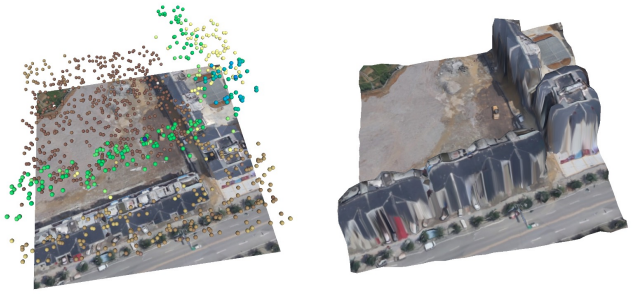

Fig. 1: This paper develops a method for 3-D mesh reconstruction (right) from aerial RGB images and noisy sparse depth measurements (left) to support outdoor terrain mapping.

We propose a learning-based method for mesh reconstruction using a single RGB image with sparse depth measurements. Fig. 1 shows an example input and mesh reconstruction. Our main **contribution** is to show that depth completion and mesh reconstruction are closely related problems. Inspired by depth completion techniques, we propose a coarse-to-fine strategy, composed of initialization and refinement stages for mesh reconstruction. In the initialization stage, we use only the sparse depth measurements to fit a coarse mesh surface by minimizing a 2-D rendered depth loss. In the refinement stage, we leverage both the RGB image and the sparse depth information. We extract deep convolutional image features and associate them with the vertices of the initial mesh through camera projection. The mesh is subsequently refined using a graph convolution model to predict vertex deformations that minimize a weighted 3-D geometric surface loss and 2-D rendered depth loss. Given the camera poses, we can fuse the optimized local meshes into a global mesh model of the environment. We build an aerial image dataset with ground-truth depth, noisy sparse depth measurements, and multiple camera trajectories based on WHU MVS/Stereo dataset [7] and details can be found on https://github.com/FengQiaojun/TerrainMesh.

II. Related Work

Depth Completion. Predicting depth from RGB images enables artificial perception systems to recover 3-D environ-

We gratefully acknowledge support from NSF NRI CNS-1830399.

The authors are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA {qjfeng,natanasov}@ucsd.edu

ment structure [8], [9]. While depth prediction from RGB images alone may be challenging, sparse depth measurements, e.g., obtained from keypoint triangulation, simplify the problem and lead to improvements in efficiency and accuracy [10]. Depth completion is the task of predicting dense depth images from sparse depth measurements and corresponding RGB images. Ma et al. [3], [11] develop methods for supervised training, relying on ground truth depth images, as well as self-supervised training, using photometric error from calibrated image pairs. Chen et al. [10] pre-process sparse depth images by generating a Euclidean Distance Transform of the depth sample locations and a nearest-neighbor depth fill map. The authors propose a multi-scale deep network that treats depth completion as residual prediction with respect to the nearest-neighbor depth fill maps. Chen et al. [4] design a 2-D convolution branch to process stacked RGB and sparse depth images and a 3-D convolution branch to process point clouds and fuse the outputs of the two branches.

Mesh Reconstruction. Online terrain mapping requires efficient storage and updates of a 3-D surface model. Storing dense depth information from aerial images requires significant memory and subsequent model reconstruction. Explicit surface representations, e.g., based on polygon meshes, may be quite memory and computationally efficient but their vertices and faces need to be optimized to fit the environment geometry. FLaME [12] performs non-local variational optimization over a time-varying Delaunay graph to obtain a real-time inverse-depth mesh of the environment. Terrian Fusion [13] performs real-time terrain mapping by generating digital surface model (DSM) meshes at selected keyframes. It converts the local meshes into grid-maps and merges them using multi-band fusion. Pixel2Mesh [14] treats a mesh as a graph and applies graph convolution [15] for vertex feature extraction and graph unpooling to subdivide the mesh for detailed refinement. Mesh R-CNN [16] simultaneously detects objects and reconstructs their 3-D mesh shape. A coarse voxel representation is predicted first and then converted into a mesh for refinement.

Our setting differs from existing work because aerial images cover large regions with significantly more subtle depth variation compared to indoor or outdoor ground settings. Our approach uses the same inputs as a depth completion problem but recovers a 3-D mesh model, which provides smoother depth estimates with fewer parameters (only vertices and faces) compared to dense depth prediction. Instead of relying on a priori known object categories, our method provides whole image mesh reconstruction.

III. Problem Formulation

Consider a UAV equipped with a camera, whose position $\mathbf{p}_k \in \mathbb{R}^3$ and orientation $\mathbf{R}_k \in SO(3)$ are estimated at discrete time steps k by a VIO or SLAM localization algorithm. Denote the RGB images corresponding to the discrete-time keyframes by $\mathbf{I}_k$. Obtaining dense depth images during outdoor flight is challenging but it is common for localization algorithms to track and estimate the depth of a sparse set of image feature points. Let $\mathbf{D}_k^*$ denote the *dense* ground-truth depth image, which is unknown during real-time operation. Let $\mathbf{D}_k$ be a *sparse* matrix that contains estimated depths at the image feature locations and zeros everywhere else. Our goal is to construct an explicit model of the camera view at time k using a 3-D triangle mesh $\mathcal{M}_k := (\mathbf{V}_k, \mathcal{E}_k, \mathcal{F}_k)$, where $\mathbf{V}_k \in \mathbb{R}^{n_k \times 3}$ are the vertex coordinates in the camera frame, $[n_k] := \{1, \dots, n_k\}$ is the set of vertex indices, $\mathcal{E}_k \subseteq [n_k] \times [n_k]$ are the edges, and $\mathcal{F}_k \subseteq [n_k] \times [n_k] \times [n_k]$ are the faces.

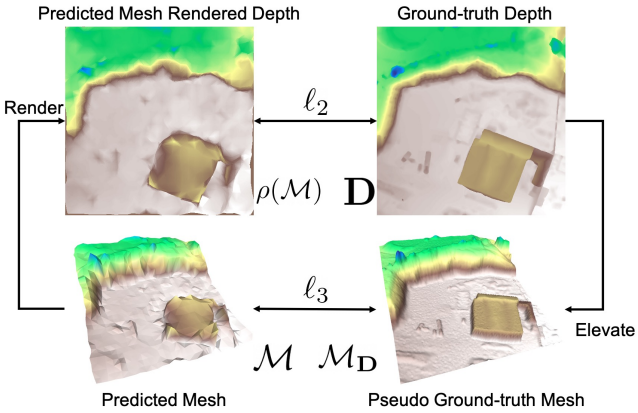

Fig. 2: Loss function illustration: ℓ_2 compares a depth image $\mathbf{D}$ to rendered mesh depth $\rho(\mathcal{M})$; ℓ_3 compares a mesh $\mathcal{M}$ to an elevated mesh $\mathcal{M}_\mathbf{D}$ obtained from a depth image $\mathcal{D}$.

Problem. Given a finite set of RGB images $\{\mathbf{I}_k\}_k$ and corresponding sparse depth measurements $\{\mathbf{D}_k\}_k$, define a mesh reconstruction function $\mathcal{M} = f(\mathbf{I}, \mathbf{D}; \boldsymbol{\theta})$ and optimize its parameters $\boldsymbol{\theta}$ to estimate the ground-truth depth $\{\mathbf{D}_k^*\}_k$:

$$\min_{\boldsymbol{\theta}} \sum_k \ell(f(\mathbf{I}_k, \mathbf{D}_k; \boldsymbol{\theta}), \mathbf{D}_k^*) \tag{1}$$

where $\ell(\mathcal{M}, \mathbf{D})$ is a loss function measuring the error between a mesh $\mathcal{M}$ and a depth image $\mathbf{D}$ representing the same camera view.

The choice of loss function ℓ is discussed in Sec. IV. We develop a machine learning approach to this problem, consisting of an offline training phase and an online mesh reconstruction phase. During training, the parameters $\boldsymbol{\theta}$ are optimized using a training set $\mathcal{D} := \{\mathbf{I}_i, \mathbf{D}_i, \mathbf{D}_i^*\}_i$ with known ground-truth depth images. During testing, given streaming RGB images $\mathbf{I}_k$ and sparse depth measurements $\mathbf{D}_k$, the optimized parameters $\boldsymbol{\theta}^*$ are used to construct mesh models $\mathcal{M}_k = f(\mathbf{I}_k, \mathbf{D}_k; \boldsymbol{\theta}^*)$. The local mesh $\mathcal{M}_k = (\mathbf{V}_k, \mathcal{E}_k, \mathcal{F}_k)$ can be converted into the global frame by transforming the vertex coordinates $\mathbf{V}_k \mathbf{R}_k^\top + \mathbf{1}\mathbf{p}_k^\top$ using the camera poses $\mathbf{p}_k$, $\mathbf{R}_k$ and multiple meshes can be assembled [17] to model the whole environment.

IV. Loss Functions for Mesh Reconstruction

We propose several loss functions to measure the error between a mesh $\mathcal{M}$ and a depth image $\mathbf{D}$ representing the same camera view. Since our problem focuses on optimizing the mesh representation $\mathcal{M}$, the loss function must be differentiable with respect to the vertices of $\mathcal{M}$.

A loss function can be defined in the 2-D image plane by rendering a depth image from $\mathcal{M}$ and comparing its pixel values to those of $\mathbf{D}$. We rely on a differentiable mesh renderer [18], [19] to generate a depth image $\rho(\mathcal{M})$ and define a 2-D loss function:

$$\ell_2(\mathcal{M}, \mathbf{D}) := \frac{\sum_{ij \in \mathcal{U}(\rho(\mathcal{M}), \mathbf{D})} |\rho_{ij}(\mathcal{M}) - \mathbf{D}_{ij}|}{|\mathcal{U}(\rho(\mathcal{M}), \mathbf{D})|}, \quad (2)$$

where $\mathcal{U}(\rho(\mathcal{M}), \mathbf{D})$ is the set of pixels where both the depth image $\mathbf{D}$ and the rendered depth $\rho(\mathcal{M})$ have valid depth information. While ℓ_2 is a natural choice of a loss function in the image plane, it does not emphasize two important properties for mesh reconstruction. First, since ℓ_2 only considers a region in the image plane where both depth images have valid information, its minimization over $\mathcal{M}$ may encourage the mesh $\mathcal{M}$ to shrink. Second, ℓ_2 does not emphasize regions of large depth gradient variation (e.g., the side surface of a building), which may lead to inaccurate reconstruction of 3-D structures.

To address these limitations, we propose a supplementary loss function, defined in the 3-D spatial domain over point clouds $\mathcal{P}_\mathcal{M}$ and $\mathcal{Q}_\mathbf{D}$ obtained from $\mathcal{M}$ and $\mathbf{D}$, respectively:

$$\ell_3(\mathcal{M}, \mathbf{D}) := \frac{1}{2}\lambda(\mathcal{P}_\mathcal{M}, \mathcal{Q}_\mathbf{D}) + \frac{1}{2}\lambda(\mathcal{Q}_\mathbf{D}, \mathcal{P}_\mathcal{M}), \quad (3)$$

where λ is the asymmetric Chamfer point cloud distance:

$$\lambda(\mathcal{P}, \mathcal{Q}) := \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \|\mathbf{p} - \arg\min_{\mathbf{q} \in \mathcal{Q}} \|\mathbf{p} - \mathbf{q}\|_2\|_2. \quad (4)$$

To generate $\mathcal{P}_\mathcal{M}$, we sample the faces of $\mathcal{M}$ uniformly. Since samples on the mesh surface can be represented using barycentric coordinates with respect to the mesh vertices, the loss function is differentiable with respect to the mesh vertices. To generate $\mathcal{Q}_\mathbf{D}$, we may sample the depth image $\mathbf{D}$ uniformly and project the samples to 3-D space but this will not generate sufficient samples in the regions of large depth gradient variation. Instead, we first generate a pseudo ground-truth mesh $\mathcal{M}_\mathbf{D}$ by densely sampling pixel locations in $\mathbf{D}$ as the mesh vertices and triangulating on the image plane to generate faces. We then sample the surface of $\mathcal{M}_\mathbf{D}$ uniformly to obtain $\mathcal{Q}_\mathbf{D}$.

Fig. 2 illustrates the loss functions ℓ_2 and ℓ_3. We also define two regularization terms to measure the smoothness of $\mathcal{M} = (\mathbf{V}, \mathcal{E}, \mathcal{F})$. The first is based on the Laplacian matrix $\mathbf{L} := \mathbf{G} - \mathbf{A} \in \mathbb{R}^{n \times n}$ of $\mathcal{M}$, where $\mathbf{G}$ is the vertex degree matrix and $\mathbf{A}$ is the adjacency matrix. We define a vertex regularization term based on the $\ell_{2,1}$-norm [20] of the degree-normalized Laplacian [21] $\mathbf{G}^{-1}\mathbf{L}$:

$$\ell_\mathbf{V}(\mathcal{M}) := \frac{1}{n} \left\|\mathbf{G}^{-1}\mathbf{L}\mathbf{V}\right\|_{2,1}. \quad (5)$$

We also introduce a mesh edge regularization term to discourage long edges in the mesh

$$\ell_\mathcal{E}(\mathcal{M}) := \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \|\mathbf{v}_i - \mathbf{v}_j\|_2, \quad (6)$$

where $\mathbf{v}_i \in \mathbb{R}^3$ are the mesh vertices. The complete loss function is:

$$\begin{aligned} \ell(\mathcal{M}, \mathbf{D}) := w_2\ell_2(\mathcal{M}, \mathbf{D}) + w_3\ell_3(\mathcal{M}, \mathbf{D}) \\ + w_\mathbf{V}\ell_\mathbf{V}(\mathcal{M}) + w_\mathcal{E}\ell_\mathcal{E}(\mathcal{M}), \end{aligned} \quad (7)$$

where the first two terms evaluate the error between $\mathcal{M}$ and $\mathbf{D}$ and the last two terms encourage smoothness of the mesh structure. The scalars $w_2, w_3, w_\mathbf{V}, w_\mathcal{E}$ allow appropriate weighting of the different terms in (7).

V. 2D-3D Learning for Mesh Reconstruction

Inspired by depth completion techniques, we approach mesh reconstruction in two stages: *initialization* and *refinement*. We initialize a coarse mesh from the sparse depth measurements and refine it by predicting vertex residuals based on RGB image features.

A. Mesh Initialization

Outdoor terrain structure can be viewed as a 2.5-D surface that is mostly flat with occasional height variations. Hence, we initialize a flat mesh and change the surface elevation based on the sparse depth measurements. The flat mesh is initialized with regular-grid vertices (1024 in our experiments) over the X-Y ground plane, orthogonal to the gravity direction (Z axis). See Fig. 3 for an illustration. Subsequently, our mesh reconstruction approach only optimizes the mesh vertices and keeps the edge and face topology fixed. We initialize the Z-axis vertex coordinates by solving an optimization problem with a weighted combination of the 2-D rendered depth loss ℓ_2 in (2) and the Laplacian loss $\ell_\mathbf{V}$ in (5) as the objective function:

$$\begin{aligned} \Delta\mathbf{V}^* = \arg\min_{\Delta\mathbf{V}} \; & w_2\ell_2(\mathcal{M}(\mathbf{V} + \Delta\mathbf{V}, \mathcal{E}, \mathcal{F}), \mathbf{D}) \\ & + w_\mathbf{V}\ell_\mathbf{V}(\mathcal{M}(\mathbf{V} + \Delta\mathbf{V}, \mathcal{E}, \mathcal{F})). \end{aligned} \quad (8)$$

In (8), $\mathbf{D}$ are sparse depth measurements so the rendered depth error is evaluated only at the sparse pixel locations. The initialized mesh $\mathcal{M}^{\text{int}} = (\mathbf{V} + \Delta\mathbf{V}^*, \mathcal{E}, \mathcal{F})$ is used as an input to the mesh refinement stage.

B. Mesh Refinement

In the refinement stage, we use a learning approach to extract features from both the 2-D image and 3-D initial mesh and regress mesh vertex offsets. The ground-truth depth maps are used for supervision.

The RGB image provides useful information for refinement since man-made objects have sharp vertical surfaces, while natural terrain has noisy but limited depth variation. The sparse depth measurements also provide information about areas with large intensity variation. Inspired by Mesh R-CNN [16], we design a network that extracts features from the 2-D image, associates them with the 3-D vertices of the initial mesh, and uses them to refine the vertex locations. Our network has 3 stages: feature extraction, vertex-image feature alignment, and vertex graph convolution.

Feature Extraction. We extract features from three sources: the RGB image $\mathbf{I}$, the rendered depth $\rho(\mathcal{M}^{\text{int}})$ from the initial mesh, and a Euclidean distance transform $\mathbf{E}(\mathbf{D})$ of the

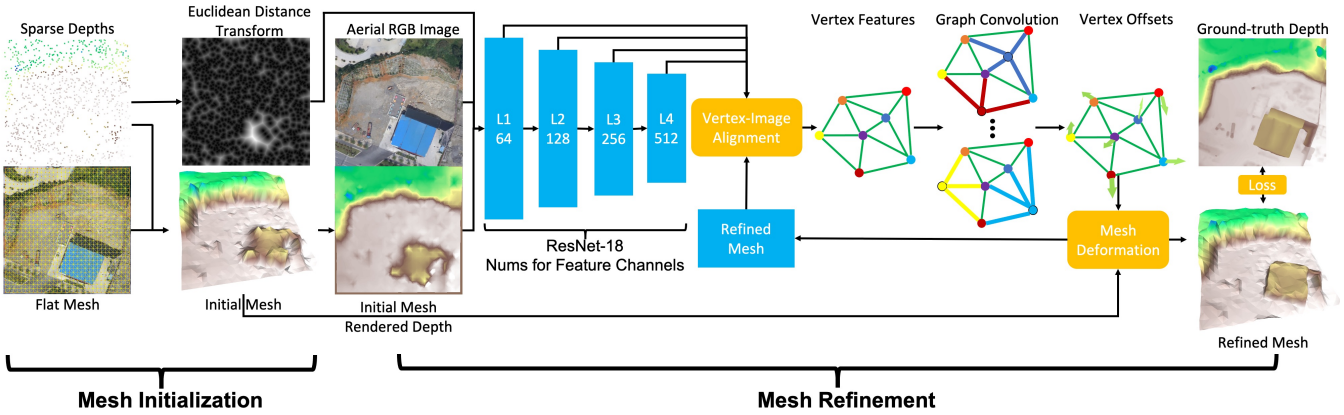

Fig. 3: Overview of the complete mesh reconstruction architecture. In the mesh initialization stage (Sec. V-A), we use sparse depths to elevate a flat mesh from the image plane to 3-D space. In the mesh refinement stage (Sec. V-B), we combine the RGB image, the initial mesh rendered depth and the Euclidean Distance Transform from the sparse depths and extract 2-D features using ResNet-18 [22]. These features are aligned to the initial mesh vertices using camera projection (Fig. 4). Vertex offsets are regressed using a graph convolution network (GCN) over the initial mesh. The ResNet-18 and GCN parameters are optimized jointly using the loss function in Sec. IV.

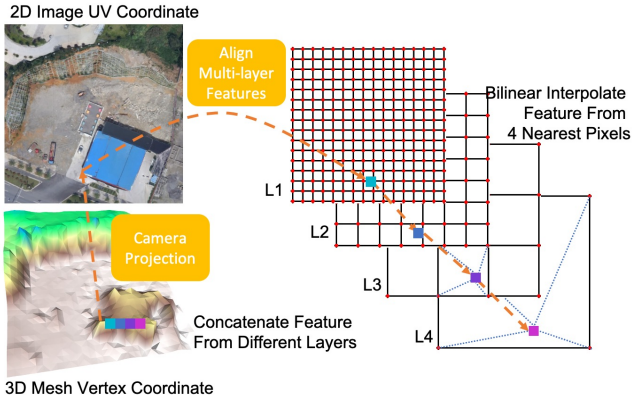

Fig. 4: Illustration of image feature to mesh vertex association. With known camera intrinsics, each mesh vertex can be projected in uv coordinates (range $[0, 1]$) onto the image plane. Bilinear interpolation is used to associate image feature maps at different resolutions with the mesh vertices. The features across different resolutions are concatenated to form a composite vertex feature.

sparse depth measurements $\mathbf{D}$, obtained by computing the Euclidean distance to the closest valid depth measurement for each pixel. The three images are concatenated to form a 5-channel input: $\mathbf{C} = \text{concat}(\mathbf{I}, \rho(\mathcal{M}^{\text{int}}), \mathbf{E}(\mathbf{D}))$. We use ResNet-18 [22] to extract image features. Since aerial images have different properties compared to ImageNet data, we learn the model weights from scratch. Four layers of features with different resolution and channels are extracted:

$$[\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \mathbf{L}_4] = \phi(\mathbf{C}), \tag{9}$$

where ϕ is the ResNet-18 model.

Vertex-Image Feature Alignment. Next, we construct 3-D features for $\mathcal{M}^{\text{int}}$ by associating the mesh vertices with the 2-D image features. This idea is inspired by Pixel2Mesh [14], which projects mesh vertices onto the image plane and extracts features at the projected coordinates. To obtain multi-scale features, we associate the projected mesh vertices with the intermediate layer feature maps $[\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \mathbf{L}_4]$ from (9). This vertex-image alignment step is illustrated in Fig. 4. All features with different channel numbers are concatenated to form composite vertex features:

$$\mathbf{V}^{g_{\text{in}}} = g^{\text{align}}(\mathcal{M}^{\text{int}}, \phi(\mathbf{C})), \tag{10}$$

where $\mathbf{V}^{g_{\text{in}}} \in \mathbb{R}^{n \times (l_1 + l_2 + l_3 + l_4 + 3)}$ are the vertex features and l_i is the number of channels in feature map $\mathbf{L}_i$. We add the 3-D vertex coordinates $\mathbf{v}_i$ as the last 3 dimensions.

Vertex Graph Convolution. The mesh can be viewed as a graph with vertex features $\mathbf{V}^{g_{\text{in}}}$. A graph convolution network [15], [16] is a suitable architecture to process the vertex features and obtain vertex offsets $\Delta\mathbf{V}$ that optimize the agreement of the refined mesh $\mathcal{M}^{\text{ref}} = (\mathbf{V}^{\text{int}} + \Delta\mathbf{V}, \mathcal{E}, \mathcal{F})$ and the ground truth depth $\mathbf{D}^*$ according to the loss in (7). To capture a larger region of feature influence, we use 3 layers of graph convolution g_1, g_2, g_3 and set the final vertex feature dimension to 64. A weight matrix $\mathbf{W}$ is applied on the final 64-D feature to derive the 3-D vertex offsets:

$$\Delta\mathbf{V} = \mathbf{W}\mathbf{V}^{g_{\text{out}}} := \mathbf{W} g_3(g_2(g_1(\mathbf{V}^{g_{\text{in}}}))), \tag{11}$$

where $\mathbf{V}^{g_{\text{out}}} \in \mathbb{R}^{n \times 64}$ and $\Delta\mathbf{V} \in \mathbb{R}^{n \times 3}$. In order to refine the mesh with more details, we concatenate 3 stages of vertex-image feature alignment and graph convolution. At stage i, last stage's refined mesh $\mathcal{M}^{\text{ref}}_{i-1}$ is set as the initial mesh $\mathcal{M}^{\text{int}}_i$ and new vertex features are extracted via vertex-image feature alignment and fed to new graph convolution layers. All 3 refined meshes in different stages $(\mathcal{M}^{\text{ref}}_1, \mathcal{M}^{\text{ref}}_2, \mathcal{M}^{\text{ref}}_3)$ are evaluated against the ground-truth depth map $\mathbf{D}^*$ using the loss functions defined in (7).

VI. Experiments

This section compares several variations of our mesh reconstruction approach to a baseline method on a dataset generated from aerial images.

A. Dataset

We build an aerial image dataset based on the WHU MVS/Stereo dataset [7]. The original dataset provides calibrated RGBD images rendered from a highly accurate 3D digital surface model which is not publicly available. Hence,

TABLE I: Quantitative evaluation of variations of the proposed method. The *SD-tri* method triangulates a mesh using all the sparse depth measurements as vertices. The Regular-n uses the mesh with n vertices. The *Initialized* model constructs a mesh from the sparse depth (Sec. V-A). The RGB, RGB+RD, RGB+RD+EDT methods refine the initialized mesh (Sec. V-B), using different inputs respectively. The loss function used by the different methods is indicated by 3D+2D (uses both ℓ_3 and ℓ_2) or 3D (uses ℓ_3 only). The second column shows the number of available sparse depth measurements per image and indicates whether the measurements are noisy (Sec. VI-A).

Error	Meshing Inputs Loss	SD-tri (vert = SD)	Regular-576			Regular-1024				
			Initialized	RGB+RD 3D+2D	RGB+RD+EDT 3D+2D	Initialized	RGB 3D+2D	RGB+RD 3D	RGB+RD 3D+2D	RGB+RD+EDT 3D+2D
ℓ_2	500	1.492	2.069	1.670	**1.637**	1.861	1.575	1.382	1.289	**1.252**
	1000	1.172	1.834	1.596	**1.546**	1.535	1.298	1.193	1.124	**1.097**
	2000	0.916	1.941	1.551	**1.511**	1.344	1.144	1.092	1.045	**1.024**
ℓ_3	500	9.815	18.278	**13.438**	13.763	13.799	7.242	**5.412**	5.647	6.352
	1000	6.494	17.762	**12.938**	13.574	11.872	5.876	**4.538**	4.911	5.703
	2000	4.649	17.130	**12.483**	13.506	10.859	5.131	**4.069**	4.477	5.291
ℓ_2	500+noise	1.865	2.294	1.809	**1.768**	2.155	1.828	1.571	1.486	**1.456**
	1000+noise	1.632	2.056	1.701	**1.685**	1.826	1.535	1.360	1.319	**1.308**
	2000+noise	1.485	1.717	1.655	**1.654**	1.629	1.364	1.243	**1.236**	1.241
ℓ_3	500+noise	19.737	18.392	**12.974**	13.532	14.887	8.351	**6.063**	6.157	6.865
	1000+noise	22.189	17.693	**12.258**	13.161	12.480	6.447	**4.904**	5.266	6.075
	2000+noise	18.545	17.256	**11.856**	12.988	11.147	5.452	**4.343**	4.793	5.620

we recover a dense point cloud from the RGBD images as a ground-truth 3D model. We generate 20 camera trajectory sequences, split into 14 for training, 2 for validation, and 4 for testing. Each camera trajectory follows a sweeping grid-pattern with 10 keyframes per row and 20 keyframes per column. The keyframes are chosen to ensure 75% row overlap and 80% column overlap. RGBD images with resolution 512×512 are rendered along each trajectory from the ground-truth point cloud using PyTorch3D [19]. We obtain camera pose estimates and sparse depth measurements $\mathbf{D}_k$ for each image by applying OpenSfM [23] to four neighbor images with known camera intrinsic parameters. Small (500), medium (1000), and large (2000) number of sparse depth measurements are obtained from SfM per image. The results from OpenSfM are treated as the data with noise, and the noise may come from the feature detection&matching as well as the bundle adjustment step. We also obtain noiseless depth measurements with the same sparsity 2-D pattern from the ground-truth depth images $\mathbf{D}_k^*$.

B. Implementation Details

During training, we use 1000 number of sparse depth measurements and the mesh vertices number is fixed to 1024. The mesh initialization optimization is performed over 100 iterations with the Adam optimizer [24] and weights $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}] = [1, 0, 0.5, 0]$ for the loss function in (7). The ResNet-18 and GCN parameters are optimized jointly during the mesh refinement training using the Adam optimizer with initial learning rate of 0.0005 for 200 epochs. The loss function in (7) with parameters $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}] = [3, 1, 0.5, 0.01]$ is used in this phase. The Chamfer distance λ in the ℓ_3 loss term is computed using 10000 samples.

C. Results

Our experiments report the ℓ_2 error in (2) and the ℓ_3 error in (3), comparing the reconstructed mesh wit. The ℓ_2 emphasizes projected depth accuracy, while the ℓ_3 pay more attention to large depth gradient region.

For comparison, we define a baseline method that triangulates the sparse depth measurements directly to build a mesh. The baseline method performs Delaunay triangulation on the 2-D image plane over the depth measurements and projects the flat mesh to 3-D using the known vertex depths. We refer to the baseline method as sparse-depth-triangulation (*SD-tri*). Note that the baseline method uses all avaiable sparse depth measurements (500, 1000, or 2000) and, hence, may has a different number of vertices from the other models. The quantitative results from the comparison are reported in Table I. Note that all the models are trained with 1024-vertex meshes and 1000 sparse depth measurements and we directly generalize them on meshes with different number of vertices and different number of sparse depth measurements.

Several variations of our approach are evaluated. We compared three different options for the input provided to the mesh refinement stage: only the 3-channel RGB image (RGB), the RGB image plus rendered depth from the initial mesh (RGB+RD, 4-channels), and the RGB image plus rendered depth from the initial mesh plus Euclidean distance transform obtained from of the sparse depth map (RGB+RD+EDT, 5-channels). The model using RGB-only does not perform as well as the other two. The RGB+RD+EDT model has the best performance according to the ℓ_2 error metric. The RGB+RD method has similar performance in the ℓ_2 metric and smaller ℓ_3 error compared to RGB+RD+EDT. The RGB+RD model is used to generate our qualitative results using 1024-vertex meshes because it offers good performance according to both error metrics.

We also compare different loss function combinations for training the RGB+RD method. The 3D+2D loss function reported in Table I corresponds to training with parameters $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}] = [3, 1, 0.5, 0.01]$ for the loss function in (7), while the 3D loss function, corresponds to parameters $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}] = [0, 1, 0.5, 0.01]$. We can see that training with the 3D+2D loss leads to balanced performance acorrding to both the ℓ_2 and ℓ_3 metrics, while training with the 3D loss only leads to good performance in the ℓ_3 loss but higher 2-D rendering error, according to the ℓ_2 metric.

Finally, we compare the mesh reconstruction accuracy when the sparse depth measurements are noiseless ver-

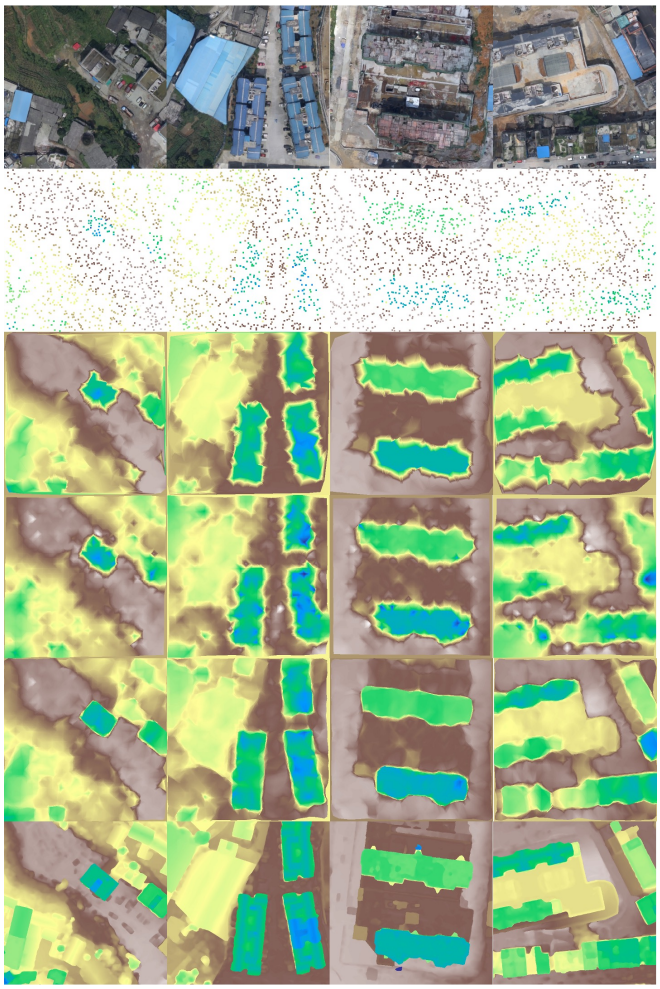

Fig. 5: Mesh reconstructions visualized as rendered depth images. The colors indicate the relative depth values. Row 1: RGB images. Row 2: sparse depth measurements (around 1000). Row 3: meshes reconstructed from sparse-depth triangulation. Row 4: meshes after initialization (Sec. V-A). Row 5: meshes after neural network refinement (Sec. V-B). Row 6: ground-truth depth images.

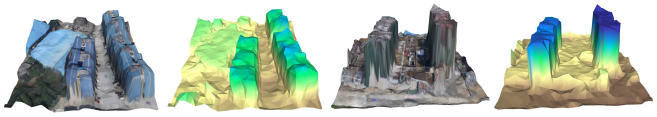

Fig. 6: Reconstructed meshes painted with RGB texture and color indicating depth. The sharp vertical transitions of the buildings are reconstructed accurately.

sus noisy. The baseline SD-tri method performs well in a noiseless setting but degenerates drastically when noise from the SfM is introduced. In contrast, our model is more robust to the noise in the sparse depth measurements. Two factors might be contributing to this. First, our mesh initialization and refinement stages both include explicit mesh regularization terms (in (5) and (6)). Second, the image features extracted during the mesh refinement process help to distinguish among different terrains and structures. The latter is clear from the improved accuracy of the refined, compared to the initialized, meshes. We also report the performance using a mesh with only 576 vertices. When the

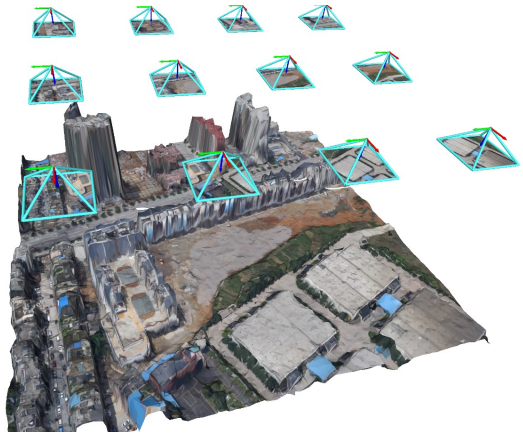

Fig. 7: Complete environment model obtained by transforming to the global frame and merging local meshes from 12 camera views.

depth measurements are noisy, it has lower ℓ_2 loss compared with the baseline method with similar number of vertices. It also has lower ℓ_3 loss even compared with meshes with more vertices generated from the baseline method.

Some qualitative results are presented in Fig. 5. Compared with the sparse-depth-triangulation and the initialized meshes, the refined meshes have smoother boundaries on the side surfaces of the buildings. The guidance from the image features allows the refined meshes to fit the 3D structure better. Fig. 7 shows a global mesh reconstruction obtained by transforming and merging 12 local camera-view reconstructions. The local meshes are transformed to global frame using the keyframe poses and no postprocessing is used to merge them into a global model.

The reconstructed mesh models are a more efficient representation than the dense depth images. A dense depth image requires 512×512 values to represent a camera view, while our mesh model (with fixed mesh faces topology) only needs to store the 3D coordinates of the 1024 vertices. Thus, our model requires only 1% of the depth image parameters to obtain a high-fidelity reconstruction of a camera view. On a desktop with NVIDIA 1080 Ti GPU, the Mesh Initialization step takes about 3 s/frame because we solve it using iterative gradient descent. However, eq. (8) can be solved much faster by treating it as a linear system. The Mesh Refinement step takes about 10 ms/frame.

VII. Conclusion

In this work, we introduce a method to reconstruct a 3D mesh from an RGB image and sparse depth measurements. We build an outdoor aerial dataset and apply our method on it for terrain mapping. Quantitative and qualitative results show that our method outperform the baseline method of triangulating the sparse depth points. Our method can also generalize to different number of sparse depth without additional storage cost. It is also robust to reconstruction noise in the sparse depth measurements. In the future, we would like to tightly merge this with a feature-based SLAM algorithm to upgrade the sparse feature-based map into a dense mesh map. We would also fuse multi-modal observations into the map such as the semantic information.

REFERENCES

[1] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[3] F. Ma and S. Karaman, "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4796–4803.

[4] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning Joint 2D-3D Representations for Depth Completion," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 10 022–10 031.

[5] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[7] J. Liu and S. Ji, "A Novel Recurrent Encoder-Decoder Structure for Large-Scale Multi-View Stereo Reconstruction From an Open Aerial Dataset," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6049–6058.

[8] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging Into Self-Supervised Monocular Depth Estimation," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3827–3837.

[9] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth From Videos in the Wild: Unsupervised Monocular Depth Learning From Unknown Cameras," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8976–8985.

[10] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, "Estimating Depth from RGB and Sparse Sensing," in *Computer Vision – ECCV*, 2018, pp. 176–192.

[11] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3288–3295.

[12] W. N. Greene and N. Roy, "FLaME: Fast Lightweight Mesh Estimation Using Variational Smoothing on Delaunay Graphs," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4696–4704.

[13] W. Wang, Y. Zhao, P. Han, P. Zhao, and S. Bu, "TerrainFusion: Real-time Digital Surface Model Reconstruction based on Monocular SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7895–7902.

[14] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," in *Computer Vision – ECCV*, 2018, pp. 55–71.

[15] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[16] G. Gkioxari, J. Johnson, and J. Malik, "Mesh R-CNN," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9784–9794.

[17] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh editing with poisson-based gradient field manipulation," *ACM Trans. Graph.*, vol. 23, no. 3, p. 644–651, Aug. 2004.

[18] S. Liu, T. Li, W. Chen, and H. Li, "A General Differentiable Mesh Renderer for Image-based 3D Reasoning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[19] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3D Deep Learning with PyTorch3D," *arXiv:2007.08501*, 2020.

[20] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 1813–1821.

[21] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian Surface Editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2004, p. 175–184.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[23] "OpenSfM," https://github.com/mapillary/OpenSfM.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2014.