TerrainMesh: Metric-Semantic Terrain Reconstruction From Aerial Images Using Joint 2-D-3-D Learning

Qiaojun Feng, *Student Member, IEEE*, and Nikolay Atanasov, *Senior Member, IEEE*

***Abstract*—This article considers outdoor terrain mapping using RGB images obtained from an aerial vehicle. While feature-based localization and mapping techniques deliver real-time vehicle odometry and sparse keypoint depth reconstruction, a dense model of the environment geometry and semantics (vegetation, buildings, etc.) is usually recovered offline with significant computation and storage. This article develops a joint 2-D-3-D learning approach to reconstruct a local metric-semantic mesh at each camera keyframe maintained by a visual odometry algorithm. Given the estimated camera trajectory, the local meshes can be assembled into a global environment model to capture the terrain topology and semantics during online operation. A local mesh is reconstructed using an initialization and refinement stage. In the initialization stage, we estimate the mesh vertex elevation by solving a least squares problem relating the vertex barycentric coordinates to the sparse keypoint depth measurements. In the refinement stage, we associate 2-D image and semantic features with the 3-D mesh vertices using camera projection and apply graph convolution to refine the mesh vertex spatial coordinates and semantic features based on joint 2-D and 3-D supervision. Quantitative and qualitative evaluation using real aerial images show the potential of our method to support environmental monitoring and surveillance applications.**

***Index Terms*—Aerial systems: Perception and autonomy, graph convolution for mesh reconstruction, mapping, semantic scene understanding.**

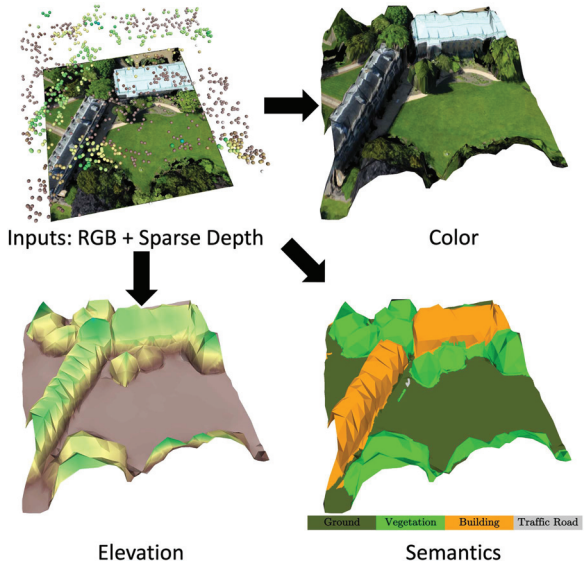


Fig. 1. This article develops a method using aerial RGB images and sparse depth measurements (top-left) to reconstruct a semantic mesh of an outdoor terrain. The color, elevation, and semantics of the mesh are visualized in the top-right, bottom-left, and bottom-right plots.

I. Introduction

RECENT advances in sensing, computation, storage, and communication hardware have set the stage for mobile robot systems to impact environmental monitoring, security and surveillance, agriculture, and many other applications. Constructing terrain maps onboard an unmanned aerial vehicle (UAV) using online sensor measurements provides critical situational awareness in such applications. This article considers the problem of building a metric-semantic terrain model, represented as a triangular mesh, of an outdoor environment using a sequence of overhead RGB images obtained onboard a UAV. Fig. 1 shows an example input and mesh reconstruction. 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. However, range sensors and, hence, dense depth information are not available during outdoor flight. One approach for terrain mapping is to recover depth images at each camera view using dense stereo matching, fuse them 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 the distances commonly present in aerial images. Moreover, due to 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 ground RGBD images used to train these models. Due to the limited availability of aerial image datasets for supervision, learning-based methods

Manuscript received 23 July 2023; revised 7 November 2023; accepted 23 December 2023. Date of publication 11 January 2024; date of current version 29 January 2024. This article was recommended for publication by Associate Editor T. A. Vidal-Calleja and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported in part by the NSF NRI under Grant CNS-1830399 and in part by ARL DCIST under Grant CRA W911NF-17-2-0181. *(Corresponding author: Qiaojun Feng.)*

The authors are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: qjfeng@ucsd.edu; natanasov@ucsd.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3353073, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3353073

have not yet been widely adopted for outdoor terrain mapping. Recently, there has also been increasing interest in supplementing geometric reconstruction with semantic information because many robotics tasks require semantic understanding. However, few algorithms exist for joint metric-semantic reconstruction. Most works treat semantic classification as a postprocessing step, decoupling it from 3-D geometric reconstruction.

This article is an extension of our 2021 IEEE ICRA conference paper [7] on mesh reconstruction from aerial RGB images with sparse depth measurements. We propose a joint 2-D-3-D learning method for metric-semantic mesh reconstruction using a novel coarse-to-fine strategy, composed of mesh initialization and mesh refinement stages. In the initialization stage, we use only the sparse depth measurements to fit a coarse mesh surface. In the refinement stage, we extract deep convolutional 2-D image features and associate them with the initial mesh 3-D vertices through perspective projection. The mesh is subsequently refined using a graph convolution model to predict both spatial coordinates and semantic features residuals of the vertices. We conduct extensive evaluation on simulated and real aerial datasets. The proposed mesh reconstruction method can be combined with any feature-based SLAM algorithm [8] to fuse local keyframe meshes into a consistent global terrain model.

The main contribution of the journal compared with the conference is the introduction and association of semantic segmentation with the mesh vertices, interpolation and projection techniques to obtain dense semantic features over the whole mesh, and graph convolution network (GCN) optimization with a joint geometric-semantic loss function to optimize the mesh vertices and semantic features. We demonstrate empirically that the joint geometric-semantic training can outperform the earlier geometric-only method proposed in the conference paper. We also derive a closed-form mesh initialization method, which is more accurate and efficient than the one in the conference paper, as well as an explicit mesh merging method to combine multiview mesh reconstruction into a single consistent global mesh of the environment. In summary, the contributions of this article are summarized as follows.

1) We introduce a joint 2-D-3-D loss function, utilizing differentiable mesh rendering, for metric-semantic mesh reconstruction.
2) We develop a two-stage coarse-to-fine mesh reconstruction approach, using a closed-form mesh vertex *initialization* from sparse depth measurements and a GCN mesh vertex *refinement* from RGB, sparse depth measurements, and semantic image features.
3) We evaluate our metric-semantic mesh reconstruction algorithm on synthetic and photo-realistic aerial image datasets.

II. Related Work

A. Depth Completion

Predicting depth from monocular RGB images allows single-camera perception systems to recover 3-D environment structure [9], [10]. However, dense depth estimation from monocular images may be challenging, especially for aerial images, where the depth variation is small compared with the absolute depth values. In contrast, the depth of sparse visual keypoints may be obtained efficiently and accurately using triangulation [11] between tracked feature points from Kanade–Lucas–Tomasi tracking [12] or visual feature matching [13].

Depth completion is the task of reconstructing a dense depth image from an RGB image with given sparse depth estimates. The authors in [3] and [14] developed a deep network for depth completion that passes the sparse depth and RGB image inputs through convolution layers, ResNet encoder layers, transposed convolution decoder layers, and a 1×1 convolution filter. The model is trained either with supervision from ground-truth depth images or via photometric error self-supervision from calibrated RGB image pairs. Instead of consuming sparse depth images directly, Chen et al. [15] preprocessed sparse depth images by generating a Euclidean distance transform (EDT) of the keypoint locations and a nearest-neighbor depth fill map. The authors proposed a multiscale deep network that treats depth completion as residual prediction with respect to the nearest-neighbor depth fill maps. We borrow the idea of densify the 2-D sparse depth inputs for our model design. 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 so that the 2-D and 3-D features are combined. Our approach also combines both the 2-D and 3-D feature extraction phases but in the form of mesh. CodeVIO [16] uses a conditional variational autoencoder to encode RGB and sparse depth inputs into a latent depth code and decode a dense depth image from the latent depth code. The sparse depth measurements are used to perform incremental depth code updates, allowing the depth reconstruction to be coupled with visual odometry estimation in the multi-state constraint Kalman filter (MSCKF) filter [17]. Our method is loosely integrated with a feature-based VIO/SLAM algorithm.

We approach depth completion using mesh reconstruction from RGB images and sparse depth measurements. While both dense depth completion and mesh reconstruction are challenging, a mesh model is more memory efficient than a depth image. For example, a 512×512 dense depth image requires 260 000 parameters. In contrast, our approach can represent the same region using only 1024 vertices.

B. Mesh Reconstruction

Online terrain mapping requires efficient storage and updates of a 3-D surface model. However, storing dense depth information from aerial images needs significant memory and subsequent model reconstruction for robot objectives, such as motion planning or environment exploration. An explicit surface representation using a polygonal mesh can be quite memory and computationally efficient. Compared with sparse point cloud models, mesh surfaces are continuous, allowing direct integration in motion planning algorithms as well as intuitive visualization for human operators. FLaME [18] performs variational optimization over a time-varying Delaunay graph to obtain an inverse-depth mesh of the environment, using sparse depth measurements from a VIO algorithm. Rosinol and Carlone [19]

extended FLaME to optimize the mesh over dense depth image measurements in real-time using a parallel implementation. Rosinol et al. [20] detected vertical and horizontal planes to regularize mesh vertices, and optimize the mesh vertices and the camera poses using a factor graph. Voxblox [21] incrementally builds a voxel-based truncated signed distance field and can reconstruct a mesh as a postprocessing step using the marching cubes algorithm [22]. Terrain Fusion [23] performs real-time terrain mapping by generating digital surface model (DSM) meshes at selected keyframes. The local meshes are converted into grid-maps and merged using multiband fusion.

Recently, learning methods have emerged as a promising approach for mesh reconstruction from limited or no 3-D information. Bloesch et al. [24] proposed a learning method to regress the image coordinates and depth of mesh vertices in a decoupled manner. This allows an in-plane 2-D mesh to capture the image structure. Pixel2Mesh [25] treats a mesh as a graph and applies graph convolution [26] for vertex feature extraction and graph unpooling to subdivide the mesh for refinement. Using differentiable mesh rendering [27], [28], the 3-D mesh structure of an object can be learned from 2-D images [29], [30], [31]. Mesh R-CNN [32] 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. Recent works [33], [34] can generate mesh reconstructions of complete scenes, including object and human meshes and their poses, from a single RGB image.

In contrast with many mesh reconstruction approaches, our method uses both visual and semantic features to refine the mesh geometry and generates mesh models with per-vertex semantic category distributions.

C. Semantic 3-D Reconstruction

3-D reconstruction from an image sequence is a fundamental problem in robotics and computer vision. Multi-view stereo (MVS) [35] aims to estimate the depth at one frame using several different frames. Classical MVS methods perform patch matching with photometric and geometric consistency [36]. These methods generalize well although the performance can be affected by low texture, lighting variation, and occlusion. Recently, learning-based methods that fuse multiview learned features across frames for depth recovery have achieved excellent performance. NeuralRecon [37] reconstructs and fuses sparse truncated signed distance function (TSDF) volumes for each frame incrementally using 3-D sparse convolutions and gated recurrent units. VoRTX [38] learns to fuse multiview frames using a transformer model and projective occupancy. SimpleRecon [39] leverages relative poses among frames to build a cost volume and uses a multilayer perceptron to reduce the volume and avoid costly 3-D convolution. Learning-based MVS [40], [41], [42], [43] can tackle challenges, such as severe occlusion but generally labeled data is needed for training. Recently, SLAM systems that integrate a learning-based MVS model to obtain dense 3-D reconstruction in real-time have been proposed [44], [45]. Besides explicit 3-D representations, implicit representations that model 3-D structures as level sets of distance or radiance functions have shown impressive performance recently. Neural radiance fields (NeRF) [46] represent the color and density field of the scene through the weights of the neural network. For any given camera pose, an RGB and depth image can be generated through volume rendering over the NeRF, providing photo-realistic novel view synthesis. Another advantage of NeRF models is that they can be trained from posed RGB images without depth supervision. However, for any new 3-D scene a NeRF needs to be trained from scratch, which may take a long time. Numerous recent works have extended the NeRF model [47], [48], [49], [50] to enable MVS pretraining, capture high-frequency details, and apply to unbounded outdoor environments. Instant neural graphics primitives (Instant-NGP) [51] and Nerfstudio [52] incorporate many of the recent NeRF model advances and offer open-source implementations.

Semantic 3-D reconstruction aims to estimate both the geometric structure and semantic content of an environment from visual observations. Extending 2-D semantic segmentation and depth prediction to a 3-D multiview consistent semantic model provides information that is critical for environmental monitoring tasks, such as observing vegetation recovery after a wildfire or controlling fuel build-up for fire prevention [53], or for terrain traversablility estimation in ground–aerial robot teaming [54]. Semantic information also allows human operators to specify tasks for mobile robots in terms of objects and concepts in the environment model. Semantic segmentation on the 2-D images can be back-projected onto 3-D space and multiview information can be fused to annotate the 3-D structure [55], [56], [57]. Besides, semantic segmentation can be directly performed on the 3-D point cloud [58], [59] or the mesh [60]. Instead of treating the semantic annotation as the postprocessing step after geometric reconstruction, researchers also investigate on how to jointly optimize geometric and semantic accuracy. Häne et al. [61] formulated a joint segmentation and dense reconstruction problem on voxels and showed that appearance likelihoods and class-specific geometric priors help each other. Cherabier et al. [62] leveraged variational energy minimization method for regularization to capture complex dependencies between the semantic labels and the 3-D geometry. Guo et al. [63] jointly optimized the geometry and semantics by predicting the implicit neural representations of the signed distance, color and semantic field.

Our approach is most closely related to the concurrent work on semantic mesh mapping [64]. Our formulation utilizes sparse depths obtained from keyframe-based SLAM and emphasizes the interaction of geometric reconstruction and semantic segmentation in 3-D mesh reconstruction.

III. Problem Formulation

Consider a UAV equipped with an RGB camera flying outdoor. Let $\mathbf{I}$ denote an RGB image. Obtaining dense depth images during outdoor flight is challenging due to the large distances and relative small variation. However, a VIO or SLAM algorithm can track and estimate the depth of a sparse set of image feature points. Let $\mathbf{D}^s$ be a *sparse* 2-D matrix that contains estimated depths at the image feature locations and zeros everywhere

else. Let $\mathbf{D}$ denote the *dense* ground-truth depth image. Let $\mathbf{S}$ denote an associated ground-truth semantic segmentation image. Assuming there are s semantic classes in total, we model $\mathbf{S}$ as a tensor with the same width and height as the RGB image $\mathbf{I}$, and third channel size s. Each element $\mathbf{S}_{i,j} \in [0,1]^s$ is a one-hot vector with 0 s in all elements, except for a single 1 indicating the true semantic class.

Our goal is to construct an explicit model of the camera view using a 3-D semantic triangle mesh $\mathcal{M} := (\mathbf{V}, \mathbf{C}, \mathcal{E}, \mathcal{F})$, where $\mathbf{V} \in \mathbb{R}^{n\times 3}$ are the vertex spatial coordinates, $\mathbf{C} \in \mathbb{R}^{n\times s}$ are the vertex semantic features, $[n] := \{1, \ldots, n\}$ is the set of vertex indices, $\mathcal{E} \subseteq [n] \times [n]$ are the edges, and $\mathcal{F} \subseteq [n] \times [n] \times [n]$ are the faces. Each row of the matrix $\mathbf{C}$ contains an unnormalized score vector for the s classes that can be converted into a probability distribution over the s classes using the softmax function [65].

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

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

where $\ell(\mathcal{M}; \mathbf{D}, \mathbf{S})$ is a loss function measuring the error between a 3-D semantic mesh $\mathcal{M}$ and a depth image $\mathbf{D}$ plus a semantic image $\mathbf{S}$.

The choice of loss function ℓ is discussed in Section IV. We develop a machine learning approach consisting of an offline training phase and an online mesh reconstruction phase. During training, the parameters $\boldsymbol{\theta}$ are optimized using a training set $\mathcal{D} := \{\mathbf{I}_k, \mathbf{D}_k^s, \mathbf{D}_k, \mathbf{S}_k\}_k$ with known ground-truth depth images and semantic segmentation images. During testing, given streaming RGB images $\mathbf{I}$ and sparse depth measurements $\mathbf{D}^s$, the optimized parameters $\boldsymbol{\theta}^*$ are used in the model $f(\mathbf{I}, \mathbf{D}^s; \boldsymbol{\theta}^*)$ to reconstruct the mesh vertex spatial coordinates $\mathbf{V}$ and semantic features $\mathbf{C}$. The mesh edges $\mathcal{E}$ and faces $\mathcal{F}$ are assumed fixed and known, and hence, are not reconstructed by the model. For notational simplicity, we write the output of model f directly as the semantic mesh $\mathcal{M} = f(\mathbf{I}, \mathbf{D}^s; \boldsymbol{\theta}^*)$. A keyframe-based VIO or SLAM algorithm estimates the positions $\mathbf{p}$ and orientations $\mathbf{R}$ of camera keyframes as well as the depth of sparse keypoint measurements associated with each keyframe. Our approach estimates a local mesh $\mathcal{M} = (\mathbf{V}, \mathbf{C}, \mathcal{E}, \mathcal{F})$ at each camera keyframe. The keyframe meshes can be converted to a global frame (with vertex coordinates $\mathbf{V}\mathbf{R}^\top + \mathbf{1}\mathbf{p}^\top$, where $\mathbf{1}$ is a $n \times 1$ vector filled with 1) and fused to obtain a complete consistent metric-semantic model of the environment.

IV. Loss Functions for Mesh Reconstruction

We develop several loss functions to measure the consistency between a semantic mesh $\mathcal{M}$ and corresponding depth image $\mathbf{D}$ and semantic segmentation image $\mathbf{S}$. Since our problem focuses on optimizing the mesh, the loss function must be differentiable with respect to the mesh vertex spatial coordinates $\mathbf{V}$ and semantic features $\mathbf{C}$. We keep the mesh edges $\mathcal{E}$ and faces $\mathcal{F}$ fixed during the mesh optimization.

A loss function can be defined in the 2-D image plane by rendering a depth image from $\mathcal{M}$ and comparing it with $\mathbf{D}$. The differentiable mesh renderer [28], [66] makes the 3-D mesh rendering, e.g., from a 3-D mesh to a 2-D image, differentiable. Therefore, we can back-propagate the loss measured on the 2-D images to the 3-D mesh vertices. We leverage a differentiable mesh renderer to generate a depth image $\rho_D(\mathcal{M})$ and define a 2-D loss function

$$\ell_2(\mathcal{M}, \mathbf{D}) := \text{mean}(|\rho_D(\mathcal{M}) - \mathbf{D}|) \tag{2}$$

where mean$(\cdot)$ is a function taking the mean over all the valid pixels where both $\mathbf{D}$ and $\rho_D(\mathcal{M})$ have a depth value.

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 to cover only a smaller image region. 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 3-D reconstruction. To address these limitations, we define an additional loss function in the 3-D spatial domain using two point clouds $\mathcal{P}_\mathcal{M}$ and $\mathcal{Q}_\mathbf{D}$ obtained from $\mathcal{M}$ and $\mathbf{D}$, respectively,

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

where d is the asymmetric Chamfer point cloud distance [67]

$$d(\mathcal{P}, \mathcal{Q}) := \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}\in\mathcal{P}} \min_{\mathbf{q}\in\mathcal{Q}} \|\mathbf{p} - \mathbf{q}\|_2^2. \tag{4}$$

Note that squared Euclidean distance is used when calculating the Chamfer distance. To generate $\mathcal{P}_\mathcal{M}$, we sample on the faces of $\mathcal{M}$ uniformly using PyTorch3-D library [66]. The loss function is differentiable with respect to the mesh vertices because the samples on the mesh faces can be represented as linear combinations of the mesh vertices using the barycentric coordinate introduced in Section V-A. 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 $\mathcal{Q}_\mathbf{D}$. The sample number is set as 10 000.

We also define two regularization terms to measure the smoothness of the mesh $\mathcal{M}$. 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 [68] of the degree-normalized Laplacian [69] $\mathbf{L}_n = \mathbf{G}^{-1}\mathbf{L} = \mathbf{I}_n - \mathbf{G}^{-1}\mathbf{A}$ where $\mathbf{I}_n$ is an identity matrix of size $n \times n$

$$\ell_\mathbf{V}(\mathcal{M}) := \frac{1}{n} \|\mathbf{L}_n \mathbf{V}\|_{2,1} \tag{5}$$

where n is the number of vertices. We also introduce a mesh edge regularization term to discourage long edges in the mesh

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

where $\mathbf{v}_i \in \mathbb{R}^3$ are the coordinates of the ith mesh vertex.

We also define a semantic loss function that relates the 3-D mesh semantic information to the 2-D semantic segmentation image by rendering the semantic mesh similar to (2). We define a differentiable semantic rendering function $\rho_S(\mathcal{M})$, which can generate a same-sized image as $\mathbf{S}$ with s channels, where s is the number of semantic classes. At each pixel, the s-dimensional vector stores the unnormalized scores representing the likelihoods of the s classes. We use a softmax function [65] $\sigma_i(\mathbf{x}) = \exp(\mathbf{x}_i)/\sum_{j=1}^{s}\exp(\mathbf{x}_j)$ to compute the probability distribution over the s classes $\sigma(\rho_S(\mathcal{M}))$. For the semantic segmentation task, we choose the Dice loss [70]

$$\ell_{\mathbf{S}}(\mathcal{M}, \mathbf{S}) := -\frac{2|\sigma(\rho_S(\mathcal{M})) \cdot \mathbf{S}|}{|\sigma(\rho_S(\mathcal{M}))| + |\mathbf{S}|} \tag{7}$$

where $|\cdot|$ sums up all the absolute values of the elements. Note that $\mathbf{S}$ contains one-hot vectors whereas $\sigma(\rho_S(\mathcal{M}))$ stores probability vectors for the s classes. Therefore, $|\sigma(\rho_S(\mathcal{M})) \cdot \mathbf{S}|$ is the probabilistic intersection between two semantic segmentation images. In Section VI-G, we compare the Dice loss with three alternative semantic loss functions (cross-entropy loss, focal loss, and Jaccard loss). Finally, we apply Laplacian smoothing (5) to the vertex semantic features

$$\ell_{\mathbf{C}}(\mathcal{M}) := \frac{1}{n} \|\mathbf{L}_n \mathbf{C}\|_{2,1} . \tag{8}$$

The complete loss function is:

$$\begin{aligned}\ell(\mathcal{M}, \mathbf{D}, \mathbf{S}) := {} & w_2\ell_2(\mathcal{M}, \mathbf{D}) + w_3\ell_3(\mathcal{M}, \mathbf{D}) \\ & + w_{\mathbf{V}}\ell_{\mathbf{V}}(\mathcal{M}) + w_{\mathcal{E}}\ell_{\mathcal{E}}(\mathcal{M}) \\ & + w_{\mathbf{S}}\ell_{\mathbf{S}}(\mathcal{M}, \mathbf{S}) + w_{\mathbf{C}}\ell_{\mathbf{C}}(\mathcal{M})\end{aligned} \tag{9}$$

where the first two terms evaluate the error between $\mathcal{M}$ and $\mathbf{D}$, the following two terms encourage smoothness of the mesh structure, and the last two terms evaluate the error between $\mathcal{M}$ and $\mathbf{S}$ and regularize the semantic features, which affects both the geometric and semantic properties of the mesh. The scalars $w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}, w_{\mathbf{S}}, w_{\mathbf{C}} \in \mathbb{R}_{\geq 0}$ allow appropriate weighting of the different terms in (9). Fig. 2 illustrates the loss functions ℓ_2 in (2), ℓ_3 in (3), and ℓ_S in (7).

V. 2-D-3-D Learning for Semantic Mesh Reconstruction

Inspired by depth completion techniques, we approach mesh reconstruction in two stages: *initialization* and *refinement*. In the initialization stage, we generate a mesh from the sparse depth measurements alone (Section V-A). In the refinement stage, we optimize the mesh vertex coordinates based on RGB image features (Section V-B) and assign semantic categories to each vertex using image segmentation features (Section V-C). An overview of our semantic mesh reconstruction model $\mathcal{M} = f(\mathbf{I}, \mathbf{D}^s; \boldsymbol{\theta})$ is shown in Fig. 3.

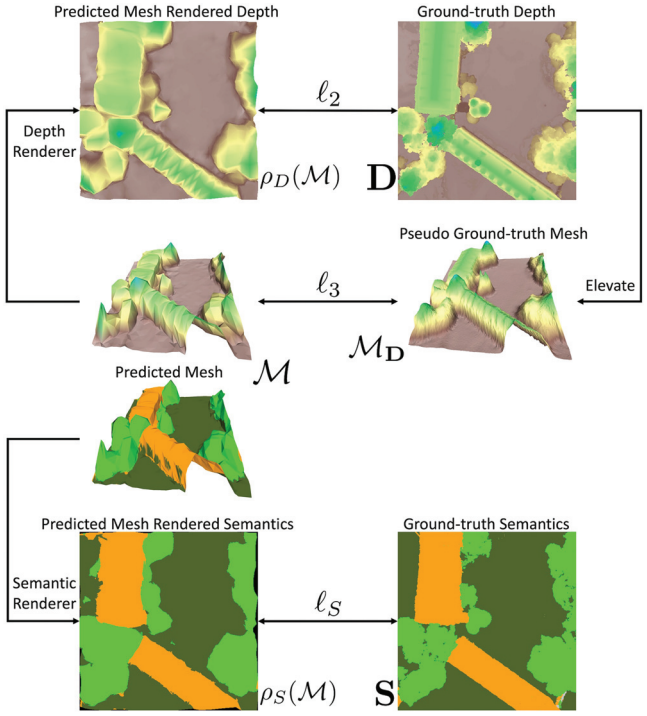


Fig. 2. Loss function visualization: ℓ_2 compares rendered mesh depth $\rho_D(\mathcal{M})$ to a depth image $\mathbf{D}$, ℓ_3 compares a mesh $\mathcal{M}$ to an elevated mesh $\mathcal{M}_{\mathbf{D}}$ obtained from a depth image, and ℓ_S compares a rendered mesh semantic image $\rho_S(\mathcal{M})$ to a semantic segmentation image $\mathbf{S}$.

A. Mesh Initialization

Outdoor terrain structure can be viewed as a 2.5-D surface with height variation. Hence, we initialize a flat mesh surface and change the surface elevation based on the sparse depth measurements. The flat mesh is initialized with regular-grid vertices ($n = 1024$ in our experiments) over the image plane, and the edges and the faces connecting the vertices. See Fig. 4 for an illustration. Subsequently, our mesh reconstruction approach only optimizes the mesh vertices and keeps the edge and face topology fixed. The initialized mesh $\mathcal{M}^{\text{int}} = (\mathbf{V}^*, \mathbf{0})$ is used as an input to the mesh refinement stage, described in Section V-B and V-C. Since we do not update $\mathcal{E}$ and $\mathcal{F}$, we will omit them for simplicity.

We constrain the mesh vertex deformation to the z-axis to change the vertex heights only. The coordinates of the ith vertex of the flat mesh, $\mathbf{v}_i = [v_i^x, v_i^y, 1]$, are divided by a scalar inverse depth λ_i to obtain the ith vertex coordinates $[v_i^x/\lambda_i, v_i^y/\lambda_i, 1/\lambda_i]$ of the initialized mesh. We concatenate λ_i to obtain a vector $\boldsymbol{\lambda} \in \mathbb{R}^n$ of all vertex inverse depths.

Any point $\mathbf{p}$ on the mesh surface that lies in a specific triangle can be represented as a convex combination $\mathbf{p} = b_i\mathbf{v}_i + b_j\mathbf{v}_j + b_k\mathbf{v}_k$ of the triangle vertices $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ with weights $b_i, b_j, b_k \in [0, 1]$, such that $b_i + b_j + b_k = 1$. The vector $[b_i, b_j, b_k]^\top$ is called the *barycentric coordinates* of $\mathbf{p}$. We use barycentric coordinates to relate the sparse depth measurements $\mathbf{D}^s$ to the vertex inverse depths $\boldsymbol{\lambda}$, which is equivalent to a linear interpolation.

Let the valid measurements in the sparse depth image $\mathbf{D}^s$ be $\{(i, j), \mathbf{D}^s_{ij}\}$, where (i, j) are the pixel coordinates and $\mathbf{D}^s_{ij}$

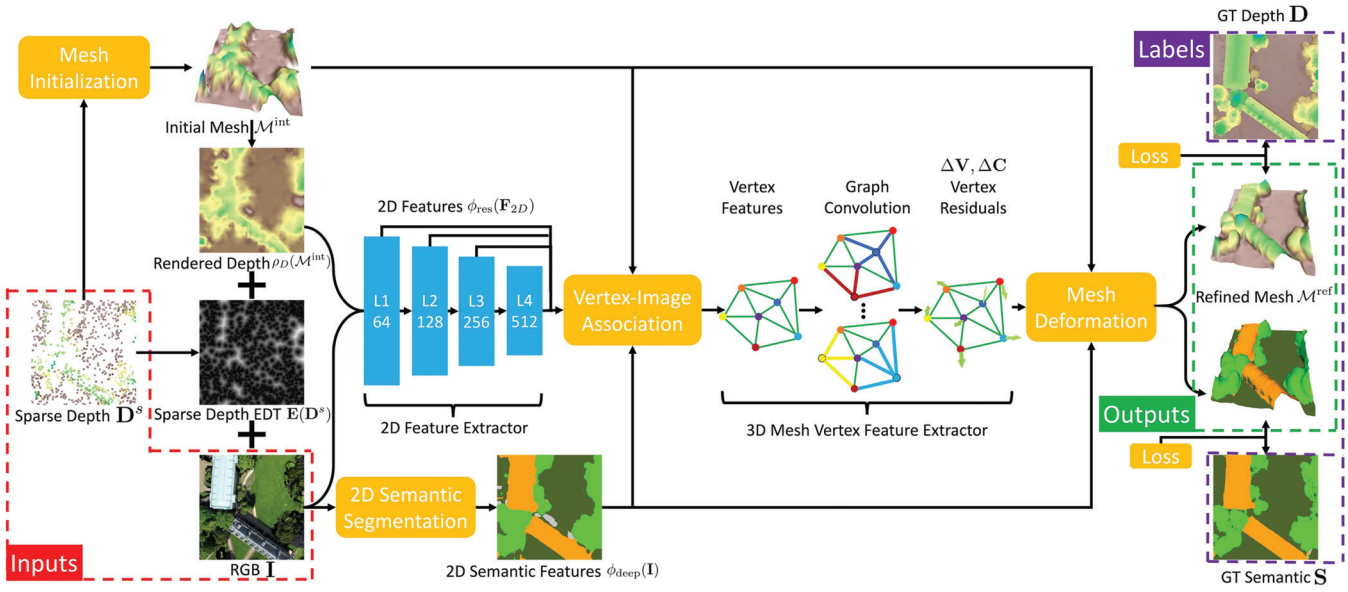


Fig. 3. Overview of our mesh reconstruction architecture. In the initialization stage (Section V-A), we use sparse depth to elevate a flat mesh from the image plane to 3-D space (Fig. 4). In the refinement stage (Section V-B and V-C), we first combine the RGB image, a depth image rendered from the initial mesh, and a EDT of the sparse depth measurements to extract features using a 2-D feature extractor. We have a 2-D semantic segmentation model to generate 2-D semantic features from the RGB image. The 2-D features and the 2-D semantic features are associated with the mesh vertices using camera projection at different stages (Figs. 5 and 6). The vertex spatial coordinates and the vertex semantic features, are regressed using GCN over the mesh. The refined output is a metric-semantic mesh (Fig. 7). The 2-D feature extractor and GCN parameters are optimized jointly using the loss function in Section IV. The 2-D semantic segmentation model is trained separately.

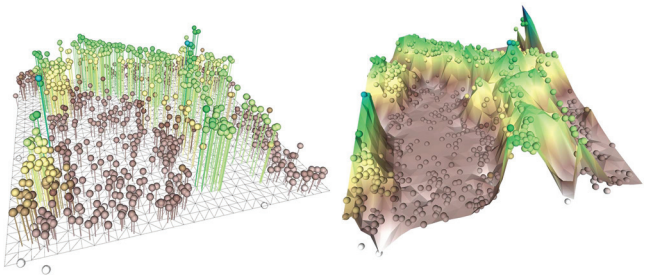


Fig. 4. Mesh initialization stage (Section V-A). Left: Sparse depth measurements (color dots) are used to determine vertex heights from a flat image-plane mesh (bottom wireframe). Right: Initialized mesh. Colors indicate elevation.

are the corresponding depth measurements. Each pixel (i, j) falls within one triangle of the flat 2-D mesh (see Fig. 4). Let $\mathbf{b}_{ij} \in \mathbb{R}^n$ be the barycentric coordinates of pixel (i, j), where at most three elements of $\mathbf{b}_{ij}$, corresponding to the three triangle vertices, are nonzero. The inverse depth $1/\mathbf{D}^s_{ij}$ is related to the vertex inverse depths $\boldsymbol{\lambda}$ through the barycentric coordinates [71], $\mathbf{b}_{ij}^\top \boldsymbol{\lambda} = 1/\mathbf{D}^s_{ij}$. Stacking these equations for all valid pixels (i, j) in $\mathbf{D}^s$, we obtain

$$\mathbf{B}\boldsymbol{\lambda} = \boldsymbol{\rho} \tag{10}$$

where $\boldsymbol{\rho}$ is a vector of the valid inverse depth measurements in $\mathbf{D}^s$ with elements $1/\mathbf{D}^s_{ij}$. Using Laplacian regularization as in (5), we formulate a least-squares problem in $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} \left(\|\mathbf{B}\boldsymbol{\lambda} - \boldsymbol{\rho}\|_2^2 + w'_{\mathbf{V}} \|\mathbf{L}_n \boldsymbol{\lambda}\|_2^2 \right). \tag{11}$$

The problem in (11) has a closed-form solution

$$\boldsymbol{\lambda}^* = (\mathbf{B}^\top \mathbf{B} + w'_{\mathbf{V}} \mathbf{L}_n^\top \mathbf{L}_n)^{-1} \mathbf{B}^\top \boldsymbol{\rho}. \tag{12}$$

The regularization term, not only makes the initialized mesh smoother, but also guarantees that the solution exists even when the number of sparse depth measurements is smaller than the number of mesh vertices. Since the 2-D mesh projection and $\mathbf{L}_n$ are predefined, the problem can be solved very efficiently, e.g., in less than 0.1 s for a mesh with 1024 vertices. Given $\boldsymbol{\lambda}^*$, we obtain an initialized mesh $\mathcal{M}^{\text{int}}$ with each vertex coordinate as $[v_i^x/\lambda_i^*, v_i^y/\lambda_i^*, 1/\lambda_i^*]$.

B. Geometric Mesh Refinement

Initialization using the sparse depth measurements only provides a reasonable mesh reconstruction but many details are missing. In the geometric 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 spatial coordinate residuals. The ground-truth depth maps are used for supervision.

The photometric image information is useful for mesh 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 [32], 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 spatial coordinates. Our network has three stages: Feature extraction, vertex-image feature association, and vertex graph convolution.

Feature Extraction: We extract features from three sources: the RGB image $\mathbf{I}$, the rendered depth $\rho_D(\mathcal{M}^{\text{int}})$ from the initial mesh, and a EDT $\mathbf{E}(\mathbf{D}^s)$ of the sparse depth measurements in the 2-D image space, obtained by computing the Euclidean

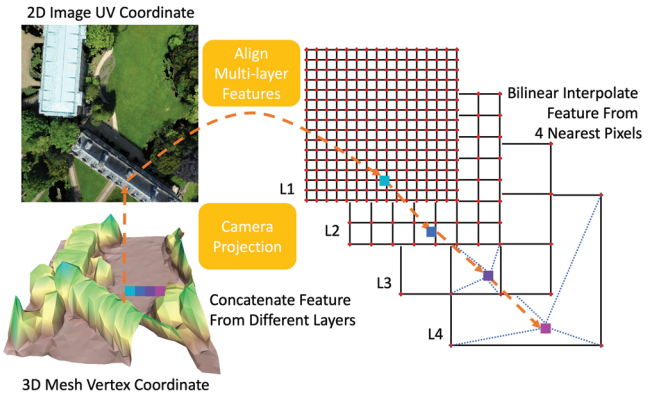


Fig. 5. Illustration of image feature to mesh vertex association. With known camera intrinsics, each mesh vertex can be projected in UV coordinates (range [0, 1]) onto the image plane. Bilinear interpolation is used to associate image feature maps at different resolutions with the mesh vertices. The features across different resolutions are concatenated to form a composite vertex feature.

distance to the closest valid depth measurement pixel from each pixel coordinate. The three images are concatenated to form a 5-channel input [3-channels in $\mathbf{I}$, 1-channel in each $\rho_D(\mathcal{M}^{\text{int}})$ and $\mathbf{E}(\mathbf{D}^s)$]

$$\mathbf{F}_{2D} = \text{concat}(\mathbf{I}, \rho_D(\mathcal{M}^{\text{int}}), \mathbf{E}(\mathbf{D}^s)). \tag{13}$$

Four layers of features with different resolution and channels are extracted

$$[\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \mathbf{L}_4] = \phi_{\text{res}}(\mathbf{F}_{2D}; \boldsymbol{\theta}_{2D}) \tag{14}$$

where ϕ_{res} is a ResNet model [72] with parameters $\boldsymbol{\theta}_{2D}$.

Vertex-Image Feature Association: Next, we construct 3-D features for the mesh vertices by projecting each vertex to the image plane and interpolating the 2-D image features. This idea is inspired by Pixel2Mesh [25], which projects mesh vertices onto the image plane and extracts features at the projected coordinates. To obtain multiscale 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 (14). The vertex–image association step is illustrated in Fig. 5. All features corresponding to different channels are concatenated to form composite vertex features. We define associate$(\cdot, \cdot)$ as the function that assigns 2-D features to 3-D mesh vertices

$$\mathbf{V}_{g_{\text{in}}} = \text{associate}(\mathcal{M}, \phi_{\text{res}}(\mathbf{F}_{2D})) \tag{15}$$

where $\mathbf{V}_{g_{\text{in}}} \in \mathbb{R}^{n \times (l_1 + l_2 + l_3 + l_4)}$ are the vertex features and l_i is the number of channels in feature map $\mathbf{L}_i$.

Vertex Graph Convolution: After the feature assignment, the mesh can be viewed as a graph with vertex features $\mathbf{V}_{g_{\text{in}}}$. Using the vertex features, a GCN [26], [32] is a suitable architecture to predict coordinate deformation $\Delta\mathbf{V}$ for the vertex spatial coordinates to optimize the agreement between the refined mesh $\mathcal{M}^{\text{ref}} = (\mathbf{V} + \Delta\mathbf{V})$ and the ground-truth depth $\mathbf{D}$ according to the loss in (9). To capture a larger region of feature influence, we

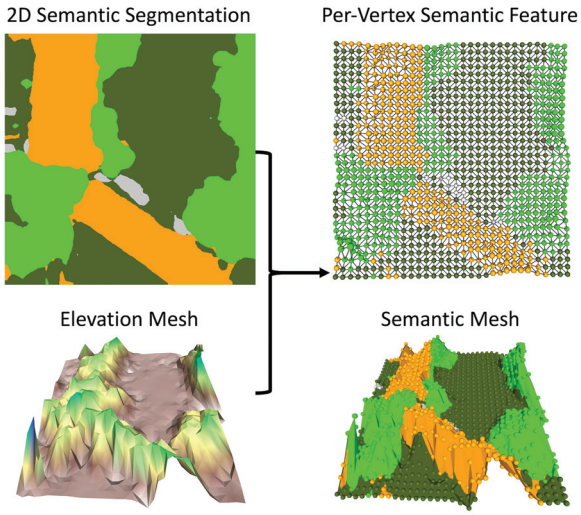


Fig. 6. 2-D semantic segmentation feature map (top left) is used to generate semantic features for the 3-D elevation mesh vertices (bottom left). Each mesh vertex is projected to the semantic segmentation feature map to retrieve an associated semantic feature (top right). Dense semantic features over the whole mesh can be obtained by interpolation on the mesh faces (bottom right).

use three layers of graph convolution $g_1^{\mathbf{V}}, g_2^{\mathbf{V}}, g_3^{\mathbf{V}}$, as follows:

$$\begin{aligned}
\mathbf{V}_1^{\text{in}} &= \text{ReLU}(\mathbf{W}_1^{\mathbf{V}} \mathbf{V}_{g_{\text{in}}}) \\
\mathbf{V}_i^{\text{in}} &= \mathbf{V}_{i-1}^{\text{out}}, && i = 2, 3 \\
\mathbf{V}_i^{\text{out}} &= \text{ReLU}(g_i^{\mathbf{V}}([\mathbf{V}_i^{\text{in}}; \mathbf{V}]; \boldsymbol{\theta}_{g\mathbf{V}i})), && i = 1, 2, 3 \\
\Delta\mathbf{V} &= \mathbf{W}_2^{\mathbf{V}}[\mathbf{V}_3^{\text{out}}; \mathbf{V}]
\end{aligned} \tag{16}$$

where $\Delta\mathbf{V} \in \mathbb{R}^{n \times 3}$ is the matrix of spatial coordinate residuals, $\mathbf{W}_1^{\mathbf{V}}$ and $\mathbf{W}_2^{\mathbf{V}}$ are weight matrices of the linear layers, $\boldsymbol{\theta}_{g\mathbf{V}i}$ are the graph convolution layer weights, and ReLU is the rectified linear unit activation function $\text{ReLU}(x) = \max(0, x)$. The trainable parameters for vertex graph convolution are $\boldsymbol{\theta}_{3D\mathbf{V}} = [\mathbf{W}_1^{\mathbf{V}}; \mathbf{W}_2^{\mathbf{V}}; \boldsymbol{\theta}_{g\mathbf{V}1}; \boldsymbol{\theta}_{g\mathbf{V}2}; \boldsymbol{\theta}_{g\mathbf{V}3}]$. It is possible to concatenate more stages of vertex–image feature association and graph convolution. At stage i, the previous stage's refined mesh $\mathcal{M}_{i-1}^{\text{ref}}$ is set as the initial mesh $\mathcal{M}_i^{\text{int}}$ and new vertex features are extracted via vertex–image feature association and fed to new graph convolution layers. All refined meshes at different stages are evaluated using the ground-truth depth map $\mathbf{D}$ using the loss functions defined in (9).

C. Semantic Mesh Reconstruction

To further enrich the environment representation, we introduce semantic information in the mesh reconstruction. By assigning per-vertex semantic features $\mathbf{C}$, we can interpolate the semantic information over the whole mesh using barycentric coordinates (see Fig. 6). To obtain a 2-D semantic segmentation image from the mesh, we use the differentiable semantic renderer $\rho_{\mathbf{S}}$ introduced in (7). Both the vertex spatial coordinates $\mathbf{V}$ and the semantic features $\mathbf{C}$ can affect the rendered 2-D semantic segmentation image $\rho_{\mathbf{S}}(\mathcal{M})$. Hence, by optimizing the semantic loss in (7), we can refine both the semantic features and the geometric structure of the mesh.

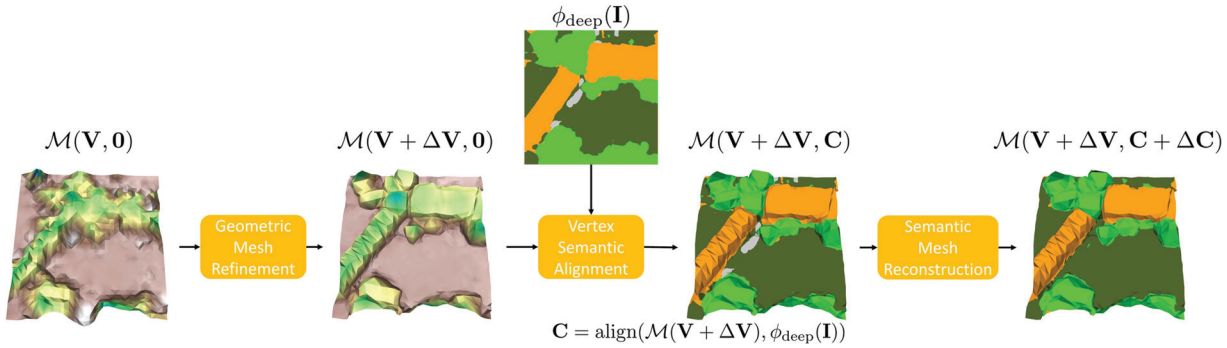


Fig. 7. Mesh refinement stage: Mesh vertex spatial coordinates are refined to $\mathbf{V}+\Delta\mathbf{V}$ using graph convolution based on the RGB image features. Then, the semantic features $\mathbf{C}$ of the mesh vertices are initialized by projecting the vertices to the image plane and associating them with 2-D semantic segmentation features. Finally, the vertex semantic features are refined to be $\mathbf{C}+\Delta\mathbf{C}$ using graph convolution.

We first obtain 2-D semantic segmentation features $\phi_{\text{deep}}(\mathbf{I};\boldsymbol{\theta}_{\text{2Dsem}})$ from the RGB image $\mathbf{I}$ using the DeepLabv3 model [73] with parameters $\boldsymbol{\theta}_{\text{2Dsem}}$. Then, we associate the mesh vertices to the 2-D semantic feature map to get initial mesh vertex semantic features

$$\mathbf{C}=\text{associate}(\mathcal{M},\phi_{\text{deep}}(\mathbf{I})). \tag{17}$$

Fig. 6 illustrates the mesh vertex association with respect to the 2-D semantic segmentation features. In the semantic refinement stage, we regress a semantic residual $\Delta\mathbf{C}$ for the semantic features. We use three layers of graph convolution $g_1^{\mathbf{C}}, g_2^{\mathbf{C}}, g_3^{\mathbf{C}}$. We also use the $\mathbf{V}_{g_{\text{in}}}$ extracted from ResNet in (15) as an input to the first graph convolution layer. In addition, we concatenate the initial mesh vertex semantic features $\mathbf{C}$ in (17) to the graph convolution input

$$\begin{aligned}
\mathbf{C}_1^{\text{in}} &= \text{ReLU}(\mathbf{W}_1^{\mathbf{C}}\mathbf{V}_{g_{\text{in}}}) \\
\mathbf{C}_i^{\text{in}} &= \mathbf{C}_{i-1}^{\text{out}}, && i=2,3 \\
\mathbf{C}_i^{\text{out}} &= \text{ReLU}(g_i^{\mathbf{C}}([\mathbf{C}_i^{\text{in}};\mathbf{V};\mathbf{C}];\boldsymbol{\theta}_{g\mathbf{C}i})), && i=1,2,3 \\
\Delta\mathbf{C} &= \mathbf{W}_2^{\mathbf{C}}[\mathbf{C}_3^{\text{out}};\mathbf{V};\mathbf{C}]
\end{aligned} \tag{18}$$

where $\Delta\mathbf{C}\in\mathbb{R}^{n\times s}$ is the matrix of semantic residuals and $\mathbf{W}_1^{\mathbf{C}}$ and $\mathbf{W}_2^{\mathbf{C}}$ are two matrices as linear layers. The trainable parameters for vertex semantic graph convolution are $\boldsymbol{\theta}_{\text{3DC}}=[\mathbf{W}_1^{\mathbf{C}};\mathbf{W}_2^{\mathbf{C}};\boldsymbol{\theta}_{g\mathbf{C}1};\boldsymbol{\theta}_{g\mathbf{C}2};\boldsymbol{\theta}_{g\mathbf{C}3}]$.

Now we can perform the joint geometric and semantic refinement. All trainable parameters for the 3-D graph convolution are $\boldsymbol{\theta}_{\text{3D}}=[\boldsymbol{\theta}_{\text{3DV}};\boldsymbol{\theta}_{\text{3DC}}]$. An illustration of the joint geometric and semantic refinement is provided in Fig. 7. For the initial mesh $\mathcal{M}(\mathbf{V},\mathbf{0})$, we first estimate the geometric residuals $\Delta\mathbf{V}$ (16) from Section V-B. On the geometrically refined mesh $\mathcal{M}(\mathbf{V}+\Delta\mathbf{V},\mathbf{0})$, we initialize the semantic features as in (17) to get $\mathcal{M}(\mathbf{V}+\Delta\mathbf{V},\mathbf{C})$. Then, we estimate the semantic residuals $\Delta\mathbf{C}$ (18). The final joint geometric and semantic refined mesh is $\mathcal{M}^{\text{ref}}=(\mathbf{V}+\Delta\mathbf{V},\mathbf{C}+\Delta\mathbf{C})$.

D. Global Mesh Merging

Given semantically annotated meshes obtained by our model from each camera view, a global mesh of the whole environment can be obtained by transforming each local mesh to the global frame using the camera pose trajectory and merging it into a combined global mesh. We design an approach to incrementally merge local meshes into a global mesh. Given a new local mesh obtained from a camera view with a known pose and the current global mesh, we update the global mesh by merging information from the local mesh.

First, we refine the global mesh vertices. We transform the global mesh to the local camera frame and project it onto the 2-D image plane. If the resulting 2-D global mesh covers an area over a certain threshold (e.g., 70%), we regard the local frame as duplicate and proceed to the next frame. Otherwise, we determine the overlapping parts between the 2-D projections of the global and local meshes. We choose the vertices of the overlapped global mesh as a source point cloud and sample a point cloud from the overlapped local mesh vertices as a target point cloud. We perform nonrigid point cloud registration between the source and target point cloud using the coherent point drift (CPD) algorithm [74]. Through this nonrigid transformation, we deform and refine the global mesh geometry based on the local mesh information.

Second, we introduce new vertices and faces into the global mesh from the nonoverlapping region of the local mesh. We remove the faces of the local mesh that overlap with the global mesh projection on the image plane. Through this step, we decouple the global mesh and the local mesh because their 2-D projections do not intersect with each other any longer. We perform 2-D constrained Delaunay triangulation [75] over the global and local mesh projections, keeping the edges of existing triangles in tact. Through this step, we connect the global mesh and the local mesh to obtain a new global mesh, which is lifted back to 3-D using the vertex depth values and the known camera pose.

Figs. 8 and 9 demonstrate the mesh merging process, which results in a single consistent global mesh and removes artifacts, such as double layers in naïve mesh merging.

VI. Experiments

In this section, we evaluate our metric-semantic mesh reconstruction approach using aerial image sequences generated from three open-source 3-D datasets: WHU MVS/Stereo [76], SensatUrban [58], and STPLS3D [77]. We evaluate the model

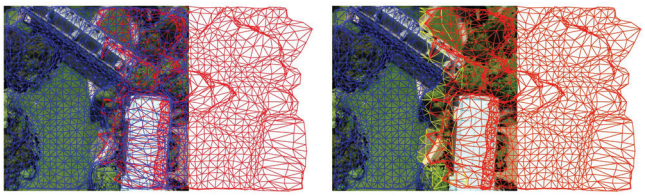


Fig. 8. Red: Global mesh. Blue: Local mesh. Yellow: New edges after merging. Left: The refined global mesh overlaps with the local mesh. Right: The global and the local meshes are separated by removing overlapping faces and a new global mesh is generated via 2-D constrained Delaunay triangulation.

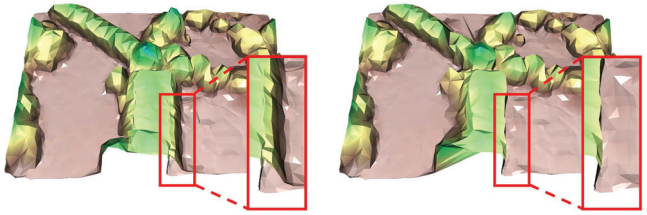


Fig. 9. Left: Stacking two local meshes directly. Right: Merging two meshes with our proposed method in Section V-D.

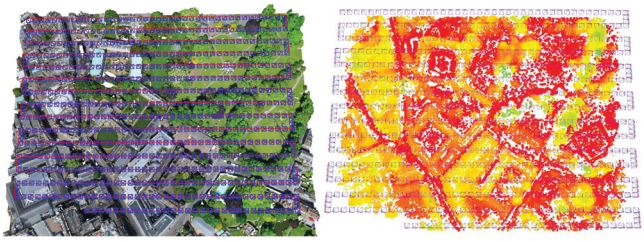


Fig. 10. Left: Camera trajectory used to render RGBD images from a point cloud model generated from the SensatUrban dataset [58]. Right: Sparse depth points and camera poses estimated by ORB-SLAM3 [8]. The color indicates elevation.

generalization ability by training and testing on different datasets. Ablation studies are included to show the effectiveness of our choices in the model design.

A. Datasets

Our mesh reconstruction approach requires ground-truth depth and semantic segmentation data for supervised training, which are generally not available and challenging to obtain from RGB aerial images. We used photo-realistic point cloud models covering several km^2 reconstructed from real aerial images in WHU MVS/Stereo and SensatUrban dataset to render RGB, depth, and semantic segmentation images. This provides accurate depth and semantic supervision data, while keeping the RGB images realistic, which is important for real-world applications of our model. We also use data from the synthetic STPLS3D dataset, which has more variation in the scene layout and the texture. We divide the large point cloud models in each dataset into different regions and generate different image sequences over them. Each camera trajectory follows a sweeping grid-pattern, which is common in drone flight planning (see Fig. 10). The camera trajectories are chosen to ensure enough image overlap for tracking and sparse depth reconstruction. RGBD images with resolution 512×512 are rendered along the trajectory from the ground-truth point cloud using PyTorch3D [66]. We keep the RGB aerial image resolution at around 0.2 m/pixel. When semantic labels are available in the point cloud model, we also render semantic segmentation images with the same size as the RGBD images. All the data sequences are available in the Supplementary Material.

WHU: The WHU MVS/Stereo dataset [76] provides geo-calibrated RGBD images rendered from a highly accurate 3-D DSM of a 6.7×2.2 km^2 area over Meitan County, Guizhou Province, China. The 3-D DSM model is not publicly available, so we recover a dense point cloud from the RGBD images as a ground-truth 3-D model. Semantic labels are also not available in this dataset so we only perform geometric reconstruction using the WHU dataset. We obtain sparse depth measurements $\mathbf{D}^s$ for each image by applying OpenSfM [78] to its four neighbor images with known camera intrinsic and extrinsic parameters. Since monocular structure from motion (SfM) suffers from scale ambiguity, we rescale the reconstructed point cloud obtained from OpenSfM to align it with the real 3-D model. In reality the scale can be recovered from other sensor measurements, such as global positioning system (GPS) or inertial measurement unit (IMU). The point features reconstructed by OpenSfM are treated as sparse noisy depth measurements. The noise is due to feature detection and matching as well as the bundle adjustment step. We also obtain noiseless depth measurements with the same 2-D sparsity pattern from the ground-truth depth images $\mathbf{D}$. We vary the number of available sparse depth measurements as 500, 1000, and 2000. We generate 20 camera trajectory sequences with 200 images in each sequence, split into 14 for training, two for validation, and four for testing.

SensatUrban: The SensatUrban dataset [58] is a point cloud dataset obtained using photogrammetry in two urban areas in Birmingham and Cambridge, U.K.. Each 3-D point in the dataset is labeled as one of 13 semantic classes. The Birmingham region covers an area of 1.2 km^2. The Cambridge region covers an area of 3.2 km^2. We only use the training set part of the data in which point cloud semantic labels are available. We keep four semantic categories (ground, vegetation, building, and traffic road) and merge or discard the remaining less-frequent categories. We used monocular ORB-SLAM3 [8] to estimate the camera poses and sparse feature depths on the SensatUrban dataset. Compared with OpenSfM, ORB-SLAM3 performs sequential optimization of the image sequences, instead of looping over all the images to find matching pairs. As a result, it runs faster (1–10 Hz) and may be deployed on an aerial robot directly. We use ORB-SLAM3 in order to ensure that our method can operate incrementally in time and handle pose and sparse depth estimation errors typical for online SLAM algorithms. We also rescale the reconstructed point cloud and camera poses to align with the real 3-D model. Finally, we project the point cloud to each camera frame to derive a sparse depth image. We vary the number of available sparse depth measurements as 500, 1000, 2000, and 4000. We generate 13 camera trajectory sequences with 660 images in each sequence, split into eight for training, two for validation, and three for testing.

STPLS3D: The STPLS3D dataset [77] is a richly-annotated synthetic 3-D aerial photogrammetry point cloud dataset with more than 16 km^2 of landscapes and up to 18 fine-grained semantic category annotations. To ensure the object placements in the virtual environments resemble real city blocks, the environments are built based on geographic information system data that are publicly available. We use the same four semantic categories as for the SensatUrban dataset. We generated 38 camera trajectory sequences with 660 images in each sequence, split into 26 for training, four for validation, and eight for testing. Camera keyframe poses and sparse depth measurements were estimated using ORB-SLAM3.

B. Implementation Details

During training, we use 1000 sparse depth measurements per image and generate a mesh model with 576/1024/2025 vertices. For the WHU dataset, the weights of the loss function in (9) are set to $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}, w_{\mathbf{S}}, w_{\mathbf{C}}] = [3, 1, 0.5, 0.01, 0, 0]$. For the SensatUrban and the STPLS3D dataset, the weights are set to $[w_2, w_3, w_{\mathbf{V}}, w_{\mathcal{E}}, w_{\mathbf{S}}, w_{\mathbf{C}}] = [5, 1, 0.5, 0.01, 5, 0.5]$ for the joint geometric-semantic training (Section V-C). For the geometric training (Section V-B), the last two weights are set to be 0. The loss weights are decided through evaluation on the validation set. We use the 2-D loss ℓ_2 in (2) as the metric to choose the best model on the validation set. The Chamfer distance d in the ℓ_3 loss (3) is computed using 10 000 samples.

For the WHU experiments in Section VI-C, we use ResNet-18 for the 2-D feature extraction. For the remaining ones including the generalization experiments, we use ResNet-34. The ResNet is initialized with the pretrained weights on ImageNet-1K. The ResNet and GCN parameters of our model are optimized jointly during the mesh refinement training using the Adam optimizer [79] with initial learning rate of 0.0005 for 100 epochs. For the semantic reconstruction task, we first train a DeepLabv3 model with ResNet-50 backbone [73] alone for 2-D semantic segmentation on the SensatUrban training set. The ResNet-50 is initialized with the pretrained weights on ImageNet-1K. We use the cross entropy loss for training and set the class weight as [ground, vegetation, building, traffic road] = [1, 2, 3, 3]. During the mesh semantic refinement step, we use the Dice loss in (7) and keep the same per-class weights. We use three graph convolution stages for the WHU dataset and two graph convolution stages for the SensatUrban and the STPLS3D dataset. For the joint geometric-semantic training, we concatenate two graph convolution stages, where the first stage predicts the geometric residual only and the second stage predicts both the geometric and the semantic residuals. All trainable parameters of the model in (1) are $\boldsymbol{\theta} = [\boldsymbol{\theta}_{2D}; \boldsymbol{\theta}_{2Dsem}; \boldsymbol{\theta}_{3D}]$, including the parameters of the 2-D features extraction, 2-D semantic segmentation, and 3-D graph convolution models.

C. Geometric Reconstruction

Our experiments report the ℓ_2 error in (2) and the ℓ_3 error in (3) for the reconstructed meshes. The ℓ_2 error emphasizes the accuracy of the projected depth, whereas ℓ_3 emphasizes the regions of large depth variation.

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 measured vertex depths. We refer to the baseline method as sparse-depth-triangulation (*SD-tri*). SD-tri defines vertices at all sparse depth measurements (500, 1000, or 2000) and, hence, may produce meshes with different number of vertices compared with other models.

First, we perform geometric reconstruction on the WHU dataset. The quantitative results from the comparison are reported in Table I. All models are trained with 1000 sparse depth measurements and directly generalize to different numbers of sparse depth measurements. We compared three options for the 2-D inputs provided to the mesh refinement stage: An RGB image only (RGB and 3-channels), an RGB image plus rendered depth (RD) from the initial mesh (RGB+RD and 4-channels), and an RGB image plus RD from the initial mesh plus EDT obtained from the sparse depth measurements (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 with RGB+RD+EDT. The RGB+RD model is used to generate our qualitative results in Figs. 11, 12, and 13 with 1024-vertex meshes because it offers good performance according to both error metrics.

At the bottom of Table I, we evaluate the mesh reconstruction accuracy with noisy sparse depth measurements obtained from OpenSfM. The measurements are noisy due to feature matching errors, local minima during bundle adjustment, and the simple projective camera model used for optimization. The average per image errors of the 500/1000/2000 sparse depth measurements were 1.011/1.017/1.023 m, respectively. The baseline SD-tri method performs well in a noiseless setting but degenerates drastically when noise from the SfM feature reconstruction is introduced. In contrast, our model is more robust to noise due to two factors. 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 distinguish among different terrains and structures. The latter is clear from the improved accuracy of the refined, compared with the initialized meshes. We also report the performance using a mesh with only 576 vertices. When the depth measurements are noisy, the 576-vertex mesh has lower ℓ_2 loss compared with the baseline method with similar number of vertices. It even has lower ℓ_3 loss compared with meshes with more vertices generated from the baseline method.

Qualitative results are presented in Figs. 11 and 12. Compared with SD-tri and 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 3-D structure better. Fig. 13 shows a global mesh reconstruction obtained by transforming and merging 12 camera-view mesh reconstructions. The local meshes are transformed to global frame using the camera keyframe poses and

TABLE I
QUANTITATIVE EVALUATION ON THE WHU DATASET [76]

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

The second column shows the number of available sparse depth measurements per image and indicates whether the measurements are noisy (Section VI-A). The *SD-Tri* method triangulates a mesh using all the sparse depth measurements as vertices. The regular-*n* model generates a regular mesh with *n* vertices and performs initialization and refinement steps as we propose in Section V. The initialized model constructs a mesh from the sparse depth (Section V-A). The RGB, RGB+RD, AND RGB+RD+EDT methods refine the initialized mesh (Section V-B), using different inputs, respectively.
The bold values indicate the best results.

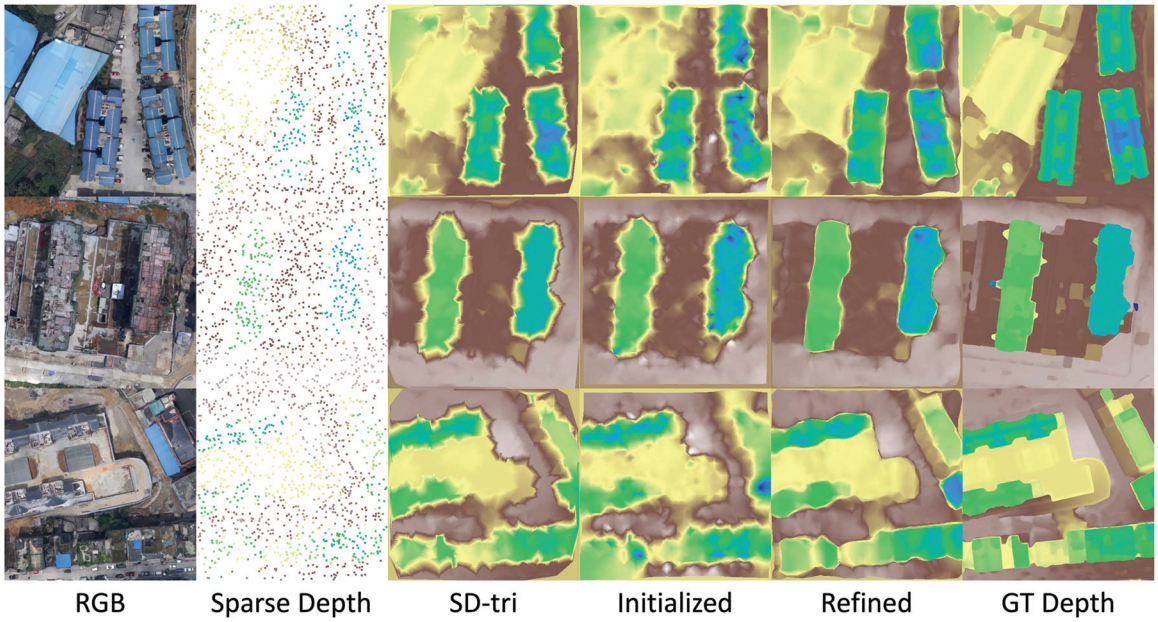


Fig. 11. Mesh reconstructions on the WHU dataset [76] visualized as RD images. The colors indicate the relative depth values. Column 1: RGB images. Column 2: Sparse depth measurements (around 1000). Column 3: Meshes reconstructed from SD-tri. Column 4: Meshes after initialization (Section V-A). Column 5: Meshes after neural network refinement (Section V-B). Column 6: Ground-truth depth images.

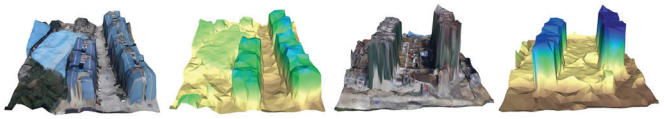


Fig. 12. Reconstructed meshes painted with RGB texture and colors indicating elevations. These are associated with the first two rows in Fig. 11. Sharp vertical transitions of the buildings are reconstructed accurately.

no postprocessing is used to merge them into a single global mesh.

D. Joint Geometric and Semantic Reconstruction

On the SemsatUrban dataset, we first perform geometric reconstruction with the same settings as in the WHU dataset. We train three models with different numbers of mesh vertices: $576 = 24^2$, $1024 = 32^2$, and $2025 = 45^2$. The quantitative results are reported in Table II. As the number of sparse depth measurements increases, the baseline SD-tri method has better accuracy because the number of mesh vertices also increases. Our initialized meshes with fewer vertices are comparable with the SD-tri mesh, and the refined meshes are much better, especially according to the 3-D metric ℓ_3. This shows that the joint 2-D-3-D loss in (9) enables our model to capture 3-D structure details. Comparing the number of input depth measurements, we find that around 2000 measurements on the 512×512 image provide the best performance, while more do not noticeably improve the results. Regarding the number of mesh vertices, all three mesh sizes perform well. While we can see that the 1024-vertex mesh is generally better than 576-vertex mesh,

TABLE II
QUANTITATIVE EVALUATION ON THE SENSATURBAN DATASET

Error	Meshing	SD-tri	Regular-576		Regular-1024		Regular-2025	
	Inputs	(vert = SD)	Initialized	Refined	Initialized	Refined	Initialized	Refined
ℓ_2	500	2.018	2.050	1.175	2.204	**1.088**	1.992	1.171
	1000	1.843	1.841	1.096	1.865	**1.000**	2.033	1.153
	2000	1.715	1.796	1.120	1.700	**0.988**	1.752	1.209
	4000	1.647	1.834	1.181	1.662	**1.026**	1.630	1.283
ℓ_3	500	8.926	7.898	2.371	9.871	**2.128**	7.645	2.371
	1000	7.796	6.353	2.075	6.725	**1.815**	8.875	2.284
	2000	7.164	5.989	2.133	5.527	**1.745**	6.200	2.500
	4000	6.908	6.176	2.339	5.217	**1.844**	5.364	2.806

The second column shows the number of available sparse depth measurements per image (Section VI-A). The baseline *SD-Tri* method triangulates a mesh using all sparse depth measurements as vertices. The regular-*n* model generates a regular mesh with *n* vertices and performs initialization and refinement steps (Section V).
The bold values indicate the best results.

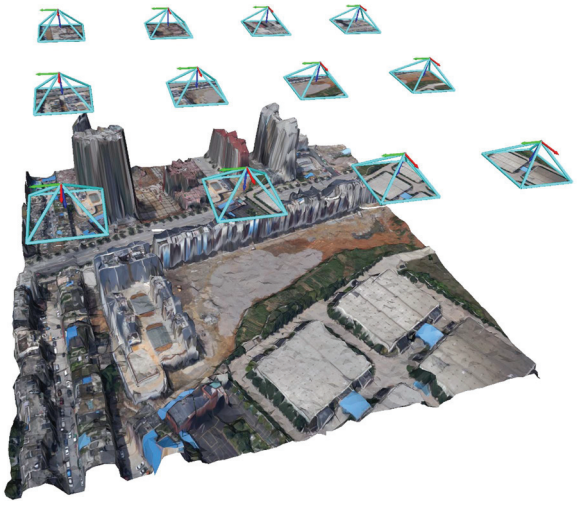


Fig. 13. Complete environment model obtained by transforming to the global frame and merging local meshes from 12 camera views.

TABLE III
SEMANTIC SEGMENTATION PER-CLASS IOU FOR DIFFERENT GEOMETRIC-SEMANTIC MODELS

Class	Ground	Vegetation	Building	Traffic Road
Geo Init	0.642	0.810	0.846	0.643
Geo Refine	0.644	0.809	0.840	0.644
Cross Entropy	0.661	0.805	0.843	0.660
Focal	0.663	0.807	0.844	0.657
Jaccard	0.664	**0.826**	**0.863**	0.653
Our Model (Dice)	**0.674**	0.824	0.860	**0.663**
2-D Seg	0.649	0.834	0.854	0.648

The definitions of the different models can be found in Sections VI-D and VI-G.
The bold values indicate the best results.

TABLE IV
GEOMETRIC ERROR FOR DIFFERENT METRIC-SEMANTIC MODELS

Method	Depth (ℓ_2)	Chamfer (ℓ_3)
Geo Init	1.866	6.725
Geo Refine	1.000	1.815
Cross Entropy	0.994	1.793
Focal	1.035	1.912
Jaccard	**0.976**	1.776
Our Model (Dice)	**0.976**	**1.763**

The definitions of the different models and loss functions can be found in Sections VI-D and VI-G.
The bold values indicate the best results.

the 2025-vertex mesh does not show an advantage over the 1024-vertex mesh. This indicates that good accuracy can be achieved with a light-weight storage-efficient mesh model.

We choose the 1024-vertex mesh to perform joint geometric-semantic reconstruction using 1000 sparse depth measurements. To evaluate the semantic reconstruction, we render a 2-D semantic image from the mesh reconstruction and calculate the per-class intersection over union (IoU). For comparison, we report the IoU of the DeepLabv3 2-D semantic segmentation model (named 2-D Seg), the direct projection of the 2-D semantic segmentation image onto the initial mesh as in (17) (named Geo Init) and the semantic segmentation projection onto the geometrically-refined mesh (named Geo Refine). Only 2-D Seg is using a dense semantic image, whereas the other methods store semantic features on the mesh vertices and interpolate through the semantic mesh renderer. As we can see in Table III, our semantic residual refinement model improves the semantic segmentation performance compared with the direct projection of the 2-D semantic segmentation image. Our approach also outperforms 2-D Seg on most of the categories (ground, building, traffic road) even though it is using only 0.4% of the points to store the semantic information (1024 mesh vertices versus 512×512 segmentation image). Using a compact mesh with few vertices improves the computation and memory efficiency but restricts the mesh from modeling small regions (e.g., small objects, such as cars, street furniture). Our approach can capture additional semantic categories if the number of mesh vertices is increased, e.g., to $64^2 = 4096$ or more, and a semantic segmentation network capable of segmenting small regions is used. Achieving this does not require any changes to the model architecture but only retraining the model parameters.

Further, we investigate whether the semantic mesh refinement affects the geometric reconstruction quality. In Table IV, we can see that our joint geometric-semantic mesh reconstruction achieves better geometric accuracy compared with purely geometric training. This can be explained by the fact that the semantic category information serves as regularization for the geometric properties. The results show that the geometric and semantic information help each other. More qualitative results for single-image reconstruction are provided in Figs. 14 and 15. Compared with SD-tri and initialized mesh, the refined mesh achieves higher reconstruction accuracy. The semantic refinement can improve the 2-D semantic segmentation results.

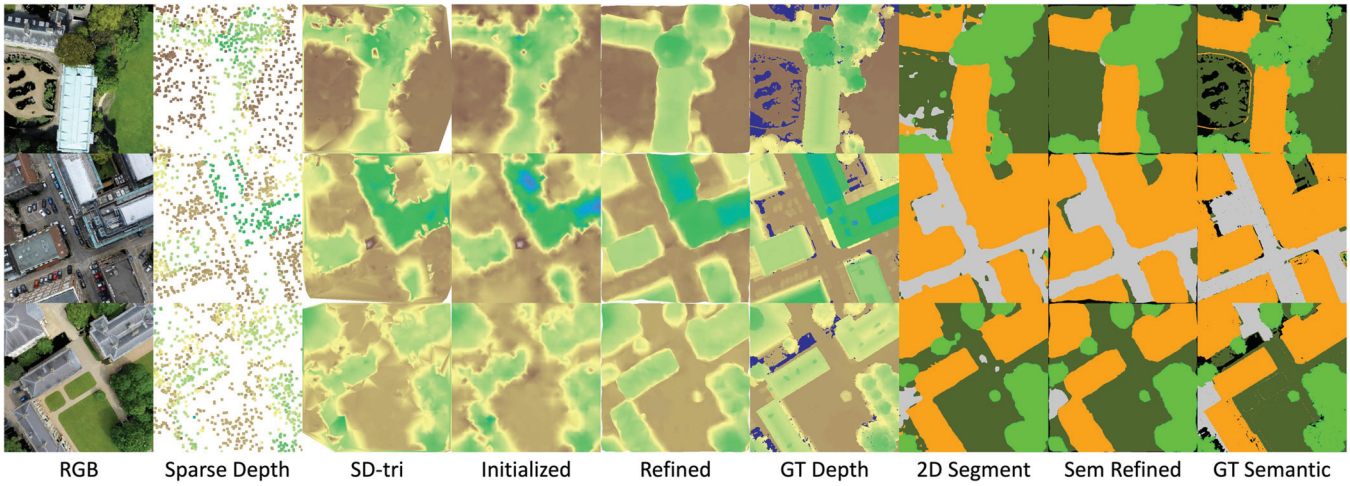


Fig. 14. Mesh reconstructions on SensatUrban dataset [58] visualized as RD (colors indicate the relative depth values) and semantic images. The original 3-D model is not fully complete so the RGB, groundtruth (GT) depth, and GT semantic may have little missing region. Column 1: RGB images. Column 2: Sparse depth measurements (1000). Column 3: Meshes reconstructed from SD-tri. Column 4: Meshes after initialization (Section V-A). Column 5: Meshes after neural network refinement (Section V-B). Column 6: Ground-truth depth images. Column 7: 2-D semantic segmentation results. Column 8: Meshes after neural network refinement (Section V-C). Column 9: Ground-truth semantic segmentation maps.

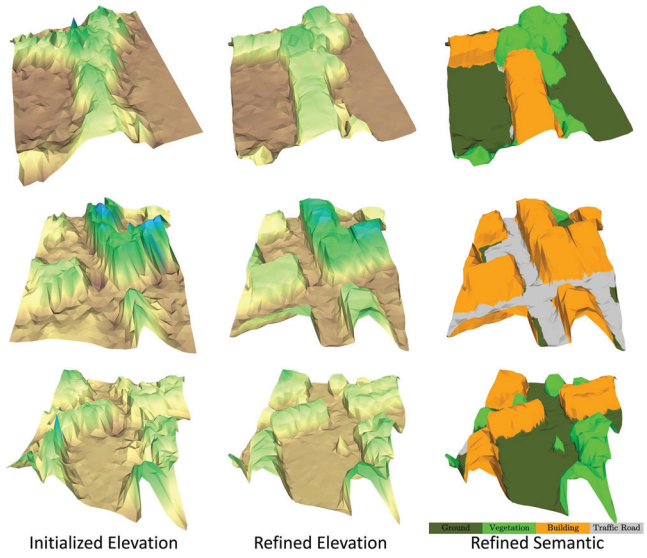


Fig. 15. Reconstructed meshes painted with colors indicating elevations and semantic labels. These are associated with the three rows in Fig. 14. Column 1: Initialized meshes. Column 2: Refined meshes colored by elevation. Column 3: Refined meshes colored by semantic categories.

We can see that some noisy classification labels are removed after the refinement.

To evaluate our global mesh-merging method based on CPD and Delaunay triangulation, we compare it with a simple stacking method that transforms the same set of local meshes to the global frame and simply treats them as a single global mesh. For each scene, about 10% of the frames are used to generate a global mesh. The global mesh is rendered with respect to all frames, including ones that were not used for its construction, to compute the error. Therefore, the error is larger than the per-frame prediction evaluation. The quantitative results are reported in Table V. The stacked global mesh has many double layers in regions where the local meshes overlap, whereas our method successfully merges the local meshes into a consistent global mesh, which improves the reconstruction accuracy, especially according to the 3-D Chamfer error metric. Fig. 16, shows the reconstruction of three global metric-semantic meshes using data from the SensatUrban dataset [58].

TABLE V
GEOMETRIC ERROR FOR GLOBAL MESH-MERGING METHODS

Method	Depth (ℓ_2)	Chamfer (ℓ_3)
Simple Stacking	1.620	6.267
Our Method	1.600	5.219

TABLE VI
GEOMETRIC ERROR ON THE STPLS3D DATASET

Method	Depth (ℓ_2)	Chamfer (ℓ_3)
SD-tri	2.039	11.832
Geo Init	1.996	10.417
Geo Refine	0.977	2.807
Sem Refine	0.972	2.725

We also evaluated our model on the synthetic STPLS3D dataset. We use a 1024-vertex mesh to perform geometric-only and joint geometric-semantic reconstruction using 1000 sparse depth measurements per image. The quantitative results are shown in Table VI. Our joint geometric-semantic mesh reconstruction method outperforms geometric-only mesh reconstruction, which verifies the effectiveness of fusing both geometric and semantic information.

E. Generalization Across Datasets

In this section, we evaluate the generalization ability of our model, trained on one dataset and applied to another. To align the datasets, we regenerated the RGB images and the sparse depth measurements on the WHU dataset to follow the same camera intrinsic parameters and trajectory patterns in the SensatUrban and the STPLS3D dataset so that there are 660 frames for each trajectory and ORB-SLAM3 is used to estimate sparse depth

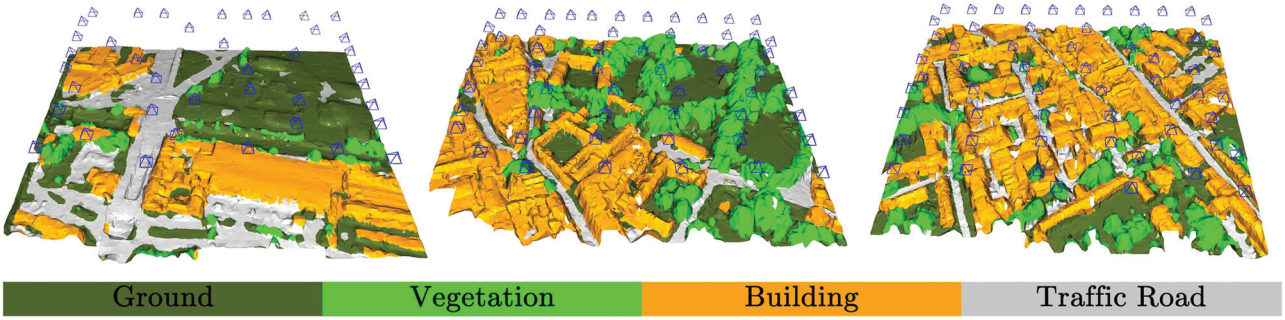


Fig. 16. Global metric-semantic meshes reconstructed from three areas in the SensatUrban dataset [58] by fusing the local mesh reconstructions at the keyframe camera poses (shown in blue). The three global meshes are obtained from 40/55/58 local keyframe meshes, respectively.

TABLE VII
GENERALIZATION EXPERIMENT: GEOMETRIC ERROR ON WHU

Method	Depth (ℓ_2)	Chamfer (ℓ_3)
SD-tri	3.628	40.431
Init	3.582	38.570
Refine (WHU)	2.253	11.332
Refine (STPLS3D)	20.043	1478.478
Refine (STPLS3D finetune)	2.047	10.796

Brackets indicate the dataset trained on.

TABLE VIII
GENERALIZATION EXPERIMENT: GEOMETRIC ERROR ON SENSATURBAN

Method	Depth (ℓ_2)	Chamfer (ℓ_3)
SD-tri	1.843	7.796
Init	1.865	6.725
Refine (SensatUrban)	1.000	1.815
Sem Refine (SensatUrban)	0.976	1.763
Refine (WHU)	1.439	3.827
Refine (WHU finetune)	1.112	2.223
Refine (STPLS3D)	1.442	3.973
Sem Refine (STPLS3D)	1.501	4.309
Refine (STPLS3D finetune)	1.021	1.992
Sem Refine (STPLS3D finetune)	1.043	2.111

Brackets indicate the dataset trained on.

TABLE IX
GENERALIZATION EXPERIMENT: SEMANTIC SEGMENTATION PER-CLASS IOU ON SENSATURBAN

Class	Ground	Vegetation	Building	Traffic Road
2-D Seg (SensatUrban)	0.649	0.834	0.854	0.648
Sem Refine (SensatUrban)	0.674	0.824	0.860	0.663
2-D Seg (STPLS3D)	0.444	0.676	0.565	0.061
Sem Refine (STPLS3D)	0.528	0.667	0.608	0.038
2-D Seg (STPLS3D finetune)	0.652	0.827	0.838	0.638
Sem Refine (STPLS3D finetune)	0.657	0.814	0.850	0.623

Brackets indicate the dataset trained on.

measurements and camera keyframe poses. It is challenging to achieve zero-shot generalization, so we also include a finetuning step. During finetuning, we only use 10% of the target domain training set and train for 30 epochs. A validation set (10% of the target domain validation set) is used to choose the best model with the 2-D loss ℓ_2 as the metric. Usually, models trained on larger datasets show better generalization ability. We choose to use a model trained on WHU to generalize to SensatUrban and a model trained on STPLS3D to generalize to both WHU and SensatUrban. The average per image sparse depth errors in meters for 1000 depth samples were 1.771 on STPLS3D, 2.460 on WHU, and 1.597 on SensatUrban. The sparse depth measurements are generated through ORB-SLAM3.

First, we evaluate how the model trained on STPLS3D generalizes to WHU. The geometric error is reported in Table VII. The WHU dataset is more challenging due to the presence of denser and taller (> 30 m) buildings. The camera intrinsics and the flight pattern and height are different compared with the data generated in Section VI-C so the numbers in Table VII are not directly comparable with Table I. Zero-shot generalization does not work for WHU, which is understandable given the large domain gap. The STPLS3D synthetic dataset uses scene layouts extracted from a U.S. Geological Survey, which covers cities in the United States, while the WHU dataset is collected in a Chinese city. After finetuning with only 10% of the original WHU training set, our model generalizes well to WHU, and even outperforms the model trained purely on WHU.

Next, we evaluate how models trained on WHU and STPLS3D generalize to SensatUrban. The results are presented in Tables VIII and IX. We report only geometric error for the model trained on WHU. We can see that zero-shot generalization from WHU to SensatUrban improves the initialized meshes, whereas a finetuned model performs even better. We train a geometric-only model and a metric-semantic model on STPLS3D. In terms of geometric loss, both STPLS3D models generalize well to SensatUrban and their performance after finetuning is close to that of a model trained on SensatUrban. However, the metric-semantic model is slightly worse than the pure geometric model. In terms of semantic segmentation performance, zero-shot generalization does not perform well and especially fails on the traffic road category. After finetuning, the metric-semantic model can largely close the gap between itself and the SensatUrban model. Given that the RGB images from the synthetic scenes in STPLS3D have very different appearance, it is understandable that the semantic model that heavily relies on the RGB image might be harder to generalize compared with the geometric-only model.

These experiments demonstrate promising generalization ability of our mesh reconstruction method, using limited data to finetune or even without finetuning in some cases. The model generalizes better in terms of geometric reconstruction than in terms of semantic classification. Nevertheless, it is exciting to see that a model trained on a synthetic dataset (STPLS3D) can generalize well to real data. This makes it possible to

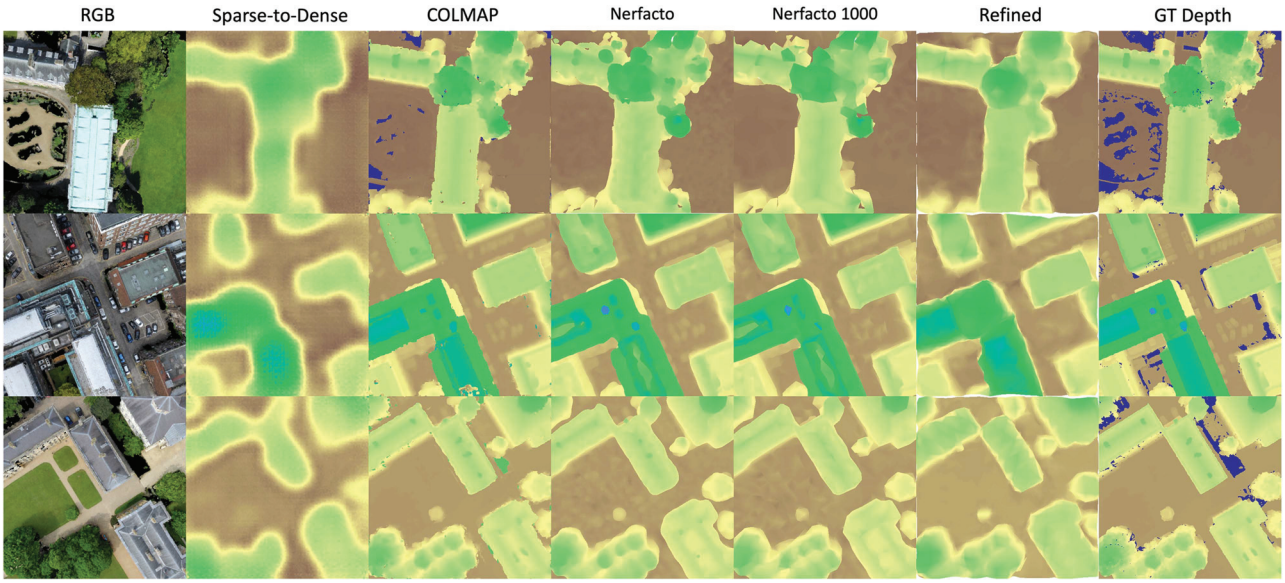


Fig. 17. Depth reconstruction comparison on the SensatUrban dataset among Sparse-to-Dense [3], COLMAP [36], Nerfacto [52], Nerfacto restricted to 1000 mesh vertices, and our method.

TABLE X
GEOMETRIC ERROR COMPARISON AMONG SPARSE-TO-DENSE [3], COLMAP [36], NERFACTO [52], AND OUR METHOD USING METRICS DEFINED IN [37]

Method	2-D				3-D					
	Abs Diff (ℓ_2) ↓	RMSE ↓	Abs Rel ↓	Sq Rel ↓	Chamfer (ℓ_3) ↓	Accuracy ↓	Completeness ↓	Precision ↑	Recall ↑	F-score ↑
Initializaiton	1.865	3.210	0.029	0.218	6.692	1.319	1.173	0.192	0.202	0.197
Refinement	1.000	1.792	0.015	0.056	1.787	0.767	0.778	0.288	0.298	0.293
Semantic Refinement	0.976	1.715	0.014	0.053	1.735	0.754	0.772	0.293	0.301	0.297
Depth Completion - Sparse-to-Dense	1.987	2.949	0.029	0.148	6.136	1.283	1.432	0.139	0.137	0.138
MVS - COLMAP	0.425	1.741	0.006	0.054	3.358	0.638	0.914	0.464	0.405	0.432
MVS - COLMAP Mesh	0.588	1.612	0.008	0.039	3.379	0.980	0.724	0.320	0.361	0.339
NeRF - Nerfacto Mesh	1.217	2.444	0.016	0.102	5.818	1.364	0.947	0.181	0.221	0.199
NeRF - Nerfacto Mesh (1000)	1.178	2.395	0.016	0.099	5.351	1.258	1.018	0.195	0.212	0.203

Nerfacto Mesh (1000) uses about 1000 mesh vertices, similar to our method. The other two mesh methods use about 20 000 vertices.

achieve good performance by training a model with inexpensive synthetic data that comes with free ground-truth labels and finetuning on a small set from the target domain.

F. Comparison With Other Methods

In this section, we compare our approach with depth completion, MVS, and NeRF techniques on the SensatUrban dataset [58] with 1000 sparse depth measurements. The qualitative results are shown in Fig. 17. The quantitative results are shown in Table X. To evaluate the reconstruction quality of the different methods comprehensively, we report multiple 2-D and 3-D reconstruction accuracy metrics [37]. The threshold distance for 3-D precision and recall is set to 0.5 m due to the large scale of the reconstructed mesh.

Depth completion methods take a sparse depth image and other inputs, such as an RGB image, and recover a dense depth image. While there are many depth completion algorithms, few focus on the aerial image domain. We compare against the sparse-to-dense method [3], an end-to-end deep learning regression model, because it considers a similar problem setting and uses similar feature extraction as our approach. The Sparse-to-Dense model consists of a ResNet feature extraction encoder and per-pixel depth regression decoder. We trained both our model and Sparse-to-Dense with a ResNet-34 feature extractor, thus focusing the comparison on the performance of the pure 2-D learning and per-pixel depth regression of Sparse-to-Dense versus the joint 2-D-3-D learning for mesh refinement of our method. The results in Fig. 17 and Table X show that the Sparse-to-Dense model is not as accurate as our method on the SensatUrban dataset, and it is beneficial to utilize our joint 2-D-3-D learning technique. At least on this dataset, it is challenging for Sparse-to-Dense to regress an accurate dense depth map, while, using the same 2-D feature extraction network, our mesh initialization and refinement method performs better at reconstructing the 3-D scene.

We also compare our method with COLMAP [36], a MVS technique that recovers 3-D structure from a series of calibrated images using pixelwise view selection for depth and normal estimation. In this comparison, we used ground-truth camera poses and skip the SfM step. For each frame, we manually select neighboring frames within a radius of 50 m for MVS matching. We use the reconstructed dense depth images to obtain a global point cloud, generate a global mesh, and crop the mesh at each camera pose to obtain local meshes associated with each frame. For the meshing step, we compared Poisson reconstruction [80] and Delaunay mesh reconstruction [81] and found that due to point cloud noise the Delaunay reconstruction performs better. The COLMAP is the dense depth estimation

result, whereas the COLMAP Mesh is the subsequent meshing result. For COLMAP, we convert the dense depth images to a point cloud and sample 10 000 points to compute the ℓ_3 error. For COLMAP Mesh, we render the local mesh to get a RD image to compute the ℓ_2 error. The results are shown in Fig. 17 and Table X. The COLMAP method generally has better depth reconstruction measured by ℓ_2 error, whereas our method has lower ℓ_3 error because of the implicit regularization in our mesh reconstruction. The 3-D error ℓ_3 can be large when outliers appear in the reconstruction. COLMAP achieves better reconstruction at the cost of heavy computation for the MVS step. It takes around 4 s per frame to recover a dense depth image using GPU, while meshing requires additional time. Our method is much faster, with 0.07 s per frame on a desktop with GeForce RTX 2080 Ti GPU and 0.45 s per frame on a Jetson AGX Xavier edge computing platform. When it comes to online mesh reconstruction on a resource-constrained platform, our method offers an advantage over MVS.

Finally, we compare with Nerfacto [52], a NeRF model that combines components from recent NeRF papers to achieve a balance between speed and quality. As a NeRF model, Nerfactor takes posed RGB images and constructs an implicit 3-D scene represented by a deep neural network. We used ground-truth camera poses for each image and trained separate Nerfacto models for each test image sequences. One in ten frames was chosen as an evaluation image during training. Notice here no depth images are used for the NeRF training. We observed that depth images rendered directly from the trained Nerfacto model have inconsistent depth across frames. Therefore, we exported a mesh model using the Poisson surface reconstruction [80] implemented in Nerfstudio [52]. Nerfacto exports meshes with different vertex density. A dense mesh has about 20 000 vertices for each camera view, while a sparse mesh has about 1000, which is similar to our method's mesh vertex density. To compute the 2-D metrics in Table X, we RD images from the mesh. The evalution results for Nerfacto are shown in Fig. 17 and Table X. Qualitatively, Nerfacto has similar reconstruction accuracy with our method but the quantitative metrics indicate that it is not as good as our method. Furthermore, Nerfacto needs to be trained for each novel environment and the training can hardly meet real-time requirement, while our method can make the inference faster.

G. Ablation Studies

Section VI-C compared the effect of using RGB, RD, and EDT as inputs for the mesh reconstruction model. This section reports additional ablation studies on the SensatUrban dataset. We use a 1024-vertex mesh model and 1000 sparse depth measurements for training and testing. We evaluate the effects of mesh initialization, types of 2-D input data, and number of graph convolution stages on the geometric mesh reconstruction accuracy. We also evaluate the performance effect of joint metric-semantic training and the choice of a semantic loss function.

1) Mesh Initialization: An important aspect of our model in Section V is the separation of the mesh initialization stage from the mesh refinement stage. The mesh initialization stage allows the data-driven refinement stage to focus on learning the mesh vertex deformation residuals instead of absolute vertex coordinates. To demonstrate the effectiveness of this design, we compare our model with a baseline model, which applies the refinement stage directly to a flat initial mesh. The baseline model, Flat Init, deforms a flat initial mesh with vertex depth specified by the mean of the sparse depth measurements. Table XI shows that the Flat Init model makes the 2-D-3-D learning problem challenging, and the model performs even worse than purely geometric initialization as in Section V-A.

TABLE XI
ABLATION STUDY ON GEOMETRIC RECONSTRUCTION ERROR FOR GEOMETRIC MODELS

Method	Depth (ℓ_2)	Chamfer (ℓ_3)
Geo Init	1.865	6.725
Flat Init	4.646	20.655
RD+EDT	1.761	5.791
RGB	1.521	3.750
RGB+RD	1.070	2.138
1 Stage	1.015	1.828
Our Model	**1.000**	**1.815**

The definitions of the different models can be found in Section VI-G.
The bold values indicate the best results.

2) 2-D Input Channels: In (13), we concatenate an RGB image $\mathbf{I}$ (RGB), RD $\rho_D(\mathcal{M}^{\text{int}})$, and a EDT $\mathbf{E}(\mathbf{D}^s)$ to form a 5-channel input image used for 2-D feature extraction. Table XI evaluates the role of the different 2-D inputs on the overall mesh reconstruction performance. The results indicate that the RGB information plays the most important role in refining the initialized mesh. The model RD+EDT that does not use RGB features performs the worst. Adding RD and EDT inputs to the RGB gives an additional boost to the accuracy.

3) Number of Graph Convolution Stages: Table XI also evaluates the effect of one (1 Stage) versus two (our model) graph convolution stages in the geometric mesh refinement (Section V-B). The first GCN stage contributes the most to the geometric refinement, whereas the second GCN stage further refines the results.

4) Semantic Loss Function: Finally, we discuss the choice of a semantic loss function $\ell_{\mathbf{S}}$ in (7). Instead of the Dice loss in (7), three other semantic loss functions may be considered.

1) The cross entropy loss is widely used for semantic segmentation. Given two stochastic vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in [0,1]^s$, the cross entropy loss is defined as

$$\text{CE}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\sum_{i=1}^{s} \boldsymbol{\beta}_i \log(\boldsymbol{\alpha}_i)$$

$$\ell_{\mathbf{S}1}(\mathcal{M}, \mathbf{S}) := \text{mean}(\text{CE}(\sigma(\rho_S(\mathcal{M})), \mathbf{S})) \tag{19}$$

where CE is applied to the elements $\sigma(\rho_{S,ij}(\mathcal{M})) \in [0,1]^s$ and $\mathbf{S}_{ij} \in [0,1]^s$ of the tensors of predicted and ground-truth semantic class probabilities.

TABLE XII
PREDICTION TIME (S) ON DIFFERENT NVIDIA DEVICES

Platform	Initialization	Refinement	Total
Jetson AGX Xavier w/o GPU	0.45	0.75	1.20
Jetson AGX Xavier GPU	0.30	0.15	0.45
GeForce RTX 2080 Ti GPU	0.05	0.02	0.07

2) The focal loss [82] is a variation of cross entropy, focusing on hard misclassified examples

$$\mathrm{FL}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\sum_i \boldsymbol{\beta}_i (1 - \boldsymbol{\alpha}_i) \log(\boldsymbol{\alpha}_i)$$

$$\ell_{\mathbf{S}2}(\mathcal{M}, \mathbf{S}) := \mathrm{mean}(\mathrm{FL}(\sigma(\rho_S(\mathcal{M})), \mathbf{S})). \tag{20}$$

3) The Jaccard loss [83] measures the negative IoU between the ground-truth and predicted semantic segmentation

$$\ell_{\mathbf{S}3}(\mathcal{M}, \mathbf{S}) := -\frac{|\sigma(\rho_S(\mathcal{M})) \cdot \mathbf{S}|}{|\sigma(\rho_S(\mathcal{M}))| + |\mathbf{S}| - |\sigma(\rho_S(\mathcal{M})) \cdot \mathbf{S}|} \tag{21}$$

where, as in (7), $|\cdot|$ sums up all the absolute values of the elements.

In Table III, we see that the Jaccard loss in (21) leads to good segmentation performance, outperforming the Dice loss in (7) for some categories. The cross entropy and the focal losses are not as good. In Table IV, we see that the cross entropy and the Jaccard loss both outperform the focal loss when considering their effect on the geometric reconstruction accuracy. The Dice loss leads to the best geometric reconstruction accuracy. Considering the joint geometric and semantic performance, we elected to use the Dice loss for our final model.

H. Memory and Computation Complexity

The reconstructed mesh model is a more efficient representation than a dense depth image. A dense depth image requires $512 \times 512 \approx 0.26\,\mathrm{M}$ parameters, and a semantic image also requires the same number of parameters. Our mesh model with fixed face topology only needs storage of the 3-D vertex coordinates and the semantic labels. With 1024 vertices, our semantic mesh model requires only 2% of the depth and semantic image parameters to obtain a high-fidelity reconstruction of a camera view. Our model has about 21 M parameters (ResNet and GCN) and takes about 3 GB GPU memory during inference. We report the inference time of our model on different computation platforms in Table XII. The results show that our mesh reconstruction algorithm can achieve 2 Hz on an embedded NVIDIA Jetson AGX Xavier computer, making it applicable for real-time deployment onboard a robot system. Regarding timing the baseline algorithm for SD-tri, we evaluated its run-time frequency to be at around 20 Hz on the same platform.

I. Limitations

Our 3-D metric-semantic mesh reconstruction algorithm can run efficiently on an embedded computer but as a result the number mesh vertices used for reconstruction is limited, which in turn affects the geometric reconstruction accuracy. Furthermore, local meshes are generated using only a single camera frame without multiview constraints, making it challenging to achieve consistent mesh merging into a global model. Potential avenues for future work that may improve the reconstruction quality include adaptively increasing the mesh vertices depending on the image feature distribution, considering techniques, such as deformable convolution [84] for associating the 3-D mesh vertices with the 2-D image features, utilizing sparse depth measurement uncertainty (e.g., keypoint covariances provided by SLAM) for weighted interpolation during the image features to vertex association, and improving the global mesh merging approach with multiview constraints.

VII. CONCLUSION

This work introduces an approach for 3-D metric-semantic mesh reconstruction from RGB image and sparse depth measurements. Compared with methods that utilize only sparse depth for mesh initialization or triangulation, our approach provides more accurate geometric reconstruction by utilizing RGB image features. Compared with 2-D semantic segmentation methods, our semantic reconstruction eliminates classification inaccuracies by inferring an underlying 3-D mesh structure. The joint metric-semantic reconstruction approach improve geometric accuracy further by utilizing semantic information and provides memory savings compared with dense image depth and segmentation techniques. Employing our method in combination with feature- and keyframe-based odometry techniques allows reconstruction of global dense metric-semantic mesh models with utility in environmental monitoring and semantic navigation applications.

REFERENCES

[1] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[2] C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.
[3] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 4796–4803.
[4] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning joint 2-D-3-D representations for depth completion," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 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 Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.
[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[7] Q. Feng and N. Atanasov, "Mesh reconstruction from aerial images for outdoor terrain mapping using joint 2-D-3-D learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 5208–5214.
[8] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.
[9] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3827–3837.
[10] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8976–8985.
[11] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[12] J. Shi and Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 593–600.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[14] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 3288–3295.

[15] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, "Estimating depth from RGB and sparse sensing," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 176–192.

[16] X. Zuo, N. Merrill, W. Li, Y. Liu, M. Pollefeys, and G. Huang, "CodeVIO: Visual-inertial odometry with learned optimizable dense depth," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 14 382–14 388.

[17] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3565–3572.

[18] W. N. Greene and N. Roy, "FLaME: Fast lightweight mesh estimation using variational smoothing on Delaunay graphs," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4696–4704.

[19] A. Rosinol and L. Carlone, "Smooth mesh estimation from depth data using non-smooth convex optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6633–6640.

[20] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone, "Incremental visual-inertial 3-D mesh generation with structural regularities," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 8220–8226.

[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3-D Euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[22] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3-D surface construction algorithm," in *Proc. 14th Annu. Conf. Comput. Graph. Interactive Techn.*, 1987, pp. 163–169.

[23] W. Wang, Y. Zhao, P. Han, P. Zhao, and S. Bu, "TerrainFusion: Real-time digital surface model reconstruction based on monocular SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 7895–7902.

[24] M. Bloesch, T. Laidlow, R. Clark, S. Leutenegger, and A. Davison, "Learning meshes for dense visual SLAM," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5854–5863.

[25] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D mesh models from single RGB images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 55–71.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[27] H. Kato, Y. Ushiku, and T. Harada, "Neural 3-D mesh renderer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3907–3916.

[28] S. Liu, T. Li, W. Chen, and H. Li, "A general differentiable mesh renderer for image-based 3D reasoning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 50–62, Jan. 2022.

[29] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik, "Learning category-specific mesh reconstruction from image collections," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 371–386.

[30] Q. Feng, Y. Meng, M. Shan, and N. Atanasov, "Localization and mapping using instance-specific mesh models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4985–4991.

[31] A. Simoni, S. Pini, R. Vezzani, and R. Cucchiara, "Multi-category mesh reconstruction from image collections," in *Proc. Int. Conf. 3D Vis.*, 2021, pp. 1321–1330.

[32] G. Gkioxari, J. Johnson, and J. Malik, "Mesh R-CNN," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9784–9794.

[33] Y. Nie, X. Han, S. Guo, Y. Zheng, J. Chang, and J. J. Zhang, "Total 3-DUnderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 52–61.

[34] Z. Weng and S. Yeung, "Holistic 3-D human and scene mesh estimation from single view images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 334–343.

[35] Y. Furukawa and C. Hernández, "Multi-view stereo: A tutorial," *Found. Trends. Comput. Graph. Vis.*, vol. 9, pp. 1–148, Jun. 2015.

[36] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 501–518.

[37] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao, "NeuralRecon: Real-time coherent 3-D reconstruction from monocular video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15 598–15 607.

[38] N. Stier, A. Rich, P. Sen, and T. Hollerer, "VoRTX: Volumetric 3-D reconstruction with transformers for voxelwise view selection and fusion," in *Proc. Int. Conf. 3D Vis.*, 2021, pp. 320–330.

[39] M. Sayed, J. Gibson, J. Watson, V. Prisacariu, M. Firman, and C. Godard, "SimpleRecon: 3D reconstruction without 3D convolutions," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 1–19.

[40] Z. Murez, T. v. As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich, "Atlas: End-to-end 3D scene reconstruction from posed images," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 414–431.

[41] A. Düzçeker, S. Galliani, C. Vogel, P. Speciale, M. Dusmanu, and M. Pollefeys, "DeepVideoMVS: Multi-view stereo on video with recurrent spatio-temporal fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15 319–15 328.

[42] J. Sun et al., "Neural 3-D reconstruction in the wild," in *Proc. ACM SIGGRAPH Conf.*, 2022, pp. 1–9.

[43] N. Stier et al., "FineRecon: Depth-aware feed-forward network for detailed 3-D reconstruction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 18377–18386.

[44] L. Koestler, N. Yang, N. Zeller, and D. Cremers, "TANDEM: Tracking and dense mapping in real-time using deep multi-view stereo," in *Proc. Conf. Robot Learn.*, 2022, vol. 164, pp. 34–45.

[45] Y. Xin, X. Zuo, D. Lu, and S. Leutenegger, "SimpleMapping: Real-time visual-inertial dense mapping with deep multi-view stereo," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2023, pp. 273–282.

[46] B. Mildenhall, P.P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 405–421.

[47] X. Zhang, S. Bi, K. Sunkavalli, H. Su, and Z. Xu, "NeRFusion: Fusing radiance fields for large-scale scene reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5449–5458.

[48] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5460–5469.

[49] Y. Wang, I. Skorokhodov, and P. Wonka, "HF-NeuS: Improved surface reconstruction using high-frequency details," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 1966–1978.

[50] Z. Li et al., "Neuralangelo: High-fidelity neural surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 8456–8465.

[51] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, Jul. 2022, Art. no. 102.

[52] M. Tancik et al., "Nerfstudio: A modular framework for neural radiance field development," in *Proc. ACM SIGGRAPH Conf.*, 2023, Art. no. 72.

[53] S. Samiappan, L. Hathcock, G. Turnage, C. McCraine, J. Pitchford, and R. Moorhead, "Remote sensing of wildfire using a small unmanned aerial system: Post-fire mapping, vegetation recovery and damage analysis in grand bay, Mississippi/Alabama, USA," *Drones*, vol. 3, no. 2, 2019, Art. no. 43.

[54] I. D. Miller, F. Cladera, T. Smith, C. J. Taylor, and V. Kumar, "Stronger together: Air-ground robotic collaboration using semantics," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9643–9650, Oct. 2022.

[55] J. P. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P. H. Torr, "Mesh based semantic modelling for indoor and outdoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2067–2074.

[56] A. Rosinol et al., "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs," *Int. J. Robot. Res.*, vol. 40, no. 12–14, pp. 1510–1546, 2021.

[57] Z. Wei, Y. Wang, H. Yi, Y. Chen, and G. Wang, "Semantic 3-D reconstruction with learning MVS and 2-D segmentation of aerial images," *Appl. Sci.*, vol. 10, no. 4, 2020, Art. no. 1275.

[58] Q. Hu, B. Yang, S. Khalid, W. Xiao, N. Trigoni, and A. Markham, "SensatUrban: Learning semantics from urban-scale photogrammetric point clouds," *Int. J. Comput. Vis.*, vol. 130, no. 2, pp. 316–343, Jan. 2022.

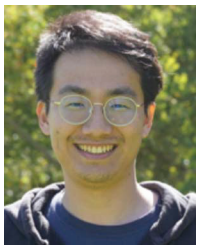
[59] W. Bing et al., "RangeUDF: Semantic surface reconstruction from 3D point clouds," 2022, *arXiv:2204.09138*.

[60] Z. Hu et al., "Voxel-Mesh network for geodesic-aware 3-D semantic segmentation of indoor scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022, early access, Feb. 28, 2022, doi: 10.1109/TPAMI.2022.3194555.

[61] C. Häne, C. Zach, A. Cohen, and M. Pollefeys, "Dense semantic 3-D reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1730–1743, Sep. 2017.

[62] I. Cherabier, J. L. Schönberger, M. R. Oswald, M. Pollefeys, and A. Geiger, "Learning priors for semantic 3D reconstruction," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 325–341.

[63] H. Guo et al., "Neural 3-D scene reconstruction with the manhattan-world assumption," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5501–5510.
[64] M. Herb, T. Weiherer, N. Navab, and F. Tombari, "Lightweight semantic mesh mapping for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6732–3738.
[65] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Proc. 2nd Int. Conf. Neural Inf. Process. Syst.*, 1989, pp. 211–217.
[66] J. Johnson et al., "Accelerating 3D deep learning with PyTorch3D," in *Proc. SIGGRAPH Asia Courses*, 2020.
[67] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proc. 5th Int. Joint Conf. Artif. Intell.*, 1977, pp. 659–663.
[68] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint $\ell_{2,1}$-Norms minimization," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1813–1821.
[69] O. Sorkine, D. C.-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proc. Eurograph./ACM SIGGRAPH Symp. Geometry Process.*, 2004, pp. 175–184.
[70] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.
[71] J. F. Hughes et al., *Computer Graphics: Principles and Practice*, 3rd ed. Reading, MA, USA: Addison-Wesley, Jul. 2013.
[72] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[73] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.
[74] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.
[75] M. V. Anglada, "An improved incremental algorithm for constructing restricted Delaunay triangulations," *Comput. Graph.*, vol. 21, no. 2, pp. 215–223, 1997.
[76] J. Liu and S. Ji, "A novel recurrent encoder-decoder structure for large-scale multi-view stereo reconstruction from an open aerial dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6049–6058.
[77] M. Chen et al., "STPLS3-D: A. large-scale synthetic and real aerial photogrammetry 3-D point cloud dataset," in *Proc. 33rd Brit. Mach. Vis. Conf.*, 2022.
[78] "OpenSfM." Accessed: Feb. 28, 2023. [Online]. Available: https://github.com/mapillary/OpenSfM
[79] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2014.
[80] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proc. 4th Eurograph. Symp. Geometry Process.*, 2006, pp. 61–70.
[81] P. Labatut, J.-P. Pons, and R. Keriven, "Robust and efficient surface reconstruction from range data," *Comput. Graph. Forum*, vol. 28, no. 8, pp. 2275–2290, 2009.
[82] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.
[83] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.
[84] J. Dai et al., "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 764–773.



Qiaojun Feng (Student Member, IEEE) received the B.Eng. degree in automation from Tsinghua University, Beijing, China, in 2017, and the M.S. and Ph.D. degrees in electrical engineering from the University of California San Diego, La Jolla, CA, USA, in 2019 and 2023, respectively.

His research interest focuses on mobile robot autonomy, and particularly on metric-semantic perception and reconstruction.

Nikolay Atanasov (Senior Member, IEEE) received the B.S. degree in electrical engineering from Trinity College, Hartford, CT, USA, in 2008, and the M.S. and Ph.D. degrees in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2012 and 2015, respectively.

He is currently an Assistant Professor of electrical and computer engineering with the University of California San Diego, La Jolla, CA, USA. He works on probabilistic models that unify geometric and semantic information in simultaneous localization and mapping (SLAM) and on optimal control and reinforcement learning algorithms for minimizing probabilistic model uncertainty. His research interests include robotics, control theory, and machine learning, applied to active perception problems for autonomous mobile robots.

Dr. Atanasov was the recipient of the Joseph and Rosaline Wolf award for the best Ph.D. dissertation in electrical and systems engineering with the University of Pennsylvania in 2015, the Best Conference Paper Award at the IEEE International Conference on Robotics and Automation (ICRA) in 2017, the NSF CAREER Award in 2021, and the IEEE RAS Early Academic Career Award in Robotics and Automation in 2023.