

Monocular Dense Visual Odometry Based on Direct Mesh Optimization

Emanuel Trabes

*Service d'électronique et de Microélectronique
University of Mons, Belgium
Department of Electronics
Universidad Nacional de San Luis, Argentina
emanuel.trabes@umons.ac.be*

Kawthar Dellel

*Service d'électronique et de Microélectronique
University of Mons, Belgium
Laboratory of Electronics and Microelectronics
Faculty of Science of Monastir, University of Monastir
kaouther.dellel@student.umons.ac.be*

Carlos Valderrama

*Service d'électronique et de Microélectronique
University of Mons, Belgium
carlos.valderrama@umons.ac.be*

Abstract—Monocular visual odometry (VO) is a cornerstone for robotic navigation, augmented reality, and autonomous systems, enabling camera motion estimation and local 3D reconstruction from a single viewpoint. In this paper, we propose a novel dense monocular VO framework that jointly optimizes both camera poses and a mesh-based scene representation. Unlike traditional methods that rely on sparse features or point-based maps, our approach represents each vertex as a ray and a depth parameter, refining both geometry and pose through direct photometric error minimization. An exponential depth parameterization ensures physically valid values while maintaining a compact optimization space. Our mesh representation naturally encodes surface smoothness and affords a detailed 3D reconstruction that is especially advantageous in low-texture regions. Experiments demonstrate that the proposed system consistently generates dense reconstructions and robust pose estimates, even in challenging scenes. This work lays the groundwork for high-fidelity, real-time 3D mapping and localization by shifting from sparse feature tracking to a dense, mesh-based, photometric optimization framework, enhancing robustness, accuracy, and 3D reconstruction quality, particularly in texture-poor environments.

Index Terms—Visual Odometry, Visual SLAM, Triangular Mesh

I. INTRODUCTION

Visual SLAM (VSLAM) [1] is a key technology behind numerous successful applications across various industries. For instance, self-driving systems for autonomous driving rely on visual SLAM, while many autonomous vacuum cleaners employ cameras coupled with VSLAM methods. Visual odometry (VO) is a sub-set of the whole visual SLAM problem, where only poses and local maps are estimated, with no loop-closure nor global map optimization.

Commonly, VSLAM (and VO) systems must meet several critical requirements. Since these systems often run on resource-constrained platforms, they must operate efficiently

with limited processing power. Errors in pose and map estimation accumulate over time, thus the system must maintain high accuracy to avoid excessive drift. Failures should be rare, and the system should tolerate diverse and challenging conditions. Lastly, many applications such as robotic manipulation require a rich and detailed dense map of the environment.

Numerous approaches have been proposed in the literature, each with specific strengths and weaknesses relative to aforementioned requirements. For example, the VSLAM method in [2] demonstrates excellent accuracy, as evidenced by its low error on the KITTI dataset. It has also been successfully deployed underwater [3] and in highly dynamic lighting conditions [4], illustrating its robustness. However, it produces only a sparse map. The approach in [5] was among the first to achieve dense reconstruction, albeit at a high computational cost. The method in [6] yields a semi-dense map but lacks robust performance across diverse scenarios. Meanwhile, [7] offers high accuracy and efficiency but again generates only a sparse representation.

There are few (if any) systems that simultaneously fulfill all VSLAM requirements. In this work, we propose a visual odometry system designed to satisfy all four key demands: low computational overhead, high accuracy, robust performance, and a dense scene reconstruction.

A common choice for representing the environment in VSLAM is a point cloud. While straightforward, this representation is inherently sparse and often redundant: points belonging to the same surface region (e.g., a wall or floor) are not entirely independent. This redundancy over-parameterizes the optimization problem, making it more challenging for Gauss-Newton or similar algorithms to converge to the global minimum. This ultimately compromises both accuracy and robustness. Over-parametrization increases computational demands. Enforcing smoothness often requires an explicit regu-

larization prior, adding computational overhead and potentially introducing artifacts when the scene deviates from smoothness assumptions.

Point cloud representations also make it challenging to model complex lighting behaviors, as they often rely on the simplifying assumption of Lambertian surfaces. In reality, scenes can exhibit reflections, highlights, or other non-Lambertian effects that degrade system performance. Another effect not modeled correctly by point clouds is occlusions. Parts of the scene which are actually hidden to the current camera will be incorrectly used by the optimization algorithm. Consequently, a more suitable native-dense representation could both reduce computational demands and improve accuracy and robustness.

Mesh-based representations offer a promising alternative. If the scene can be approximated by a continuous surface, a mesh can capture its geometry more precisely. Many human-made environments (e.g., building interiors or city streets) consist largely of piecewise-continuous surfaces, which meshes can represent exactly. Moreover, meshes reduce the number of parameters: rather than storing every 3D point explicitly, a mesh can be described by a sparser set of vertices. This lower-dimensional parameter space simplifies optimization and obviates the need for additional smoothness priors. Normals are also implicitly encoded, facilitating more advanced appearance modeling (e.g., non-Lambertian lighting). Occlusions can also be correctly included in the optimization loop. Another practical advantage is that modern embedded hardware typically includes native acceleration for mesh-based rendering operations.

Several works have used triangular meshes in the context of Visual SLAM. Early works, such as [9], integrate single-image depth estimates from a CNN with features from a sparse monocular VSLAM system into a mesh. The authors show that the system has improved reconstruction accuracy. Having both a depth estimation and a VSLAM system running simultaneously adds more computational requirements. Furthermore, the VSLAM system does not benefit from the depth estimation, so the pose accuracy and the robustness of the system remains the same. A similar method is developed in [10], where the authors probabilistically fuse the depth maps coming from a deep dense VSLAM system. Again, this system requires more computational resources. The authors in [8] propose a method to learn triangular meshes for dense scene representation in VSLAM. They identify scene representation as a critical challenge in VSLAM systems and conclude that triangular meshes are a promising alternative. However, they do not tightly couple mesh refinement with pose estimation, leaving room for further integration and performance improvements.

In this paper, we propose a monocular dense visual odometry system that optimizes camera poses and a triangular mesh representation in a single, unified framework. The method parameterizes each vertex by an exponential depth term (and a fixed viewing direction) to maintain a lower-dimensional search space and ensure physically valid depth values. This design yields dense reconstructions and accurate pose estimates,

even under challenging conditions, while keeping computational overhead moderate. The following sections present a detailed account of our formulation, system implementation, and experimental validation.

II. METHOD

Let $v \in \mathbb{R}^3$ be a vertex in 3D space. We decompose v into a direction (ray) and a depth as follows:

$$v = r \cdot d(r), \quad (1)$$

where $r \in \mathbb{R}^3$ is a ray emanating from the camera’s focal point and $d(r)$ is the corresponding depth. Concretely, we

$$\text{define } r = \pi(v) = \begin{pmatrix} v_1/v_3 \\ v_2/v_3 \\ 1 \end{pmatrix},$$

v_1 and v_2 are the x and y coordinates, so that v_3 corresponds to the depth of v . Hence, v is fully determined by

$$(r, d)$$

A. Projection Model

A 3D point v can be projected into a pixel location u of a camera with pose $P \in \text{SE}(3)$ ¹ via

$$u = K \pi(P \cdot v), \quad (2)$$

where $K \in \mathbb{R}^{3 \times 3}$ is the intrinsic calibration matrix, and $\pi(\cdot)$ represents the homogeneous projection to the image plane.

B. Mesh Representation

We define a triangle t to be a set of three vertices $\{v_a, v_b, v_c\}$. A mesh T is then the collection of N such triangles $\{t_1, \dots, t_N\}$, along with the set of M vertices $V = \{v_1, \dots, v_M\}$.

For an arbitrary ray r that lies inside a triangle t , we use barycentric coordinates $\lambda = (\lambda_a, \lambda_b, \lambda_c)$ to express r as a linear combination of the triangle’s vertex rays:

$$U \lambda = u, \quad (3)$$

where $U = [r_a \ r_b \ r_c]$ and r_a, r_b, r_c are the rays corresponding to the triangle’s vertices. If $\lambda_a, \lambda_b, \lambda_c \in [0, 1]$ and $\lambda_a + \lambda_b + \lambda_c = 1$, then $r = \lambda_a r_a + \lambda_b r_b + \lambda_c r_c$. This provides a way to interpolate vertex-related quantities (including depth) inside the triangle.

¹The Special Euclidean group, i.e. the space of rigid transformations in 3D.

C. Unknowns and Parametrization

We adopt the vertex decomposition of Equation (1) and fix r for each vertex to the values observed from a reference frame (defined below). Consequently, the unknowns in our system are: $\{P_f\}_{f \in F}$ and $\{d_m\}_{v_m \in V}$, where $P_f \in \text{SE}(3)$ is the pose of the f -th frame (or camera), and d_m is the depth of the m -th vertex. We enforce $d_m > 0$ by using

$$d_m = e^{p_m}, \quad (4)$$

so the actual optimization variables for each vertex are $\{p_m\}_{v_m \in V}$.

D. Objective Function

Let I_{kf} be a reference image (with pose $P_{kf} = I$) and let I_f be the image associated with pose P_f . We define a multi-scale photometric error (using a Huber norm $\|\cdot\|_h$) across all frames, triangles, and valid rays:

$$L(\{P_f\}, \{p_m\})^{\text{LoD}} = \sum_{f \in F} \sum_{t \in T} \sum_{r \in U} \|I_f^{\text{LoD}}(u_f(r(t))) - I_{kf}^{\text{LoD}}(u_{kf}(r(t)))\|_h \quad (5)$$

where $u_f(r)$ and $u_{kf}(r)$ follow the projection model of Equation (2), using P_f and P_{kf} respectively. The Level of Detail, superscript LoD, indicates that this loss is applied at multiple resolutions, including the original image and progressively downsampled versions.

E. Optimization

We aim to find $\{P_f\}_{f \in F}$ and $\{p_m\}_{v_m \in V}$ that minimize Equation (5). To do so, we use the Levenberg–Marquardt algorithm. At each iteration, we solve the linear system

$$H \delta = G \quad (6)$$

for the update vector δ , where

$$G = \sum_{f \in F} \sum_{t \in T} \sum_{r \in U} J(f, t, r) \cdot e(f, t, r) \quad (7)$$

$$H = \sum_{f \in F} \sum_{t \in T} \sum_{r \in U} J(f, t, r)^\top J(f, t, r).$$

Here, $e(f, t, r) = I_f(u_f(r)) - I_{kf}(u_{kf}(r))$ is the photometric residual, and J is its Jacobian with respect to the unknowns $\{P_f\}$ and $\{p_m\}$.

F. Jacobian Computation

The Jacobians with respect to p_m and P_f are derived using the chain rule.

For p_m ,

$$\frac{\partial I_f}{\partial p_m} = \frac{\partial I_f}{\partial u} \frac{\partial u}{\partial r_f} \frac{\partial r_f}{\partial d_{kf}} \frac{\partial d_{kf}}{\partial d_m} \frac{\partial d_m}{\partial p_m} \quad (8)$$

with

$$\frac{\partial u}{\partial r_f} = K \cdot \begin{bmatrix} 1/v_3 & 0 & -v_1/v_3^2 \\ 0 & 1/v_3 & -v_2/v_3^2 \\ 0 & 0 & 0 \end{bmatrix} \quad (9)$$

$$\frac{\partial r_f}{\partial d_{kf}} = R \cdot r_f \quad (10)$$

$$\frac{\partial d_m}{\partial p_m} = d_m \quad (11)$$

and, since the rays r corresponding to each $v \in V$ are fixed with respect to I_{kf}

$$\frac{\partial d_{kf}}{\partial d_n} = \lambda \quad (12)$$

The partial derivatives with respect to the camera poses are given by:

$$\frac{\partial I_f}{\partial P_f} = \frac{\partial I_f}{\partial u} \frac{\partial u}{\partial r_f} \frac{\partial r_f}{\partial P_f} \quad (13)$$

where $\frac{\partial r_f}{\partial P_f}$ follows from standard rigid-body transformation derivatives:

$$\frac{\partial I_f}{\partial P_f} = \frac{\partial I_f}{\partial u} \frac{\partial u}{\partial r_f} \frac{\partial r_f}{\partial P_f} \quad (14)$$

By combining (i) a mesh-based representation and (ii) a direct photometric error term, we can jointly optimize the camera poses $\{P_f\}$ and log-depth parameters $\{p_m\}$ for all vertices in V . This approach enables dense reconstruction by exploiting the inherent constraints of a mesh, while ensuring physically valid depths through exponential parametrization.

III. VISUAL ODOMETRY

Our visual odometry pipeline consists of map initialization, pose initialization, incremental pose refinement, keyframe management, and joint optimization. Below, we detail each component.

A. Map Initialization

To initialize the keyframe depth, we sample each depth value from a Gaussian distribution with mean 1 and standard deviation 0.5. Despite its simplicity, this approach is effective, as our robust optimization procedure rapidly refines the initial estimate.

B. Pose Initialization

When a new frame f arrives, we initialize its pose P_f based on the transformation from the previous two frames:

$$P_f = M_{f-1} P_{f-1}, \quad \text{where} \quad M_{f-1} = P_{f-1} P_{f-2}^{-1}$$

For the first frame in a sequence, we simply set $P_f = I$, the identity transformation.

C. Pose Estimation

Upon receiving each new frame, we refine its pose P_f by holding the current depth parameters $\{p_m\}$ fixed and solving the optimization system in Equation (6) where

$$J = \frac{\partial I_f}{\partial P_f}.$$

This process starts at a coarse LoD and progresses to finer resolutions, initializing each level using the final estimate from the previous level. This multi-scale strategy improves both speed and robustness.

D. New Keyframe Selection

A new keyframe is selected if either of the following conditions is met:

- 1) Over 20% of pixels in the current keyframe are unobserved.
- 2) The mean observation angle between the new frame f and the current keyframe f_{kf} surpasses $\pi/10$. This angle is computed as:

$$\frac{1}{N} \sum_{v \in V} ((v - T_{f_1})^\top (v - T_{f_2})), \quad (15)$$

where $f_1 = f$, $f_2 = f_{kf}$, and N is the number of vertices in V .

E. Frame Stack

Depending on the camera trajectory, some frames may have viewpoints very similar to the current keyframe or recently processed frames. Including these in optimization can result in near-degenerate configurations, degrading solution quality. To address this, we maintain a *frame stack* containing only “good” frames. A frame is considered “good” if its mean observation angle (Equation (15)) relative to the most recently added frame exceeds $\pi/128$. Additionally, we limit the stack size to F frames to control computational and memory costs.

F. Joint Optimization

Whenever a new keyframe is selected, we jointly optimize both the depth parameters $\{p_m\}$ and the poses $\{P_f\}$ of all frames in the stack (see Section II). This simultaneous refinement improves overall accuracy and ensures consistency in both geometric reconstruction and camera pose estimation.

IV. IMPLEMENTATION

Our system is implemented in C++, using the Eigen library for core linear algebra and Lie group operations. Image processing is handled by OpenCV, and Pangolin enables real-time visualization of the reconstructed mesh and camera poses.

A. Rasterizer

The computation of the loss function (5) and its derivatives ((8), (14), (7)) involves summing contributions over all triangles and their constituent rays:

$$\sum_{t \in T} \sum_{r \in U} (\dots).$$

This summation essentially serves as a *rasterization* process. One significant motivation for using a mesh-based representation is that rasterization is a standard operation for which most hardware platforms offer dedicated acceleration.

In anticipation of a future FPGA-based hardware implementation, we chose a pure C++ approach on the CPU. This design enables straightforward porting to High-Level Synthesis (HLS) tools later. Future work will investigate accelerating rasterization with GPUs and FPGAs, comparing throughput and energy consumption under various constraints.

B. Thread Management

Our system employs a multi-threaded design typical of SLAM frameworks. A *localization* thread processes incoming images, estimating the camera pose in real time as outlined in Section III. The *mapping* thread receives each new frame and its estimated pose from the localization thread. If the received frame meets the criteria in Section III, it is appended to the stack of “good” frames and possibly triggers a new keyframe for joint optimization. Finally, a *visualization* thread receives the optimized keyframes (along with their corresponding mesh) from the mapping thread and renders them using Pangolin, providing real-time feedback on the reconstructed scene and camera trajectory.

V. RESULTS

We evaluated our system qualitatively on three different datasets to assess its ability to estimate dense depth for every keyframe pixel and to track camera pose accurately.

A. Food-Court and Dining Hall Datasets

Our first two datasets, originally introduced in [6], depict a *food-court* and a *dining hall*, respectively. These scenes were recorded under favorable conditions (good lighting, smooth camera motion, and rich textures), enabling us to validate the core estimation techniques from Section II without additional complications such as extreme lighting changes or severe texture deficits.

Figures 1 and 2 show representative results for the food-court sequence. The keyframe images and their corresponding inverse-depth visualizations indicate that the system reconstructs most surfaces effectively. Minor artifacts appear where object boundaries (e.g., chairs and tables) do not align perfectly with the triangular edges in our mesh. However, as shown in Figure 2, these artifacts do not disrupt consistent pose estimation over time.

The dining hall dataset presents more challenging surfaces, including non-Lambertian materials on floors and tables. In addition, the ceiling and floor exhibit repetitive patterns with

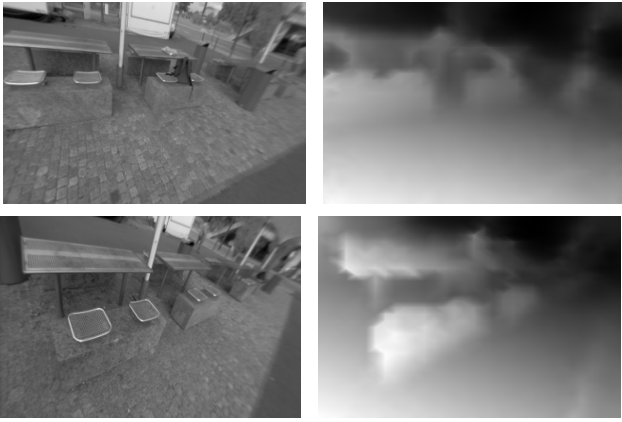


Fig. 1: Depth estimation results in the food-court dataset. Left: keyframe image. Right: estimated inverse depth

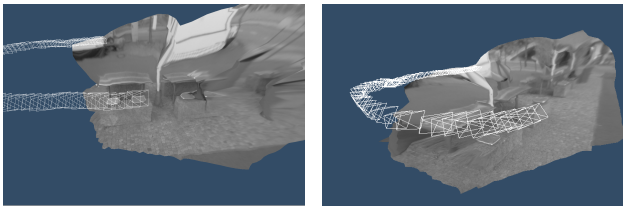


Fig. 2: Examples of maps from the food-court dataset. The keyframe frustum is shown in white, while the mesh and corresponding keyframe image are displayed in grayscale

low texture contrast, which can introduce local minima during optimization. Despite this, Figures 3 and 4 show that our method recovers mostly accurate depth maps, demonstrating the robustness of our mesh-based parameterization even under suboptimal texture conditions.

B. Challenging Outdoor Dataset

The third dataset, from [7], presents greater challenges due to varying lighting conditions and camera motion. Large



Fig. 3: Depth estimation results in the dining hall dataset. Left: keyframe image. Right: estimated inverse depth

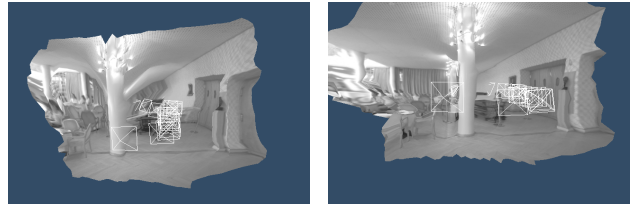


Fig. 4: Examples of maps from the dining hall dataset. The keyframe frustum is shown in white, while the mesh and corresponding keyframe image are displayed in grayscale

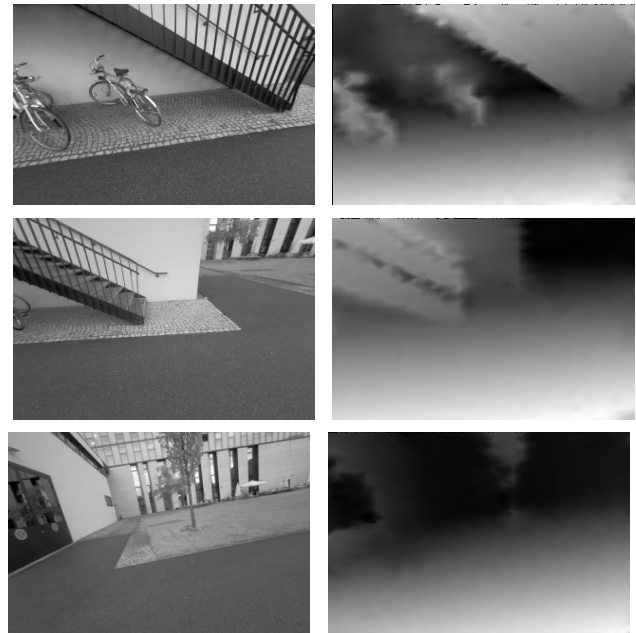


Fig. 5: Depth estimation results from the [7] dataset. Left: keyframe image. Right: estimated inverse depth

viewpoint changes and occasional poor illumination result in regions with weak or repetitive textures. As shown in Figures 5 and 6, our system handles these difficulties by producing plausible depth maps even on surfaces such as roads, which appear largely uniform but with subtle, high-frequency patterns. While these regions are difficult to optimize, the triangular mesh’s flexibility ensures accurate reconstruction of the scene structure.

Overall, these preliminary results suggest that our system consistently generates dense reconstructions and stable pose estimates in diverse environments, including those with smooth textures, repetitive patterns, and challenging illumi-

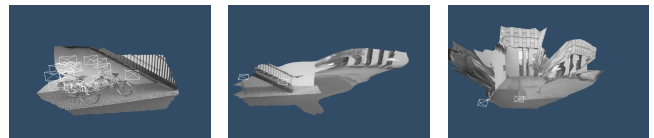


Fig. 6: Map examples from the [7] dataset

nation. The combination of mesh-based scene modeling and direct photometric optimization proves to be effective for a wide range of real-world conditions.

VI. CONCLUSIONS

In this work, we introduced a direct monocular visual odometry system that employs a triangular mesh as the primary scene representation. Unlike traditional approaches, our method encourages a closer coupling between geometry and pose through structured optimization over a continuous surface model. This not only enhances reconstruction quality in texture-scarce areas but also shifts the paradigm toward more semantically and physically meaningful mapping. However, this formulation introduces questions about how to manage scale, continuity across keyframes, and efficient data fusion as scenes grow. Mesh-based VO shows promise not only for dense mapping, but also as a stepping stone toward deeper scene understanding. Future work will focus on benchmarking the system on established datasets, analyzing its performance under varying lighting and motion conditions, and extending it to handle more complex scenarios (e.g., dynamic objects, multi-camera setups). Overall, our results highlight the benefits of a mesh representation for dense monocular SLAM, providing a strong foundation for high-fidelity, real-time 3D mapping and localization systems.



Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska Curie grant agreement No 101034383

REFERENCES

- [1] B. Al-Tawil, T. Hempel, A. Abdelrahman, and A. Al-Hamadi, "A review of visual SLAM for robotics: Evolution, properties, and future applications," *Frontiers in Robotics and AI*, vol. 11, 2024.
- [2] C. Campos, R. Elvira, J. Gomez, J. Montiel, and J. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [3] F. Hidalgo, C. Kahlefendt, and T. Braunl, "Monocular ORB-SLAM application in underwater scenarios," in *Proc. OCEANS - Kobe*, 2018, pp. 1–4, doi: 10.1109/OCEANSKOB.2018.8559435.
- [4] Y. Fang, G. Shan, X. Li, W. Liu, T. Wang, and H. Snoussi, "HE-SLAM: A stereo SLAM system based on histogram equalization and ORB features," in *Proc. Chinese Automation Congress (CAC)*, 2019, doi: 10.1109/CAC.2018.8623424.
- [5] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2011, pp. 2320–2327, doi: 10.1109/ICCV.2011.6126513.
- [6] V. Usenko, J. Engel, J. Stueckler, and D. Cremers, "Reconstructing street-scenes in real-time from a driving car," in *Proc. Int. Conf. 3D Vision (3DV)*, 2015.
- [7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

- [8] 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. (ICCV)*, 2019.
- [9] T. Mukasa, J. Xu, and S. Bjorn, "3D scene mesh from CNN depth predictions and sparse monocular SLAM," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, 2017.
- [10] A. Rosinol, J. J. Leonard, and L. Carlone, "Probabilistic volumetric fusion for dense monocular SLAM," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, 2023, pp. 3097–3105.