R3D3: Dense 3D Reconstruction of Dynamic Scenes from Multiple Cameras

Aron Schmied[1*] Tobias Fischer[1*] Martin Danelljan[1] Marc Pollefeys[1,2] Fisher Yu[1]
[1] ETH Zürich [2] Microsoft
https://www.vis.xyz/pub/r3d3/

Abstract

Dense 3D reconstruction and ego-motion estimation are key challenges in autonomous driving and robotics. Compared to the complex, multi-modal systems deployed today, multi-camera systems provide a simpler, low-cost alternative. However, camera-based 3D reconstruction of complex dynamic scenes has proven extremely difficult, as existing solutions often produce incomplete or incoherent results. We propose R3D3, a multi-camera system for dense 3D reconstruction and ego-motion estimation. Our approach iterates between geometric estimation that exploits spatial-temporal information from multiple cameras, and monocular depth refinement. We integrate multi-camera feature correlation and dense bundle adjustment operators that yield robust geometric depth and pose estimates. To improve reconstruction where geometric depth is unreliable, e.g. *for moving objects or low-textured regions, we introduce learnable scene priors via a depth refinement network. We show that this design enables a dense, consistent 3D reconstruction of challenging, dynamic outdoor environments. Consequently, we achieve state-of-the-art dense depth prediction on the DDAD and NuScenes benchmarks.*

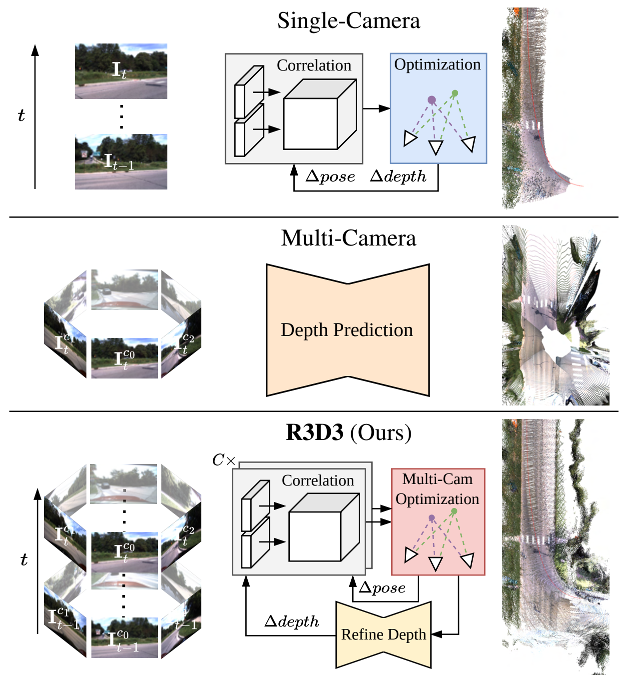

Figure 1. Many methods use temporal context but neglect inter-camera information, leading to incomplete results (**top**). Other works focus on exploiting inter-camera context but neglect temporal information, yielding incoherent predictions (**middle**). In contrast, our method achieves a consistent, dense 3D reconstruction by iteratively integrating geometric depth estimation from multiple cameras with monocular depth refinement (**bottom**).

1. Introduction

Translating sensory inputs into a dense 3D reconstruction of the environment and tracking the position of the observer is a cornerstone of robotics and fundamental to the development of autonomous vehicles (AVs). Contemporary systems rely on fusing many sensor modalities like camera, LiDAR, RADAR, IMU and more, making hardware and software stacks complex and expensive. In contrast, multi-camera systems provide a simpler, low-cost alternative already widely available in modern consumer vehicles. However, image-based dense 3D reconstruction and ego-motion estimation of large-scale, dynamic scenes is an open research problem as moving objects, uniform and repetitive textures, and optical degradations pose significant algorithmic challenges.

Existing works approaching the aforementioned task can be divided into two lines of research. Many methods have focused on recovering 3D scene structure via structure-from-motion (SfM). In particular, simultaneous localization and mapping (SLAM) methods focus on accurate ego-motion estimation and usually only recover sparse 3D structure [9, 29, 34, 12, 11]. They typically treat dynamic objects or uniformly textured regions as outliers yielding an incomplete 3D reconstruction result, which makes them less suitable for AVs and robotics. In addition, only a few works have focused on multi-camera setups [24, 8, 33, 30, 26]. In contrast, multi-view stereo (MVS) methods [31, 32, 41, 28,

*Equal contribution.

57, 27, 17] aim to estimate dense 3D geometry but focus on static scenes and highly overlapping sets of images with known poses.

The second line of research focuses on dense depth prediction from monocular cues, such as perspective object appearance and scene context [65, 15, 16, 59, 49, 61, 19]. However, due to the injective property of projecting 3D structures onto a 2D plane, reconstructing depth from a single image is an ill-posed problem, which limits the accuracy and generalization of these methods. Recent methods [51, 18, 21], inspired by MVS literature, combine monocular cues with temporal context but are focused on front-facing, single-camera setups. A handful of recent works extend monocular depth estimation to multi-camera setups [22, 52, 54]. These methods utilize the spatial context to improve accuracy and realize absolute scale depth learning. However, those works neglect the temporal domain which provides useful cues for depth estimation.

Motivated by this observation, we introduce R3D3, a system for dense 3D reconstruction and ego-motion estimation from multiple cameras of dynamic outdoor environments. Our approach combines monocular cues with geometric depth estimates from both spatial inter-camera context as well as inter- and intra-camera temporal context. We compute accurate geometric depth and pose estimates via iterative dense correspondence on frames in a co-visibility graph. For this, we extend the dense bundle adjustment (DBA) operator in [48] to multi-camera setups, increasing robustness and recovering absolute scene scale. To determine co-visible frames across cameras, we propose a simple yet effective multi-camera algorithm that balances performance and efficiency. A depth refinement network takes geometric depth and uncertainty as input and produces a refined depth that improves the reconstruction of, *e.g.*, moving objects and uniformly textured areas. We train this network on real-world driving data without requiring any LiDAR ground-truth. Finally, the refined depth estimates serve as the basis for the next iterations of geometric estimation, thus closing the loop between incremental geometric reconstruction and monocular depth estimation.

We summarize our **contributions** as follows. **1)** we propose R3D3, a system for dense 3D reconstruction and ego-motion estimation in dynamic scenes, **2)** we estimate geometric depth and poses with a novel multi-camera DBA formulation and a multi-camera co-visibility graph, **3)** we integrate prior geometric depth and uncertainty with monocular cues via a depth refinement network.

As a result, we achieve state-of-the-art performance across two widely used multi-camera depth estimation benchmarks, namely DDAD [19] and NuScenes [3]. Further, we show that our system exhibits superior accuracy and robustness compared to monocular SLAM methods [48, 5].

2. Related Work

Multi-view stereo. MVS methods aim to recover dense 3D scene structure from a set of images with known poses. While earlier works focused on classical optimization [31, 41, 4, 13, 14, 62, 32], more recent works capitalize on the success of convolutional neural networks (CNNs). They use CNNs to estimate features that are matched across multiple depth-hypothesis planes in a 3D cost volume [28, 57, 64, 27, 17, 58, 60]. While early approaches adopt multiple cost volumes across image pairs [64], recent approaches use a single cost volume across the whole image set [57]. These works assume a controlled setting with many, highly overlapping images and known poses to create a 3D cost volume. Instead, we aim to achieve robust, dense 3D reconstruction from an arbitrary multi-camera setup on a moving platform with an unknown trajectory.

Visual SLAM. Visual SLAM approaches focus on jointly mapping the environment and tracking the trajectory of the observer from visual inputs, *i.e*. one or multiple RGB cameras. Traditional SLAM systems are often split into different stages, where first images are processed into keypoint matches, which subsequently are used to estimate the 3D scene geometry and camera trajectory [9, 10, 6, 34, 35, 5, 39]. Another line of work focuses on directly optimizing 3D geometry and camera trajectory based on pixel intensities [12, 11, 56, 55]. A handful of works have focused on multi-camera SLAM systems [24, 8, 33, 30, 26]. Recent methods integrate CNN-based depth and pose predictions [45, 56, 55] into the SLAM pipeline. The common challenge these methods face is outliers in the pixel correspondences caused by the presence of low-textured areas, dynamic objects, or optical degradations. Hence, robust estimation techniques are used to filter these outliers, yielding an incomplete, sparse 3D reconstruction result.

In contrast, dense 3D reconstruction and ego-motion estimation has proven to be more challenging. Early works utilize active depth sensors [36, 64] to alleviate the abovementioned challenges. Recently, several works on dense visual SLAM from RGB input have emerged [1, 44, 7, 46, 48]. While these methods are able to produce high-quality depth maps and camera trajectories, they inherit the limitations of their traditional counterparts, *i.e*. their predictions are subject to noise when there are outliers in the pixel correspondences. This can lead to artifacts in the depth maps, inaccurate trajectories, or even a complete failure of the system. On the contrary, we achieve robust, dense 3D reconstruction and ego-motion estimates by jointly leveraging multi-camera constraints as well as monocular depth cues.

Self-supervised depth estimation. The pioneering work of Zhou *et al*. [65] learns depth estimation as a proxy task while minimizing a view synthesis loss that uses geometric constraints to warp color information from a reference to a target view. Subsequent research has focused on improving network architectures, loss regularization, and training schemes [15, 16, 20, 59, 49, 61, 19]. Recent methods draw inspiration from MVS and propose to use 3D cost volumes in order to incorporate temporal information [51, 18, 53, 21]. While these methods achieve promising results, they still focus on single-camera, front-facing scenarios that do not reflect the real-world sensor setups of AVs [3, 43, 19]. Another recent line of work focuses on exploiting spatial information across overlapping cameras in a multi-camera setup [22, 52, 54]. These works propose to use static feature matches across cameras at a given time to guide depth learning and estimation, and to recover absolute scale. Our work aims to exploit both spatial and temporal cues in order to achieve a robust, dense 3D reconstruction from multiple cameras in dynamic outdoor environments.

3. Method

Problem formulation. At each timestep $t \in T$ we are given a set of images $\{\mathbf{I}_t^c\}_{c=1}^C$ from C cameras that are mounted on a moving platform with known camera intrinsics $\mathbf{K}_c$ and extrinsics $\mathbf{T}_c$ with respect to a common reference frame. We aim to estimate the depth maps $\mathbf{d}_t^c \in \mathbb{R}_+^{H \times W}$ and ego-pose $\mathbf{P}_t \in SE(3)$ at the current time step.

Overview. Our system is composed of three stages. First, given a new set of images $\{\mathbf{I}_t^c\}_{c=1}^C$, we extract deep features from each $\mathbf{I}_t^c$. We maintain a co-visibility graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where a frame $\mathbf{I}_t^c$ represents a node $v \in \mathcal{V}$ and co-visible frames are connected with edges. For each edge $(i, j) \in \mathcal{E}$ in this graph, we compute the feature correlation $\mathbf{C}_{ij}$ of the two adjacent frames. Second, given initial estimates of the depth $\mathbf{d}_i$ and the relative pose $\mathbf{G}_{ij} = \left(\mathbf{P}_{t_j}\mathbf{T}_{c_j}\right)^{-1}\mathbf{P}_{t_i}\mathbf{T}_{c_i}$ between the two frames, we compute the induced flow $\mathbf{f}_{ij}^*$. We correct the induced flow with a recurrent update operator using the feature correlations $\mathbf{C}_{ij}$. We then globally align the updated flow estimates at each edge (i, j) with the current estimates of $\mathbf{G}_{ij}$ and $\mathbf{d}_i$ across the co-visibility graph $\mathcal{G}$ with our proposed multi-camera DBA. Third, we introduce a depth refinement network that refines the geometric depth estimates with monocular depth cues to better handle scenarios that are challenging for geometric estimation methods. We illustrate our architecture in Fig. 2.

3.1. Feature Extraction and Correlation

Given a new observation $\{\mathbf{I}_t^c\}_{c=1}^C$, we first extract deep image features, update the co-visibility graph, and compute feature correlations. We describe these steps next.

Feature extraction. We follow [47, 48] and extract both correlation and context features from each image individually via deep correlation and context encoders g_ϕ and g_ψ.

Co-visibility graph. We store correlation and context features in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node corresponds to an image $\mathbf{I}_t^c$ and an edge in the graph indicates that two images are considered a pair in the feature correlation and bundle adjustment steps. Contrary to [48], we construct $\mathcal{G}$ with three kinds of edges: temporal, spatial, and spatial-temporal. Since the number of possible edges is $|\mathcal{V}|^2$ and the system runtime scales linearly with $|\mathcal{E}|$, the degree of sparsity in the co-visibility graph is crucial. We thus carefully design a simple yet effective co-visibility graph construction algorithm in multi-camera setups (see Fig. 3).

For temporal edges (Fig. 3 in green), we examine the time window $t - \Delta t_{\text{intra}}$ and connect all frames of the same camera that are less than r_{intra} time steps apart. For spatial and spatial-temporal edges we exploit three priors. 1) we know camera adjacency from the given calibration, 2) we assume motion in the direction of the forward-facing camera and 3) the observer undergoes limited motion within the local time window. We establish spatial edges (Fig. 3 in red) between adjacent cameras within time step t. We further establish spatial-temporal edges (Fig. 3 in orange) given the static camera adjacency within $t - \Delta t_{\text{inter}}$. In particular, we connect camera c_i and c_j from any t' to $t' - r_{\text{inter}}$, if c_i and c_j connect within t and if c_j is closer to the forward-facing camera than c_i under forward motion assumption. Finally, we remove an edge (i, j) if node i or j is outside the local time windows $t - \Delta t_{\text{inter}}$ or $t - \Delta t_{\text{intra}}$. We remove unconnected nodes from the graph. We provide pseudo-code and additional discussion in the supplemental material.

Feature correlation. For each edge $(i, j) \in \mathcal{E}$ we compute the 4D feature correlation volume via the dot-product

$$\mathbf{C}_{ij} = \langle g_\phi(\mathbf{I}_i), g_\phi(\mathbf{I}_j) \rangle \in \mathbb{R}^{H \times W \times H \times W}. \quad (1)$$

As in [47, 48], we constrain the correlation search region to a radius r with a lookup operator

$$L_r : \mathbb{R}^{H \times W \times H \times W} \times \mathbb{R}^{H \times W \times 2} \rightarrow \mathbb{R}^{H \times W \times (r+1)^2}, \quad (2)$$

that samples a $H \times W$ grid of coordinates from the correlation volume using bilinear interpolation.

3.2. Depth and Pose Estimation

Given the correlation volume $\mathbf{C}_{ij}$, we here describe how to estimate the relative pose $\mathbf{G}_{ij}$ and depth $\mathbf{d}_i$ for each edge $(i, j) \in \mathcal{E}$ in the co-visibility graph.

Flow correction. Given an initial estimate of $\mathbf{G}_{ij}$ and $\mathbf{d}_i$, we first compute the induced flow $\mathbf{f}_{ij}^*$ to sample the correlation volume $\mathbf{C}_{ij}$ using (2). We then feed the sampled correlation features, the context features $g_\psi(\mathbf{I}_i)$, and the induced flow $\mathbf{f}_{ij}^*$ into a convolutional GRU. The GRU predicts a flow residual $\mathbf{r}_{ij}$ and confidence weights $\mathbf{w}_{ij} \in \mathbb{R}^{H \times W \times 2}$ as in [48]. A challenge of correspondence estimation in a multi-camera setup is that the flow magnitude varies greatly among edges in the co-visibility graph due to inter-camera edges with large relative rotation. To this end, we propose to subtract the rotational part of $\mathbf{f}_{ij}^*$ in those edges via

$$\mathbf{f}'_{ij} = (\mathbf{x}_i + \mathbf{f}_{ij}^*) - \Pi_{c_j}(\mathbf{R}_{c_j}^{-1}\mathbf{R}_{c_i} \circ \Pi_{c_i}^{-1}(\mathbf{1})), \quad (3)$$

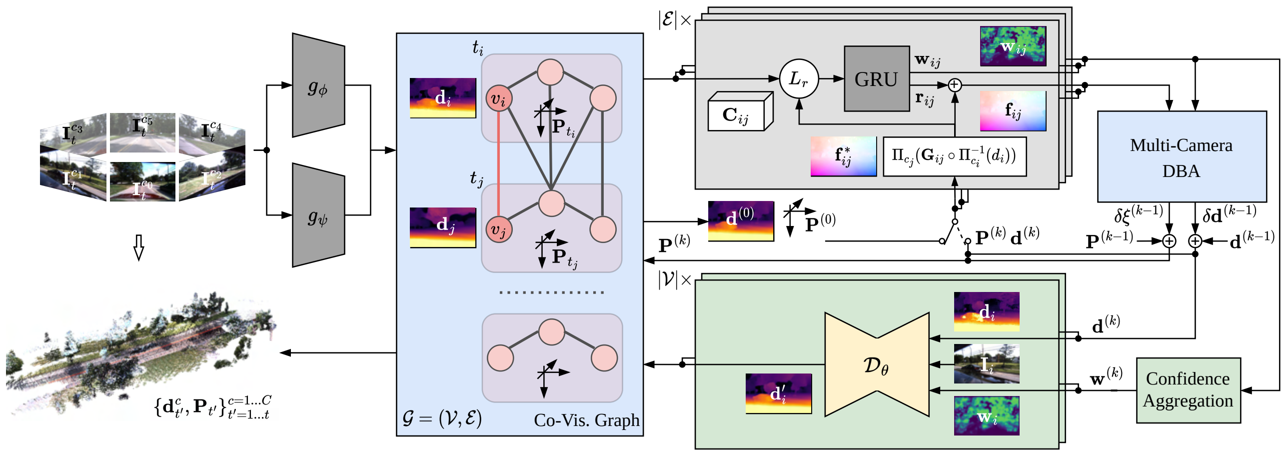

Figure 2. **Method overview.** First, frames $\{\mathbf{I}_t^c\}_{c=1}^C$ from C cameras at time t are encoded and integrated into the co-visibility graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with initial guesses of depth maps $\mathbf{d}_t^c$ and ego-pose $\mathbf{P}_t$ (Sec. 3.1). Second, for each edge $(i, j) \in \mathcal{E}$, we compute the induced flow $\mathbf{f}_{ij}^*$ from $\mathbf{d}_i$ and relative camera pose $\mathbf{G}_{ij}$ derived from ego-poses $\mathbf{P}$ and camera extrinsics $\mathbf{T}$. Given $\mathbf{f}_{ij}^*$, we pool feature correlations from $\mathbf{C}_{ij}$ with operator L_r as input to a GRU that estimates flow update $\mathbf{r}_{ij}$ and confidence $\mathbf{w}_{ij}$. We globally align depths $\mathbf{d}$ and poses $\mathbf{P}$ with the new flow estimates $\mathbf{f}$ via our multi-camera DBA operator in k iterations (Sec. 3.2). Finally, for each node $i \in \mathcal{V}$, we refine the initial geometric estimate $\mathbf{d}_i$ given the aggregated confidence $\mathbf{w}_i$ via $\mathcal{D}_\theta$ (Sec. 3.3). We highlight our contributions in color.

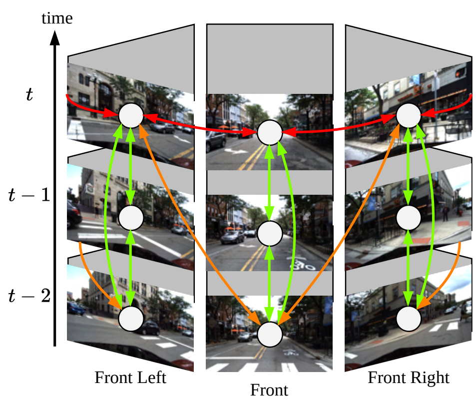

Figure 3. **Co-visibility graph.** We illustrate an example of the connectivity pattern of the co-visibility graph using our multi-camera graph construction algorithm. We use $\Delta t_{\text{intra}} = 3$, $\Delta t_{\text{inter}} = 2$, $r_{\text{intra}} = 2$, $r_{\text{inter}} = 2$ with six cameras. We depict temporal, spatial, and spatial-temporal edges.

where $\mathbf{x}_i$ are pixel coordinates of frame i, Π_c denotes the projection operator under camera c and $\mathbf{R}$ is the rotation of transformation $\mathbf{T} = (\mathbf{R}\,\mathbf{t})$. This aligns the flow magnitude with temporal, intra-camera edges. Note that, while we input $\mathbf{f}'_{ij}$ instead of $\mathbf{f}^*_{ij}$ to the GRU, we still sample the correlation volume $\mathbf{C}_{ij}$ based on $\mathbf{f}^*_{ij}$.

Pose and depth correction. We update our pose and depth estimates given the updated flow $\mathbf{f}_{ij} = \mathbf{f}^*_{ij} + \mathbf{r}_{ij}$ by minimizing the re-projection error, defined as

$$E = \sum_{(i,j)\in\mathcal{E}} \left\| (\mathbf{x}_i + \mathbf{f}_{ij}) - \Pi_{c_j}(\mathbf{G}_{ij} \circ \Pi_{c_i}^{-1}(\mathbf{d}_i)) \right\|^2_{\Sigma_{ij}} , \quad (4)$$

where $\|.\|_{\Sigma_{ij}}$ is the Mahalanobis norm and $\Sigma_{ij} = \text{diag}(\mathbf{w}_{ij})$. Thus, only matched pixels where the confidence $\mathbf{w}_{ij}$ is greater than zero contribute to the total cost. We extend the dense bundle adjustment (DBA) proposed in [48] to include the known extrinsics of the multi-camera setup. In particular, we decompose the relative poses between two frames $\mathbf{G}_{ij}$ into the unknown, time-varying poses $\mathbf{P}$ in the reference frame and the known static relative transformations $\mathbf{T}$ between the cameras and the reference frame via

$$\mathbf{G}_{ij} = \left(\mathbf{P}_{t_j}\mathbf{T}_{c_j}\right)^{-1}\mathbf{P}_{t_i}\mathbf{T}_{c_i}. \quad (5)$$

We linearize the residual of the energy function in Eq. 4 with a first-order Taylor expansion. We use Eq. 5 and treat $\mathbf{T}_{c_j}$ and $\mathbf{T}_{c_i}$ as constants when calculating the Jacobians, thus computing updates only for ego-pose $\mathbf{P}$. We apply the Gauss-Newton step to compute the updated $\mathbf{P}^{(k)} = \exp(\delta\xi^{(k-1)})\mathbf{P}^{(k-1)}$ and depth $\mathbf{d}^{(k)} = \delta\mathbf{d}^{(k-1)} + \mathbf{d}^{(k-1)}$ with $\delta\xi^{(k-1)} \in \mathfrak{se}(3)$. We provide a detailed derivation in the supplementary material.

Our formulation has two advantages over DBA [48]. First, the optimization is more robust since C frames are connected through a single ego-pose $\mathbf{P}$. If one or multiple frames have a high outlier ratio in feature matching, *e.g.* through lens-flare, low textured regions, or dynamic objects, we can still reliably estimate $\mathbf{P}$ with at least one undisturbed camera. Second, if there are not only pixels matched across the temporal context but also pixels matched across the static, spatial context, we can recover the absolute scale of the scene. This is because for static, spatial matches the relative transformation $\mathbf{G}_{ij} = \mathbf{T}_{c_j}^{-1}\mathbf{T}_{c_i}$ is known to scale so that E is minimized if and only if $\mathbf{d}$ is in absolute scale.

Training. We train the networks g_ϕ and g_ψ as well as the flow correction GRU on dynamic scenes following the procedure in [48]. As shown in [48], the geometric features learned from synthetic data can generalize to real-world scenes. We can therefore leverage existing synthetic driving datasets without relying on real-world ground-truth measurements from sensors like LiDAR or IMU to adjust our method to dynamic, multi-camera scenarios.

3.3. Depth Refinement

SfM relies on three assumptions: accurate pixel matches, sufficient camera movement, and a static scene. These assumptions do not always hold in the real world due to, *e.g.*, low-textured areas, a static ego vehicle, or many dynamic agents. Still, we would like the system to produce reasonable scene geometry to ensure safe operation.

On the contrary, monocular depth cues are inherently not affected by these issues. However, they usually lack the accurate geometric detail of SfM methods. Hence, for each node $i \in \mathcal{V}$, we complement the accurate, but sparse geometric depth estimates with monocular cues.

Network design. We use a CNN $\mathcal{D}_\theta$ parameterized by θ. We input the depth $\mathbf{d}_i$, confidence $\mathbf{w}_i$, and the corresponding image $\mathbf{I}_i$. The network predicts an improved, dense depth $\mathbf{d}'_i = \mathcal{D}_\theta(\mathbf{I}_i, \mathbf{d}_i, \mathbf{w}_i)$. We obtain the per-frame depth confidence $\mathbf{w}_i$ for frame i by using the maximum across the per-edge confidence weights $\mathbf{w}_{ij}$. We compute $\mathbf{w}_i = \max_j \frac{1}{2}(\mathbf{w}^x_{ij} + \mathbf{w}^y_{ij})$ where x and y are the flow directions. We use $\max(\cdot)$ because we observe that depth triangulation will be accurate if at least one pixel is matched with high confidence. We sparsify input depth and confidence weights by setting regions with confidence $\mathbf{w}_i < \beta$ to zero. We concatenate these with the image $\mathbf{I}_i$. We further concatenate depth and confidence with features at 1/8$^{\text{th}}$ scale. As in [16], the output depth is predicted at four scales. To accommodate different focal lengths among cameras in the sensor setup, we do focal length scaling of the output [45].

Training. Contrary to geometric approaches, monocular depth estimators infer depth from semantic cues, which makes it hard for them to generalize across domains [23]. Hence, instead of relying on synthetic data, we train $\mathcal{D}_\theta$ on raw, real-world video in a self-supervised manner by minimizing a view synthesis loss [65]. We minimize the photometric error $\mathrm{pe}(\mathbf{I}^c_t, \mathbf{I}^{c' \to c}_{t' \to t})$ between a target image $\mathbf{I}^c_t$ and a reference image $\mathbf{I}^{c'}_{t'}$ warped to the target viewpoint via

$$\mathbf{I}^{c' \to c}_{t' \to t} = \Phi(\mathbf{I}^{c'}_{t'}; \Pi_{c'}(\mathbf{G}_{(t,c)(t',c')} \circ \Pi^{-1}_c(\mathbf{d}^c_t)))\,, \qquad (6)$$

with Φ the bi-linear sampling function, $\mathbf{d}^c_t$ the predicted depth, $c' = c \pm 1$ and $t' = t \pm 1$. We compute photometric similarity with SSIM [50] and L_1 distances. We use $\mathbf{G}_{(t,c)(t',c')} = (\mathbf{P}_{t'}\mathbf{T}_{c'})^{-1}\mathbf{P}_t\mathbf{T}_c$ generated by the first two stages of our system (see Sec. 3.2). Further, self-supervised depth estimation is well-studied, and we follow the common practice of applying regularization techniques to filter the photometric error [16, 63, 20]. First, we mask areas where $\mathbf{M}^{\text{st}} = \left[\mathrm{pe}(\mathbf{I}^c_t, \mathbf{I}^{c' \to c}_{t' \to t}) < \mathrm{pe}(\mathbf{I}^c_t, \mathbf{I}^{c'}_{t'})\right]$, *i.e.* we filter regions where assuming a stationary scene would induce a lower loss than re-projection. Second, we compute an induced flow consistency mask from $\mathbf{f}^*$,

$$\mathbf{M}^{\text{fc}} = \left[\left\| \mathbf{f}^*_{(t,c)(t',c')} + \Phi(\mathbf{f}^*_{(t',c')(t,c)}; \mathbf{x} + \mathbf{f}^*_{(t,c)(t',c')}) \right\|_2 < \gamma\right]. \qquad (7)$$

This term warps pixel coordinates $\mathbf{x}$ from target to reference view, looks up their value in $\mathbf{f}^*_{(t',c')(t,c)}$, and compares them to $\mathbf{f}^*_{(t,c)(t',c')}$. Therefore, pixels not following the epipolar constraint like dynamic objects are masked. Third, to handle the self-occlusion of the ego-vehicle, we manually draw a static mask $\mathbf{M}^{\text{oc}}$. We can use a single mask for all frames since the cameras are mounted rigidly. Further, to handle occlusions due to change in perspective, we take the minimum loss $\min_{t',c'} \mathrm{pe}(\mathbf{I}^c_t, \mathbf{I}^{c' \to c}_{t' \to t})$ across all reference views. The overall loss for $\mathcal{D}_\theta$ is thus

$$\mathcal{L} = \min_{t',c'} \left(\mathbf{M}^{\text{st}}\mathbf{M}^{\text{fc}}\mathbf{M}^{\text{oc}} \cdot \mathrm{pe}(\mathbf{I}^c_t, \mathbf{I}^{c' \to c}_{t' \to t}) \right) + \lambda \mathcal{L}_{\text{smooth}}\,, \quad (8)$$

with $\mathcal{L}_{\text{smooth}}$ a spatial smoothness regularization term [15].

3.4. Inference Procedure

Given an input stream of multi-camera image sets, we perform incremental 3D reconstruction in three phases.

Warmup phase. First, we add frames to the co-visibility graph if a single GRU step of a reference camera yields a large enough mean optical flow until we reach n_{warmup} frames. At this stage, we infer depth with our refinement network $\mathbf{d}^c_t = \mathcal{D}_\theta(\mathbf{I}^c_t, \mathbf{0}, \mathbf{0})$. We write the depth into the co-visibility graph so that it can be used during initialization.

Initialization phase. We initialize the co-visibility graph with the available frames and run $n_{\text{itr-wm}}$ GRU and bundle adjustment iterations. For the first $n_{\text{itr-wm}}/2$ iterations, we fix depths in the bundle adjustment and only optimize poses, which helps with inferring the absolute scale of the scene.

Active phase. We now add each incoming frame to the co-visibility graph. We initialize the new frames with $\mathbf{d}^c_t = \mathrm{mean}(\mathbf{d}^c_{t-4:t-1})$ and $\mathbf{P}_t = \mathbf{P}_{t-1}$. Then, we perform n_{iter1} GRU and bundle adjustment updates, and apply the refinement network to all updated frames. If the mean optical flow of the reference camera between t and $t-1$ is low, we remove all frames at $t-1$ from the co-visibility graph. This helps to handle situations with little to no ego-motion. In case no frame is removed, we do another n_{iter2} iterations. Finally, we write the updated depths and poses into the co-visibility graph.

4. Experiments

4.1. Experimental Setup

We perform extensive experiments on large-scale driving datasets that contain recordings from vehicles equipped

Table 1. **Method ablation.** We ablate our method components on the DDAD [19] dataset. We examine the influence of each component given the geometric estimation as in [48] with naive DBA as the baseline. Further, we show the influence of VKITTI [2] fine-tuning on the geometric estimation. We enumerate each variant for better reference. * median scaled depth, ** scaled trajectory.

No.	Geom. Est.	VKITTI	Multi-Cam DBA	Refinement	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta_{1.25}$ ↑	ATE [m] ↓	ATE** [m] ↓
1a*	✓	✓			0.438	26.97	19.109	0.636	-	2.134
1b	✓		✓		0.442	36.39	18.664	0.736	1.672	0.435
1c	✓	✓	✓		0.320	15.45	16.303	0.727	2.356	0.922
2a				✓	0.211	3.806	12.668	0.715	-	-
3a	✓	✓	✓	✓	**0.162**	**3.019**	**11.408**	**0.811**	**1.235**	**0.433**

Table 2. **Co-visibility graph ablation.** We compare our co-visibility graph construction algorithm to the original algorithm in [48] on the DDAD [19] dataset. We observe that our algorithm improves the overall runtime by an order of magnitude while maintaining the same level of performance.

Graph	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta_{1.25}$ ↑	ATE [m] ↓	t_{inf} [s] ↓
Original	**0.162**	**2.968**	**11.156**	**0.816**	**0.982**	3.23
Ours	**0.162**	3.019	11.408	0.811	1.235	**0.35**

Table 3. **Refinement network ablation.** We show on the DDAD [19] dataset that both adding confidence weights $\mathbf{w}_i$ and sparsifying the depth input via $\mathbf{w}_i < \beta$ to filter outliers in the input of depth refinement network $\mathcal{D}_\theta$ is vital to depth accuracy.

$\mathbf{w}_i$	$\mathbf{w}_i < \beta$	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta_{1.25}$ ↑
		0.181	4.078	12.108	0.789
✓		0.173	3.497	11.924	0.792
✓	✓	**0.162**	**3.019**	**11.408**	**0.811**

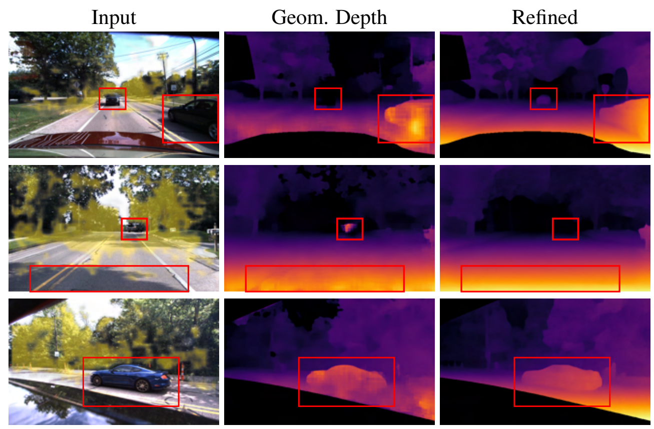

Figure 4. **Depth refinement.** We plot images overlayed with confidence $\mathbf{w}_i$, the geometric depth estimate $\mathbf{d}$, and the refined depth $\mathbf{d}'$ from $\mathcal{D}_\theta$. We observe a smoother ground plane, corrected dynamic objects, and filtered outliers.

with multiple cameras, LiDARs, and other sensors. Note that we only use camera data for training and inference, and use LiDAR data only as ground-truth for evaluation. We evaluate our method in inference mode (cf. Sec. 3.4), *i.e.* we obtain predictions in a fully online (causal) manner to mimic real-world deployment.

DDAD. This dataset consists of 150 training and 50 validation scenes from urban areas. Each sequence has a length of 50 or 100 time steps at a frame rate of 10Hz. The sensor setup includes six cameras in a surround-view setup with up to 20% overlap. We follow [19] and evaluate depth up to 200m averaged across all cameras. We use self-occlusion masks from [52] to filter the ego-vehicle in the images. Images have a resolution of 1216×1936 and are downsampled to 384×640 in our experiments.

NuScenes. This dataset contains 700 training, 150 validation, and 150 testing sequences of urban scenes with challenging conditions such as nighttime and rain. Each scene is composed of 40 keyframes at 2Hz synchronized across sensors. The six cameras have a sampling rate of 12Hz and are arranged in a surround-view setup with up to 10% overlap. We follow [22] and evaluate depth averaged across all cameras on the validation set. We use a single self-occlusion mask for all scenes. While the raw images have a resolution of 900×1600, we use 768×448.

Implementation details. We implement our system in PyTorch [37]. Our multi-camera DBA is implemented in CUDA. We fine-tune g_ϕ, g_ψ and the flow correction GRU on VIKITTI2 [2] for 10 epochs with a batch size of 1, sequence length of 7, and learning rate 10^{-4} with the pre-trained model in [48]. For $\mathcal{D}_\theta$, we use a U-Net [38] with ResNet18 [25] encoder and initialize it with ImageNet [40] pre-trained weights. We use $\beta = 0.5$, $\gamma = 3$ and $\lambda = 10^{-3}$. We train for 20 epochs with batch size of 6 and the Adam optimizer with learning rate 10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We reduce the learning rate by a factor of 10 after 15 epochs. For inference, the runtime is measured for a set of six images on a single RTX3090 GPU. Each sequence is initialized with 3 warmup frames filtered at a mean flow threshold of 1.75. We build the co-visibility graph with $\Delta t_{\text{intra}} = 3$, $r_{\text{intra}} = 2$, $\Delta t_{\text{inter}} = 2$ and $r_{\text{inter}} = 2$.

4.2. Ablation Studies

Method ablation. We ablate our method components on the DDAD dataset in Tab. 1. The geometric estimation baseline (1a) consists of [48] with the naive DBA formulation applied to all cameras and the co-visibility graph as described in Sec. 3.1. It performs poorly in depth and pose estimation, with an Abs. Rel. score of 0.438 and an absolute trajectory error (ATE) [42] of 2.134m, even when adjusting for scale. Next, we add our multi-camera DBA (1b). We observe that, even without VKITTI fine-tuning, the pose esti-

Table 4. **Comparison to state-of-the-art on DDAD**. We compare favorably to existing methods on both scale-aware and scale-invariant depth prediction. * median scaled depth.

Method	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta_{1.25}$ ↑
FSM [22]	0.201	-	-	-
SurroundDepth [52]	0.208	3.371	12.977	0.693
Ours	**0.162**	**3.019**	**11.408**	**0.811**
FSM* [22]	0.202	-	-	-
SurroundDepth* [52]	0.200	3.392	12.270	0.740
MCDP* [54]	0.193	3.111	12.264	**0.811**
Ours*	**0.169**	**3.041**	**11.372**	0.809

Table 5. **Per-camera evaluation on DDAD.** We show a per-camera breakdown of previous works, our $\mathcal{D}_\theta$ only (cf. 2a in Tab. 1), and our full method. Our full method performs the best with a particularly significant improvement on the side-view cameras while $\mathcal{D}_\theta$ performs similarly to previous works.

	Abs Rel ↓					
Method	*Front*	*F.Left*	*F.Right*	*B.Left*	*B.Right*	*Back*
FSM [22]	0.130	0.201	0.224	0.229	0.240	0.186
SurroundDepth [52]	0.152	0.207	0.230	0.220	0.239	0.200
$\mathcal{D}_\theta$ only	0.154	0.213	0.237	0.231	0.237	0.194
Ours	**0.128**	**0.160**	**0.168**	**0.172**	**0.174**	**0.169**

Table 6. **Comparison to multi-frame methods on DDAD.** We compare to methods that exploit temporal context from a single camera [51, 18]. We evaluate only the front camera since these methods were trained only on this camera.

	Front Camera			
Method	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta_{1.25}$ ↑
ManyDepth [51]	0.146	3.258	14.098	0.822
DepthFormer [18]	0.135	**2.953**	**12.477**	0.836
Ours	**0.128**	3.168	13.214	**0.868**

Table 7. **Pose evaluation on DDAD.** We compare our method to state-of-the-art monocular SLAM methods for which we report the average ATE across the cameras that successfully tracked the camera path. ** scaled trajectory.

	DROID-SLAM [48]	ORB-SLAMv3 [5]	**Ours**
ATE** [m] ↓	7.500	6.179	**0.433**

mation accuracy improves dramatically over the naive DBA baseline. The relative scale ATE drops from 2.134m to only 0.435m. The absolute scale ATE is only about 1.2m higher, indicating the scene scale is recovered accurately.

Using VKITTI fine-tuning (1c), we achieve significant gains in depth accuracy, where Abs. Rel. drops to 0.32 and $\delta_{1.25}$ increases to 72.7%. However, note that VKITTI fine-tuning does not affect the $\delta_{1.25}$ score, *i.e.* it helps mostly with depth outliers. We attribute this to the absence of dynamic objects in the training data of [48] so that fine-tuning helps to adjust the confidence prediction to handle such outliers (cf. Sec. 3.2). We further test the depth refinement network (Sec. 3.3) without geometric estimation prior (2a). The depth obtained from $\mathcal{D}_\theta$ is less prone to outliers, as evidenced by its low Abs. Rel., Sq. Rel. and RMSE scores. However, note that its $\delta_{1.25}$ is only 71.5%, *i.e.* its

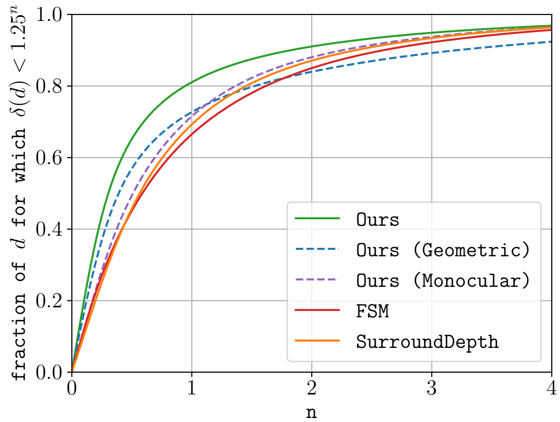

Figure 5. **Depth in $\delta_{1.25^n}$ range for different thresholds on DDAD.** We depict the ratio of depth estimates d for which $\delta(d) < 1.25^n$ where $\delta(d) = \max\left(\frac{d}{d^*}, \frac{d^*}{d}\right)$ and d^* is ground-truth depth. Note that values at $n = 1, 2, 3$ represent $\delta_{1.25}, \delta_{1.25^2}, \delta_{1.25^3}$.

Table 8. **Comparison to state-of-the-art on NuScenes.** Evaluated up to 80m for scale-aware depth as in [22, 52] and 60m for median-scaled depth (denoted with *) as in [54].

Method	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta_{1.25}$ ↑
FSM [22]	0.297	-	-	-
SurroundDepth [52]	0.280	**4.401**	7.467	0.661
Ours	**0.253**	4.759	**7.150**	**0.729**
PackNet*[19]	0.303	3.154	7.014	0.655
FSM* [22]	0.270	3.185	6.832	0.689
SurroundDepth* [52]	0.245	3.067	6.835	0.719
MCDP* [54]	0.237	**3.030**	6.822	0.719
Ours*	**0.235**	3.332	**6.021**	**0.749**

depth accuracy compared to the geometric depth estimates is actually lower when disregarding outliers. Finally, we test our full system (3a). Compared to geometric estimation (1d), we observe a significant improvement in outlier sensitive metrics like RMSE, but also in outlier robust metrics like $\delta_{1.25}$. Additionally, the pose estimation accuracy is increased and the difference between scaled and non-scaled ATE is smaller. Compared to depth obtained from $\mathcal{D}_\theta$ (2a), the increase in outlier sensitive metrics is smaller but still significant, *e.g.* 1.26m in RMSE. Further, we observe a sizable gain in $\delta_{1.25}$ of 9.6 percentage points.

Further, in Fig. 5, we compare the $\delta_{1.25^n}$ scores at different accuracy thresholds n of our geometric depth estimation without refinement ('Ours (Geometric)'), our refinement network only ('Ours (Monocular)') and our full method ('Ours'). The results confirm our hypothesis that while geometric estimates are more accurate than monocular depth estimates, they are also noisier. Importantly, the results further illustrate that our full method can effectively combine the strengths of both approaches while not suffering from their weaknesses.

Co-visibility graph construction. In Tab. 2, we compare our co-visibility graph construction algorithm (Sec. 3.1) to a multi-camera adaption of the algorithm in [48]. Ours

Table 9. **Per-camera evaluation on NuScenes**. We compare our method with previous works on scale-aware depth estimation. Evaluated up to 80m as in Tab. 8. We observe the same trend as on DDAD, *i.e.* our method performs best across all cameras, with a particularly high improvement on the side views.

	Abs Rel ↓					
Method	*Front*	*F.Left*	*F.Right*	*B.Left*	*B.Right*	*Back*
FSM [22]	0.186	0.287	0.375	0.296	0.418	0.221
SurroundDepth [52]	0.179	0.260	0.340	0.282	0.403	0.212
Ours	**0.174**	**0.230**	**0.302**	**0.249**	**0.360**	**0.201**

Table 10. **Cross-dataset transfer.** We use models trained on DDAD to evaluate scale-aware depth prediction on the NuScenes dataset. We compare to state-of-the-art methods [22, 52] and observe that our method exhibits better generalization ability (cf. Tab. 8).

Method	Abs Rel ↓	Sq Rel ↓	RMSE ↓	$\delta_{1.25}$ ↑
FSM [22]	0.349	5.064	8.785	0.499
SurroundDepth [52]	0.364	5.476	8.447	0.525
Ours	**0.292**	**4.800**	**7.677**	**0.660**

yields similar performance in depth estimation, with the exact same Abs. Rel. score and only minor difference in other metrics. Further, the absolute scale ATE is only slightly (25cm) higher. In contrast, using our algorithm, we achieve a runtime reduction of nearly $10\times$ on the full system.

Depth refinement network. We verify the choice of our inputs to $\mathcal{D}_\theta$ in Tab. 3. If we do not input the confidence $\mathbf{w}_i$ and do not sparsify the depth via $\mathbf{w}_i < \beta$ before feeding it to $\mathcal{D}_\theta$, we observe the lowest performance. If we input the confidence $\mathbf{w}_i$ but do not sparsify the depth, we see an improvement over not using them. We achieve the best performance with the inputs described in Sec. 3.3.

Moreover, we show qualitative examples of our depth refinement in Fig. 4. The confidence $\mathbf{w}_i$ (overlaid with the image) segments regions where geometric estimation is ambiguous, *i.e.* dynamic objects and uniformly textured areas like the sky and the road surface. The geometric depth maps exhibit severe artifacts in such regions. After refinement, these artifacts are corrected.

4.3. Comparison to State-of-the-Art

DDAD. In Tab. 4, we compare our method to existing self-supervised multi-camera depth estimation methods. We achieve significant improvements in scale-aware and scale-invariant depth prediction with only minor differences in absolute and relative scale results. In both settings, we achieve state-of-the-art results with the lowest Abs. Rel. scores of 0.162 and 0.169, respectively.

In Fig. 5 we compare the $\delta_{1.25^n}$ scores of our method to the state-of-the-art approaches on DDAD at different n. For small values of n, we observe a particularly pronounced performance gain over existing methods. This shows the advantage of combining highly accurate geometric estimates with monocular cues. Further, our method consistently outperforms the baselines over all values of n for $\delta < 1.25^n$.

In Fig. 6, we show a qualitative comparison to existing works. On the left side, we illustrate depth maps across three cameras. Compared to FSM [22] and SurroundDepth [52], we produce sharper depth boundaries. On the right side, we illustrate point clouds accumulated over time. We produce a more consistent reconstruction, as can be seen when focusing on the vehicles or the fences beside the road.

Tab. 5 shows the per-camera breakdown of the scale-aware depth prediction comparison. The advantage of our method is particularly pronounced on the side cameras. Other methods struggle with the side views because the static spatial context is limited and they do not exploit any temporal or spatial-temporal context. The temporal and spatial-temporal contexts are conducive in the side views since sideward motion leads to better triangulation angles than forward motion. This is also evidenced by the results of $\mathcal{D}_\theta$ without geometric estimation, which performs similarly to the baseline methods, struggling with the side views. In contrast, our full system performs well across all cameras.

We further compare to self-supervised single-camera methods that leverage temporal information in Tab. 6. For a fair comparison, we evaluate only the front camera. We compare favorably to DepthFormer [18] and ManyDepth [51]. In particular, we substantially improve in the outlier robust metric $\delta_{1.25}$ and perform competitively in all others. Our Abs. Rel. score is the lowest at 0.128.

To evaluate ego-motion estimation, we compare our system to state-of-the-art monocular SLAM methods [48, 5] in Tab. 7. We run the baselines on each camera and report the average ATE across the cameras that successfully tracked the camera path. Note that, contrary to our approach, the competing methods frequently failed to produce a trajectory. We observe that state-of-the-art monocular SLAM methods exhibit a large error (ATE), while our method recovers accurate trajectories.

NuScenes. In Tab. 8, we compare our system to previous self-supervised multi-camera depth estimation methods. We outperform previous methods by a significant margin, corroborating our findings on DDAD. Notably, we observe substantial improvements in scale-aware depth estimation, while previous works usually struggle with recovering absolute scale on this dataset since the overlap between cameras is smaller than in DDAD [22]. Therefore, previous works like MCDP [54] report results on scale-invariant depth prediction. We outperform existing methods also in this setting, although the performance gap is smaller. Note that the gap between scale-aware and scale-invariant evaluation is only 2 percentage points in $\delta_{1.25}$ for our method, while for SurroundDepth [52] the gap is much larger with 5.8 percentage points. We show a per-camera breakdown of the comparison to the state-of-the-art in scale-aware depth estimation in Tab. 9. The results corroborate our findings

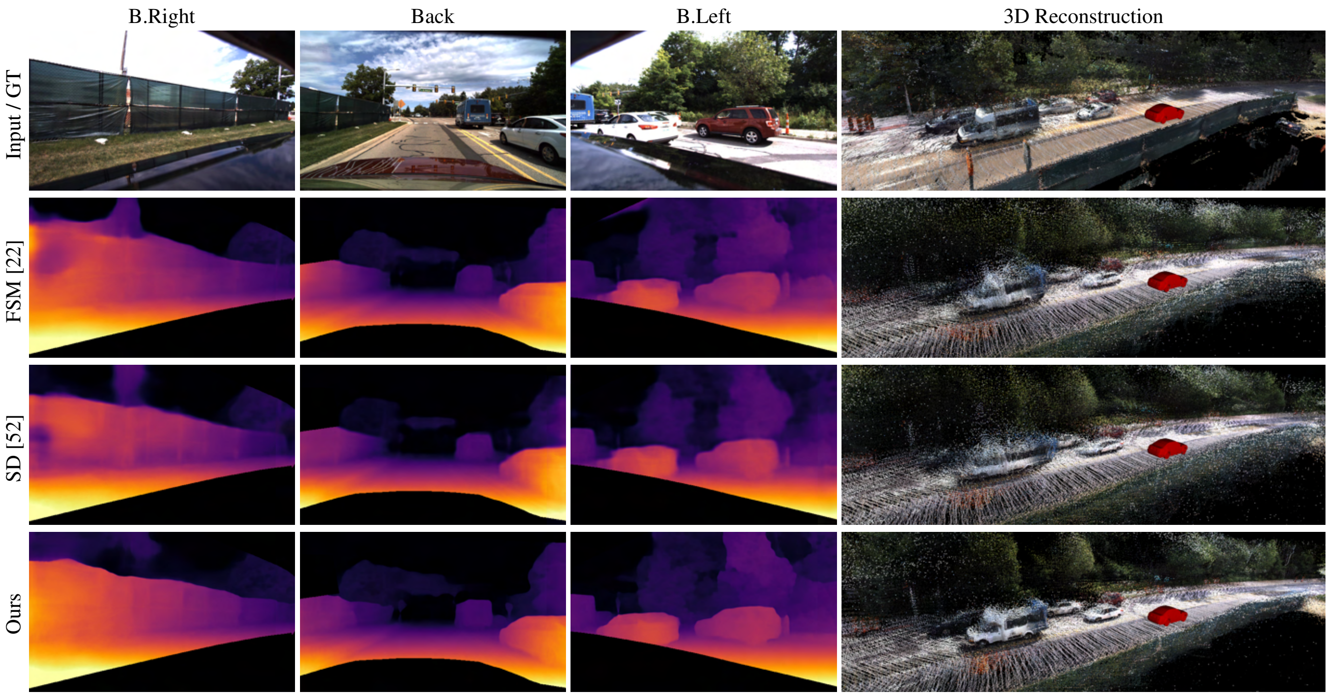

Figure 6. **Qualitative comparison on DDAD.** We show depth maps and point clouds accumulated over time along with input images, ground-truth LiDAR 3D Reconstruction, and the ego-vehicle in red. Our depth maps are sharper and more accurate. Our accumulated point clouds yield a more consistent 3D reconstruction. Note that we use our pose predictions for [22, 52] since these do not predict pose.

on DDAD, namely that our method outperforms previous works across all cameras with a particularly pronounced improvement on the side views.

Cross-dataset transfer. We finally test the cross-dataset generalization of our method versus FSM [22] and SurroundDepth [52]. In Tab. 10, we show *scale-aware* depth estimation results of models trained on DDAD, evaluated on NuScenes. We observe that the gap in Abs. Rel. widens compared to Tab. 8, with our method outperforming SurroundDepth by 0.072 in Abs. Rel. and FSM by 0.057. Further, our method maintains much higher levels of $\delta_{1.25}$ and Sq. Rel., while SurroundDepth drops significantly in both metrics. Note that we apply focal length scaling [45] to all methods to accommodate differences in camera intrinsics.

5. Conclusion

We introduced R3D3, a multi-camera system for dense 3D reconstruction of dynamic outdoor environments. The key ideas we presented are a multi-camera DBA operator that greatly improves geometric depth and pose estimation, a multi-camera co-visibility graph construction algorithm that reduces the runtime of our system by nearly $10\times$ without significant performance drop, and a depth refinement network that effectively fuses geometric depth estimates with monocular cues. We observe that our design choices enable dense 3D mapping in challenging scenes presenting many dynamic objects, uniform and repetitive textures, and complicated camera phenomena like lens flare and auto-exposure. We achieve state-of-the-art performance in dense depth prediction across two multi-camera benchmarks.

References

[1] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *CVPR*, 2018.

[2] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *CoRR*, 2020.

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[4] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*, 2008.

[5] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *T-RO*, 2021.

[6] Laura A Clemente, Andrew J Davison, Ian D Reid, José Neira, and Juan D Tardós. Mapping large loops with a single hand-held camera. In *RSS*, 2007.

[7] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *RA-L*, 2020.

[8] Arun Das and Steven L Waslander. Entropy based keyframe selection for multi-camera visual slam. In *IROS*, 2015.

[9] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, 2003.

[10] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE TPAMI*, 2007.

[11] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE TPAMI*, 2017.

[12] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.

[13] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE TPAMI*, 2009.

[14] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *ICCV*, 2015.

[15] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[16] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *ICCV*, 2019.

[17] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2020.

[18] Vitor Guizilini, Rares Ambrus, Dian Chen, Sergey Zakharov, and Adrien Gaidon. Multi-frame self-supervised depth with transformers. In *CVPR*, 2022.

[19] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *CVPR*, 2020.

[20] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. In *ICLR*, 2020.

[21] Vitor Guizilini, Kuan-Hui Lee, Rares Ambrus, and Adrien Gaidon. Learning optical flow, depth, and scene flow without real-world labels. In *RA-L*, 2022.

[22] Vitor Guizilini, Igor Vasiljevic, Rares Ambrus, Greg Shakhnarovich, and Adrien Gaidon. Full surround monodepth from multiple cameras. In *RA-L*, 2022.

[23] Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang. Learning monocular depth by distilling cross-domain stereo networks. In *ECCV*, 2018.

[24] Adam Harmat, Inna Sharf, and Michael Trentini. Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle. In *ICRA*, 2012.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[26] Lionel Heng, Benjamin Choi, Zhaopeng Cui, Marcel Geppert, Sixing Hu, Benson Kuan, Peidong Liu, Rang Nguyen, Ye Chuan Yeo, Andreas Geiger, et al. Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system. In *ICRA*, 2019.

[27] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *CVPR*, 2018.

[28] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *NeurIPS*, 2017.

[29] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, 2007.

[30] Juichung Kuo, Manasi Muglikar, Zichao Zhang, and Davide Scaramuzza. Redesigning slam for arbitrary multi-camera systems. In *ICRA*, 2020.

[31] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. In *ICCV*, 1999.

[32] Maxime Lhuillier and Long Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE TPAMI*, 2005.

[33] Peidong Liu, Marcel Geppert, Lionel Heng, Torsten Sattler, Andreas Geiger, and Marc Pollefeys. Towards robust visual odometry with a multi-camera system. In *IROS*, 2018.

[34] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *T-RO*, 2015.

[35] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *T-RO*, 2017.

[36] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, 2011.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[39] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *ICRA*, 2020.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[41] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016.

[42] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *RSS*, 2012.

[43] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020.

[44] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. In *ICLR*, 2019.

[45] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *CVPR*, 2017.

[46] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. In *ICLR*, 2020.

[47] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[48] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *NeurIPS*, 2021.

[49] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *CVPR*, 2018.

[50] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 2004.

[51] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel Brostow, and Michael Firman. The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth. In *CVPR*, 2021.

[52] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Yongming Rao, Guan Huang, Jiwen Lu, and Jie Zhou. Surrounddepth: Entangling surrounding views for self-supervised multi-camera depth estimation. In *CoRL*, 2022.

[53] Felix Wimbauer, Nan Yang, Lukas Von Stumberg, Niclas Zeller, and Daniel Cremers. Monorec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera. In *CVPR*, 2021.

[54] Jialei Xu, Xianming Liu, Yuanchao Bai, Junjun Jiang, Kaixuan Wang, Xiaozhi Chen, and Xiangyang Ji. Multi-camera collaborative depth prediction via consistent structure estimation. In *ACM MM*, 2022.

[55] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *CVPR*, 2020.

[56] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, 2018.

[57] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

[58] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multiview stereo depth inference. In *CVPR*, 2019.

[59] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018.

[60] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *CVPR*, 2020.

[61] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *CVPR*, 2018.

[62] Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. Patchmatch based joint view selection and depthmap estimation. In *CVPR*, 2014.

[63] Yiran Zhong, Pan Ji, Jianyuan Wang, Yuchao Dai, and Hongdong Li. Unsupervised deep epipolar flow for stationary or dynamic scenes. In *CVPR*, 2019.

[64] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *ECCV*, 2018.

[65] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.