DeFlowSLAM: Self-Supervised Scene Motion Decomposition for Dynamic Dense SLAM

Weicai Ye*, Xingyuan Yu*, Xinyue Lan, Yuhang Ming, Jinyu Li, Hujun Bao, Zhaopeng Cui and Guofeng Zhang

***Abstract*—We present a novel dual-flow representation of scene motion that decomposes the optical flow into a static flow field caused by the camera motion and another dynamic flow field caused by the objects' movements in the scene. Based on this representation, we present a dynamic SLAM, dubbed DeFlowSLAM, that exploits both static and dynamic pixels in the images to solve the camera poses, rather than simply using static background pixels as other dynamic SLAM systems do. We propose a dynamic update module to train our DeFlowSLAM in a self-supervised manner, where a dense bundle adjustment layer takes in estimated static flow fields and the weights controlled by the dynamic mask and outputs the residual of the optimized static flow fields, camera poses, and inverse depths. The static and dynamic flow fields are estimated by warping the current image to the neighboring images, and the optical flow can be obtained by summing the two fields. Extensive experiments demonstrate that DeFlowSLAM generalizes well to both static and dynamic scenes as it exhibits comparable performance to the state-of-the-art DROID-SLAM in static and less dynamic scenes while significantly outperforming DROID-SLAM in highly dynamic environments. The code and pre-trained model will be available on the project webpage: `https://zju3dv.github.io/deflowslam/`.**

***Index Terms*—Dual-Flow Representation, Dynamic Dense SLAM, Dynamic Update Module, Motion Estimation.**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) is fundamental to the field of computer vision and robotics, with many applications ranging from augmented reality (AR), and virtual reality (VR) to autonomous driving. In AR applications, SLAM is often leveraged to provide accurate localization for agents, which facilitates users to place virtual objects [1], while the dense reconstruction is urgently needed to better interact with the surrounding environments. Due to the simplicity of monocular video acquisition, monocular dense SLAM [2], [3] has attracted wide attention, yet it is still a more complicated task compared to RGB-D SLAM [4]–[8].

Impressive progress has been seen in geometry-based [9]–[12], learning-based [13]–[15], and hybrid approaches [16]–[18]. However, it is still a challenging problem to develop robust and reliable SLAM methods for real-world AR applications, especially in dynamic scenarios. To meet the challenges of such dynamic environments, some approaches [19], [20] filter out dynamic objects in advance by introducing information such as semantics, e.g., using Mask R-CNN [21] to segment out potentially moving vehicles and pedestrians, and then run monocular SLAM systems such as ORB-SLAM [10]. However, due to the slow running speed and high memory consumption of Mask-RCNN, it is difficult to meet the real-time requirement of SLAM. What's more, there are limits in practical applications, for not all dynamic objects exist in the detector's training data and may lead to catastrophic system failure [4]. Besides, simply throwing away dynamic information and only constructing sparse feature maps makes such SLAM methods incapable of handling tasks like planning and interaction. Some approaches [22], [23] combine multi-object detection and SLAM, making it possible to add target-aware constraints to eliminate the interference of dynamic objects, but the generalization of such SLAM systems is limited as well due to the detector.

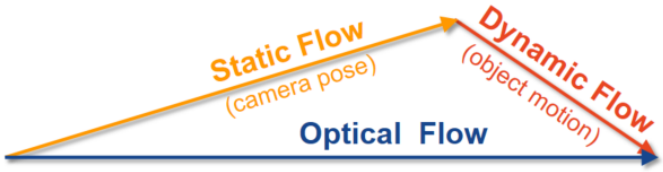

Fig. 1. **Dual-Flow Representation.** The estimated optical flow is decomposed into a static flow caused by the camera pose and a dynamic flow caused by the dynamic object's motion.

Recently, a learning-based dense SLAM system, DROID-SLAM [24], has been proposed, which demonstrates better accuracy and robustness than traditional methods. However, we find that it has relatively large errors in the pose estimation in some stronger dynamic scenarios such as sequence 09 of KITTI [25]. DROID-SLAM [24] presents a dense bundle adjustment layer to iteratively update the residual of inverse depths and camera poses using the estimated optical flow and weights, while the presence of dynamic objects may lead to ambiguity in optical flow estimation if the system failed to discover them. For this reason, we propose a novel scene motion representation, named dual-flow, that decomposes the estimated optical flow into a static flow caused by the camera poses and a dynamic flow of the dynamic object's own motion, as shown in Fig. 1. Such a representation largely mimics the way humans perceive the real world [26]. In addition, DROID-SLAM introduces massive supervision for camera pose, optical flow and depth estimation during training, which severely restricts the possibility of fine-tuning new scenes if it performs poorly in new scenes. On the contrary, we explore to

W. Ye, X. Yu, X. Lan, J. Li, H. Bao, Z. Cui and G. Zhang are with the State Key Lab of CAD&CG, Zhejiang University. W. Ye, X. Yu, X. Lan, J. Li, H. Bao and G. Zhang are also affiliated with ZJU-SenseTime Joint Lab of 3D Vision. E-mails: {weicaiye, xinyuelan, baohujun, zhpcui, zhangguofeng}@zju.edu.cn and mail@jinyu.li. Xinyuan Yu is also with Wuhan University. E-mail: RickyYXY@whu.edu.cn. Yuhang Ming is with Visual Information Laboratory, University of Bristol. E-mail: yuhang.ming@bristol.ac.uk.
*: indicates equal contribution. G. Zhang is the corresponding author.

train our dual-flow-based dynamic SLAM in a self-supervised way.

Our insight is that for the vast majority of scenes, dynamic objects may not be dominant and their motion can be decomposed by the dominant static objects. The representation based on scene motion decomposition has natural advantages: 1) we can easily optimize static fields in a similar way as DROID-SLAM [24] does for the poses and depths. 2) For dynamic fields, we can obtain consistent luminosity by warping the current frame to adjacent frames. 3) This holds promise for obtaining a self-supervised network model, which is also more interpretable. Based on these observations, we propose a dynamic update module for DeFlowSLAM, in which the dual-flow representation is cleverly embedded, as shown in Fig. 2, and detailed in Sec III. Directly warping the dynamic object field to the neighbor images may result in occlusion, shown in Fig. 4, we introduce a dynamic mask aggregation operator in the dynamic update module, named Mask-Agg to remove the incorrect alignment effect. Specifically, the Mask-Agg operator iteratively updates the residuals of the dynamic mask via convolutional gated recurrent unit (ConvGRU) [27]. The final dynamic mask can be obtained by summing the aggregated dynamic mask residuals with the original mask. The obtained dynamic mask will be combined with the estimated weights and then fed into the dense bundle adjustment (DBA) Layer [24] to optimize the residuals of pose and depth, shown in Fig. 3.

We first verify our hypothesis on a highly dynamic dataset, Virtual KITTI2 [28], where ablation experiments demonstrate that a dual-flow representation can achieve better performance. In addition, DeFlowSLAM can make full use of information from all pixels to solve the camera pose, resulting in gains compared to potential approaches that filter dynamic objects by masking. Applying our knowledge to the challenging SLAM dataset, such as TartanAir [29], the trained model exhibits good generalization and achieves comparable accuracy to DROID-SLAM in both static and slightly dynamic scenes. While in highly dynamic scenes, such as KITTI [25], our method significantly outperforms DROID-SLAM, and even the error in pose estimation is sharply reduced to half of DROID-SLAM. Then, we simply extended it to stereo and RGB-D datasets, showing the high scalability of the system. In AR applications, DeFlowSLAM shows relatively accurate pose estimation results, as shown in Fig. 9. In summary, our contributions are three-fold:

- We propose a novel dual-flow scene motion representation that decomposes the optical flow into a static flow field and a dynamic flow field.
- We develop a dynamic dense SLAM system based on dual-flow representation with a dynamic update module, which outperforms state-of-the-art methods in dynamic scenes.
- We propose a self-supervised training method instead of the strong supervision in DROID-SLAM [24].

II. Related Work

A. Scene Motion Decomposition

Scene motion estimation aims to obtain the 3D structure and 3D motion of dynamic scenes, and has received increasing attention for 3D perception. Recently, several scene motion estimation methods have been proposed according to the type of input data, such as 3D point clouds [30], [31], stereo images [32]–[34], or RGB-D images [35]–[38]. Comparatively few approaches have been studied for monocular scene motion estimation so far, as it is a highly ill-posed problem [39]. Here, we focus on monocular videos and the scene motion estimation can be regarded as optical flow [40]–[43]. Recently, a state-of-the art optical flow estimation method, RAFT [27] has been proposed, but it does not consider the scene motion decomposition. EffiScene [44] presents an unsupervised scene flow estimation method by jointly learning optical flow, depth, camera pose, and motion segmentation. Different from the forward and backward flow proposed by DF-VO [45], we attempt to decompose the optical flow to a static field caused by the camera pose and a dynamic field caused by the objects' own motion, exploiting the properties [46] of the scene flow itself, which inherently separates every pixel into multiple moving agents and a large class of points that follow a same rigid motion.

B. Dynamic SLAM

SLAM is a fundamental capability of many intelligent systems to perform accurate pose estimation and mapping. Humans live in a dynamic environment, intelligent systems should also have the ability to deal with dynamic environments, which are required to recognize the dynamic contents from the static environments. Traditional approaches largely filter out the interference of dynamic objects by introducing a prior [47] or RANSAC methods [48]. Some recent approaches try to use segmentation to filter out potential dynamic objects [19], [20], [49] and then run the SLAM system, or unify object detection and SLAM into a multi-task system [23], [50], [51] or add the object constraint to the SLAM system [22], [52], [53]. Unlike existing approaches, this paper uses a learning-based approach to identify dynamic fields at the pixel level, which does not require explicit supervision of object detection and better simulates the way humans perceive the world.

C. Dense Reconstruction

Reconstructing dense volumetric scenes is a cornerstone in robotics with many applications such as city modeling, augmented reality navigation [54], and cultural heritage preservation [55]. Most of the existing systems [56]–[60] need to use depth or lidar information for dense reconstruction, this paper does not have this limitation, and can support data formats such as monocular video, stereo, and RGB-D, and can maintain a good reconstruction effect in the dynamic environment. To improve the reconstruction performance, some approaches propose collaborative reconstruction by multiple robots [61]–[63]. As an example, Coxgraph [63] extends the single-robot

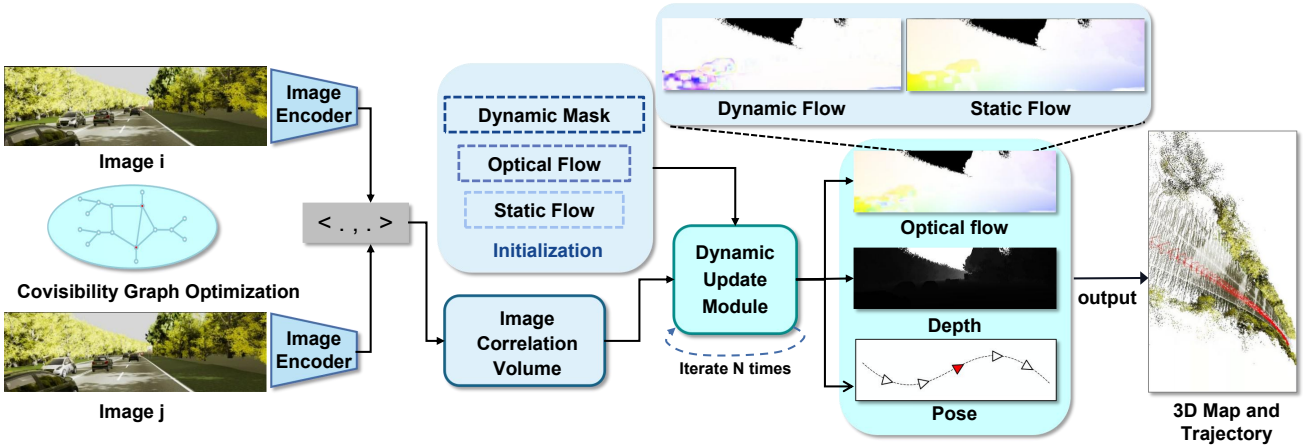

Fig. 2. **DeFlowSLAM Overview.** DeFlowSLAM takes the image sequence as input, extracts features to construct a correlation volume, and feeds it with the initial static flow, optical flow, and dynamic mask into the dynamic update module to iteratively optimize the residual of the pose, inverse depth, static flow, and dynamic flow, and finally outputs the estimation of the camera pose and 3D structure. The optimization process is performed by creating a covisibility graph and updating the existing covisibility graph.

reconstruction system, Voxgraph [56], to a multi-robot dense reconstruction system, and additionally adds more constraints to ensure low-error reconstruction. Our system is able to maintain robust dense reconstruction with less constraint. Some methods focus on dense depth reconstruction [64]–[66], while our DeFlowSLAM focuses on pose estimation although it can produce the intermediate depth and flow.

III. Methodology

Fig. 2 depicts an overview of our novel dual-flow-based SLAM method, DeFlowSLAM, which takes as input a sequence of images and outputs the camera pose estimation and the 3D map of the environments. DeFlowSLAM has an end-to-end differentiable architecture as DROID-SLAM [24] that leverages the strengths of both classical approaches and deep networks. It can robustly cope with challenging scenarios such as dynamic scenes, thanks to our proposed dual-flow representation, iterative dynamic update module, and factor graph optimization based on co-visibility between frames. Specifically, unlike DROID-SLAM, which iteratively updates the camera pose and depth, we additionally update a dynamic mask and a dynamic flow. And we perform each update of camera pose by the optimization of the estimated static flow field rather than the optical flow in DROID-SLAM. Next, we will first review DROID-SLAM [24] for understanding and then elaborate on the details of our approach.

A. Preliminaries: DROID-SLAM

DROID-SLAM [24] operates on a sequence of images $\{\mathbf{I}_t\}_{t=0}^{N}$, and maintains two state variables: camera pose $\mathbf{G}_t \in SE(3)$ and inverse depth $\mathbf{d}_t \in \mathbb{R}_{+}^{H\times W}$ for each image t, which are updated iteratively as new frames are processed. A frame graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is adopted to indicate co-visibility between frames in DROID-SLAM, where the nodes are input images and the edge $(i, j) \in \mathcal{E}$ implies that the images I_i and I_j have overlapped views.

1) Feature Extraction and Correlation: Following RAFT [27], the input images are first fed to the feature extraction module, and the relationship between the two images will be computed.

Feature Extraction. First, an image encoder, containing six residual blocks and three downsampling layers, takes each image as input and produces a dense image feature map, which is 1/8 of the original resolution, and feature maps from pairs of input images are used for later correlation volume construction.

Correlation Pyramid. For each edge $(i, j) \in \mathcal{E}$ in the frame graph $\mathcal{G}$, DROID-SLAM [24] computes the correlation volume $\mathbf{C}^{ij}$ as the dot product of feature vector pairs taken from $\mathbf{f}_\theta(I_i)_{u_i v_i}$ and $\mathbf{f}_\theta(I_j)_{u_j v_j}$:

$$C^{ij}_{u_i v_i u_j v_j} = \langle \mathbf{f}_\theta(I_i)_{u_i v_i},\ \mathbf{f}_\theta(I_j)_{u_j v_j} \rangle, \tag{1}$$

where u_i, v_i, u_j, v_j are the pixel coordinates for image I_i, I_j respectively and $<\ .\ >$ stands for the dot product. The last two dimensions of the correlation volume are fed to the average pooling layers with four different kernel sizes (1,2,4,8), forming a 4-level correlation pyramid [27].

Correlation Lookup. DROID-SLAM [24] defines a correlation lookup operator that uses a coordinate grid with radius r to index the correlation volume $L_r : \mathbb{R}^{H\times W\times H\times W} \times \mathbb{R}^{H\times W\times 2} \mapsto \mathbb{R}^{H\times W\times (r+1)^2}$. This correlation operator takes an $H \times W$ grid of coordinates of optical flow field as input, and retrieves the values from the correlation volume by using a bi-linear interpolation, which are concatenated to compute the final feature vector. The lookup function is applied to every correlation volume in the pyramid [24].

2) Update Module: DROID-SLAM [24] proposes an update operator, in which a 3×3 convolutional GRU is used to update the hidden state $\mathbf{h}$, the camera pose $\mathbf{G}$ and depth measurements $\mathbf{d}$. The final pose and depth measurement can be obtained by applying the incremental updates $\Delta\boldsymbol{\xi}^{(k)}$ and $\Delta\mathbf{d}^{(k)}$ to the current pose and depth estimates through retrac-

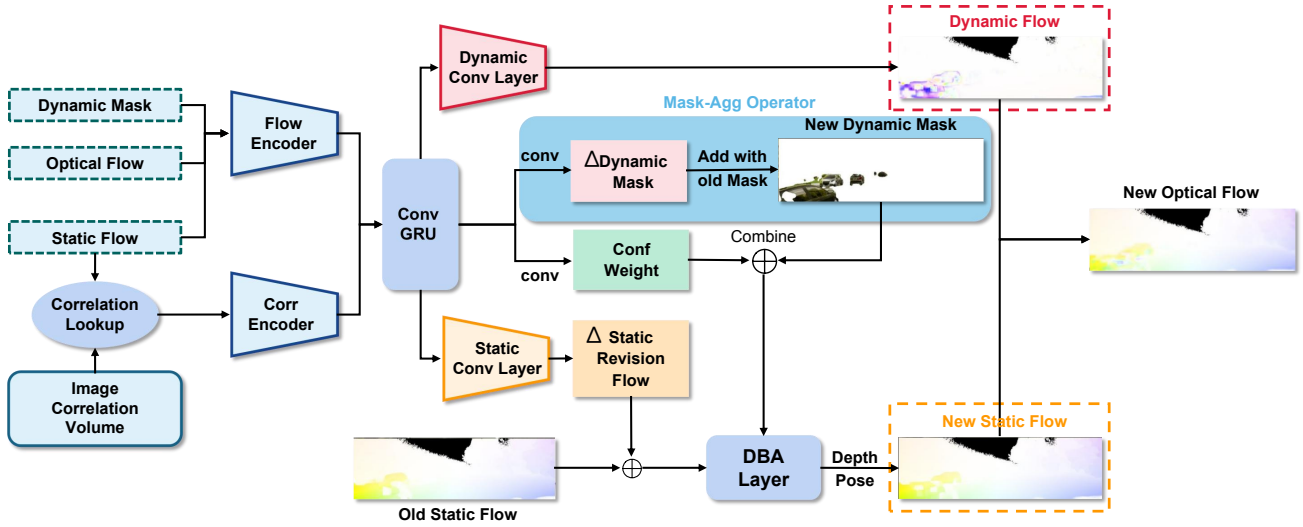

Fig. 3. **Dynamic Update Module.** The correlation feature of the static flow is looked up by correlation volumes. The obtained features will be fed into two convolutional layers together with the optical flow and dynamic mask, resulting in intermediate features. These features will be fed to ConvGRU, followed by two convolution layers with dynamic mask residual and confidence. The iterative dynamic mask residual plus the original mask to obtain the new dynamic mask termed Mask-Agg Operator. In addition, static flow revision and dynamic flow are obtained from the static and dynamic convolution layers. The static revision flow plus the original static flow is fed into the DBA layer that combines the dynamic mask and confidence to optimize the depth and pose. Finally, the new static and dynamic flows from the dynamic convolutional layer are summed to the optical flow.

tion on the SE3 manifold and vector addition respectively:

$$\mathbf{G}^{(k+1)} = \mathrm{Exp}(\Delta\boldsymbol{\xi}^{(k)}) \circ \mathbf{G}^{(k)}, \qquad \mathbf{d}^{(k+1)} = \Delta\mathbf{d}^{(k)} + \mathbf{d}^{(k)}. \tag{2}$$

The update operator [24] iteratively produces a sequence of poses and depths with the goal of converging to a fixed point: $\{\mathbf{G}^{(k)}\} \to \mathbf{G}^*$, $\{\mathbf{d}^{(k)}\} \to \mathbf{d}^*$.

B. Dual-Flow Representation

The core concept of the proposed dynamic SLAM network is the dual-flow representation and the self-supervised training scheme with a dynamic update module. In contrast to DROID-SLAM [24], which uses optical flow as an intermediate motion representation, we propose a novel scene motion representation by decomposing optical flow into a static flow caused by camera motion and a dynamic flow caused by the motion of dynamic objects themselves, as shown in Fig. 1. Such a representation can distinguish between static and dynamic object motions, thus having better interpretability and making the network traceable during the training process. We can directly use the dense static flow to estimate camera motion without masking out the pixels belonging to dynamic objects. The optical flow $\mathbf{F}_{\mathbf{o}t} \in \mathbb{R}^{H\times W\times 2}$, static flow $\mathbf{F}_{\mathbf{s}t} \in \mathbb{R}^{H\times W\times 2}$ and dynamic flow $\mathbf{F}_{\mathbf{d}t} \in \mathbb{R}^{H\times W\times 2}$ are a set of vectors, where the static flow plus the dynamic flow equals the optical flow:

$$\mathbf{F}_{\mathbf{o}t} = \mathbf{F}_{\mathbf{s}t} + \mathbf{F}_{\mathbf{d}t}. \tag{3}$$

Our network also operates on a sequence of images $\{\mathbf{I}_t\}_{t=0}^N$ as DROID-SLAM [24]. As new frames being processed, different from DROID-SLAM which updates the set of camera poses $\{\mathbf{G}_t\}_{t=0}^N \in SE(3)$ and inverse depths $\{\mathbf{d}_t\}_{t=0}^N \in \mathbb{R}_+^{H\times W}$, in DeFlowSLAM, the static flows $\{\mathbf{F}_{\mathbf{s}t}\}_{t=0}^N \in \mathbb{R}_+^{H\times W\times 2}$, the dynamic flows $\{\mathbf{F}_{\mathbf{d}t}\}_{t=0}^N \in \mathbb{R}^{H\times W\times 2}$ and the binary dynamic masks $\{\mathbf{M}_{\mathbf{d}t}\}_{t=0}^N \in \mathbb{R}_+^{H\times W\times 2}$ are additionally update iteratively. We let 0 indicate dynamic while 1 indicate static in $\mathbf{M}_{\mathbf{d}t}$.

Similar to DROID-SLAM [24], a frame graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is also adopted to represent co-visibility between frames. For example, as shown in the co-visibity graph optimization in Fig. 2, the white nodes indicate each image, and the edges indicate two images with a co-visibility relationship, where the red points indicate image I_i and image I_j, and the blue edges indicate the co-visibility relationship in the frame graph. The frame graphs are built and updated dynamically during the training and inference as DROID-SLAM [24]. After each pose or depth update with the revision static flow field, the frame graph with new co-visibilities will be updated. When a loop closure occurs, a long-distance connection is added to the frame graph.

C. Dynamic Update Module

Fig. 3 demonstrates the dynamic update module of our DeFlowSLAM, which is a 3×3 ConvGRU with a hidden state $\mathbf{h}$. Different from the update module in DROID-SLAM [24], which works directly on the revision optical flow, our dynamic update module works on the decomposed static and dynamic flow fields respectively. We update the revision static flow field in a similar way to DROID-SLAM, while for the dynamic flow field, we add it to the static flow field to obtain the optical flow, which is fed into the flow encoder as a new optimization term in the next iteration. In every iteration during updating, the module will produce a pose increment, depth increment, dynamic mask increment, and dynamic flow. The pose increments are applied to the current pose through retraction on the SE3 manifold as DROID-SLAM [24]:

$$\mathbf{G}^{(k+1)} = \mathrm{Exp}(\Delta\boldsymbol{\xi}^{(k)}) \circ \mathbf{G}^{(k)}. \tag{4}$$

While the depth and the dynamic mask increments are added to the current depth and dynamic mask respectively [51]:

$$\Xi^{(k+1)} = \Delta\Xi^{(k)} + \Xi^{(k)}, \Xi \in \{\mathbf{d}, \mathbf{M_d}\}. \tag{5}$$

where ${\mathbf{F_d}}^{(k+1)}$ is directly given a new value in every iteration. With the updated static flow ${\mathbf{F_s}}^{(k+1)}$ transformed from $\mathbf{G}^{(k+1)}$ and $\mathbf{d}^{(k+1)}$, the final optical flow can be computed using Eq. 3.

Different from iterative operations of the update module in DROID-SLAM [24], which produces a sequence of poses, and depths, our dynamic update module additionally produces dynamic masks, dynamic flows, and complete optical flows with the expectation of converting to an optimal point, such as $\{\mathbf{G}^{(k)}\} \rightarrow \mathbf{G}^*$, $\{\mathbf{d}^{(k)}\} \rightarrow \mathbf{d}^*$, $\{{\mathbf{M_d}}^{(k)}\} \rightarrow {\mathbf{M_d}}^*$, $\{{\mathbf{F_d}}^{(k)}\} \rightarrow {\mathbf{F_d}}^*$, $\{{\mathbf{F_o}}^{(k)}\} \rightarrow {\mathbf{F_o}}^*$, representing the true reconstruction.

1) Correspondence: At the beginning of each iteration, we use the current pose and depth estimates to search for correspondence like DROID-SLAM [24]. Take edge $(i, j) \in \mathcal{E}$ in the frame graph as an example, we set frame i as the current frame. So $\mathbf{p}_i$ is actually a group of grid coordinates related to the shape of frames. To get the dense corresponding coordinates $\mathbf{p}_{ij}$ in frame j, we can use two poses and a depth map to complete the projecting process:

$$\mathbf{p}_{ij} = \Pi_c(\mathbf{G}_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}_i)), \ \mathbf{G}_{ij} = \mathbf{G}_j \circ \mathbf{G}_i^{-1}. \tag{6}$$

where Π_c is the camera model that maps a set of 3D points onto the image, and Π_c^{-1} is the inverse projection function that maps the inverse depth map $\mathbf{d}$ and $\mathbf{p}_i$ to the 3D point cloud [51].

2) Inputs: Following DROID-SLAM [24], we use the same way to retrieve the correlation features. In addition, we use the dense correspondence field to derive the static optical flow revision caused by the camera motion, i.e., the difference between $\mathbf{p}_{ij} - \mathbf{p}_j$, which is $\mathbf{F}_{\mathbf{s}ij}$. For the initial dynamic mask $\mathbf{M}_{\mathbf{d}ij}$ and the initial dynamic flow $\mathbf{F}_{\mathbf{d}ij}$, we simply initialize them to zeros.

3) Dynamic Update: Similar to DROID-SLAM [24], we also use a ConvGRU network for the iteration of the dynamic update module. Different from DROID-SLAM, which only produces a revised optical flow field and a correlation confidence map $\mathbf{w}_{ij} \in \mathbb{R}^{H \times W \times 2}$, our dynamic update module instead produces several outputs mentioned below: (1) a revised static flow field $\mathbf{r}_{\mathbf{s}ij} \in \mathbb{R}^{H \times W \times 2}$, (2) an updated dynamic flow field $\mathbf{F}_{\mathbf{d}ij} \in \mathbb{R}^{H \times W \times 2}$, (3) a correlation confidence map $\mathbf{w}_{ij} \in \mathbb{R}^{H \times W \times 2}$, (4) an updated dynamic mask increment field $\Delta\mathbf{M}_{\mathbf{d}ij} \in \mathbb{R}^{H \times W \times 2}$. The revision $\mathbf{r}_{\mathbf{s}ij}$ is used to correct errors in the dense correspondence fields, which can be expressed as $\mathbf{p}^*_{\mathbf{s}ij} = \mathbf{r}_{\mathbf{s}ij} + \mathbf{p}_{\mathbf{s}ij}$ [24]. The maps predicted by the module are in low resolution. To obtain the original resolution maps, we follow the mask upsample method mentioned in DROID-SLAM to get better upsample results and predict a pixel-level damping factor map λ.

4) Dense Bundle Adjustment Layer (DBA): After we get the revised static flow field, we optimize the poses and depth maps using a dense bundle adjustment algorithm in DROID-SLAM [24]. The cost function is defined as follows:

$$\mathbf{E}(\mathbf{G}', \mathbf{d}') = \sum_{(i,j)\in\mathcal{E}} \left\| \mathbf{p}^*_{\mathbf{s}ij} - \Pi_c(\mathbf{G}'_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}'_i)) \right\|^2_{\Sigma_{ij}}, \tag{7}$$

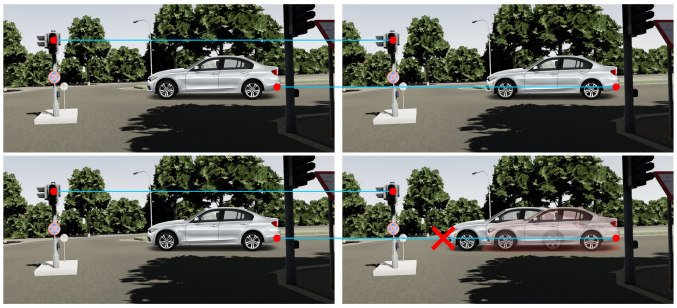

Fig. 4. **Mask-Agg Illustration**. If there are dynamic objects in the scene, directly matching pixels using static flow will cause erroneous results, making photometric loss invalid. Using the aggregated mask predicted by the network, we can filter out these invalid pixels (e.g. the pink mask) in geometry photometric loss, which is shown in the right bottom picture.

$$\Sigma_{ij} = \operatorname{diag} \mathbf{w}_{\mathbf{d}ij}, \tag{8}$$

$$\mathbf{w}_{\mathbf{d}ij} = \operatorname{sigmoid}(\mathbf{w}_{ij} + (1 - \mathbf{M}_{\mathbf{d}ij}) \cdot \eta), \tag{9}$$

where η is set as 10. And $\|\cdot\|_\Sigma$ is the Mahalanobis distance, which weights the error term according to the combined confidence $\mathbf{w}_{\mathbf{d}ij}$ [51]. Note that we introduce the dynamic mask $\mathbf{M}_{\mathbf{d}ij}$ here to make sure the optimization considers all needed points. It can make the optimization process work better in dynamic environments. The addition of these points can effectively improve the accuracy of BA operation in dynamic scenes. During BA optimization, we use the same Gauss-Newton algorithm to solve the linear system, like DROID-SLAM. We also use Schur's complement to accelerate the solving process. The DBA layer will not influence the gradient backpropagation as it is also in the computation graph. More details can be found in [24].

D. Loss Function

In this section, we elaborate on the self-supervised training scheme for DeFlowSLAM in detail.

1) Geometry Photometric Loss: To supervise the geometric predictions using images, we introduce a photometric reprojection loss to guide the network's optimization. We refer to [67] to build our photometric loss function. Given the predicted pose G_{ij} and the predicted depth $\hat{\mathbf{d}}_i$, we can get the corresponding coordinates of pixels in image I_i in image I_j. We then use bi-linear sampling to sample the image I_j, getting a new-sampled image $I_{j\rightarrow i}$:

$$I_{j\rightarrow i} = I_j \left\langle \Pi_c(\mathbf{G}_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \hat{\mathbf{d}}_i)) \right\rangle. \tag{10}$$

Then we can use the photometric loss on source image I_i and new image $I_{j\rightarrow i}$:

$$\mathcal{L}_{\text{geo\_ph}} = \frac{1}{N} \sum_{ij} pe\left(I_i, I_{j\rightarrow i}\right). \tag{11}$$

We leverage L_1 loss and $SSIM$ [68] loss to form our geometry photometric loss with $\alpha = 0.85$:

$$pe\left(I_a, I_b\right) = \frac{\alpha}{2}\left(1 - \operatorname{SSIM}\left(I_a, I_b\right)\right) + (1 - \alpha)\left\|I_a - I_b\right\|_1. \tag{12}$$

2) Mask Aggregation Operator for Geometry: In the self-supervised training policy, when using the photometric loss to supervise camera poses and depths, the direct use of the dual-flow representation may result in pixel mismatches due to the object's motion, making the geometry photometric loss less accurate. So we propose a mask aggregation operator, i.e. Mask-Agg, to filter out the wrong pixel matches to enhance our dual-flow representation, as shown in Fig. 4. The aggregated mask $\mathbf{M}_{\mathbf{d_i}}^{\mathbf{Agg}}$ for frame I_i is computed by gathering the dynamic masks estimated for all the frames which are connected to frame I_i in the frame graph. Then, the final geometry photometric loss function is:

$$\mathcal{L}_{\text{geo\_ph}} = \frac{1}{N'} \sum_{ij} pe\left(I_i, I_{j\to i}\right) \cdot \mathbf{M}_{\mathbf{d_i}}^{\mathbf{Agg}}. \tag{13}$$

where N' means the count of pixels whose $\mathbf{M}_{\mathbf{d_i}}^{\mathbf{Agg}}$ value is 1. Specifically, the Mask-Agg operator iteratively updates the residuals of the dynamic mask via ConvGRU. The final dynamic mask can be obtained by summing the aggregated dynamic mask residuals with the original old mask. Tab. I shows the Mask-Agg module helps filter out the ambiguous matches in self-supervised training, obtaining better results.

3) Optical Flow Photometric Loss: The geometry photometric loss is used to supervise the static flow caused by camera motion. So we also introduce the optical flow photometric loss to supervise the complete scene motion, including camera motion and objects' motion. Through the update module, we can get the optical flow results $\mathbf{F}_{\mathbf{o}ij}$ by adding static flow and dynamic flow. Similar to the geometry photometric loss, we use $\mathbf{F}_{\mathbf{o}ij}$ to generate corresponding coordinates between images:

$$I_{j\to i} = I_j \left\langle \mathbf{F}_{\mathbf{o}ij} + \mathbf{p}_{\mathbf{ij}} \right\rangle. \tag{14}$$

Then we still use bi-linear sampling to sample from the source image, evaluate their photometric errors:

$$\mathcal{L}_{\text{flow\_ph}} = \sum_{ij} pe\left(I_i, I_{j\to i}\right). \tag{15}$$

where the pe function here is just L_1 loss:

$$pe\left(I_a, I_b\right) = \left\|I_a - I_b\right\|_1. \tag{16}$$

4) Supervised Mask Loss: When dynamic mask labels are available, we can directly supervise our predicted masks using a simple cross-entropy classification loss [69]:

$$\mathcal{L}_{\text{gt\_mask}} = -\frac{1}{|\mathcal{N}|} \sum_{\mathbf{p}_i \in \mathcal{N}} \mathbf{M}_i \log \hat{\mathbf{M}}_i + (1 - \mathbf{M}_i) \log\left(1 - \hat{\mathbf{M}}_i\right). \tag{17}$$

where $\mathbf{M}_i$ is the ground truth mask labels, $\hat{\mathbf{M}}_i$ is the predicted masks.

5) Artificial Mask Loss: When the dynamic mask labels are not available, we design a method to artificially build the referred masks for self-supervised dynamic mask learning. Referred to SLIM [46], we propose artificial mask loss to achieve this goal. Using the camera pose, depth, and optical flow we have already gotten, we can first infer the target coordinate of the pixel p following the equations below:

$$\mathbf{p}_{\text{cam}} = \Pi_c(\mathbf{G}_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \hat{\mathbf{d}}_i)), \mathbf{p}_{\text{flow}} = \mathbf{p}_i + \hat{\mathbf{F}}_{\mathbf{o}ij}. \tag{18}$$

Fig. 5. **Qualitative Results of DeFlowSLAM**. DeFlowSLAM can generalize to new datasets, such as ETH3D, TUM RGB-D, ScanNet and Virtual KITTI2.

where $\mathbf{p}_{\text{cam}}$ is the target coordinate calculated by projection, $\mathbf{p}_{\text{flow}}$ is the target coordinate calculated by optical flow. Then we use the difference between these two coordinates to form our artificial mask labels:

$$\mathbf{M}_{\mathbf{i}}^{\mathbf{art}} = [\|\mathbf{p}_{\text{cam}} - \mathbf{p}_{\text{flow}}\|_2 \leq \mu], \tag{19}$$

where μ is set as 0.5. This artificial mask label has the same format compared with ground truth mask labels, so our final loss function should be:

$$\mathcal{L}_{\text{art\_mask}} = -\frac{1}{|\mathcal{N}|} \sum_{\mathbf{p}_i \in \mathcal{N}} \mathbf{M}_{\mathbf{i}}^{\mathbf{art}} \log \hat{\mathbf{M}}_i + \left(1 - \mathbf{M}_{\mathbf{i}}^{\mathbf{art}}\right) \log\left(1 - \hat{\mathbf{M}}_i\right). \tag{20}$$

6) Final Loss Function: We use $\mathcal{L}_{\text{geo\_ph}}$, $\mathcal{L}_{\text{flow\_ph}}$ and $\mathcal{L}_{\text{art\_mask}}$ for self-supervised training. The supervised final loss function is

$$\mathcal{L}_{\text{self-sup}} = \lambda_1 \mathcal{L}_{\text{geo\_ph}} + \lambda_2 \mathcal{L}_{\text{flow\_ph}} + \lambda_3 \mathcal{L}_{\text{art\_mask}}. \tag{21}$$

where $\lambda_1 = 100$, $\lambda_2 = 5$, and $\lambda_3 = 0.05$. Since the network has several update iterations, we use $\gamma = 0.9$ to apply the loss to the output of each iteration with exponentially increasing weights. For semi-supervised learning, we modify the $\mathcal{L}_{\text{art\_mask}}$ to the $\mathcal{L}_{\text{gt\_mask}}$.

E. Implementation Details

1) Training Details: We implement DeFlowSLAM in PyTorch and use the LieTorch extension [70] to perform back-propagation in the tangent space of all group elements. In the ablation study, we trained DeFlowSLAM on the Virtual KITTI2 dataset [28] with 2 RTX-3090 GPUs for 80,000 steps, which took about 2 days. While for the TartanAir dataset [29], it takes 5 days for 250k steps with a batch size of 4 and resolution of 344×464 on 4 RTX-3090 GPUs. In the monocular setup, we fix the first two poses as the true pose for each training sequence as DROID-SLAM [24], for the network can only recover the trajectory of the camera to the similarity transformation. In addition, there are still normative degrees of uncertainty during training, which may have an impact on the stability in terms of regulation and gradients. During training,

TABLE I
ABLATION STUDY OF DEFLOWSLAM TRAINED AND TESTED ON VITUAL KITTI2 (VK) DATASET. SS MEANS SELF-SUPERVISED, SM MEANS SEMI-SUPERVISED, SF MEANS SINGLE FLOW, DF MEANS DUAL FLOW, AND MA MEANS MASK-AGG.

Monocular	VK01	VK02	VK06	VK18	VK20
DROID-SLAM* [24]	1.091	**0.025**	0.113	1.156	8.285
Ours (SM, DF)	**0.761**	0.069	0.11	0.737	**2.546**
Ours (SS, SF)	4.278	0.187	0.142	1.248	10.487
Ours (SS, DF)	1.099	0.094	1.984	0.983	6.918
Ours (SS, DF, MA)	1.341	0.089	**0.092**	**0.376**	5.975

TABLE II
ABLATION STUDY OF DYNAMIC THRESHOLD μ OF DEFLOWSLAM TRAINED AND TESTED ON VITUAL KITTI2 (VK) DATASET IN SELF-SUPERVISED SETTING. $\mu = 0.5$ ACHIEVES THE BEST RESULTS.

μ	VK01	VK02	VK06	VK18	VK20
0.3	0.763	0.117	0.126	36.10	114.5
0.5	1.341	**0.089**	**0.092**	**0.376**	**5.975**
0.7	**0.738**	0.274	0.324	27.41	104.1

we follow DROID-SLAM [24], where each training example consists of a 6-frame video sequence, and the average distance between adjacent frames is between 8 pixels and 96 pixels.

2) SLAM System Details: For better comparison with DROID-SLAM [24], we keep the same system setting as DROID-SLAM in initialization, front-end, and backend. In the initialization, DeFlowSLAM continuously receives new incoming frames until 12 in total, constructs frame graphs for them, and uses our dynamic update module to compute their initial pose and inverse depth maps. In the frontend, as a new frame comes, the system will use 3 nearest neighbor frames and the new frame itself to create a temporary graph, where the hidden state of the new frame such as pose and inversed depth will be optimized. In the backend, the system will create a new graph that contains every preserved keyframes. The edge between keyframes is generated in a certain rule in order to eliminate redundant edges. We use the dynamic update module to optimize the whole graph for the final poses and depth. For more specific details about the SLAM system, please refer to DROID-SLAM [24].

3) Different Input Modes: Following DROID-SLAM [24], our system also supports different kinds of input data such as stereo images or RGB-D images, just to add crossed camera edges for stereo or add the depth constraint for RGB-D, compared with the monocular mode. The implementing details can be found in DROID-SLAM [24].

IV. EXPERIMENTS

We first validate the effectiveness of our method in highly dynamic scenarios, such as Virtual KITTI2 [28] in the ablation study. Further, we train DeFlowSLAM from scratch with the same strategy on a larger dataset, TartanAir [29], and test the generalization of our method on different dynamic datasets, such as Virtual KITTI2 [28], KITTI [25], and dynamic sequences of TUM-RGBD [71]. We also test

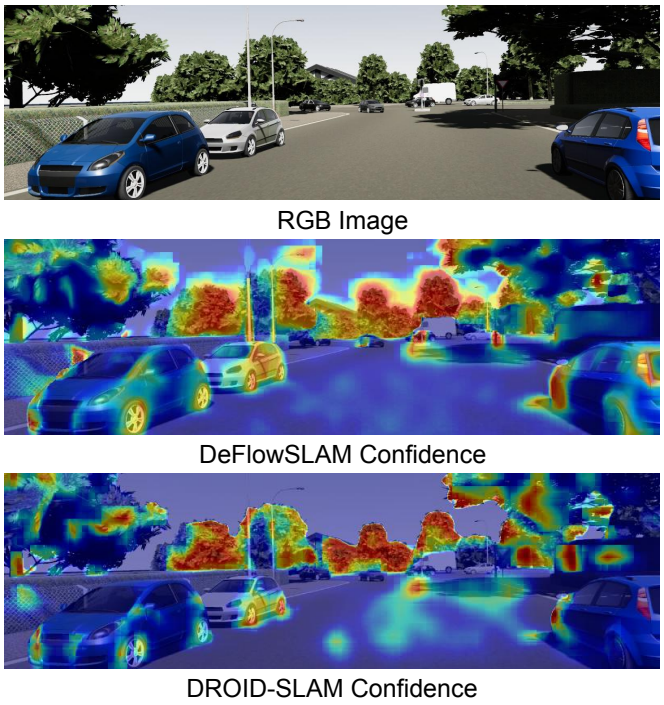

Fig. 6. **Confidence Comparison.** We visualize the confidence of our DeFlowSLAM vs. DROID-SLAM. Our method can exploit the dynamic pixels to solve the camera pose. The darker red color means greater weight. DeFlowSLAM also takes advantage of more information about the parked vehicles on the roadside.

TABLE III
DYNAMIC SLAM RESULTS ON KITTI (K) & VIRTUAL KITTI2 (VK) DATASETS WITH METRIC: ATE[M] TRAINED ON TARTANAIR DATASET. WE ACHIEVE THE BEST RESULTS.

Method	K09	K10	VK01	VK02	VK06	VK18	VK20
DSO [12]	28.1	24.0	-	-	-	-	-
DynaSLAM* [19]	41.91	7.519	27.830	X	X	X	2.807
DROID-SLAM* [24]	47.1	11.0	2.259	0.049	0.136	1.170	6.998
Ours	**27.8**	**4.2**	**0.591**	**0.021**	**0.13**	**0.400**	**1.039**

monocular or stereo datasets, such as static scenes of TUM-RGBD [71] and EuRoc [72]. Following DROID-SLAM [24], we use absolute trajectory error (ATE) [73] to evaluate the accuracy of the estimated camera trajectories. In particular, we compare DeFlowSLAM to DynaSLAM [19], etc in dynamic scenes, demonstrating the effectiveness of our method. We also compare DeFlowSLAM to DROID-SLAM in AR applications, the results further prove the robustness of our pose estimation.

A. Datasets

Virtual KITTI2 [28] is derived from the KITTI tracking benchmark [25] and consists of 5 sequences augmented with various weather conditions (e.g. fog, rain) and different camera configurations in terms of the camera orientations. In the ablation study, we use the camera configurations with the default camera orientation as the training set, the configurations with *15-degree* as the validation set, and the ones *30-degree* as the test set.

TartanAir [29] is a challenging dataset that is collected in photo-realistic simulation environments with the presence of

TABLE IV
DYNAMIC SLAM RESULTS ON TUM DYNAMIC SEQUENCES WITH METRIC: ATE[M]. THE BEST RESULTS ARE SHOWN IN BOLD. DEFLOWSLAM ACHIEVES COMPETITIVE AND EVEN BEST PERFORMANCE. NOTE THAT DVO SLAM, ORB-SLAM2, AND POINTCORR USE THE RGB-D DATASET, WHILE OUR METHOD AND DROID-SLAM ONLY USE THE MONOCULAR RGB DATASET.

Sequences		Trans. RMSE of trajectory alignment [m]				
		DVO SLAM [74]	ORB-SLAM2 [75]	PointCorr [7]	DROID-SLAM [24]	Ours
slightly dynamic	fr2/desk-person	0.104	**0.006**	0.008	0.017	0.013
	fr3/sitting-static	0.012	0.008	0.010	**0.007**	**0.007**
	fr3/sitting-xyz	0.242	0.010	**0.009**	0.016	0.015
	fr3/sitting-rpy	0.176	0.025	**0.023**	0.029	0.027
	fr3/sitting-halfsphere	0.220	0.025	0.024	**0.022**	0.025
highly dynamic	fr3/walking-static	0.752	0.408	0.011	0.016	**0.007**
	fr3/walking-xyz	1.383	0.722	0.087	0.019	**0.018**
	fr3/walking-rpy	1.292	0.805	0.161	0.059	**0.057**
	fr3/walking-halfsphere	1.014	0.723	**0.035**	0.312	0.42

TABLE V
MONOCULAR SLAM RESULTS ON TARTANAIR MONOCULAR BENCHMARK WITH METRIC: ATE[M]. BOLD STANDS FOR BEST RESULTS AND UNDERLINED FOR THE SECOND BEST. * MEANS THE RESULTS ARE GENERATED BY RUNNING THE OFFICIAL PRETRAINED MODEL IN OUR ENVIRONMENT TO ENSURE EVALUATION CONSISTENCY. X MEANS THE SYSTEM HAS FAILED HERE. - MEANS LACK OF RESULTS. WE ACHIEVE THE BEST RESULTS.

Monocular	MH000	MH001	MH002	MH003	MH004	MH005	MH006	MH007	Avg
ORB-SLAM [10]	1.30	**0.04**	2.37	2.45	X	X	21.47	2.73	-
DeepV2D [76]	6.15	2.12	4.54	3.89	2.71	11.55	5.53	3.76	5.03
TartanVO [77]	4.88	0.26	2.00	0.94	**1.07**	3.19	1.00	2.04	1.92
DROID-SLAM* [24]	**0.04**	0.69	0.03	0.02	3.73	0.62	0.38	**0.07**	0.70
Ours	0.63	0.06	**0.02**	**0.01**	2.80	**0.20**	**0.31**	0.45	**0.56**

TABLE VI
MONOCULAR SLAM RESULTS ON EUROC DATASET WITH METRIC: ATE[M]. † DENOTES VISUAL ODOMETRY METHODS. X MEANS THE SYSTEM HAS FAILED HERE. - MEANS LACK OF RESULTS. WE ACHIEVE COMPARABLE RESULTS ON PAR WITH DROID-SLAM.

		MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg
Deep/Hyb.	DeepFactors [2]	1.587	1.479	3.139	5.331	4.002	1.520	0.679	0.900	0.876	1.905	1.021	2.040
	DeepV2D [76]†	0.739	1.144	0.752	1.492	1.567	0.981	0.801	1.570	0.290	2.202	2.743	1.298
	DeepV2D (Tartan Air)†	1.614	1.492	1.635	1.775	1.013	0.717	0.695	1.483	0.839	1.052	0.591	1.173
	TartanVO [77]†	0.639	0.325	0.550	1.153	1.021	0.447	0.389	0.622	0.433	0.749	1.152	0.680
	D3VO + DSO [78]†	-	-	0.08	-	0.09	-	-	0.11	-	0.05	0.19	-
Classical	ORB-SLAM [10]	0.071	0.067	0.071	0.082	0.060	**0.015**	0.020	X	0.021	0.018	X	-
	DSO [79]†	0.046	0.046	0.172	3.810	0.110	0.089	0.107	0.903	0.044	0.132	1.152	0.601
	SVO [80]†	0.100	0.120	0.410	0.430	0.300	0.070	0.210	X	0.110	0.110	1.080	-
	DSM [81]	0.039	0.036	0.055	0.057	0.067	0.095	0.059	0.076	0.056	0.057	0.784	0.126
	ORB-SLAM3 [82]	0.016	0.027	0.028	0.138	0.072	0.033	0.015	0.033	0.023	0.029	X	-
	DROID-SLAM* [24]	**0.013**	**0.014**	0.022	**0.043**	**0.043**	0.037	0.012	**0.020**	0.017	**0.013**	**0.014**	**0.022**
	Ours	0.018	0.037	**0.020**	0.060	0.048	0.031	**0.008**	0.083	**0.016**	0.035	1.137	0.136

moving objects, changing light and various weather conditions. We use the official train/val/test split [29] for SLAM, and calculate ATE in all sequences.

KITTI [25] captures datasets in real-world traffic situations and ranges from freeways over rural areas to inner-city scenes with many static and dynamic objects, which is leveraged to perform online benchmarks for stereo, optical flow, object detection, and visual odometry [25].

TUM-RGBD [71] consists of notoriously difficult indoor scene datasets captured with a handheld camera due to rolling shutter artifacts, motion blur, and heavy rotation [24]. We use the dynamic sequences to evaluate the effectiveness of DeFlowSLAM in dynamic scenes and exploit the static sequences in the monocular setting.

EuRoc [72] contains 11 sequences with significant illumination changes and strong camera motion, which is widely used to evaluate SLAM systems.

B. Ablation Study

We conduct an ablation study to verify the effectiveness of our dual-flow representation in dynamic scenes and further explore the self-supervised training mechanism. We chose the Virtual KITTI2 dataset [28] for the ablation study.

Framework Design. To remove the interference of dynamic objects, an easy way to implement this is to add a module to DROID-SLAM that removes dynamic objects, which is considered a mask-removal approach. Instead, we use our dual-flow representation to keep more pixels for solving the pose, decomposing the static components even for regions of dynamic objects. Fig. 6 shows that DeFlowSLAM can exploit the dynamic pixels to solve the camera pose, compared with DROID-SLAM [24]. Tab. I shows that the dual-flow representation is better than the rough single-flow method. Notice that SS means self-supervised, SM means semi-supervised, SF means single flow, and DF means dual flow.

Supervision Policy. We explore the possibility of training DeFlowSLAM in a self-supervised manner. Compared with the supervised approach, which requires a large number of

TABLE VII
MONOCULAR SLAM RESULTS ON TUM-RGBD DATASET WITH METRIC: ATE[M]. X MEANS THE SYSTEM HAS FAILED HERE. - MEANS LACK OF RESULTS. WE ACHIEVE COMPARABLE RESULTS ON PAR WITH DROID-SLAM.

	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	avg
ORB-SLAM2 [75]	X	0.071	X	0.023	X	X	X	X	0.010	-
ORB-SLAM3 [82]	X	0.017	0.210	X	0.034	X	X	X	**0.009**	-
DeepTAM [83]	**0.111**	0.053	0.103	0.206	0.064	0.239	0.093	0.144	0.036	0.116
TartanVO [77]	0.178	0.125	0.122	0.349	0.297	0.333	0.049	0.339	0.062	0.206
DeepV2D [76]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
DeepV2D (TartanAir)	0.182	0.652	0.633	0.579	0.582	0.776	0.053	0.602	0.150	0.468
DeepFactors [2]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
DROID-SLAM* [24]	**0.111**	0.018	0.042	**0.021**	**0.016**	**0.049**	0.026	0.048	0.012	**0.038**
Ours	0.159	**0.016**	**0.030**	0.169	0.048	0.538	**0.021**	**0.039**	**0.009**	0.114

TABLE VIII
STEREO SLAM RESULTS ON TARTANAIR STEREO BENCHMARK WITH METRIC: ATE[M]. X MEANS THE SYSTEM HAS FAILED HERE. - MEANS LACK OF RESULTS. WE ACHIEVE COMPARABLE RESULTS ON PAR WITH DROID-SLAM.

Stereo	SH000	SH001	SH002	SH003	SH004	SH005	SH006	SH007	Avg
ORB-SLAM2 [75]	**0.05**	6.67	X	X	X	X	0.10	X	-
TartanVO [77]	2.52	1.61	3.65	0.29	3.36	4.74	3.72	3.06	2.87
DROID-SLAM* [24]	0.44	**0.08**	**0.13**	0.20	0.16	**3.29**	0.38	0.18	**0.61**
Ours	0.14	0.10	**0.13**	**0.08**	**0.09**	7.60	**0.03**	**0.02**	1.02

TABLE IX
STEREO SLAM RESULTS ON EUROC DATASET WITH METRIC: ATE[M]. - MEANS LACK OF RESULTS. WE ACHIEVE COMPARABLE RESULTS ON PAR WITH DROID-SLAM.

	MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg
D3VO + DSO [78]	-	-	0.08	-	0.09	-	-	0.11	-	0.05	-	-
ORB-SLAM2 [75]	0.035	0.018	0.028	0.119	0.060	0.035	0.020	0.048	0.037	0.035	-	-
VINS-Fusion [84]	0.540	0.460	0.330	0.780	0.500	0.550	0.230	-	0.230	0.200	-	-
SVO [80]	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.050	0.090	0.790	0.159
ORB-SLAM3 [82]	0.029	0.019	0.024	0.085	0.052	0.035	0.025	0.061	0.041	0.028	0.521	0.084
DROID-SLAM* [24]	**0.015**	0.013	0.035	**0.048**	**0.040**	0.037	**0.011**	**0.020**	**0.018**	**0.015**	**0.017**	**0.024**
Ours	0.018	**0.011**	**0.020**	0.061	0.042	**0.031**	**0.011**	0.090	0.039	0.037	0.478	0.075

complicated loss functions and is difficult to adjust the appropriate loss weights, and the semi-supervised approach, which requires additional masks for dynamic objects, the self-supervised approach achieves the transformation of different loss functions, allowing a less restricted adaptation to new scenarios and demonstrating better generalization ability. Tab. I demonstrates that the self-supervised approach can achieve comparable or better accuracy than the supervised approach.

Mask-Agg Setting. In self-supervised training, we introduce the Mask-Agg technique to reduce the influence of error matching of the pixels. Because the predicted dynamic masks in self-supervised training are not converged in the early stage of training, we only introduce Mask-Agg at the last $1/10$ epochs. Tab. I demonstrates that Mask-Agg helps to improve pose estimation, where MA means Mask-Agg.

Dynamic Threshold μ. In the self-supervised setting, we perform an ablation study of dynamic threshold μ on the Virtual KITTI2 (VK) dataset. Experiments demonstrate that $\mu = 0.5$ achieves better performance, as shown in Tab. II.

C. Generalization

After verifying the validity of our hypothesis, we trained our DeFlowSLAM on TartanAir dataset [29] from scratch and test our DeFlowSLAM on other popular SLAM datasets, such as Virtual KITTI2 (VKITTI2) [28] and KITTI [25] for autonomous driving scenarios with dynamic objects, EuRoc [72] for drones with strong motion and significant illumination changes, and TUM RGB-D [71] for hand-held SLAM with motion blur and heavy rotations. The results demonstrate that our model generalizes well to different datasets, shown in Fig. 5. Next, we show the effectiveness of DeFlowSLAM in several settings.

TABLE X
MOTION SEGMENTATION. WE SHOW THE MOTION SEGMENTATION RESULTS OF DEFLOWSLAM ON VIRTUAL KITTI2 DATASET.

Monocular	VK01	VK02	VK06	VK18	VK20
Ours	0.538585	0.528356	0.562788	0.739282	0.654025

D. Dynamic SLAM

We test the performance of the proposed DeFlowSLAM on sequences 09 and 10 from the KITTI dataset [25] and all sequences from the Virtual KITTI2 dataset [28]. Here we provide an additional trajectory for this experiment. The ATE results are shown in Table III. Compared with DynaSLAM [19] which uses Mask-RCNN for the dynamic environments and DROID-SLAM [24], our DeFlowSLAM is far more accurate and robust in dynamic scenes. We also perform evaluations on TUM RGB-D dynamic sequences with different dynamic proportions and the comparison results in Tab. IV

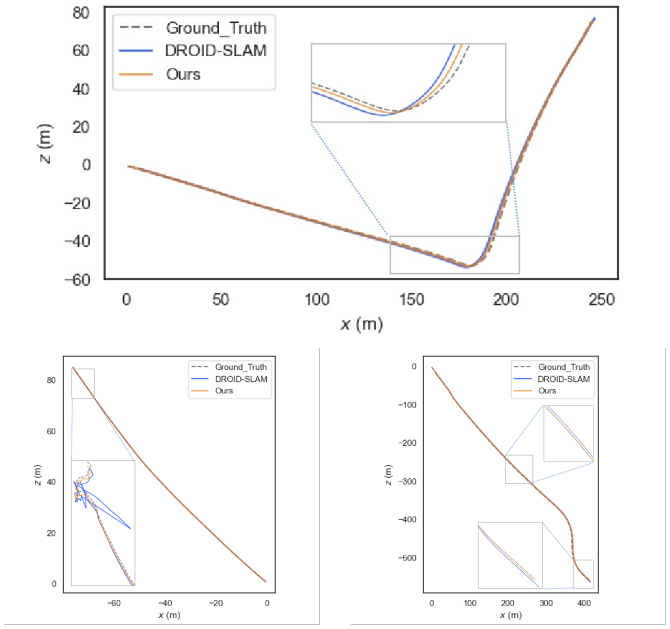

Fig. 7. **Comparison Trajectory Results of Our Method with DROID-SLAM on VKITTI2 Sequences 01 (Top), 02 (Bottom-Left), and 20 (Bottom-right).** In these dynamic sequences, our method performs better than DROID-SLAM, with better trajectory estimation results.

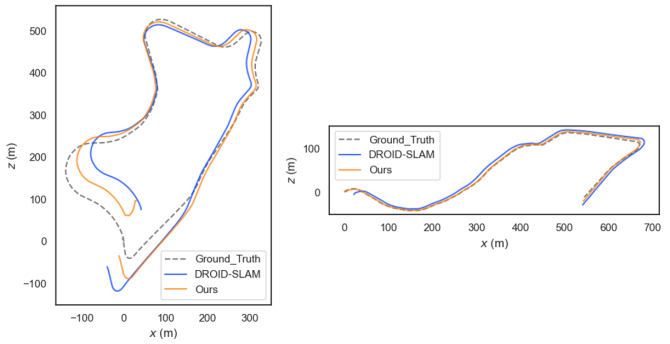

Fig. 8. **Trajectory Comparison between Our Method and DROID-SLAM.** In KITTI sequences 09 (Left) and 10 (Right), our trajectories are closer to the ground truth.

show that DeFlowSLAM achieves competitive and even best performance. Note that DVO SLAM [74], ORB-SLAM2 [75], and PointCorr [7] use the RGB-D dataset, while our method and DROID-SLAM [24] only use the monocular RGB dataset.

E. Monocular SLAM

In the monocular setting, we test our trained DeFlowSLAM on TartanAir test sets, EuRoC, and TUM RGB-D dataset. As shown in Tab. V, DeFlowSLAM achieves the best results. Tab. VI and Tab. VII show that our method achieves comparable even better results than the SOTA supervised method, DROID-SLAM [24] in most sequences. The results also demonstrate our DeFlowSLAM is more robust than the classical SLAM algorithms as they failed in many sequences. Specifically, we achieve an average ATE of 0.136m on EuRoC dataset in the monocular setting, and an average ATE of 0.114m on TUM-RGBD static sequences, outperforming most supervised methods.

F. Stereo SLAM

Under stereo setup, our trained DeFlowSLAM is also tested on TartanAir test dataset and EuRoC stereo dataset. Tab. VIII illustrates DeFlowSLAM achieves comparable results on par with DROID-SLAM [24], with an average ATE of 1.02m on the TartanAir stereo test dataset, outperforming TartanVO [85]. Tab. IX shows that DeFlowSLAM exhibits comparable results on the EuRoc dataset in the stereo setting with DROID-SLAM [24], outperforming most supervised methods and traditional SLAM, ORB-SLAM3 [82]. In most sequences, our method is on the same order of magnitude as DROID-SLAM [24], which shows the effectiveness of our method.

G. AR Applications

We conduct extensive experiments on AR applications to demonstrate the robustness of DeFlowSLAM. As shown in Fig. 9, we augment the original video with a virtual tree, a car, and a street lamp. Our DeFlowSLAM can deal with the dynamic objects in the scene very well while DROID-SLAM exhibits significant drifts (the red boxes).

H. Motion Segmentation

Although our method focuses on the SLAM system, our dual-flow representation can be well applied to motion segmentation. We simply set a threshold value (such as $\mu = 0.5$) for motion and visualize the pixel points of the dynamic field larger than this threshold to obtain the result of motion segmentation, as shown in Fig. 10 and Tab. X.

V. Discussion and Limitations

Although our system is versatile to many SLAM settings and is more robust to the challenges of dynamic scenes, it can be further improved in the following directions. DeFlowSLAM performs slightly weaker in some scenarios than DROID-SLAM, probably because we use a certain fixed dynamic threshold. We can explore dynamic threshold estimation methods for the challenges of different scenarios. As with droid-SLAM, we have high memory requirements for longer sequences and larger scenes, and our DeFlowSLAM system needs to run in segments. A lightweight and efficient SLAM system is a potential research direction. For some loop closure sequences, we can add loop closure constraints to reduce the drift. Our system focuses more on solving the camera pose, and the depth and optical flow obtained are only 1/8 of the original image size, which is not ideal for tasks like depth estimation and optical flow estimation. An efficient and versatile SLAM system is worthwhile researching. We can also explore dynamic object reconstruction with dynamic dense SLAM.

VI. Conclusions

We present a novel dual-flow representation that decomposes the optical flow into a static flow field caused by the camera poses and a dynamic field caused by the dynamic objects' motion. We present a dynamic update module with dual-flow representation to compose the full SLAM system,

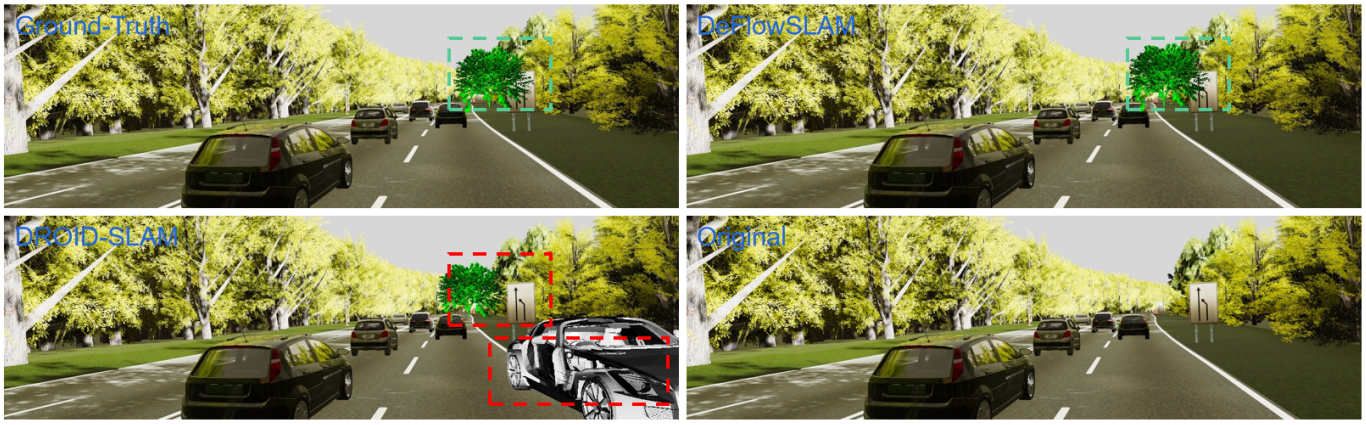

Fig. 9. **AR Application.** DeFlowSLAM deals with the dynamic objects in the scene very well while DROID-SLAM exhibits significant drifts (the red boxes). We augment the original video with a virtual tree, a car, and a street lamp. From left to right and from top to down: Ground-Truth, DeFlowSLAM, DROID-SLAM, Original Image.

Fig. 10. **Motion Segmentation.** From left to right: Ground-Truth dynamic mask, Predicted dynamic mask larger than the dynamic threshold $\mu = 0.5$.

DeFlowSLAM. We explore a self-supervised method to train our dual-flow-based dynamic dense SLAM system, which outperforms DROID-SLAM in highly dynamic scenes and achieves comparable performance in static and slightly dynamic scenes.

REFERENCES

[1] W. Ye, H. Li, T. Zhang, X. Zhou, H. Bao, and G. Zhang, "SuperPlane: 3D Plane Detection and Description from a Single Image," in *IEEE Virtual Reality and 3D User Interfaces*, 2021.

[2] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[3] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6243–6252.

[4] C. Wang, B. Luo, Y. Zhang, Q. Zhao, L. Yin, W. Wang, X. Su, Y. Wang, and C. Li, "Dymslam: 4d dynamic scene reconstruction based on geometrical motion segmentation," *IEEE Robotics and Automation Letters*, 2021.

[5] L. Yan, X. Hu, L. Zhao, Y. Chen, P. Wei, and H. Xie, "Dgs-slam: A fast and robust rgbd slam in dynamic environments combined by geometric and semantic information," *Remote Sensing*, 2022.

[6] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[7] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "Rgb-d slam in dynamic environments using point correlations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 373–389, 2020.

[8] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic slam: the need for speed," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2123–2129.

[9] J. Engel, T. Schops, and D. Cremers, "LSD-SLAM: Large-scale direct monocular slam," in *ECCV*, 2014.

[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[11] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *ICRA*. IEEE, 2014, pp. 15–22.

[12] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[13] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017.

[14] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 513–542, 2018.

[15] X. Wang, D. Maturana, S. Yang, W. Wang, Q. Chen, and S. Scherer, "Improving learning-based ego-motion estimation with homomorphism-based losses and drift correction," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 970–976.

[16] C. Wang, J. Yuan, and L. Xie, "Non-iterative SLAM," in *International Conference on Advanced Robotics (ICAR)*. IEEE, 2017, pp. 83–90.

[17] C. Wang, M.-C. Hoang, L. Xie, and J. Yuan, "Non-iterative RGB-D-inertial Odometry," *arXiv preprint arXiv:1710.05502*, 2017.

[18] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[19] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[20] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019.

[21] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[22] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[23] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2168–2177.

[24] Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," *Advances in neural information processing systems*, 2021.

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

[26] J. A. Villacorta-Atienza, C. C. Tapia, S. Díez-Hermano, A. Sánchez-Jiménez, S. Lobov, N. Krilova, A. Murciano, G. E. López-Tolsa, R. Pellón, and V. A. Makarov, "Static internal representation of dynamic situations reveals time compaction in human cognition," *Journal of advanced research*, vol. 28, pp. 111–125, 2021.

[27] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European conference on computer vision*. Springer, 2020, pp. 402–419.

[28] Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," *arXiv preprint arXiv:2001.10773*, 2020.

[29] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, "Tartanair: A dataset to push the limits of visual slam," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[30] X. Liu, C. R. Qi, and L. J. Guibas, "Flownet3d: Learning scene flow in 3d point clouds," *CVPR*, 2019.

[31] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang, "Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds," in *Computer Vision and Pattern Recognition (CVPR), 2019 IEEE International Conference on*, 2019.

[32] F. Huguet and F. Devernay, "A variational method for scene flow estimation from stereo sequences," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–7.

[33] C. Vogel, K. Schindler, and S. Roth, "Piecewise rigid scene flow," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1377–1384.

[34] Y. Zhang and C. Kambhamettu, "On 3d scene flow and structure estimation," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 2. IEEE, 2001, pp. II–II.

[35] M. Hornacek, A. Fitzgibbon, and C. Rother, "Sphereflow: 6 dof scene flow from rgb-d pairs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3526–3533.

[36] S. Hadfield and R. Bowden, "Kinecting the dots: Particle based scene flow from depth sensors," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2290–2295.

[37] Z. Lv, K. Kim, A. Troccoli, D. Sun, J. Rehg, and J. Kautz, "Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation," in *ECCV*, 2018.

[38] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, "Dense semi-rigid scene flow estimation from rgbd images," in *European Conference on Computer Vision*. Springer, 2014, pp. 567–582.

[39] F. Brickwedde, S. Abraham, and R. Mester, "Mono-sf: Multi-view geometry meets single-view depth for monocular scene flow estimation of dynamic traffic scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2780–2790.

[40] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[41] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[42] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.

[43] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, "Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 240–12 249.

[44] Y. Jiao, T. D. Tran, and G. Shi, "Effiscene: Efficient per-pixel rigidity inference for unsupervised joint learning of optical flow, depth, camera pose and motion segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 5538–5547.

[45] H. Zhan, C. S. Weerasekera, J.-W. Bian, R. Garg, and I. Reid, "Df-vo: What should be learnt for visual odometry?" 2021.

[46] S. Baur, D. Emmerichs, F. Moosmann, P. Pinggera, B. Ommer, and A. Geiger, "Slim: Self-supervised lidar scene flow and motion segmentation," in *International Conference on Computer Vision (ICCV)*, 2021.

[47] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular slam in dynamic environments," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013, pp. 209–218.

[48] H. Huang, W.-Y. Lin, S. Liu, D. Zhang, and S.-K. Yeung, "Dual-slam: A framework for robust single camera navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4942–4949.

[49] J. Vincent, M. Labbé, J.-S. Lauzon, F. Grondin, P.-M. Comtois-Rivet, and F. Michaud, "Dynamic object tracking and masking for visual slam," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4974–4979.

[50] G. B. Nair, S. Daga, R. Sajnani, A. Ramesh, J. A. Ansari, K. M. Jatavallabhula, and K. M. Krishna, "Multi-object monocular slam for dynamic environments," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 651–657.

[51] W. Ye, X. Lan, S. Chen, Y. Ming, X. Yu, H. Bao, Z. Cui, and G. Zhang, "Pvo: Panoptic visual odometry," 2022.

[52] M. Strecke and J. Stuckler, "Em-fusion: Dynamic object-level slam with probabilistic data association," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5865–5874.

[53] N. Brasch, A. Bozic, J. Lallemand, and F. Tombari, "Semantic monocular slam for highly dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 393–400.

[54] O. Wasenmüller, M. Meyer, and D. Stricker, "Augmented reality 3D discrepancy check in industrial applications," *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 125–134, 2016.

[55] M. Zollhöfer, C. Siegl, M. Vetter, B. Dreyer, M. Stamminger, S. Aybek, and F. Bauer, "Low-cost real-time 3D reconstruction of large-scale excavation sites," *J. Comput. Cult. Herit.*, vol. 9, no. 1, Nov. 2015. [Online]. Available: https://doi.org/10.1145/2770877

[56] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 227–234, 2019.

[57] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 Ieee/rsj International Conference on Intelligent Robots and Systems (iros)*. IEEE, 2017, pp. 1366–1373.

[58] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A scalable and consistent TSDF-based dense mapping approach," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 995–1002.

[59] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic, "Killingfusion: Non-rigid 3d reconstruction without correspondences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1386–1395.

[60] A. Bozic, M. Zollhofer, C. Theobalt, and M. Nießner, "Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7002–7012.

[61] F. Li, S. Yang, X. Yi, and X. Yang, "CORB-SLAM: a collaborative visual slam system for multiple robots," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer, 2017, pp. 480–490.

[62] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, vol. 36, no. 4, pp. 763–781, 2019.

[63] X. Liu, W. Ye, C. Tian, Z. Cui, H. Bao, and G. Zhang, "Coxgraph: Multi-robot collaborative, globally consistent, online dense reconstruction system," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8722–8728.

[64] F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers, "MonoRec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[65] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun, "Dense monocular depth estimation in complex dynamic scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4058–4066.

[66] C. Russell, R. Yu, and L. Agapito, "Video pop-up: Monocular 3d reconstruction of dynamic scenes," in *European conference on computer vision*. Springer, 2014, pp. 583–598.

[67] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2020.

[68] W. Zhou, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans Image Process*, 2004.

[69] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2020, pp. 1–7.

[70] Z. Teed and J. Deng, "Tangent space backpropagation for 3d transformation groups," in *Conference on Computer Vision and Pattern Recognition*, 2021.

[71] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The tum vi benchmark for evaluating visual-inertial odometry," in *IROS*. IEEE, 2018, pp. 1680–1687.

[72] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

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

[74] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 3748–3754.

[75] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[76] Z. Teed and J. Deng, "Deepv2d: Video to depth with differentiable structure from motion," *arXiv preprint arXiv:1812.04605*, 2018.

[77] W. Wang, Y. Hu, and S. Scherer, "Tartanvo: A generalizable learning-based vo," 2020.

[78] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1281–1292.

[79] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[80] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[81] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct sparse mapping," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1363–1370, 2020.

[82] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam," *arXiv preprint arXiv:2007.11898*, 2020.

[83] H. Zhou, B. Ummenhofer, and T. Brox, "Deeptam: Deep tracking and mapping," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 822–838.

[84] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3662–3669.

[85] W. Wang, Y. Hu, and S. Scherer, "Tartanvo: A generalizable learning-based vo," *arXiv preprint arXiv:2011.00359*, 2020.