Self-Supervised Monocular Scene Decomposition and Depth Estimation

Sadra Safadoust Fatma Güney
KUIS AI Center, Koç University
ssafadoust20@ku.edu.tr fguney@ku.edu.tr

Abstract

*Self-supervised monocular depth estimation approaches either ignore independently moving objects in the scene or need a separate segmentation step to identify them. We propose **MonoDepthSeg** to jointly estimate depth and segment moving objects from monocular video without using any ground-truth labels. We decompose the scene into a fixed number of components where each component corresponds to a region on the image with its own transformation matrix representing its motion. We estimate both the mask and the motion of each component efficiently with a shared encoder. We evaluate our method on three driving datasets and show that our model clearly improves depth estimation while decomposing the scene into separately moving components.*

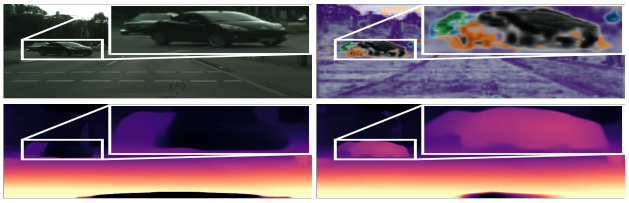


Figure 1. **Joint Scene Decomposition and Depth Estimation.** Monocular depth estimation methods assume a static scene by relying on the ego-motion to explain the scene and fail in foreground regions with independently moving objects (**bottom-left**, [16]). By decomposing the scene into a set of components, we estimate a separate rigid transformation for each component, representing its motion. This improves the results in regions with moving objects (**bottom-right**) while simultaneously recovering a decomposition of the scene, mostly corresponding to moving regions (**top-right**).

1. Introduction

Humans are remarkably good at decomposing a scene into semantically meaningful components and inferring the 3D structure of the scene. This ability is crucial for self-driving vehicles while navigating complex environments to avoid collision. In computer vision, we often address the problems of inferring 3D structure and semantics of the scene separately. Years of research in geometric computer vision has perfected matching pixels and using this as a cue for self-supervision recently in deep learning. Yet, most of these methods fail to attribute a semantic meaning to discovered geometry with some notable exceptions [22, 36, 26, 19, 35]. Similarly, semantic segmentation of the scene often solely relies on RGB images while complicated occlusion relationships between objects can easily be untangled with the help of 3D information.

There has been a great progress in learning of scene structure from a moving camera without using any labels. Self-supervised models can be trained to accurately estimate depth and the ego-motion of the vehicle by matching regularities in the scene across time. By following the progress in self-supervised monocular depth and ego-motion estimation [43, 16], we use view synthesis as an objective. In view synthesis, two models are trained to predict both the scene structure from a reference view and the relative ego-motion from the source view to the reference view. Then, the reference view can be synthesized by sampling pixels from the source view according to these estimations. While monocular methods need to associate semantics of the scene with structure to estimate depth, semantics are not typically represented explicitly in the model.

The lack of semantics become apparent when there are independently moving objects whose motion cannot be explained by the ego-motion alone (Fig. 1). The previous work either introduces additional models to estimate optical flow [41, 44, 7] and also motion masks [31, 28] or assumes a given segmentation mask marking semantic regions [7, 18] or instances of objects [5] on the image. While separate modelling of residual motion with flow or object motion improves the performance, it also increases the computational cost. In case of given segmentation masks, static objects are also segmented and processed separately. Furthermore, errors in the segmentation can propagate to the scene structure. In this paper, we propose to efficiently learn a decomposition of the scene into components with their own motion. As shown in Fig. 1, components correspond to moving regions by successfully separating moving objects.

Inspired by the success of SE3-Nets on point clouds [2, 3], we introduce **MonoDepthSeg** to segment a scene

into a set of components and predict the motion of each component from monocular image sequences. We fix the maximum number of components for a model and assume rigid body motion for each component. Instead of one transformation corresponding to ego-motion, we estimate a set of **SE**(3) transformations to account for the motion of independently moving objects in the scene as well. For this purpose, we modify the pose network's decoder to estimate a fixed number of poses, i.e. one per component. In parallel, a mask decoder learns to segment the image into regions whose motion is encoded by the corresponding pose. In order to model the close relationship between the structure and the semantics, mask and pose estimations are tightly correlated through a shared encoder. This also provides efficiency in comparison to a separate segmentation network entirely. We apply a regularization on the masks by encouraging a depth ordering between them.

We build on top of the recent self-supervised learning approaches for monocular depth estimation and show the effectiveness of learning to decompose the scene on KITTI [14, 13], Cityscapes [8], and DDAD [17] datasets. Without using an additional network other than the mask decoder, we improve depth estimation and obtain a decomposition of the scene into moving regions which is coherent both semantically and structurally. Our extensive experiments demonstrate that our joint formulation is able to resolve problems of the baseline Monodepth2 [16], especially in challenging foreground regions with moving objects on KITTI and significantly outperforms it on more complex Cityscapes and DDAD due to a large number of independently moving objects. In addition, our method is able to extract a decomposition of the scene into moving components which is consistent with the estimated structure (Fig. 1).

In summary, our contributions are as follows: (1) We introduce a framework for jointly estimating depth and a scene decomposition from monocular video without any ground-truth labels. (2) We estimate the motion of each independently moving object in addition to ego-motion to achieve a more accurate depth estimation.

2. Related Work

View synthesis was first proposed by enforcing consistency between two different views in the stereo setup [12]. By using Spatial Transformer Networks (STNs) [23], Monodepth [15] proposes a fully differentiable image reconstruction process by also optimizing for left-right consistency check. According to the depth estimation from a reference view, the source view can be geometrically warped in a differentiable way using STNs to synthesize the reference view. Then, the difference between the synthesized reference view and the original one can be used as a supervisory signal. Starting with SfMLearner by Zhou et al. [43], monocular depth estimation approaches generalize view synthesis to adjacent frames by also estimating the relative pose from one frame to the next. We generalize [43] to scenes with moving objects by applying the idea of K motion models to real-world image sequences that was initially proposed for point clouds in [2].

2.1. Self-Supervised Monocular Depth

There is an inherent scale ambiguity in monocular depth estimation and several approaches address this issue differently. Zhan et al. [42] use stereo pairs for joint training of single view depth and two-frame visual odometry. Wang et al. [39] argue the necessity of handling the scale problem for monocular input and apply an effective normalization on the depth estimation before computing the loss. Bian et al. [1] enforce scale-consistent predictions with a loss minimizing normalized differences in predictions over the entire sequence. Mahjourian et al. [29] propose to go beyond pixel-wise losses by minimizing the alignment error between the inferred 3D geometries of the scene.

There are a number of approaches that propose to jointly estimate monocular depth, optical flow and camera motion. The idea behind these approaches is to handle the camera motion as rigid flow and the remaining non-rigid object motion as residual flow as proposed in GeoNet [41]. DF-Net [44] proposes to enforce the consistency between depth and flow estimations geometrically in order to eliminate the propagation of errors from pose and depth estimation to flow. GLNet [7] proposes to capture multiple geometric constraints and further improves the results, especially with the epipolar constraint for flow.

Ranjan et al. [31] learn to segment static parts of the scene where optical flow is estimated using depth and camera motion. EPC++ [28] proposes a holistic 3D motion parser to segment moving objects and estimate motion maps for each as well as the background. These methods explicitly compute optical flow and motion masks with separate networks at the expense of increased computational complexity. We assume that the most common objects in outdoor scenes such as cars move rigidly but differently than the ego-motion. Based on this, we segment them and model their motion separately, but still in a unified architecture.

Recent approaches [16, 17] report improved results without using flow or motion masks. The improvements are due to architectural changes, better loss functions, and some careful consideration of the main assumption in self-supervised monocular training, i.e. a moving camera in a static scene. An auto-masking strategy is used to mask out pixels with the same motion as camera, which violate this assumption. Monodepth2 [16] proposes to handle occlusions by computing the photometric loss as the minimum over source views instead of averaging which leads to blurred results. In addition, while using multi-scale predictions, photometric error is always computed in the full

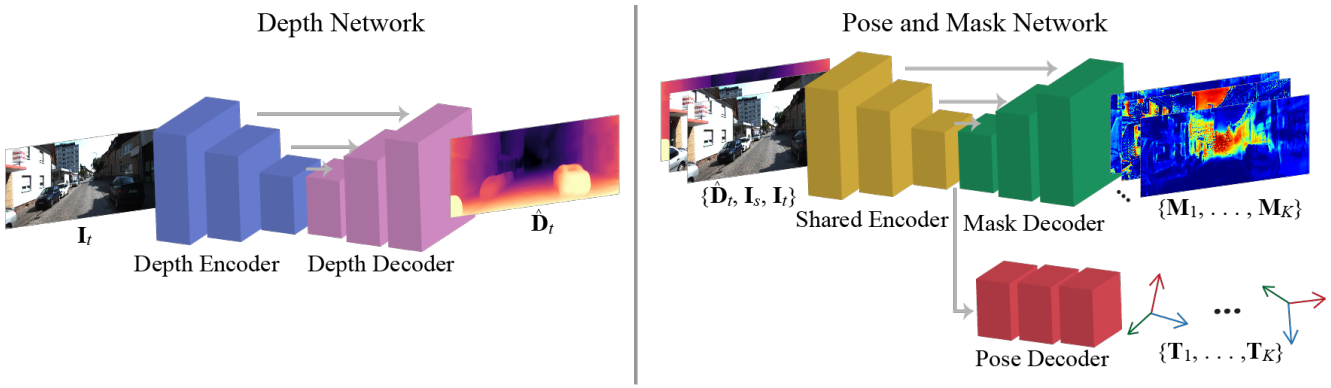


Figure 2. **MonoDepthSeg.** *Left:* The depth network outputs depth estimate $\hat{\mathbf{D}}_t$ for input $\mathbf{I}_t$. *Right:* Given $\hat{\mathbf{D}}_t$ and two consecutive frames $\mathbf{I}_s$ and $\mathbf{I}_t$, the shared encoder maps the input frames and the depth estimate to a common representation shared by the following two decoders. The mask decoder produces the same resolution K masks $\{\mathbf{M}_1, \ldots, \mathbf{M}_K\}$ with skip connections between the corresponding layers of the encoder. The pose decoder takes the same encoded representation and converts it into rigid transformations $\{\mathbf{T}_1, \ldots, \mathbf{T}_K\}$ corresponding to the masks. Given the set of rigid transformations and corresponding masks, we transform a 3D point as a convex combination of estimated transformations weighted according to estimated masks. The number of components, K, is a hyper-parameter of our model.

input resolution to avoid ambiguities at low resolution in low-texture regions. PackNet [17] uses 3D convolutions to learn detail-preserving representations via symmetric packing and unpacking blocks. In this paper, we follow the improvements proposed by recent approaches, specifically Monodepth2 [16], in terms of architecture and loss design.

2.2. Depth with Semantics

Joint training of semantics and depth has been explored before, especially in the context of indoor data where ground-truth depth is available for training [24]. For outdoor scenes, Cao et al. [4] propose to predict object motion and depth from stereo motion sequences by using object proposals as input. The proposed method can factor the scene into independently moving objects but require stereo sequences and limited supervision for object bounding boxes. In this paper, our goal is to generalize their success to monocular setup without any supervision.

For monocular data, Casser et al. [5] propose to estimate the motion of each object in addition to the ego-motion based on an initial instance segmentation. Motion estimation and the following warping are performed in masked regions according to given segmentation. We follow a similar approach by estimating the motion of each component but we also estimate the masks instead of relying on an initial segmentation. This way, we save computation not only by removing the need to initially segment the instances but also by focusing on the dynamic instances which require a separate modelling. While they train separate networks for ego-motion and object motion, our motion estimation is shared across components, leading to better efficiency.

Another two-stage training process for incorporating semantics is proposed by Guizilini et al. [18]. A pre-trained semantic segmentation network is used to guide the depth network via pixel-adaptive convolutions between the decoders. Semantic-aware features produce better estimations especially for dynamic objects. This proves the tight relationship between the semantics and the depth as we also explore in this paper. While [18] uses a pre-trained and fixed segmentation network for semantic guidance, we train a pose and mask network to predict corresponding poses and masks, with a shared encoder but separate decoders. Our approach can produce masks corresponding to separately moving regions on the image. We also perform a detailed evaluation according to motion segmentation on KITTI and Cityscapes and semantic classes on DDAD and show meaningful improvements for moving regions.

3. Methodology

Our monocular depth and segmentation network, **MonoDepthSeg**, jointly estimates the structure and a decomposition of the scene into moving components from monocular video sequences without using any labels. An overview of our approach is presented in Fig. 2. Our solution couples these two related tasks efficiently within a single framework. Given a source frame $\mathbf{I}_s$ and a target frame $\mathbf{I}_t$ as input where $\mathbf{I}_s, \mathbf{I}_t \in \mathbb{R}^{W \times H \times 3}$, we decompose the scene into K components where each component is represented with a mask and a rigid body motion:

$$\hat{\mathbf{I}}_s = \theta(\mathbf{I}_s, \mathbf{I}_t) \;\mid\; \theta := \{\mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_i\},\; i = 1 \ldots K \qquad (1)$$

where $\hat{\mathbf{I}}_s$ is the warped source image according to estimated parameters θ. For K components, θ denotes the set of rigid body transformations $[\mathbf{R}_i, \mathbf{t}_i] \in \mathbf{SE}(3)$ and corresponding masks $\mathbf{M}_i \in [0, 1]^{W \times H}$ representing the component i. In our setting, we assume that K is a hyper-parameter which limits the number of components that can be estimated.

Our framework consists of two parts: (i) a depth network which estimates per-pixel depth values (ii) a pose and mask network which divides the image into components and estimates a separate pose for each component. Finally, every pixel on the target frame is transformed according to estimated parameters, i.e. inverse warped by sampling from the source frame, and compared to the original target frame. The two networks are trained jointly and end-to-end.

3.1. Scene Decomposition

The depth network estimates the depth $\hat{\mathbf{D}}_t$ for every pixel on the target image $\mathbf{I}_t$. The pose network solves two related problems based on a shared representation, i.e. identifying components that move together and estimating transformation parameters for each component. We use the depth network of Monodepth2 [16] as explained in Section 3.4. In this section, we focus on the pose and mask network starting with the assumptions in our model.

In this work, we make two assumptions. First, we assume rigidly moving components whose motion can be explained with a rigid body transformation. This assumption, as explored before [5], typically holds in outdoor scenes where most of the moving objects are rigidly moving vehicles. Otherwise, it is an approximation for deformable objects such as pedestrians. Second, we have a pre-defined maximum number of components, K. Based on the second assumption, we can estimate the segmentation mask $\mathbf{M}$ by assigning each pixel into one of the K components. In order to make this process differentiable, we assign soft weights by allowing pixels to belong to more than one component.

Formally, $\mathbf{M}_i(\mathbf{p})$ denotes the probability of the pixel $\mathbf{p}$ belonging to the component i where the sum of all possible K values for the pixel $\mathbf{p}$ is equal to 1:

$$\mathbf{M}(\mathbf{p}) = \{\mathbf{M}_1(\mathbf{p}), \dots, \mathbf{M}_K(\mathbf{p})\} \mid \sum_{i=1}^{K} \mathbf{M}_i(\mathbf{p}) = 1 \quad (2)$$

Similar to [11, 34], we encourage masks to be layered according to a pre-defined depth order, e.g. the first mask appears first, then the second, and so on. The ordering helps to account for occlusions by putting the foreground objects before the background. We assign a scalar number d_i to each mask from 1 to K denoting its order. Then, we weight the mask logits by this number before applying softmax across K channels at every pixel:

$$\mathbf{M}_i(\mathbf{p}) = \frac{\mathrm{e}^{d_i \mathbf{M}_i'(\mathbf{p})}}{\sum_{j=1}^{K} \mathrm{e}^{d_j \mathbf{M}_j'(\mathbf{p})}} \quad (3)$$

where $\mathbf{M}'$ is the output of the mask decoder. With the help of this regularization, probabilities over K values are ordered according to a pre-defined depth ordering. As a baseline to the depth ordering, we experiment with another regularization to reduce the noise in the masks. We simply encourage mask values to be locally smooth by using an edge-aware smoothness term in the loss function.

3.2. Pose Estimation

We represent the motion of each component using a 3D rigid transformation $\mathbf{T} = [\mathbf{R}, \mathbf{t}] \in$ **SE(3)** which is composed of a rotation matrix $\mathbf{R} \in$ **SO(3)** and a translation vector $\mathbf{t} \in \mathbb{R}^3$. We first shortly introduce the preliminaries regarding the back-projection of a pixel into 3D, the application of transformation $\mathbf{T}$ in 3D, and the projection of the transformed 3D point into the other view. After introducing the general framework for establishing correspondences between different viewpoints under a rigid transformation, we generalize it to our framework with a set of rigid transformations and corresponding masks.

Preliminaries: Assuming a known intrinsic camera matrix $\mathbf{K}$, for a pixel $\mathbf{p}$ on the target image $\mathbf{I}_t$ and its depth value on $\hat{\mathbf{D}}_t$, the corresponding 3D point $\mathbf{x}$ can be computed as follows:

$$\mathbf{x} = \hat{\mathbf{D}}_t(\mathbf{p})\, \mathbf{K}^{-1}\, \mathbf{p} \quad (4)$$

Given an estimated rigid transformation $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$, we can transform the 3D point $\mathbf{x}$ and project it to find the corresponding point $\mathbf{p}'$ on the source image $\mathbf{I}_s$ as follows:

$$\mathbf{x}' = \mathbf{T}\, \mathbf{x} = \mathbf{R}\, \mathbf{x} + \mathbf{t} \qquad \mathbf{p}' = \mathbf{K}\, \mathbf{x}' \quad (5)$$

All operations are performed in homogeneous coordinates which are removed for simplicity.

Decomposition of Transformation: In this paper, we estimate K rigid transformations corresponding to the motion of K components. We build a tight relationship between the region of a component on the image, i.e. the mask, and the motion of that component through a shared encoder. Based on this shared representation, there are two separate decoders for predicting the masks and the transformations.

Given the set of predicted rigid transformations $\{\mathbf{T}_i\}$ and corresponding masks $\{\mathbf{M}_i\}$, we write (5) as a weighted combination of K predictions in a differentiable manner:

$$\mathbf{x}' = \sum_{i=1}^{K} \mathbf{M}_i(\mathbf{p})\, \mathbf{T}_i\, \mathbf{x} = \sum_{i=1}^{K} \mathbf{M}_i(\mathbf{p})\, (\mathbf{R}_i\, \mathbf{x} + \mathbf{t}_i) \quad (6)$$

where $\mathbf{x}'$ is the transformed 3D point $\mathbf{x}$ corresponding to the pixel $\mathbf{p}$ on the target image. As stated in [2], the resulting transformation as a convex combination of the transformed 3D points is not in **SE(3)** anymore. However, rigid transformations as frequently observed in outdoor scenes can be approximated using (6) as shown in our experiments.

3.3. Self-Supervised Training

For a pixel $\mathbf{p}$ on the target image $\mathbf{I}_t$, we first obtain the corresponding 3D point $\mathbf{x}$ as shown in (4). Then, we transform the 3D point $\mathbf{x}$ to $\mathbf{x}'$ according to the set of transformations and masks predicted by the decoders as shown in

(6). Finally, we project the transformed 3D point $\mathbf{x}'$ to the corresponding pixel $\mathbf{p}'$ on the source image. We repeat this process for all the pixels on the target image.

Following the monocular depth estimation approaches [43], we use a differentiable inverse warping process to reconstruct the target image $\mathbf{I}_t$ by sampling pixels from the source image $\mathbf{I}_s$. Since the projected pixel coordinates are continuous, we perform bilinear interpolation by using differentiable bilinear sampling mechanism proposed in the Spatial Transformer Networks [23]. This way, we can approximate the value of the warped image $\hat{\mathbf{I}}_s$ at location $\mathbf{p}$ in a differentiable manner. As a result, we obtain the corresponding points on the target image $\mathbf{I}_t$, i.e. the warped source image $\hat{\mathbf{I}}_s$, and the source image $\mathbf{I}_s$:

$$\mathbf{I}_t(\mathbf{p}) \approx \hat{\mathbf{I}}_s(\mathbf{p}) = \mathbf{I}_s(\mathbf{p}') \tag{7}$$

Our loss function minimizes the difference between the corresponding points over all pixels for pairs of images from a target view and a couple of source views, i.e. the previous and the next frame. For the photometric reconstruction loss, we follow [16] and use per-pixel minimum reprojection loss, and also incorporate the structural similarity (SSIM) loss [40] by setting α to 0.85:

$$\begin{aligned}\mathcal{L}_{\text{photo}}(\mathbf{p}) = \min_s \Big[& (1-\alpha)\,|\mathbf{I}_t(\mathbf{p}) - \hat{\mathbf{I}}_s(\mathbf{p})| \\ & + \frac{\alpha}{2}\Big(1 - \text{SSIM}(\mathbf{I}_t, \hat{\mathbf{I}}_s)(\mathbf{p})\Big)\Big]\end{aligned} \tag{8}$$

$\mathcal{L}_{\text{photo}}$ is minimized over the two source views considered, i.e. the previous and the next frames.

Similar to [39, 16], we define an edge-aware smoothness loss $\mathcal{L}_{smooth}$ over the mean-normalized inverse depth values. Our final loss $\mathcal{L}$ is a weighted sum of the loss functions defined over all pixels, divided by N, the number of pixels:

$$\mathcal{L} = \frac{1}{N}\sum_{\mathbf{p}} \mathcal{L}_{photo}(\mathbf{p}) + \lambda\, \mathcal{L}_{smooth}(\mathbf{p}) \tag{9}$$

3.4. Network Architecture

Depth network is based on the U-Net architecture [32], where we have an encoder-decoder network with skip connections. The encoder for the depth network is a ResNet50 architecture [20], which is pre-trained on ImageNet [33]. We use the same depth decoder proposed in Monodepth2 [16] where the output layer is a sigmoid converted to depth.

Given two consecutive frames, $\mathbf{I}_s$ and $\mathbf{I}_t$, and the depth estimate $\hat{\mathbf{D}}_t$ for $\mathbf{I}_t$, a second ResNet50 architecture is modified to accept two RGB images and the depth estimate as input. We concatenate the depth estimate as input to the pose and mask network similar to [27]. As shown on the right in Fig. 2, the output of this encoder is shared between the pose and mask decoders [2]. This way, we learn a shared representation for the structure and the decomposition of the scene in an efficient way. The pose decoder outputs K 6-DoF relative poses corresponding to K masks. Following [39, 16], we predict the rotation in axis-angle representation, and scale the rotation and translation outputs by 0.01.

Based on the same encoded representation as the pose decoder, the mask decoder outputs K masks, i.e. a K channel image where each channel represents the probability of each pixel belonging to that component [2]. We use transposed convolutions in the mask decoder to compute pixel-wise masks at the input resolution. We also add skip connections between the encoder and the mask decoder to better preserve boundaries. For the pose and mask network, we also experiment by initializing it from a pre-trained segmentation model with ResNet50 backbone, DeepLabv3+ [6].

4. Experiments

We evaluate **MonoDepthSeg** on KITTI [13], Cityscapes [8], and DDAD [17] datasets. In this section, we provide details of training and present our results. We first perform some ablation studies to justify our design choices and to show the effect of different values of hyper-parameters in our framework. We then present our results both quantitatively and qualitatively.

4.1. Datasets

KITTI: We train our model on the Eigen split [10] of KITTI dataset [14, 13]. We follow the pre-processing proposed in [43] and remove the static scenes which results in 39810 samples for training, 4424 for validation, and 697 for test set. We also present results using the improved ground-truth obtained with the technique proposed in [37] to compare our results to the latest approaches [17]. In the improved ground-truth, Uhrig et al. [37] remove the outliers in the LiDAR scans and increase the density of the ground-truth, resulting in high quality 652 depth maps, especially improving the original ground-truth for dynamic objects. We train only on KITTI dataset, without pre-training on Cityscapes as done by some previous work, and compare to the other approaches under the same training conditions.

Cityscapes: We also train and evaluate on the Cityscapes dataset [8]. For training, we choose 12 of 18 cities containing 30-frame videos from the training split, which results in 48802 samples. For validation, we choose 2 of the 3 cities containing 30-frame videos from the validation split, resulting in 6058 samples. We evaluate on all 1525 samples in the original test set. We use the provided disparity ground-truths obtained using the SGM stereo method [21], and convert them to depth for evaluation. Similar to KITTI, we evaluate depth for distances up to 80 m.

DDAD: We also conduct experiments on the recent DDAD dataset [17] capturing precise structure across images at

Components	Lower Better				Higher Better		
(K)	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
2	0.091	0.832	4.092	0.167	0.919	**0.967**	0.982
3	0.090	0.801	4.123	0.167	0.920	**0.967**	0.982
5	**0.084**	0.719	**3.943**	**0.163**	**0.925**	**0.967**	**0.983**
10	0.085	**0.620**	3.984	0.164	0.917	0.966	**0.983**

Table 1. **Number of Components.** This table shows our results according to changing the number of components (K). The best in each column is shown in bold and the second best is underlined. Increasing the number of components to $K = 5$ performs favorably compared to smaller number of components with $K = 2$ and $K = 3$ and a large number of components with $K = 10$.

Components	**Depth Ordering**	Lower Better				Higher Better		
(K)		Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
5	No	0.089	0.832	3.983	0.164	0.924	**0.967**	0.982
10	No	0.089	0.827	4.044	0.166	0.922	0.966	0.982
5	Yes	**0.084**	0.719	**3.943**	**0.163**	**0.925**	**0.967**	**0.983**
10	Yes	0.085	**0.620**	3.984	0.164	0.917	0.966	**0.983**

Table 2. **Depth Ordering.** This table shows the results of using depth ordering compared to simply encouraging smooth masks for regularization. Depth ordering improves the performance of our models over all metrics, especially for squared relative error (Sq Rel).

longer ranges up to 250 m, making it a more challenging and realistic benchmark for the task of depth estimation. We use 12350 training and 3850 validation samples from the front camera and evaluate on 3080 test images using the online evaluation server [9].

4.2. Training Details

We use Adam optimizer [25] with parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, with a learning rate of 10^{-4}. For KITTI, we fix the input resolution to 640×192 for efficiency and compare to recent approaches trained on resolutions not higher than that, and for Cityscapes we resize the images to 512×256. On both of these datasets, we train for 20 epochs where the learning rate is reduced to 10^{-5} for the last 5 epochs. For DDAD, we use images of size 640×384 (similar to [17]) and train for 30 epochs.

The weighting in the loss, i.e. λ is set to $0.001 \times$ scale where scale $\in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\}$. We use depth ordering for regularization by default, but also experiment with local smoothing of masks. We use the ResNet50 architecture [20] by default for both the depth network and the pose and mask network. For the pose and mask network, we experiment by using the pre-trained DeepLabv3+ architecture for the mask part [6]. We note that architectural improvements [17] or improvements due to higher input resolutions as observed in recent work [16, 17] are complementary to our approach and can be used to improve the results.

4.3. Ablation Study

In this section, we analyze different parts of our model including the maximum number of components, depth ordering for regularization, and using a pre-trained segmentation network for the pose and mask network. We train each model for 20 epochs on KITTI and present the results on the validation set using the original ground truth.

Number of Components: In order to evaluate the effect of scene decomposition, we perform an ablation study by changing the maximum number of components in the model. Specifically, we compare different values of K ranging over $2, 3, 5,$ and 10 components. As can be seen from Table 1, the number of components affects the results. A small number of components can be the limiting factor for the model as observed for $K = 2$ and $K = 3$. While increasing it to $K = 5$ improves the results for all the metrics, $K = 10$ performs a little worse than $K = 5$. The number of components clearly depends on the complexity of scenes in the dataset. This parameter might need to be adjusted for other datasets with more foreground objects but on KITTI, $K = 5$ components results in the best performance in almost all metrics among the values compared.

Depth Ordering: We investigate the effect of depth ordering on the performance of our model by comparing the results with and without depth ordering. When we remove the depth ordering, we still apply a regularization on the masks with simple mask smoothing in order to reduce the noise. As can be seen in Table 2, depth ordering improves the results for both $K = 5$ and $K = 10$. The difference between the depth ordering and smoothing masks is local versus global regularization. Depth ordering relates the masks over the channels corresponding to components by forcing similar depth-level pixels to cluster on the same mask while mask smoothing only locally enforces similar values on each mask independently. For this reason, we use depth ordering for regularization in the following experiments.

Pre-Trained Segmentation Network: In order to push the limits for the pose and mask network, we use a successful

Components (K)	**DeepLabv3+ Pre-Training** [6]	Lower Better				Higher Better		
		Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
5	No	0.087	0.697	4.031	0.169	0.918	0.964	0.981
10	No	0.092	**0.595**	4.111	0.174	0.908	0.961	0.981
5	Yes	**0.084**	0.719	**3.943**	**0.163**	**0.925**	**0.967**	**0.983**
10	Yes	0.085	0.620	3.984	0.164	0.917	0.966	**0.983**

Table 3. **Pre-trained Segmentation Network.** This table compares the results of using a pre-trained DeepLabv3+ model for semantic segmentation [6] as the pose and mask network. We modify it to add the pose decoder in addition to the mask decoder and train within our framework. For different number of components, using the modified DeepLabv3+ improves the results.

segmentation model, DeepLabv3+ [6], with the ResNet50 backbone and first pre-train it on Cityscapes dataset [8]. Then, we modify the DeepLabv3+ architecture by adding a pose decoder in addition to the mask decoder. Following our initial design, the encoder is still shared between the pose and the mask decoders. We train our framework with the modified DeepLabv3+ for the pose and mask network. Note that we initialize the encoder and the mask decoder from the DeepLabv3+ model pre-trained in a supervised manner on Cityscapes but our training is still self-supervised. As shown in Table 3, using the modified DeepLabv3+ model outperforms the original ResNet50 model for the pose and mask network. In the following experiments, we use the modified DeepLabv3+ model.

4.4. Results

We compare the performance of our method to the state-of-the-art by choosing the best performing configuration according to our ablation study, i.e. $K = 5$ with depth ordering and pre-trained segmentation network. For KITTI, we evaluate our method both with respect to the original ground-truth [10] and the improved ground-truth [37]. As shown in Table 5, our method performs comparably to the state-of-the-art methods PackNet [17] and Monodepth2 [16] in the monocular setting according to both the original and the improved ground-truth. Furthermore, we generate a scene decomposition jointly without conditioning on any segmentation map. Our improvements are particularly pronounced in foreground regions with moving objects (Fig. 3 and Table 4) but the overall performance is also improved with respect to the methods which use stereo pairs and rely on the output of a semantic [7] or instance segmentation network [5]. Furthermore, our joint scene decomposition formulation can compute the masks and poses of images in a single pass (9.057 samples per second), whereas using pre-computed masks as in [5], requires a separate pass for each object in the image (4.124 samples per second). While the state-of-the-art approach [17] uses an architecture with a large number of parameters due to 3D convolutions, we achieve similar results with a simpler architecture based on ResNet50. The followup work by the same authors improve the performance further by using supervision for semantic segmentation [18]. Note that we do not condition on the output of a segmentation network but automatically discover components which move independently (Fig. 3).

We compare our method with Monodepth2 [16] in moving regions. We extract moving regions using motion segmentation methods on KITTI [30] and Cityscapes [38]. Our method achieves significantly better results than Monodepth2 on Cityscapes in overall, and clearly outperforms it in moving regions on both KITTI and Cityscapes (Table 4). Please check the supplementary for the full tables.

Finally, we evaluate the performance of our model on the more complex DDAD dataset. Note that because the online evaluation server [9] does not allow any semantic supervision, our model is not initialized by a pre-trained segmentation network on this dataset. Despite this, our method outperforms the state-of-the-art method [17] and Monodepth2 [16] in all metrics including car and person classes, as can be seen in Table 6.

Data	Method	Abs Rel		RMSE	
		Moving	All	Moving	All
KITTI	Md2 [16]	0.143	**0.110**	5.949	**4.642**
	Ours	**0.138**	**0.110**	**5.796**	4.700
City-scapes	Md2 [16]	0.158	0.170	8.043	8.155
	Ours	**0.143**	**0.142**	**7.649**	**7.361**

Table 4. **Quantitative Results in Moving Regions and Overall.** Comparing our approach to Monodepth2 [16] (denoted as Md2) for moving regions and overall on KITTI and Cityscapes datasets.

5. Conclusion and Future Outlook

We proposed a joint scene decomposition and depth estimation model. We showed that joint formulation improves depth results, especially in moving foreground regions, compared to Monodepth2 as well as the other recent methods which condition on the output of a semantic or instance segmentation. Our model can discover components on the image which correspond to moving regions without any conditioning. The maximum number of components is fixed, however, outdoor scenes are not equally complex. For most scenes, ego-motion alone can explain the structure of the whole scene, as evidenced by the success of previous monocular depth estimation approaches. As we showed in

	Method	Supervision	Lower Better				Higher Better		
			Abs Rel	Sq Rel	RMSE	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original [10]	Godard et al. [16]	M	**0.110**	0.831	4.642	**0.187**	**0.883**	**0.962**	**0.982**
	Guizilini et al. [17]	M	0.111	**0.785**	**4.601**	0.189	0.878	0.960	**0.982**
	MonoDepthSeg (Ours)	M	**0.110**	0.792	4.700	0.189	0.881	0.960	**0.982**
	Casser et al. [5]	S+Ins	0.141	1.025	5.290	0.215	0.816	0.945	0.979
	Chen et al. [7]	S+Sem	0.118	0.905	5.096	0.211	0.839	0.945	0.977
	Guizilini et al. [18]	M+Sem	0.102	0.698	4.381	0.178	0.896	0.964	0.984
Improved [37]	Luo et al. [28]	M	0.120	0.789	4.755	0.177	0.856	0.961	0.987
	Godard et al. [16]	M	0.085	0.468	3.672	0.128	0.921	0.985	0.995
	Guizilini et al. [17]	M	**0.078**	**0.420**	**3.485**	**0.121**	**0.931**	**0.986**	**0.996**
	MonoDepthSeg (Ours)	M	0.085	0.458	3.779	0.131	0.919	0.985	**0.996**

Table 5. **Quantitative Results on KITTI.** This table compares our proposed approach **MonoDepthSeg** to previous approaches on KITTI dataset. The **Supervision** column shows different types of supervision used by different methods, **M** stands for monocular sequences, and **S** for methods that train on stereo pairs. We mark the methods which require semantic (**Sem**) or instance-level (**Ins**) information. We show the results for the input resolution 640×192, with pre-trained segmentation network, and by training on KITTI for depth. The best self-supervised method in each column is shown in bold and the second best is underlined.

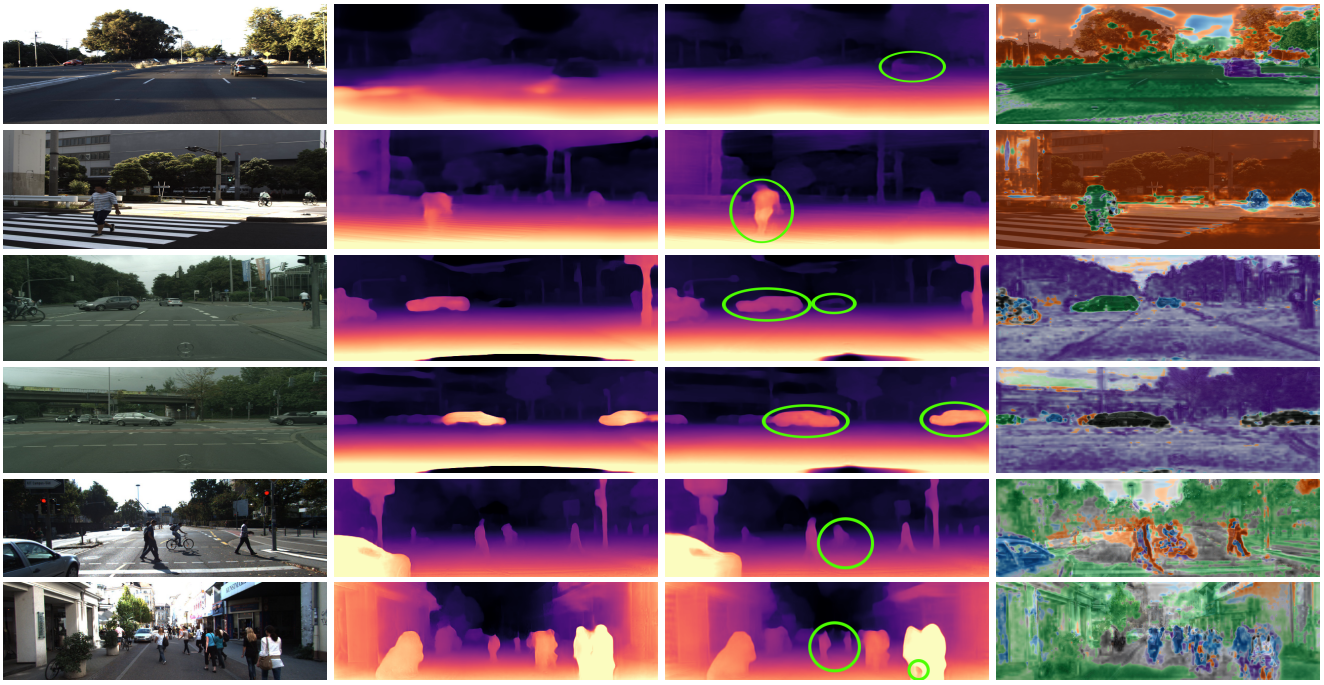


Figure 3. **Qualitative Results.** Each row shows from top-to-bottom: The input image (the first column), Monodepth2 [16] results as the baseline (the second column), our results with decomposition (the last two columns). In the last column, each channel of the mask is assigned a random color and the brightness level of the color is adjusted according to our estimation. The moving regions (circled in green) containing vehicles, bikes, and pedestrians are clearly segmented out as a separate component and assigned a different color. By learning a separate transformation for each component, we obtain more accurate depth estimations especially in moving foreground regions.

this paper, their performance can be improved, especially in moving foreground regions, by decomposing the scene and allowing other types of motion estimations. We plan to work on extending our formulation to an adaptive number of components according to scene complexity in the future.

Acknowledgements. This project has received funding from KUIS AI Center, TÜBİTAK (118C256), EU Horizon 2020 under Marie Skłodowska-Curie grant (898466).

Method	Abs Rel	RMSE	Car	Person
Guizilini et al. [17]	0.23	17.92	0.38	0.20
Godard et al. [16]	0.22	17.63	0.25	0.21
Ours	**0.19**	**16.61**	**0.24**	**0.17**

Table 6. **Quantitative Results on DDAD.** The *Car* and *Person* are shown using *Abs Rel*. Our method achieves the state-of-the-art results without using a pre-trained segmentation network.

References

[1] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 35–45, 2019.

[2] Arunkumar Byravan and Dieter Fox. SE3-Nets: Learning rigid body motion using deep neural networks. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 173–180, 2017.

[3] Arunkumar Byravan, Felix Leeb, Franziska Meier, and Dieter Fox. SE3-Pose-Nets: Structured deep dynamics models for visuomotor control. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 1–8, 2018.

[4] Zhe Cao, Abhishek Kar, Christian Hane, and Jitendra Malik. Learning independent object motion from unlabelled stereoscopic videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5594–5603, 2019.

[5] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, pages 8001–8008, 2019.

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[7] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 7063–7072, 2019.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] Dense depth for autonomous driving (DDAD) challenge. 2021. https://eval.ai/web/challenges/challenge-page/902/overview.

[10] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2366–2374, 2014.

[11] Matheus Gadelha, Rui Wang, and Subhransu Maji. Shape reconstruction using differentiable projections and deep priors. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[12] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 740–756, 2016.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017.

[16] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[17] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3D packing for self-supervised monocular depth estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[18] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. In *Proc. of the International Conf. on Learning Representations (ICLR)*, April 2020.

[19] Christian Häne, Nikolay Savinov, and Marc Pollefeys. Class specific 3d object shape priors using surface normals. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[21] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):328–341, 2007.

[22] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2005.

[23] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2017–2025, 2015.

[24] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 53–69, 2018.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv.org*, 1412.6980, 2014.

[26] Lubor Ladicky, Jianbo Shi, and Marc Pollefeys. Pulling things out of perspective. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[27] Hanhan Li, Ariel Gordon, Hang Zhao, Vincent Casser, and Anelia Angelova. Unsupervised monocular depth learning in dynamic scenes. *arXiv.org*, 2010.16404, 2020.

[28] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 42(10):2624–2641, 2019.

[29] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5667–5675, 2018.

[30] Eslam Mohamed, Mahmoud Ewaisha, Mennatullah Siam, Hazem Rashed, Senthil Yogamani, and Ahmad El-Sallab. Instancemotseg: Real-time instance motion segmentation for autonomous driving. *arXiv preprint arXiv:2008.07008*, 2020.

[31] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12240–12249, 2019.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[34] Sara Sabour, Andrea Tagliasacchi, Soroosh Yazdani, Geoffrey E. Hinton, and David J. Fleet. Unsupervised part representation by flow capsules. *arXiv.org*, 2011.13920, 2021.

[35] Nikolay Savinov, Christian Haene, Lubor Ladicky, and Marc Pollefeys. Semantic 3D reconstruction with continuous regularization and ray potentials using a visibility consistency constraint. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Ashutosh Saxena, Jamie Schulte, and Andrew Ng. Depth estimation using monocular and stereo cues. In *Proc. of the International Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2197–2203, 2007.

[37] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017.

[38] Johan Vertens, Abhinav Valada, and Wolfram Burgard. Smsnet: Semantic motion segmentation using deep convolutional neural networks. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, 2017.

[39] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2022–2030, 2018.

[40] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing (TIP)*, 13(4):600–612, 2004.

[41] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1983–1992, 2018.

[42] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 340–349, 2018.

[43] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1851–1858, 2017.

[44] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 36–53, 2018.