Addressing the Static Scene Assumption and the Scale Ambiguity in Self-Supervised Monocular Depth Estimation

by

Sadra Safadoust

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

August 11, 2022

Addressing the Static Scene Assumption and the Scale Ambiguity in Self-Supervised Monocular Depth Estimation

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Sadra Safadoust

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assist. Prof. Fatma Güney (Advisor)

_____

Prof. Yücel Yemez

_____

Assist. Prof. Matteo Poggi

Date: _____

To all the people that I love.

ABSTRACT

Addressing the Static Scene Assumption and the Scale Ambiguity in Self-Supervised Monocular Depth Estimation
Sadra Safadoust
Master of Science in Computer Science and Engineering
August 11, 2022

Self-supervised monocular depth estimation is the task of estimating per-pixel depth from a single image without any supervision. Typically, there are two networks to estimate depth and camera pose between consecutive frames, which are then used to reconstruct one view from another for self-supervision. There are two problems with this approach. Firstly, the scene is assumed to be static and the only motion is due to the camera, namely the static scene assumption, however, this is frequently violated in real-world driving scenarios. Due to this assumption, monocular depth methods struggle to produce accurate predictions in the moving regions of the scene. Current methods either ignore moving regions or require an additional instance segmentation input to identify and separately process moving regions. In this thesis, we first propose **MonoDepthSeg** to jointly estimate the depth and decompose the scene into moving regions to model the motion of dynamic objects. We show that going beyond the static scene assumption improves the accuracy of depth prediction, especially in moving regions. The second problem of self-supervised monocular depth estimation methods is the scale ambiguity. The estimated depth values are in an unknown scale which is typically handled with normalization with respect to the ground truth scale value during inference. We revisit the traditional paradigm of plane and parallax to address this issue and propose **DepthP+P** to estimate depth in metric scale. Our method shows promising results that are metric scale without any additional normalization.

ÖZETÇE

Yüksek Lisans Tez Başlığı
Sadra Safadoust
Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans
11 ağustos 2022

Kendi kendini denetleyen monoküler derinlik tahmini, herhangi bir denetim olmaksızın tek bir görüntüden piksel başına derinliği tahmin etme görevidir. Tipik olarak, ardışık kareler arasındaki derinliği ve kamera pozunu tahmin etmek için iki ağ vardır ve bunlar daha sonra kendi kendini denetleme için bir görünümü diğerinden yeniden elde etmek için kullanılır. Bu yaklaşımla ilgili iki sorun vardır. İlk olarak, sahnenin statik olduğu ve tek hareketin kameradan kaynaklandığı varsayılır, bu varsayım statik sahne varsayımı olarak isimlendirilir, ancak bu, gerçek dünyadaki sürüş senaryolarında sıklıkla ihlal edilir. Bu varsayım nedeniyle, monoküler derinlik yöntemleri, sahnenin hareketli bölgelerinde doğru tahminler üretmekte zorlanır. Mevcut yöntemler ya hareketli bölgeleri yok sayar ya da hareketli bölgeleri belirlemek ve ayrı ayrı işlemek için ek bir örnek segmentasyonu girdisi gerektirir. Bu tezde, ilk olarak **MonoDepthSeg**'in derinliği tahmin etmesini ve bununla beraber dinamik nesnelerin hareketini modellemek için sahneyi hareketli bölgelere ayırmasını öneriyoruz. Statik sahne varsayımının ötesine geçmenin, özellikle hareketli bölgelerde derinlik tahmininin doğruluğunu geliştirdiğini gösteriyoruz. Kendi kendini denetleyen monoküler derinlik tahmin yöntemlerinin ikinci sorunu ise ölçek belirsizliğidir. Tahmini derinlik değerleri, genellikle sonuç elde etme esnasında asıl referans ölçeği değerine göre normalleştirme ile ele alınan bilinmeyen bir ölçektedir. Bu sorunu ele almak için geleneksel düzlem ve paralaks paradigmasını yeniden ele alıyoruz ve metrik ölçekte derinliği tahmin etmek için **DepthP+P**'yi öneriyoruz. Yöntemimiz, herhangi bir ek normalizasyon olmaksızın metrik ölçekte sonuçlar üretebilmektedir.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my Advisor, Asst. Professor Fatma Güney for her continuous support and guidance. Working with her has been a delightful experience.

I would also like to thank Professor Yücel Yemez and Asst. Professor Matteo Poggi for agreeing to participate in my thesis committee and evaluating my work.

I would like to thank the members of the AVG Lab, especially Kaan Akan, who helped me a lot in my studies and research.

I want to thank my dear and lovely friend Müge Kural who has always been there for me and whose friendship has assisted me in my difficult times.

I want to thank all my friends in KUIS AI Lab, especially Ahmet Canberk Baykal, Burak Can Biner, Hilal Güven, Kerem Özfatura, and Mert Çökelek for all the fun times that we had together.

I want to thank Ata Arjomandi Fard, Aylar Arani Fard, Saba Aghdam Tabar, Amirhossein Pashaei and Reza Sabouri for all the beautiful and unforgettable memories we shared.

I would like to thank my mother, Ashraf, my father, Mehdi, my brother Sina, and my sister-in-law, Elshan, who have always supported me unconditionally. Their love and support was the only thing that I could always count on.

And finally, I want to thank my favourite musician Anneke Van Giersbergen. The sound of her voice and music has always been a relief. I hope she knows that her music is to stay forever.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Network
SSIM	Structural Similarity Index Measure
DoF	Degrees of Freedom
P+P	Plane and Parallax
3D	Three-Dimensional
2D	Two-Dimensional
Abs Rel	Absolute Relative Error
Sq Rel	Squared Relative Error
RMSE	Root Mean Squared Error
SILog	Scale Invariant Logarithmic Error

Chapter 1

INTRODUCTION

It is fairly easy for human beings to understand the 3D structure of a scene. We are capable of reasoning about around environment and can decompose it into regions in a semantically meaningful way. It is crucial for autonomous vehicles to have this ability in order to drive safely in different environments. Research in computer vision has proven the success of training deep neural networks for estimating depth. However, many of the current methods are supervised and rely on ground truth depth for training, which is costly to obtain. Another line of work uses a stereo setup that must be carefully calibrated. Both of these approaches are unable to utilise the huge amount of easily available unlabeled videos for training. On the other hand, self-supervised monocular depth estimation methods that do not rely on stereo supervision do not suffer from the mentioned limitations and, in practice, have been closing the gap with their supervised or stereo counterparts.

The main idea behind the current self-supervised monocular methods is to have a depth network for estimating the depth of the target view (the image for which we want to estimate depth). In addition, they utilise a pose network for estimating the ego-motion between a source view and the target view. The estimated depth and ego-motion are then used for sampling pixels from the source image to reconstruct the target frame. The difference between the reconstructed image and the original target image is then used as the supervisory signal to train the networks jointly. The main assumption in this approach is that the scene is static, which breaks down when there are dynamic objects in the scene. The estimated ego-motion cannot explain the motion of the moving objects, which will cause the depth estimations to be incorrect in these regions. To solve this issue, one of the early ideas was to estimate

the dynamic regions in the images and ignore them when calculating the loss [Zhou et al., 2017]. Recent studies use additional models to estimate motion masks [Ranjan et al., 2019, Luo et al., 2019] or optical flow [Yin and Shi, 2018, Zou et al., 2018, Chen et al., 2019] to explain the residual motion of the objects. Although these methods work better than simply ignoring the moving pixels, they come with an additional computational cost. Another approach is to use precomputed instance segmentation masks to identify objects of the scene [Casser et al., 2019]. Then another network is used to estimate the motion of each of the objects. However, these instance masks cannot differentiate between static and dynamic objects, and therefore, static objects are also processed needlessly. Moreover, any possible errors in instance masks will negatively affect the estimated scene structure.

Our approach to solving the dynamic objects in the scene is inspired by SE3-Nets on point clouds [Byravan and Fox, 2017, Byravan et al., 2018]. We propose "MonoDepthSeg" [Safadoust and Güney, 2021], a model that segments the scene into a predefined number of components and estimates each component's motion. Each component is assumed to have a rigid body motion. Instead of a single transformation for ego-motion, we estimate a fixed number of rigid body transformations corresponding to each component. This is easily implemented by having the decoder of the pose network output multiple poses. For estimating the components, we add a mask decoder to the pose network. The mask decoder uses depth ordering as a regularisation. The pose decoder and the mask decoder both share the same encoder. The shared encoder makes the architecture more efficient than having a separate segmentation network and also helps model the close relationship between the semantics of the scene and its structure. Intuitively, each component corresponds to an independently moving region in the scene. Therefore we can identify the moving regions in the scene and explain their motion instead of ignoring them. As a result, we are able to address the issue of dynamic objects in the scene.

A common limitation of the current methods is their inability to estimate metric depth. Their predicted depth map is in an unknown scale which is usually recovered using ground truth values at test time for each input image separately. We propose "DepthP+P" for metrically accurate depth estimation, which uses another

approach for synthesizing the target image. We use the traditional planar parallax framework [Sawhney, 1994, Irani and Anandan, 1996], decomposing the motion of the pixels into a planar homography and a residual parallax. Consider a plane in the scene and the homography associated with aligning it across the source and target images. The idea is that if we warp the source image using the homography so that the motion between the warped image and the target image is cancelled for the plane, then the residual image motion depends on two factors: (1) only the translation component of the camera motion and (2) the structure of the scene, represented by the depth of points and their perpendicular distance to the plane, i.e., height from the plane. Self-driving cars are a perfect application for this approach since a planar surface, i.e., is visible to the vehicle. It is important to note that in this paradigm, the planar surface can also be a virtual plane and does not necessarily need to be a real plane; however, using the road in the scene as the plane makes the implementations easier in practice.

In DepthP+P, we first align the road plane between the source and target images. This is achieved by calculating the homography between the road regions in two frames and then warping the source frame according to the homography to obtain the aligned image. As a result, the road region in the aligned image will match that of the target image. We can then calculate the remaining motion between the aligned image and the target image as follows. Using a depth network, we estimate a depth map and back-project the pixels to 3D coordinates. Assuming a known camera height, we can therefore compute the perpendicular distance of points to the road. Additionally, we have another network that, unlike the pose network, estimates only the translation between camera origins. Using the estimated values for depth, height from the plane, and the estimated camera translation, we calculate the residual parallax and synthesise the target frame from the aligned frame according to the calculated parallax.

The planar parallax paradigm for self-supervised monocular estimations has a number of advantages over the previous approach which estimates the 6-DoF motion. First, having no ambiguities associated with the rotation component of camera motion makes it much easier to optimise. Second, using this approach results in the

outputs being in the metric scale. Previous self-supervised monocular depth estimation methods typically estimate depth in an unknown scale. Then during the evaluation, they use ground truth depth data to scale their depth estimations such that the median of their estimated depth values is equal to the median of ground truth depth [Zhou et al., 2017]. However, our approach is able to output metric depth and translations due to the known height of the camera.

The structure of the thesis is organised as follows:

Chapter 2 discusses the related work on self-supervised monocular depth estimation approaches.

Chapter 3 introduces our MonoDepthSeg approach for handling the problem of dynamic objects in the scene.

Chapter 4 describes our DepthP+P model for estimating metrically accurate depth using the Planar Parallax paradigm.

Chapter 5 concludes the thesis and presents possible future research directions.

Chapter 2

RELATED WORK

2.1 Self-Supervised Monocular Depth

[Garg et al., 2016] were the first to propose a model for single image depth estimations using view synthesis as an objective. Instead of using ground truth depth for supervision, they use a CNN to estimate a depth map for the left image in the stereo setup. The estimated depth map and the known camera baseline are then used to inverse warp the right image to reconstruct the left. The photometric error between the reconstructed image and the original left image is used to train their network. Monodepth [Godard et al., 2017] proposes to synthesise images in a fully differentiable way using Spatial Transformer Networks (STNs) [Jaderberg et al., 2015]. Using only the left image in the stereo setup as its input to the CNN network, it estimates the disparities for both left and right images. Subsequently, it reconstructs both images using the other image's estimated disparities and also enforces left-right consistency between disparities. SfmLearner [Zhou et al., 2017] used another network to predict the relative pose between the source and target image, extending view synthesis to temporally consecutive images. Since it does not use any stereo supervision, its predicted depth is defined up to an unknown scale factor. Therefore, during evaluation, the estimated depth map is scaled such that its median is equal to the median of the ground truth depth values. In MonoDepthSeg, we extend this framework to estimate multiple poses corresponding to different moving regions in the scene. While this idea had been explored before for point clouds by SE3-Nets [Byravan and Fox, 2017], we apply it to real-world image sequences.

[Zhan et al., 2018] use stereo sequences and perform view synthesis using left-right pairs as well as temporally consecutive pairs, allowing them to benefit from stereo and monocular supervision at the same time. They also propose to use feature

reconstruction, in addition to frame reconstruction, as supervision. [Mahjourian et al., 2018] is another work that goes beyond pixel-wise reconstruction error for training. They propose to use a 3D point cloud alignment loss to enforce the estimated point clouds and the camera pose to be consistent temporally. [Wang et al., 2018] propose to compute the pose using the estimated depth via direct visual odometry in a differentiable way. They also normalise the estimated inverse depth map before calculating the loss to prevent the output scale from decreasing throughout the training.

Several methods propose to estimate optical flow, in addition to depth camera pose. After predicting the ego-motion, GeoNet [Yin and Shi, 2018] uses optical flow to estimate the remaining motion of objects. In order to prevent the errors of depth and ego-motion predictions from propagating to flow estimations, DF-Net [Zou et al., 2018] enforces the estimated optical flow and the flow induced by the depth and pose predictions to be consistent. GLNet [Chen et al., 2019] improves the performance by enforcing structural consistency in 3D space between the estimated depths for multiple views. They use the epipolar constraint for optical flow as an additional geometric constraint.

EPC++ [Luo et al., 2019] proposes a holistic 3D motion parser that uses predicted depth, pose, and optical flow to estimate segmentation masks for dynamic objects and their motion as well as background motion. [Ranjan et al., 2019] jointly train networks for estimating depth, ego-motion, optical flow, and motion segmentation. On the static regions, they utilise geometric constraints, while on dynamic objects, they use generic optical flow. Explicit estimation of motion masks and optical flow increases the computational complexity of the approach. Because our MonoDepthSeg approach focuses on the driving cases, we assume that moving objects, which will mainly be cars, have a rigid motion different than the ego-motion. As a result, in MonoDepthSeg, we segment the moving objects and estimate their motions in a unified architecture.

Some methods maintain the original framework with a depth and a pose network but propose innovative design choices, improved network architectures, and better loss functions for improving the performance. When predicting depth at multiple

scales, Monodepth2 [Godard et al., 2019] upsamples the low-scale predicted depths to the size of the input image and calculates the loss at that original image scale afterward. Another consideration by Monodepth2 [Godard et al., 2019] is that some pixels that are close to image boundaries in the target view may not be visible in the source view because of ego-motion. Moreover, some pixels might be occluded in the source view but visible in the target view. Estimating the correct depth for such pixels will probably not reconstruct them correctly, leading to a high reconstruction loss. To address this issue, when synthesizing the target image from multiple views, Monodepth2 [Godard et al., 2019] calculates the minimum of reprojection errors per pixel instead of averaging them. This modification also helps prevent blurry predictions. PackNet [Guizilini et al., 2020a] proposes to learn to preserve spatial information by modifying the depth network's architecture and using symmetrical 3D packing and unpacking blocks for depth estimation.

2.2 Depth with Semantics

[Jiao et al., 2018] have explored joint training of depth and semantics on the indoor data with available ground truth depth and semantic labels. [Cao et al., 2019] use stereo sequences and object proposals as input and estimate scene depth and object motion. Although their approach can decompose the scene into independently moving objects, they rely on annotations for object bounding boxes and stereo supervision. On the other hand, MonoDepthSeg achieves that goal without any supervision in a monocular framework.

For monocular sequences, struct2depth [Casser et al., 2019] uses a trained Mask R-CNN [He et al., 2017] to obtain object masks and then estimates the motion of those objects in addition to ego-motion. Our MonoDepthSeg approach is similar in that we also estimate masks for dynamic objects and their motion; however, instead of relying on an external network to compute the masks, we learn to estimate them. As a result, our method is computationally cheaper because it does not need to segment the objects initially. Moreover, unlike struct2depth, which predicts ego-motion and object motion using different networks, we use a single network and

share the motion estimation across the estimated components, which again is more efficient. [Guizilini et al., 2020b] use a fixed pretrained segmentation network to guide the depth network to learn semantic-dependent geometric representations. This guidance is achieved using pixel-adaptive convolutions between the decoders of the semantic segmentation network and the depth network.

2.3 Scale Ambiguity

Self-supervised monocular depth prediction methods suffer from the scale ambiguity issue, where the depth and pose outputs are in an ambiguous and unknown scale. The median scaling technique used by many previous is not actually a solution to this issue because it depends on ground truth depth in test time. [Bian et al., 2019] propose a loss function to minimise normalised differences of depth maps across the whole sequence, making the predictions globally consistent in their scale. However, this only forces the predictions to be at the same scale, but that particular scale is still unknown, and the median scaling is still needed when evaluating.

There are a number of monocular methods capable of producing metrically accurate depth estimations. [Roussel et al., 2019] use a network which was pretrained with stereo pairs on a dataset and finetune it on another dataset while maintaining the correct scale. [Guizilini et al., 2020a] propose using ground truth camera velocity and the frame timestamps to enforce the predictions to be scale aware. [Bartoccioni et al., 2021] use sparse LiDAR to supervise the depth estimations. However, all of these methods depend on ground truth data from extra sensors during training.

A few other methods do not need additional supervision and can output metrically accurate estimations only using the camera height information, similar to our DepthP+P approach. DNet [Xue et al., 2020] estimates the ground plane at test time and, using the camera height, can recover the correct scale of the predictions. However, it relies on a ground plane being visible during inference. In other words, their model does not learn to generate outputs at the metric scale. Instead, they estimate depths at an unknown scale similar to other methods and then use a module at test time to recover the scale of the predictions. [Wagstaff and Kelly, 2021] propose

a method that uses the known height of the camera to learn the correct scale during training. Apart from the depth and the pose networks, they use another network for plane segmentation. Their architecture relies on a three-staged training procedure for obtaining metric depth. First, their depth and pose networks are trained to predict unscaled outputs. These trained networks are then used for training the plane segmentation network. Finally, using the pretrained plane segmentation network, they train their depth and pose networks again to produce metrically accurate outputs. Our DepthP+P method is similar to [Wagstaff and Kelly, 2021] in that we also learn to produce metrically accurate outputs during training. However, our approach is much simpler and does not require a multi-stage process, nor does it depend on the ground plane being visible at test time, differently than DNet [Xue et al., 2020].

2.4 Planar Parallax

The Planar Parallax paradigm, also called Plane + Parallax (P+P), has been used for understanding the 3D scene structure from multiple views by decomposing the motion into a planar homography and a residual parallax. [Sawhney, 1994] derived a formulation for calculating the residual parallax using the depth of the points and their distance to a common plane. [Irani and Anandan, 1996] use this formulation to derive a rigidity constraint between pairs of points over multiple views. Using this constraint, they detect the dynamic objects in the scene. [Irani et al., 1998] obtain trifocal constraints and use them to propose a simple method for new view synthesis. [Irani et al., 2002] propose a method to generalise the planar parallax method to more than two uncalibrated views.

There are a number of more recent approaches that use the traditional planar parallax paradigm. MR-Flow [Wulff et al., 2017] uses this paradigm for refining the optical flow estimations. [Chaney et al., 2019] use P+P for predicting the height of points in the scene with event-based cameras.

In DepthP+P, we propose to use the P+P formulation for view synthesis to train a metrically accurate depth estimation model in a self-supervised way.

Chapter 3

SELF-SUPERVISED MONOCULAR SCENE DECOMPOSITION AND DEPTH ESTIMATION

Existence of moving objects in the scene violates the static scene assumption of self-supervised monocular depth estimation network and results in their degraded performance. To address this limitation, we propose MonoDepthSeg, a method that, by using monocular sequences jointly estimates the depth and decomposes the scene into independently moving components without any ground truth annotations, as shown in Figure 3.1. Our model is able to estimate more accurate depth results for moving regions in the scene by identifying the dynamic objects and their motion. We illustrate the overview of our architecture in Figure 3.2. Our architecture consists of a depth network, and a mask and pose network. Similar to other works, the depth network outputs per-pixel depth values. The mask and pose networks is responsible for segmentation of the scene into different components and estimating the motion of each component. In our approach, the two related tasks of segmentation and depth estimation are coupled in a unified framework efficiently.

3.1 Methodology

Given a target image $\mathbf{I}_t$ and a source image $\mathbf{I}_s$ as input where $\mathbf{I}_t, \mathbf{I}_s \in \mathbb{R}^{H \times W \times 3}$, with H and W being the height and width of the images, the scene is decomposed into K components where each component is represented with a mask and a rigid body motion:

$$\hat{\mathbf{I}}_s = \theta(\mathbf{I}_t, \mathbf{I}_s) \;\; | \;\; \theta \coloneqq \{\mathbf{M}_i, \mathbf{R}_i, \mathbf{t}_i\}, \;\; i = 1 \dots K \tag{3.1}$$

where θ are the predicted parameters, denoting the set of K masks $\mathbf{M}_i \in [0,1]^{H \times W}$ and the corresponding rigid body transformations $[\mathbf{R}_i, \mathbf{t}_i] \in \mathbf{SE}(3)$, with $\mathbf{R}_i \in \mathbb{R}^{3 \times 3}$

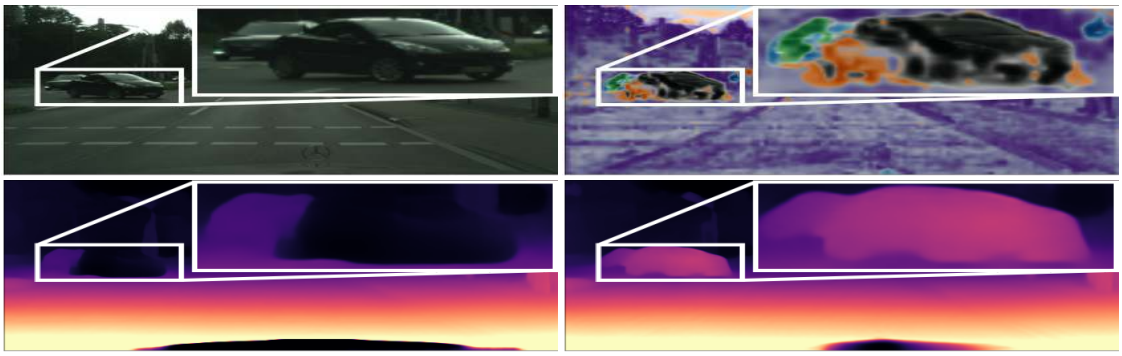

Figure 3.1: **Joint Scene Decomposition and Depth Estimation.** The main assumption in most monocular depth estimation methods is that the scene is static. They use the ego-motion to explain the whole scene, resulting in incorrect estimations for the dynamic objects whose motion cannot be explained by the ego-motion alone (**bottom-left**, [Godard et al., 2019]). MonoDepthSeg decomposes the scene into a number of components, predicting a rigid-body transformation representing its motion for each one. This leads to improved depth estimation results in dynamic regions (**bottom-right**) and, at the same time, produces a decomposition of the scene, which mostly corresponds to dynamic regions (**top-right**).

and $\mathbf{t}_i \in \mathbb{R}^3$ being the rotation and translation associated with component i. $\hat{\mathbf{I}}_s$ is the source image warped according to the predicted parameters θ. In our method, K is a hyper-parameter limiting the number of components that our model can predict.

The architecture of MonoDepthSeg consists of two networks: (1) a depth network that outputs a depth map for a given target image and (2) a mask and pose network that, by taking the source and the target image as input, decomposes the scene into components and predicts a pose separately for each of them. We synthesise the target image by sampling pixels from the source image according to the outputs of these two networks. The difference between the target image and the synthesised image is used as supervision for training both networks jointly in an end-to-end fashion.

3.1.1 Scene Decomposition

In this section, we discuss our mask and pose network in detail. MonoDepthSeg has two main assumptions. First, we assume the scene is composed of components

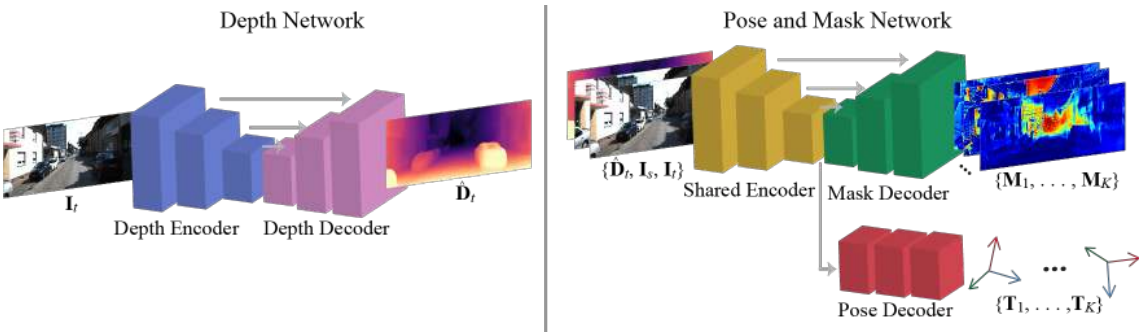

Figure 3.2: **MonoDepthSeg.** *Left:* The depth network takes the target image $\mathbf{I}_t$ as input and outputs depth estimate $\hat{\mathbf{D}}_t$. *Right:* Given the two consecutive frames $\mathbf{I}_s$ and $\mathbf{I}_t$ and the depth estimate $\hat{\mathbf{D}}_t$, the shared encoder maps the depth estimate and input images into a common representation that is shared by the following two decoders. The mask decoder outputs the same resolution K masks $\{\mathbf{M}_1, \ldots, \mathbf{M}_K\}$. The pose decoder takes the same encoded representation and outputs K rigid-body transformations $\{\mathbf{T}_1, \ldots, \mathbf{T}_K\}$ corresponding to the masks. Given the masks and the corresponding rigid-body transformations, a 3D point is transformed as a convex combination of transformations weighted by masks. K, is a hyper-parameter controlling the number of estimated components in our model.

whose motion is explainable with $\mathbf{SE}(3)$ transformations. In driving scenarios, this assumption usually holds because most of the dynamic objects are cars and vehicles that move rigidly. For objects such as pedestrians, which are deformable, rigid-body motions can also be used as an approximation. The second assumption is that there is a predefined maximum number of components, K. Based on this assumption, each pixel can be assigned to one of the K components. By soft assignments and allowing pixels to belong to more than only one component, we make this process differentiable.

Concretely, $\mathbf{M}_i(\mathbf{p})$ represents the probability that the pixel $\mathbf{p}$ belongs to the i'th component. Note that the K probabilities for the pixel $\mathbf{p}$ sum to 1. Therefore we have:

$$\mathbf{M}(\mathbf{p}) = \{\mathbf{M}_1(\mathbf{p}), \ldots, \mathbf{M}_K(\mathbf{p})\} \mid \sum_{i=1}^{K} \mathbf{M}_i(\mathbf{p}) = 1 \tag{3.2}$$

Our masks are encouraged to be layered based on a predefined ordering [Gadelha et al., 2019, Sabour et al., 2021]. This has the benefit of encouraging the foreground objects to be put before the background and therefore handling the occlusion issue.

Each mask is assigned a scalar d_i representing its order. The mask logits are weighted by this scalar before applying the softmax for calculating the probabilities for each pixel:

$$\mathbf{M}_i(\mathbf{p}) = \frac{e^{d_i \mathbf{M}_i'(\mathbf{p})}}{\sum_{j=1}^{K} e^{d_j \mathbf{M}_j'(\mathbf{p})}} \tag{3.3}$$

where $\mathbf{M}'$ is the output of the mask decoder. We experiment with another regularisation technique in Section 3.2.3 as an alternative to this depth ordering approach.

3.1.2 Pose Estimation

In our model, the motion of each component is represented with a rigid-body transformation $\mathbf{T} = [\mathbf{R}, \mathbf{t}] \in \mathbf{SE(3)}$, consisting of a rotation matrix $\mathbf{R} \in \mathbf{SO(3)}$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$. This section will briefly discuss the preliminaries of back projecting pixels from the target view to 3D coordinates, transforming them using the rigid-body transformation T, and projecting the 3D points into the source view. We will extend it to our framework, where we have multiple components with the corresponding transformations.

Preliminaries: Consider a target image $\mathbf{I}_t$ and a source image $\mathbf{I}_s$. A pixel $\mathbf{p}$ on the target image $\mathbf{I}_t$ can be back-projected to its corresponding 3D point $\mathbf{x}$ as follows:

$$\mathbf{x} = \mathbf{D}(\mathbf{p})\mathbf{K}^{-1}\ \mathbf{p}. \tag{3.4}$$

where $\mathbf{K}$ and $\mathbf{D}$ are the 3×3 camera intrinsic matrix and the depth map for the target image, respectively. The 3D point $\mathbf{x}$ can be transformed and projected on the source frame $\mathbf{I}_s$ as follows:

$$\mathbf{x}' = \mathbf{T}\ \mathbf{x} = \mathbf{R}\ \mathbf{x} + \mathbf{t} \qquad\qquad \mathbf{p}' = \mathbf{K}\ \mathbf{x}' \tag{3.5}$$

where $\mathbf{T}$ is the rigid body transformation between corresponding to the relative pose between $\mathbf{I}_t$ and $\mathbf{I}_s$ which consists of the 3×3 rotation matrix $\mathbf{R}$ and the 3D translation vector $\mathbf{t}$. The point $\mathbf{x}'$ is the 3D point obtained by applying the transformation $\mathbf{T}$ on $\mathbf{x}$. By projecting $\mathbf{x}'$ using the camera intrinsic $\mathbf{K}$, we obtain pixel $\mathbf{p}'$ on the source view $\mathbf{I}_s$ which corresponds to $\mathbf{p}$ on the target image $\mathbf{I}_t$. Note

that the homogenous coordinates have been removed for brevity. Assuming a static scene, $\mathbf{I}_t$ can be reconstructed by sampling pixels from $\mathbf{I}_s$ and obtaining the warped image $\hat{\mathbf{I}}_s$ such that $\hat{\mathbf{I}}_s(\mathbf{p}) = \mathbf{I}_s(\mathbf{p}')$.

Decomposition of Transformation: MonoDepthSeg estimates K components and predicts K rigid-body transformations explaining their motion. Using a shared encoder, the components and their motion are tightly correlated. The output of the shared encoder is fed into two separate decoders for estimating masks and their poses.

Using the estimated masks $\{\mathbf{M}_i\}$ and the corresponding transformations $\{\mathbf{T}_i\}$, Equation (3.5) is written in a differentiable way as weighted combination of K estimations:

$$\mathbf{x}' = \sum_{i=1}^{K} \mathbf{M}_i(\mathbf{p})\ \mathbf{T}_i\ \mathbf{x} = \sum_{i=1}^{K} \mathbf{M}_i(\mathbf{p})\ (\mathbf{R}_i\ \mathbf{x} + \mathbf{t}_i) \tag{3.6}$$

Although the final transformation for obtaining $\mathbf{x}'$ is not in **SE(3)** [Byravan and Fox, 2017], our experiments show that the rigid transformations that usually happen in driving scenarios can be approximated using Equation (3.6).

3.1.3 Self-Supervised Training

For every pixel $\mathbf{p}$ on the $\mathbf{I}_t$, we use our estimated depth to backproject it according to (3.4), and then transform the 3D point $\mathbf{x}$ using Equation (3.6) according to predicted masks and transformations to obtain the $\mathbf{x}'$. We then project $\mathbf{x}'$ to the corresponding pixel $\mathbf{p}'$ on $\mathbf{I}_s$. Similar to previous methods, we will inverse warp the source frame $\mathbf{I}_s$ to synthesise the target frame $\mathbf{I}_t$. Therefore, for each pixel $\mathbf{p}$ on the $\mathbf{I}_t$, we need to sample the pixel $\mathbf{p}'$ from $\mathbf{I}_s$. However, the calculated $\mathbf{p}'$ is not necessarily an integer number. To solve this issue, we use differential bilinear sampling [Jaderberg et al., 2015]. Therefore, the value of the pixel $\mathbf{p}$ for the warped image $\hat{\mathbf{I}}_s$ is approximated in a differentiable way:

$$\mathbf{I}_t(\mathbf{p}) \approx \hat{\mathbf{I}}_s(\mathbf{p}) = \mathbf{I}_s(\mathbf{p}') \tag{3.7}$$

The difference between $\mathbf{I}_t$ and $\hat{\mathbf{I}}_s$ will act the supervisory signal for training our model. We define our photometric loss function as the linear combination of the

L1 distance and the structural similarity (SSIM) [Wang et al., 2004] to minimise the difference between the target image $\mathbf{I}_t$ and the reconstructed image $\hat{\mathbf{I}}_s$. For every target image we use two source images, i.e. the previous and the next frame. Following [Godard et al., 2019], we use the per-pixel minimum reprojection loss, so our photometric loss for every pixel is defined as follows:

$$\mathcal{L}_{\text{photo}}(\mathbf{p}) = \min_s \Big[(1-\alpha) \left| \mathbf{I}_t(\mathbf{p}) - \hat{\mathbf{I}}_s(\mathbf{p}) \right| + \frac{\alpha}{2} \Big(1 - \text{SSIM}(\mathbf{I}_t, \hat{\mathbf{I}}_s)(\mathbf{p}) \Big) \Big] \tag{3.8}$$

where we set $\alpha = 0.85$. We also define $\mathcal{L}_{smooth}$ as an edge-aware smoothness loss over the mean-normalised inverse depth estimates [Wang et al., 2018] to encourage the depth predictions to be locally smooth. Our total loss function is a combination of $\mathcal{L}_{smooth}$, $\mathcal{L}_{photo}$ averaged over all N pixels:

$$\mathcal{L} = \frac{1}{N} \sum_{\mathbf{p}} \lambda \; \mathcal{L}_{smooth}(\mathbf{p}) + \mathcal{L}_{photo}(\mathbf{p}) \tag{3.9}$$

where λ is hyperparameter balancing the effect of the loss terms.

3.1.4 Architecture of Networks

Our depth network in MonoDepthSeg is based on the U-Net architecture [Ronneberger et al., 2015]. We use a ResNet [He et al., 2016] pretrained on the ImageNet dataset [Russakovsky et al., 2015] as the encoder for our depth network, and for the decoder we use the architecture proposed in [Godard et al., 2019]. The depth network takes as input a single target image $\mathbf{I}_t$ and outputs the per-pixel depth estimates. 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 ResNet 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 [Li et al., 2020]. The pose decoder outputs K 6-DoF relative poses corresponding to K masks. Following [Wang et al., 2018], we predict the rotation in axis-angle representation and multiply the network outputs by 0.01, following [Godard et al., 2019].

Using the same representation encoded by the shared encoder, the mask decoder estimates the K masks as an image with K channels corresponding to the components [Byravan and Fox, 2017]. The value of a channel i for a pixel $\mathbf{p}$ denotes the

probability that the pixel **p** belongs to the i'th component. In order to capture spatial boundaries, the mask decoder utilises skip connections with the shared encoder. To be able to calculate the masks at the same resolution as the input image, transposed convolutions are used. We also experiment with initializing our mask and pose network with a pretrained DeepLabv3+ [Chen et al., 2018] with ResNet-50 as its backbone. The details are discussed in Section 3.2.3.

3.2 Experiments

We train and evaluate our model on KITTI [Geiger et al., 2013], DDAD [Guizilini et al., 2020a] datasets and Cityscapes [Cordts et al., 2016] datasets. This section introduces the datasets and explains our training details. Then ablation studies are presented, justifying our design choices. Finally, the quantitative and qualitative results are provided.

3.2.1 Datasets

KITTI: We use the Eigen split [Eigen et al., 2014] of the KITTI dataset [Geiger et al., 2013, Geiger et al., 2012] to train and evaluate MonoDepthSeg. Following [Zhou et al., 2017], we remove the static images in the split, resulting in 39810 training and 4424 validation samples. We evaluate our model on the 697 test images in the split using the original ground truth provided by LiDAR. We also report results using the improved ground truth for 652 test images provided by [Uhrig et al., 2017]. They use a stereo-reconstruction method to remove the outliers in LiDAR points and increase the ground truth density by accumulating laser scans.

Cityscapes: Another dataset that we use for training and evaluation of MonoDepthSeg is the Cityscapes dataset [Cordts et al., 2016]. Out of the 18 cities containing frame sequences from the left camera in the original training split, we choose 12 cities for training. We choose 2 of the three cities in the validation split as our validation set. This yields 48802 and 6058 samples for training and validation, respectively. Our test set comprises all 1525 left camera images from the original test split. The ground truths are computed by converting the provided disparity

ground truth, which is obtained by the SGM method [Hirschmuller, 2007], to depth. Similar to the KITTI dataset, our depth estimations are evaluated for distances of up to 80 metres.

DDAD: The DDAD dataset [Guizilini et al., 2020a] is a more recent dataset containing a diverse set of scenes and dense ground truth depth data, making it a more realistic and challenging depth estimation benchmark. Our training and validation set contains 12350 and 3850 samples from the front camera, respectively. We evaluate the MonoDepthSeg method on the 3080 test images from the front camera using the online evaluation server [DDAD Challenge, 2021] for longer ranges of up to 250 metres.

3.2.2 Training Details

We train our model using the Adam optimiser [Kingma and Ba, 2014] with a learning rate of 10^{-4} and the default values of $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. For efficiency, the input images are resized to 640×192 on KITTI, 512×256 on Cityscapes and 640×384 on the DDAD dataset. Our model is trained for 20 epochs on the KITTI and Cityscapes datasets and for 30 epochs on the DDAD dataset.

Since our model estimates the depth at multiple scales, the hyperparameter λ in the loss function (Equation (3.9)) depends on the scale and is defined to be $\lambda = 0.001 \times \text{scale}$ where $\text{scale} \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\}$. By default, The ResNet-50 [He et al., 2016] architecture is used in both the depth network and the mask and pose network of our model. We also experiment with adopting a pretrained DeepLabv3+ [Chen et al., 2018] architecture for estimating masks. Note that our approach can benefit from using larger input sizes, as demonstrated by [Gadelha et al., 2019, Guizilini et al., 2020a]. Architectural improvements, such as the one proposed in [Guizilini et al., 2020a] are also complementary to our method.

3.2.3 Ablation Studies

This section analyses various aspects of MonoDepthSeg, including regularising the masks with the depth ordering technique, the number of components to be estimated,

and the use of a pretrained segmentation for the pose and mask network. The results in this section are obtained by training each model for 20 epochs and reporting the results using the original ground truth on the validation set.

Table 3.1: **Number of Components.** This table reports the results of our model by changing K, the number of components. In each column, the best result is written in bold and the second best has been underlined. Using $K = 5$ components results in the best performance in comparison to using smaller or larger number of components for the tested K values.

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**	<u>0.982</u>
3	0.090	0.801	4.123	0.167	<u>0.920</u>	**0.967**	<u>0.982</u>
5	**0.084**	<u>0.719</u>	**3.943**	**0.163**	**0.925**	**0.967**	**0.983**
10	<u>0.085</u>	**0.620**	<u>3.984</u>	<u>0.164</u>	0.917	<u>0.966</u>	**0.983**

Number of Components: We analyse the effect of K, the maximum number of components, on the performance of MonoDepthSeg. Concretely, we experiment with values of K being $2, 3, 5$, and 10. Table 3.1 shows how the number of components affects the results. A small value for K, such as $K = 2$ and $K = 3$, limits the model's performance while increasing it to $K = 5$ leads to better results in every metric. Further increasing it to $K = 10$ worsens the results slightly, resulting in $K = 5$ as the optimal value. Please note that this value clearly depends on the dataset and how complex it is. However, we conclude that among the values compared, $K = 5$ achieves the best results on the KITTI dataset in every metric.

Depth Ordering: We study the effect of depth ordering in our MonoDepthSeg model. We compare the performance of the model that uses depth ordering with the model that uses a simple mask smoothing to reduce the noise instead. We evaluate the performance with $K = 5$ and $K = 10$ and provide the results in Table 3.2.

In both $K = 5$ and $K = 10$ settings, using the depth ordering regularisation as described in Equation (3.3), clearly improves the results. Intuitively, depth ordering is global regularisation, while mask smoothing is local regularisation. Depth ordering

Table 3.2: **Depth Ordering Regularisation.** In this table, the depth ordering technique is compared to simply encouraging masks to be smooth as regularisation. In all metrics and for different number of components, the depth ordering method results in an improvement in the performance, especially for the Sq Rel (squared relative error) metric.

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

forces the pixels whose depth values are close to each other to cluster on the same mask. On the other hand, mask smoothing enforces similar values on each mask locally and independently than other masks. Therefore, we use depth ordering in our experiments unless otherwise specified.

Table 3.3: **Pretrained Segmentation Network.** This table shows the results of adopting a pretrained DeepLabv3+ model [Chen et al., 2018] as the mask and pose network. The DeepLabv3+ is modified so that a pose decoder is also added to it. This modified version of DeepLabv3+ achieves better results.

Components	**DeepLabv3+**	Lower Better				Higher Better		
(K)	**Pretraining**	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	<u>0.918</u>	0.964	<u>0.981</u>
10	No	0.092	**0.595**	4.111	0.174	0.908	0.961	<u>0.981</u>
5	Yes	**0.084**	0.719	**3.943**	**0.163**	**0.925**	**0.967**	**0.983**
10	Yes	<u>0.085</u>	<u>0.620</u>	<u>3.984</u>	<u>0.164</u>	0.917	<u>0.966</u>	**0.983**

Pretrained Segmentation Network: For our pose and mask network, we use DeepLabv3+ [Chen et al., 2018] with ResNet-50 backbone and pretrain it on Cityscapes [Cordts et al., 2016] first, to push its limits. We change the DeepLabv3+ and add a pose decoder to its mask decoder. As was previously mentioned, the mask and pose decoder use a shared encoder. Note that although our mask decoder and the shared encoder have been pretrained on the Cityscapes dataset in a supervised

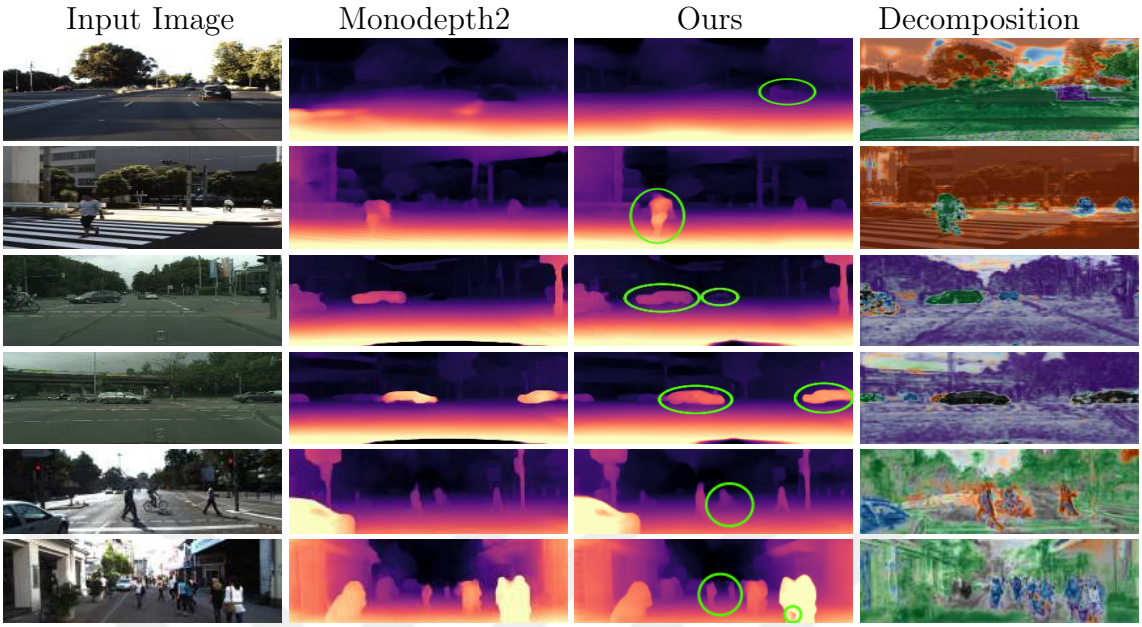

Figure 3.3: **Qualitative Results for MonoDepthSeg.** In each row, we show the qualitative depth estimations results for Monodepth2 [Godard et al., 2017] and MonoDepthSeg for a given image. The last column shows our scene decomposition results, where a colour is randomly assigned to each mask channel and the brightness of the colour has been adjusted according to the predicted mask value. MonoDepthSeg can clearly segment the dynamic regions (circled in green) which contain cars, pedestrians and bicycles. We obtain more accurate depth estimations in moving foreground regions by learning individual transformations for each component. The top two rows are from KITTI, the next two rows are from DDAD and the last two rows are from Cityscapes.

manner, our training process is still self-supervised. In Table 3.3, We compare the effect of using the modified DeepLabv3+ against the original ResNet-50 model for the pose and mask network under $K = 5$ and $K = 10$ settings. It can be seen that the modified DeepLabv3+ version outperforms the ResNet-50 architecture.

3.2.4 Results

In Table 3.4, we report the depth estimation results of our method on the KITTI Eigen split using both the original and the improved ground truth. The configuration of MonoDepthSeg is based on our ablation studies. Concretely, we use $K = 5$ as the number of components, depth ordering as regularisation for masks, and a pretrained segmentation network based on DeepLabv3+ for our mask and pose net-

Table 3.4: **Quantitative Results on the KITTI Dataset.** This table compares our proposed approach, **MonoDepthSeg** , to previous approaches on the KITTI dataset using the original and improved ground truth. The **Supervision** column shows different types of supervision used by different methods and **M** stands for monocular sequences. **Sem** and **Ins** refer to using semantic or instance information, respectively. **v** refers to velocity supervision. We show the results for the input resolution 640×192. The best self-supervised method in each column is shown in bold and the second best is underlined.

	Method	**Supe-rvision**	Lower Better				Higher Better		
			Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original Ground Truth	[Zhou et al., 2017]	M	0.183	1.595	6.709	0.270	0.734	0.902	0.959
	[Yang et al., 2018b]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
	[Mahjourian et al., 2018]	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
	[Yin and Shi, 2018]	M	0.149	1.060	5.567	2.226	0.796	0.935	0.975
	[Wang et al., 2018]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
	[Zou et al., 2018]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
	[Yang et al., 2018a]	M	0.162	1.352	6.276	0.252	-	-	-
	[Ranjan et al., 2019]	M	0.148	1.149	5.464	0.226	0.815	0.935	0.973
	[Luo et al., 2019]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
	[Chen et al., 2019]	M	0.118	0.905	5.096	0.211	0.839	0.945	0.977
	[Godard et al., 2019]	M	**0.110**	0.831	4.642	**0.187**	**0.883**	**0.962**	**0.982**
	[Guizilini et al., 2020a]	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., 2019]	M+Ins	0.141	1.026	5.291	0.215	0.816	0.945	0.979
	[Guizilini et al., 2020a]	M+v	0.111	0.829	4.788	0.199	0.864	0.954	0.980
	[Guizilini et al., 2020b]	M+Sem	0.102	0.698	4.381	0.178	0.896	0.964	0.984
Improved GT [Uhrig et al., 2017]	[Zhou et al., 2017]	M	0.176	1.532	6.129	0.244	0.758	0.921	0.971
	[Mahjourian et al., 2018]	M	0.134	0.983	5.501	0.203	0.827	0.944	0.981
	[Yin and Shi, 2018]	M	0.132	0.994	5.240	0.193	0.833	0.953	0.985
	[Wang et al., 2018]	M	0.126	0.866	4.932	0.185	0.851	0.958	0.986
	[Ranjan et al., 2019]	M	0.123	0.881	4.834	0.181	0.860	0.959	0.985
	[Luo et al., 2019]	M	0.120	0.789	4.755	0.177	0.856	0.961	0.987
	[Godard et al., 2019]	M	0.085	0.468	3.672	0.128	0.921	0.985	0.995
	[Guizilini et al., 2020a]	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**

work. It can be seen that our MonoDepthSeg method performs comparably to the PackNet [Guizilini et al., 2020a] and Monodepth2 [Godard et al., 2019] methods according to both the original ground truth and the improved ground truth [Uhrig

et al., 2017]. Furthermore, MonoDepthSeg decomposes the scene without relying on any kind of segmentation masks. In fact, it performs better than the method that use instance segmentation masks as input [Casser et al., 2019]. Moreover, using our joint scene decomposition framework, masks and poses of images can be estimated in a single pass (9.057 samples per second) instead of requiring a separate pass for each object in an image (4.124 samples per second) when using precomputed masks [Casser et al., 2019]. In other words, MonoDepthSeg is more efficient and performs better than the method that depend on a segmentation network's output. Using a simple ResNet-50 architecture, it performs similarly to PackNet [Guizilini et al., 2020a], which has many parameters because of using 3D convolutions in its architecture. [Guizilini et al., 2020b], which is a follow-up of [Guizilini et al., 2020a], improves the results by using semantic segmentation supervision. It is important to note that MonoDepthSeg does not condition on the output of a segmentation network but instead automatically discovers independently moving components.

Table 3.5: **Quantitative Results in Moving Regions on KITTI and Cityscapes.** Comparing MonodepthSeg to Monodepth2 [Godard et al., 2019] for moving regions on the KITTI and Cityscapes datasets. Our method outperforms Monodepth2 [Godard et al., 2019] in all metrics on the Cityscapes dataset, and in all but one metric on the KITTI dataset.

Dataset	Method	Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
KITTI	[Godard et al., 2019]	0.143	1.826	5.949	0.229	0.823	**0.913**	0.951
	MonoDepthSeg (Ours)	**0.138**	**1.709**	**5.796**	**0.223**	**0.837**	0.912	**0.953**
Cityscapes	[Godard et al., 2019]	0.158	3.148	8.043	0.241	0.820	0.940	0.972
	MonoDepthSeg (Ours)	**0.143**	**2.018**	**7.649**	**0.225**	**0.824**	**0.944**	**0.977**

The model's improvements are especially noticeable in the dynamic objects (Figure 3.3 and Table 3.5). We use the motion segmentation methods [Mohamed et al., 2020] and [Vertens et al., 2017] to extract the moving regions of the KITTI and Cityscapes datasets, respectively, and evaluate MonoDepthSeg on these regions. As can be seen in Table 3.5, on moving regions, our approach clearly outperforms Monodepth2 [Godard et al., 2019] on both of the KITTI and Cityscapes datasets in all but one metric where the results are very close. Also, Table 3.6 shows that on the

Table 3.6: **Quantitative Results Overall on Cityscapes.** Comparing MonoDepthSeg to Monodepth2 [Godard et al., 2019] on Cityscapes in all regions. MonoDepthSeg clearly outperforms Monodepth2 [Godard et al., 2019] in all of the metrics in this dataset. Cityscapes is more complex than KITTI and has more moving objects, and therefore can better show our model's strength.

Method	Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Godard et al. [Godard et al., 2019]	0.170	3.936	8.155	0.243	0.822	0.940	0.971
MonoDepthSeg (Ours)	**0.142**	**1.942**	**7.361**	**0.218**	**0.827**	**0.946**	**0.978**

more complex Cityscapes dataset, MonoDepthSeg performs significantly better than Monodepth2 [Godard et al., 2019] on all of the metrics.

Table 3.7: **Quantitative Results on DDAD.** Results obtained using the online evaluation server [DDAD Challenge, 2021]. The *Car* and *Person* are shown using the *Abs Rel* metric. The last column is the average of *Abs Rel* errors over all of the semantic classes. For every metric, lower is better, except for $\delta < 1.25$, where higher is better. MonoDepthSeg achieves the state-of-the-art results in every metric, without being initialized by a pre-trained segmentation network.

Method	Abs Rel	RMSE	SILog	$\delta < 1.25$	Car	Person	Semantic Classes (Average)
[Guizilini et al., 2020a]	0.23	17.92	32.56	0.72	0.38	0.20	0.24
[Godard et al., 2019]	0.22	17.63	31.93	0.70	0.25	0.21	0.24
MonoDepthSeg (Ours)	**0.19**	**16.61**	**29.37**	**0.75**	**0.24**	**0.17**	**0.22**

Next, we evaluate MonoDepthSeg on the more complex DDAD dataset. This evaluation was done using the online server [DDAD Challenge, 2021], and we did not initialise our model with a pretrained segmentation as the server does not allow any kind of semantic supervision. Table 3.7 shows that, in spite of not using a pretraining, we achieve remarkably better results than the state-of-the-art PackNet [Guizilini et al., 2020a] and Monodepth2 [Godard et al., 2019] in all metrics. It also shows that our performance is better in the person and car classes. We provide additional qualitative results for MonoDepthSeg for each dataset in Figures 3.6, 3.7, and 3.8. To better demonstrate different parts and aspects of our mode, we provide additional experiments in the next parts.

Shared versus Separate Encoders: To investigate the effect of using a shared

encoder, we experiment with using separate encoders for estimating masks and poses. As can be seen in Table 3.8, not only does using a shared encoder reduce the complexity of the architecture and the training time by having fewer parameters and, but it also achieves better performances due to relating the mask and pose estimations. The pose and mask of the components are interleaved with each other through the shared encoder to produce corresponding pose and mask predictions for each component via separate decoders.

Table 3.8: **Effect of Using a Shared Encoder.** Using separate encoders for mask and pose estimation results in worse performance while increasing the complexity of the model on the KITTI dataset using both the original and improved ground truth. In the **GT** column, **O** refers to using original ground truth and **Imp** refers to improved ground truth.

Mask & Pose Encoder	**GT**	Lower Better				Higher Better		
		Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Separate	**O**	0.112	**0.747**	4.855	0.194	0.870	0.957	0.981
Shared	**O**	**0.110**	0.792	**4.700**	**0.189**	**0.881**	**0.960**	**0.982**
Separate	**Imp**	0.089	0.481	4.100	0.140	0.906	0.980	0.995
Shared	**Ipm**	**0.085**	**0.458**	**3.779**	**0.131**	**0.919**	**0.985**	**0.996**

Effect of Input Image Resolution: Similar to [Godard et al., 2019] we also report the results of our model using smaller (416×128) and bigger (1024×320) input sizes on the KITTI dataset. The quantitative results along with the training time is reported in Table 3.9. It can be seen that using larger images results in better performances. However, as expected, the training takes longer.

Qualitative Analysis of Number of Components: In this part, we qualitatively evaluate the effect of different values of K on our predictions. We train models with $K = 2, 3, 5,$ and 10 components and visualise the estimated scene decomposition of each model on the same input images in Figure 3.4. We can see that using $K = 2$ results in the model dividing the scene only into the road region and the remaining part. For $K = 3$, the model starts to separate the moving objects better, however, the results are still not desirable. By increasing the number of components to $K = 5$ our model is able to divide the scene into road, dynamic objects and

Table 3.9: **Effect Of Image Resolution on KITTI.** This table compares the performances and training times for our models using different image resolutions. The timing for the high resolution model comprises 10 epochs of of training of the 640×192 model and 10 epochs of training at 1024×320 resolution. In the **GT** column, **O** refers to using original ground truth and **Imp** refers to improved ground truth. The training times are reported in hours.

Resolution	**GT**	**Train Time**	Lower Better				Higher Better		
			Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
416×128	**O**	11	0.119	0.827	5.053	0.201	0.856	0.952	0.979
640×192	**O**	22	0.110	0.792	4.700	0.189	0.881	0.960	0.982
1024×320	**O**	11 + 32	**0.107**	**0.727**	**4.520**	**0.184**	**0.888**	**0.964**	**0.983**
416×128	**Imp**	11	0.094	0.553	4.328	0.149	0.896	0.974	0.993
640×192	**Imp**	22	0.085	0.458	3.779	0.131	0.919	0.985	0.996
1024×320	**Imp**	11 + 32	**0.083**	**0.405**	**3.562**	**0.125**	**0.925**	**0.987**	**0.997**

regions in the scene that are far away, validating our choice of $K = 5$. As can be seen for $K = 10$, using a large value for K, results in a single object being divided into multiple parts, worsening the performance. In Figure 3.5 we visualise the depth estimations for the models with these values of K.

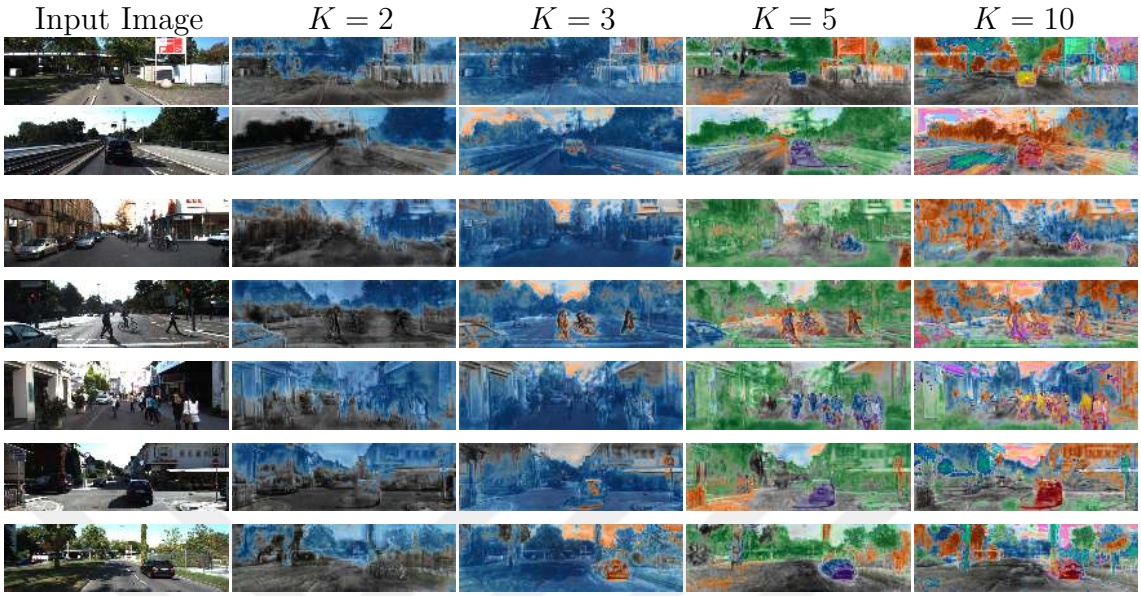

Figure 3.4: **Effect of K on Scene Decomposition.** Choosing a small K makes the model unable to decompose the dynamic objects. A too large value for K results in objects being divided unnecessarily.

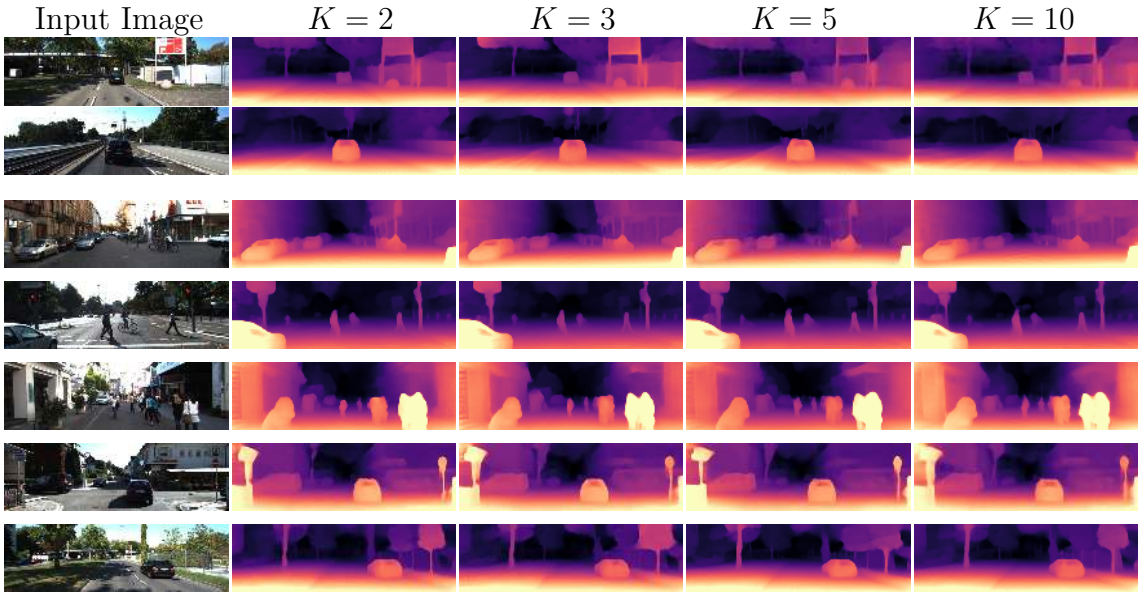

Figure 3.5: **Effect of K on Depth Estimates.** In this figure, we visualise the depth predictions for our models with $K = 2, 3, 5,$ and 10.

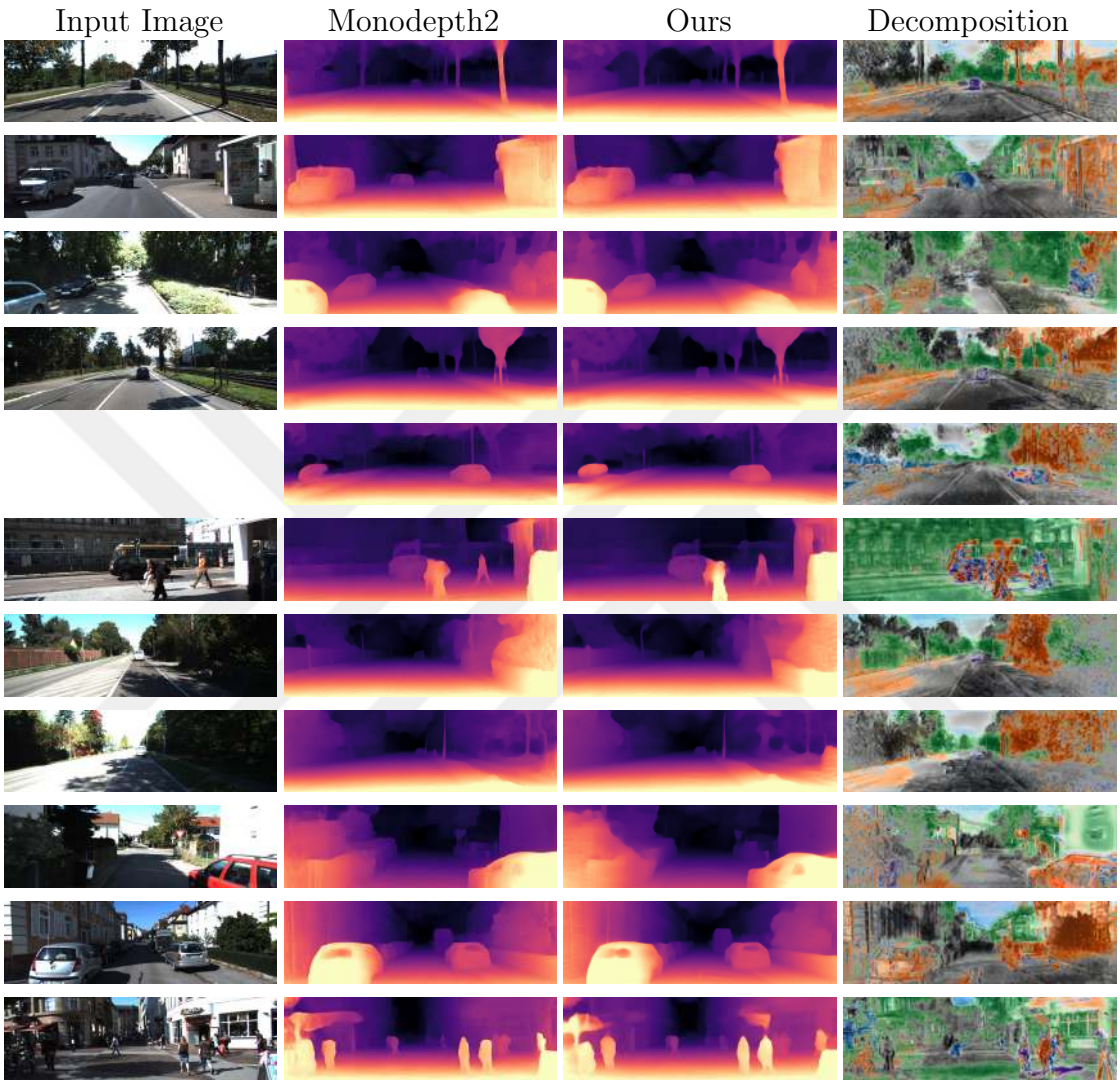

Figure 3.6: **Additional Qualitative Results on KITTI.** In each row, we show the qualitative depth estimations results for Monodepth2 [Godard et al., 2017] and MonoDepthSeg for a given image. The last column shows our scene decomposition results, where a colour is randomly assigned to each mask channel and the brightness of the colour has been adjusted according to the predicted mask value. We obtain more accurate depth estimations in moving foreground regions by learning individual transformations for each component. In the last row, MonoDepthSeg clearly outperforms Monodepth2 on pedestrians, especially for the person on the right.

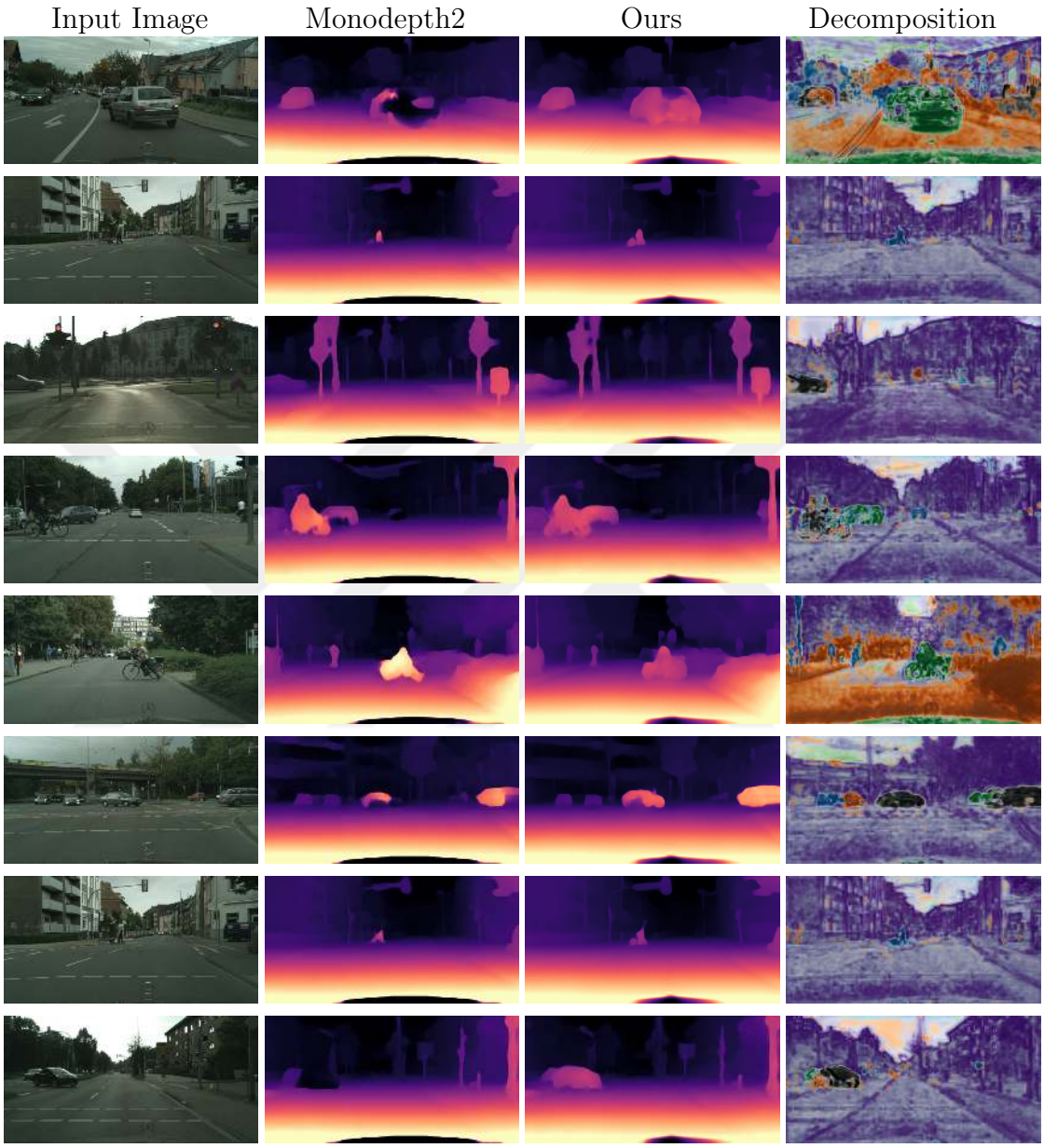

Figure 3.7: **Additional Qualitative Results on Cityscapes.** In each row, we show the qualitative depth estimations results for Monodepth2 [Godard et al., 2017] and MonoDepthSeg for a given image. The last column shows our scene decomposition results, where a colour is randomly assigned to each mask channel and the brightness of the colour has been adjusted according to the predicted mask value. We obtain more accurate depth estimations in moving foreground regions by learning individual transformations for each component.

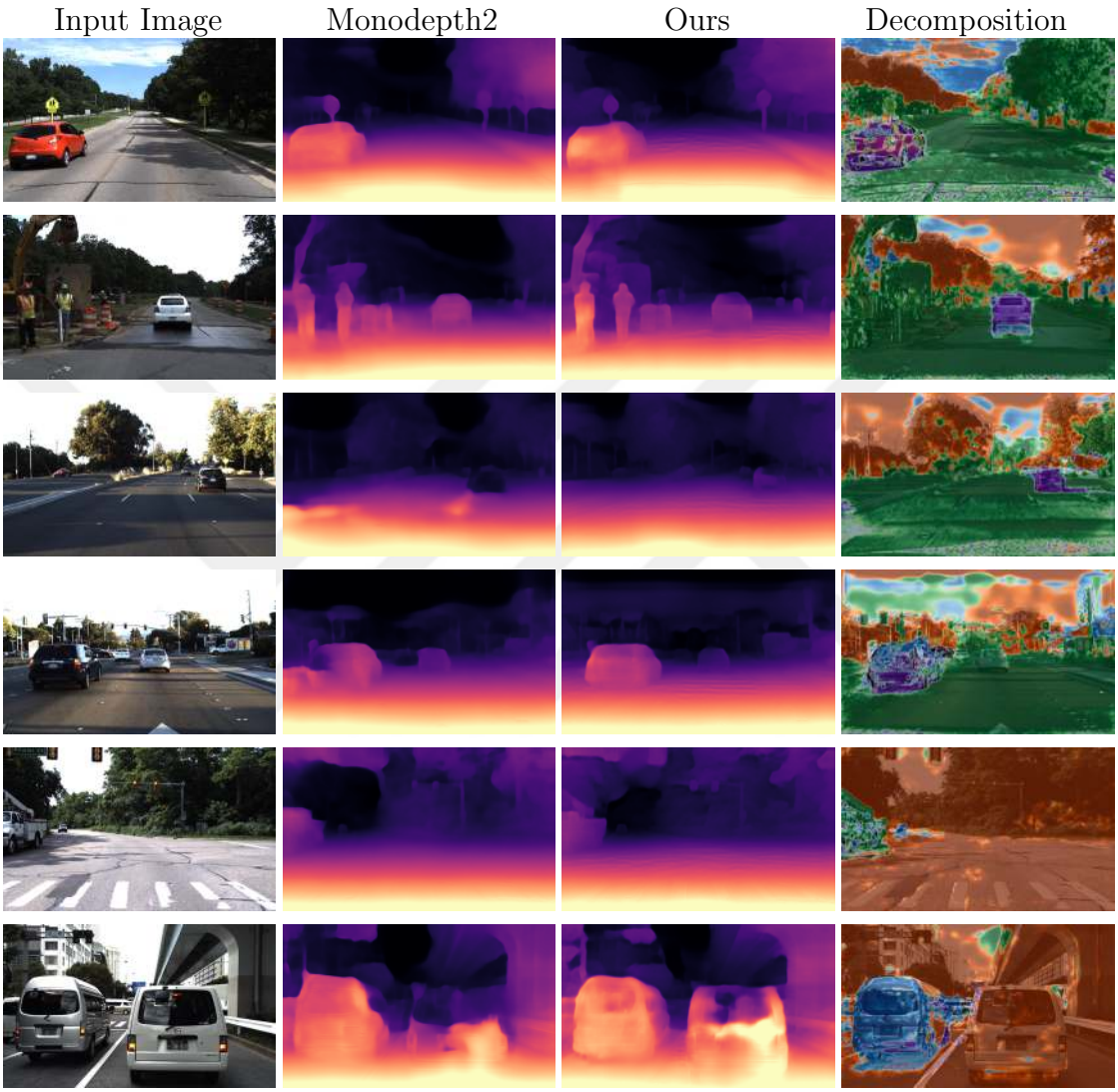

Figure 3.8: **Additional Qualitative Results on DDAD.** In each row, we show the qualitative depth estimations results for Monodepth2 [Godard et al., 2017] and MonoDepthSeg for a given image. The last column shows our scene decomposition results, where a colour is randomly assigned to each mask channel and the brightness of the colour has been adjusted according to the predicted mask value. We obtain more accurate depth estimations in moving foreground regions by learning individual transformations for each component. In the last row, the car on the right is not moving while the car on the left is moving.

Chapter 4

METRIC ACCURATE MONOCULAR DEPTH ESTIMATION USING PLANAR PARALLAX

Current self-supervised monocular depth estimation methods are all based on back projecting pixels of the target image to 3D, transforming the points to the coordinate system of another view using an estimated rigid-body motion, and projecting them to the pixels on that view for reconstructing the target image from this view. Despite their success, these models, including our MonoDepthSeg, suffer from the scale ambiguity problem. i.e., the predicted depth and translations are in an unknown scale. This is because there is nothing in their formulation forcing the predictions to be in a specific scale. An easy way to see this is that, if we scale the predicted depth map D and the translation T of such a model by a constant, then for a pixel $\mathbf{p}$, Equations (3.4) and (3.5) will result in the same $\mathbf{p}'$ before and after scaling. In this chapter we propose DepthP+P, depicted in Figure 4.1, a method that without using any ground truth depth supervision is able to estimate depth in metric scale.

4.1 Methodology

4.1.1 Planar Parallax for Depth Estimation

Our approach is based on the plane and parallax decomposition. The P+P paradigm has been studied in detail [Sawhney, 1994, Irani and Anandan, 1996]. This section starts by introducing P+P and establishing our notation and is then followed by the description of how we use it for estimating depth.

Notation: Let $\mathbf{\Pi}$ be a 3D plane in the scene and $\mathbf{H}$ be the homography aligning the plane $\mathbf{\Pi}$ between the source frame $\mathbf{I}_s$ and the target frame $\mathbf{I}_t$. Let $\mathbf{p}$ and $\mathbf{p}'$ be

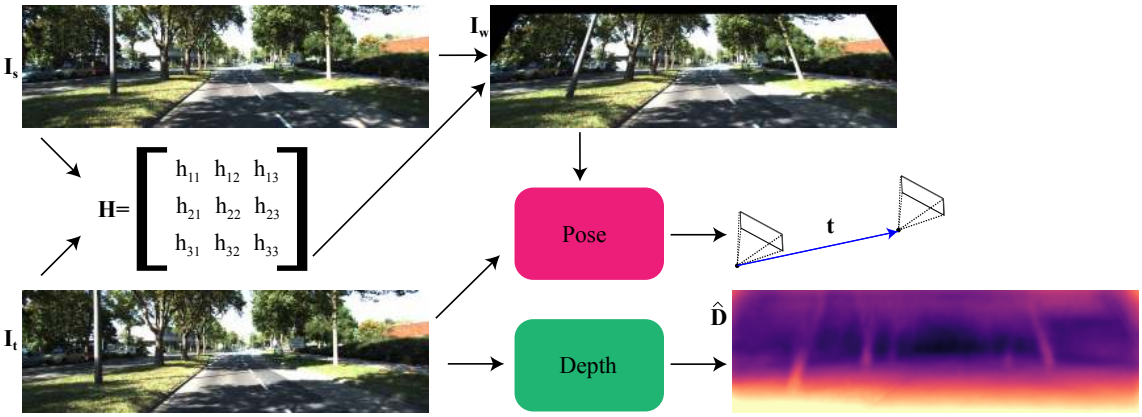

Figure 4.1: **Overview of our Approach.** Using the source image $\mathbf{I}_s$ and the target image $\mathbf{I}_t$, we first calculate the homography $\mathbf{H}$ that aligns the road plane across these two images. We then warp $\mathbf{I}_s$ according to $\mathbf{H}$ and obtain the aligned image $\mathbf{I}_w$. The aligned image $\mathbf{I}_w$ and the target image $\mathbf{I}_t$ are input to the pose network which estimates the camera translation $\mathbf{t}$ only. The depth network takes the $\mathbf{I}_t$ and produces a metric accurate depth map $\hat{\mathbf{D}}$.

the images of the 3D point $\mathbf{x} = [\mathbf{X}, \mathbf{Y}, \mathbf{Z}]^T$ on the target frame $\mathbf{I}_t$ and the source frame $\mathbf{I}_s$, respectively. As demonstrated on the left in Figure 4.2, the homography matrix $\mathbf{H}$ can be used to warp $\mathbf{p}'$ and obtain $\mathbf{p}_w$:

$$\mathbf{p}_w \sim \mathbf{H}\mathbf{p}' \tag{4.1}$$

where we have omitted the conversion to the homogenous coordinates. Note that by warping $\mathbf{I}_s$, we obtain the aligned image $\mathbf{I}_w$ such that the planar surface $\boldsymbol{\Pi}$ is aligned between $\mathbf{I}_w$ and $\mathbf{I}_t$. We can calculate the displacement between $\mathbf{p}_w$ and $\mathbf{p}$ as:

$$\mathbf{p}_w - \mathbf{p} = \frac{\boldsymbol{\gamma}}{\mathbf{d}_c - \boldsymbol{\gamma}\mathbf{t}_z}(\mathbf{t}_z\mathbf{p} - \mathbf{K}\mathbf{t}) \tag{4.2}$$

where $\mathbf{t} = [\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z]^T$ is the translation vector between the $\mathbf{I}_t$ and $\mathbf{I}_s$, $\mathbf{K}$ is the intrinsics matrix of the camera, and $\mathbf{d}_c$ is the distance to the plane for camera centre of the source view to the plane $\boldsymbol{\Pi}$. The parameter $\boldsymbol{\gamma} = \frac{\mathbf{h}}{\mathbf{Z}}$ represents the structure, with $\mathbf{h}$ being the distance of $\mathbf{x}$ to $\boldsymbol{\Pi}$. Note that when $\mathbf{x}$ is on $\boldsymbol{\Pi}$, we will have $\mathbf{h} = 0$, resulting in $\mathbf{p}_w = \mathbf{p}$ which conforms to the fact that the plane $\boldsymbol{\Pi}$ is aligned between $\mathbf{I}_w$ and $\mathbf{I}_t$. The derivation of Equation (4.2) is presented in Section 4.1.2.

DepthP+P: We have two networks in our framework, similar to the typical self-supervised monocular depth approaches. One network predicts depth and the other

is used to predict the translation between views. The important thing to note here is that, as opposed to other approaches, our model does not require predicting the rotation between two views. Concretely, our network takes the target image $\mathbf{I}_t$ and estimates a depth map $\hat{\mathbf{D}}$ for $\mathbf{I}_t$. The pose network in our approach takes the aligned image $\mathbf{I}_w$ and the target image $\mathbf{I}_t$ and predicts the translation vector $\mathbf{t}$. .

Let $\hat{\mathbf{D}}(\mathbf{p})$ be the predict depth for a pixel $\mathbf{p} = [x, y]$, The pixel $\mathbf{p}$ is backprojected using the intrinsics matrix of the camera and the predicted depth to obtain the corresponding 3D point $\hat{\mathbf{x}}$ in the camera coordinate system of the target view $\mathbf{I}_t$:

$$\hat{\mathbf{x}} = \hat{\mathbf{D}}(\mathbf{p})\ \mathbf{K}^{-1}\ [x, y, 1]^T. \tag{4.3}$$

Therefore, as shown on the right in Figure 4.2, we have:

$$\hat{\mathbf{h}} = \mathbf{d}_c - \mathbf{N}^T\hat{\mathbf{x}}, \qquad \hat{\boldsymbol{\gamma}} = \frac{\hat{\mathbf{h}}}{\hat{\mathbf{D}}(\mathbf{p})} \tag{4.4}$$

where $\hat{\mathbf{h}}$ is the estimated distance of the point $\hat{\mathbf{x}}$ to the plane $\boldsymbol{\Pi}$, $\mathbf{N}$ is the normal vector of $\boldsymbol{\Pi}$, and $\hat{\boldsymbol{\gamma}}$ is our estimate of the structure parameter $\boldsymbol{\gamma}$. This means that we have obtained all the parameters that is necessary to use Equation (4.2) for warping the aligned frame $\mathbf{I}_w$ to synthesise the target frame $\mathbf{I}_t$ that will result in $\hat{\mathbf{I}}_w$. In other words, for each pixel $\mathbf{p}$ on the target frame $\mathbf{I}_t$, we use Equation (4.2) to compute $\mathbf{p}_w$ according to our estimated depth and translation. The aligned image $\mathbf{I}_w$ is then inverse warped to obtain $\hat{\mathbf{I}}_w$ for reconstructing $\mathbf{I}_t$:

$$\mathbf{I}_t(\mathbf{p}) \approx \hat{\mathbf{I}}_w(\mathbf{p}) = \mathbf{I}_w(\mathbf{p}_w) \tag{4.5}$$

As explained in Section 4.1.3, the difference between the reconstructed image $\hat{\mathbf{I}}_w$ and the target image $\mathbf{I}_t$ is used for supervision.

A preprocessing step is performed on the dataset for obtaining the aligned images. Using the road as the planar surface $\boldsymbol{\Pi}$, a homography associated with that plane is calculated for every consecutive pairs of frames, and the frames are warped according the computed homographies. In other words, a homography $\mathbf{H}$ is computed for every pair of $\mathbf{I}_t$ and $\mathbf{I}_s$. Then, $\mathbf{I}_s$ is warped according to $\mathbf{H}$ to obtain the aligned source image $\mathbf{I}_w$. This preprocessing step is explained in detail in Section 4.2.1.

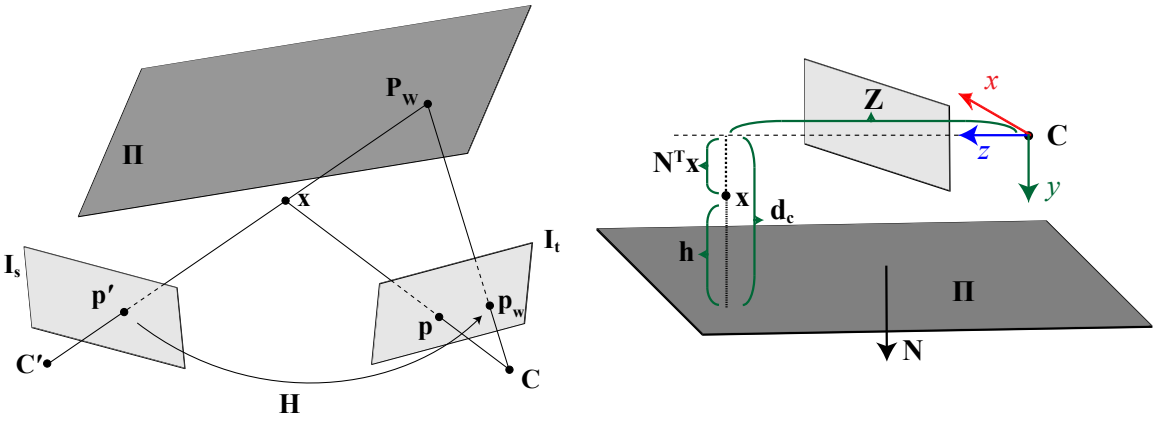

Figure 4.2: **Visualisation of the Planar Parallax.** *Left:* The 3D point $\mathbf{x}$ is projected to points $\mathbf{p}$ and $\mathbf{p}'$ on the target image $\mathbf{I}_t$ and the source image $\mathbf{I}_s$ respectively. Using the homography $\mathbf{H}$ induced by the plane $\mathbf{\Pi}$, the point $\mathbf{p}'$ will be transformed to point $\mathbf{p}_w$ on the target image. *Right:* Calculating $\mathbf{h}$, distance of $\mathbf{x}$ to the $\mathbf{\Pi}$ using the camera height $\mathbf{d}_c$ and the normal vector $\mathbf{N}$ of the plane. $\mathbf{C}$ and $\mathbf{C}'$ are the camera centers of $\mathbf{I}_t$ and $\mathbf{I}_s$ and $\mathbf{Z}$ is the depth of the point.

4.1.2 Derivation of Planar Parallax Formulation

In this section, we derive the planar parallax formulation (Equation (4.2)) that is the basis for our DepthP+P approach. Let $\mathbf{C}$ be the camera centre of the target frame $\mathbf{I}_t$ and $\mathbf{C}'$ be the camera centre of the source frame $\mathbf{I}_s$. Let $\mathbf{x} = [\mathbf{X}, \mathbf{Y}, \mathbf{Z}]^T$ and $\mathbf{x}' = [\mathbf{X}', \mathbf{Y}', \mathbf{Z}']^T$ be the coordinates of a 3D point with respect to the camera centres $\mathbf{C}$ and $\mathbf{C}'$, respectively. The 3D points $\mathbf{x}$ and $\mathbf{x}'$ can be related as:

$$\mathbf{x} = \mathbf{R}\mathbf{x}' + \mathbf{t} \tag{4.6}$$

where $\mathbf{R}$ is the 3×3 rotation matrix and $\mathbf{t} = [\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z]^T$ is the translation vector between the camera centres $\mathbf{C}$ and $\mathbf{C}'$. According to the Figure 4.2 (right), the perpendicular distance $\mathbf{h}$ of the point $\mathbf{x}$ to the plane $\mathbf{\Pi}$, can be calculated as:

$$\mathbf{h} = \mathbf{d}_c - \mathbf{N}^T\mathbf{x}' \tag{4.7}$$

where $\mathbf{d}_c$ is the height of the camera $\mathbf{C}'$, the camera for the source view $\mathbf{I}_s$. However, $\mathbf{h}$ is invariant with respect to the two cameras. The above equation can be rewritten as:

$$1 = \frac{\mathbf{h} + \mathbf{N}^T\mathbf{x}'}{\mathbf{d}_c} \tag{4.8}$$

Substituting this into Equation (4.6), we have the following:

$$\mathbf{x} = \mathbf{R}\mathbf{x}' + \mathbf{t}\left(\frac{\mathbf{h} + \mathbf{N}^T\mathbf{x}'}{\mathbf{d}_c}\right) = \left(\mathbf{R} + \frac{\mathbf{t}\mathbf{N}^T}{\mathbf{d}_c}\right)\mathbf{x}' + \frac{\mathbf{h}}{\mathbf{d}_c}\mathbf{t} \tag{4.9}$$

Let $\mathbf{K}$ and $\mathbf{K}'$ denote the intrinsic matrices for the target view and the source view. Then $\mathbf{p} = [x, y, 1]^T = \frac{1}{\mathbf{Z}}\mathbf{K}\mathbf{x}$ and $\mathbf{p} = [x', y', 1] = \frac{1}{\mathbf{Z}'}\mathbf{K}'\mathbf{x}'$ represent the pixel coordinates of $\mathbf{x}$ in the target view and source view, respectively. Therefore, we can rewrite Equation (4.9) as:

$$\mathbf{Z}\mathbf{K}^{-1}\mathbf{p} = \left(\mathbf{R} + \frac{\mathbf{t}\mathbf{N}^T}{\mathbf{d}_c}\right)\mathbf{Z}'\mathbf{K}'^{-1}\mathbf{p}' + \frac{\mathbf{h}}{\mathbf{d}_c}\mathbf{t} \tag{4.10}$$

Multiplying both sides by $\frac{1}{\mathbf{Z}'}\mathbf{K}$ we obtain:

$$\frac{\mathbf{Z}}{\mathbf{Z}'}\mathbf{p} = \underbrace{\mathbf{K}\left(\mathbf{R} + \frac{\mathbf{t}\mathbf{N}^T}{\mathbf{d}_c}\right)\mathbf{K}'^{-1}}_{\mathbf{H}}\mathbf{p}' + \frac{\mathbf{h}}{\mathbf{Z}'\mathbf{d}_c}\mathbf{K}\mathbf{t} \tag{4.11}$$

$$\frac{\mathbf{Z}}{\mathbf{Z}'}\mathbf{p} = \mathbf{H}\mathbf{p}' + \frac{\mathbf{h}}{\mathbf{Z}'\mathbf{d}_c}\mathbf{K}\mathbf{t} \tag{4.12}$$

where $\mathbf{H} = \mathbf{K}\left(\mathbf{R} + \frac{\mathbf{t}\mathbf{N}^T}{\mathbf{d}_c}\right)\mathbf{K}'^{-1}$ is the 3×3 homography matrix associated with the plane $\mathbf{\Pi}$ between $\mathbf{I}_s$ and the $\mathbf{I}_t$. In Equation (4.12), the third component of the vector in both sides of the equation should be equal. So we have:

$$\frac{\mathbf{Z}}{\mathbf{Z}'} = \mathbf{H}_3\mathbf{p}' + \frac{\mathbf{h}}{\mathbf{Z}'\mathbf{d}_c}\mathbf{K}_3\mathbf{t} \tag{4.13}$$

where $\mathbf{K}_3$ and $\mathbf{H}_3$ are the third rows of $\mathbf{K}$ and $\mathbf{H}$, respectively. Because $\mathbf{K}_3\mathbf{t} = \mathbf{t}_z$, we have:

$$\frac{\mathbf{Z}}{\mathbf{Z}'} = \mathbf{H}_3\mathbf{p}' + \frac{\mathbf{h}}{\mathbf{Z}'\mathbf{d}_c}\mathbf{t}_z \tag{4.14}$$

Dividing each side of Equation (4.12) by each side of Equation (4.14) results in:

$$\mathbf{p} = \frac{\mathbf{H}\mathbf{p}' + \frac{\mathbf{h}}{\mathbf{Z}'\mathbf{d}_c}\mathbf{K}\mathbf{t}}{\mathbf{H}_3\mathbf{p}' + \frac{\mathbf{h}}{\mathbf{Z}'\mathbf{d}_c}\mathbf{t}_z} \tag{4.15}$$

By adding and subtracting the term $\frac{\mathbf{Hp'}}{\mathbf{H_3p'}}$ to the right-hand side we obtain:

$$\mathbf{p} = \frac{\mathbf{Hp'}}{\mathbf{H_3p'}} - \frac{\mathbf{Hp'}}{\mathbf{H_3p'}} + \frac{\mathbf{Hp'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{Kt}}{\mathbf{H_3p'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z} \tag{4.16}$$

$$= \frac{\mathbf{Hp'}}{\mathbf{H_3p'}} - \frac{\mathbf{Hp'}\left(\mathbf{H_3p'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z\right)}{\mathbf{H_3p'}\left(\mathbf{H_3p'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z\right)} + \frac{\mathbf{H_3p'}\left(\mathbf{Hp'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{Kt}\right)}{\mathbf{H_3p'}\left(\mathbf{H_3p'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z\right)} \tag{4.17}$$

$$= \frac{\mathbf{Hp'}}{\mathbf{H_3p'}} - \frac{\mathbf{Hp'}\left(\frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z\right)}{\mathbf{H_3p'}\left(\mathbf{H_3p'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z\right)} + \frac{\mathbf{H_3p'}\left(\frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{Kt}\right)}{\mathbf{H_3p'}\left(\mathbf{H_3p'} + \frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z\right)} \tag{4.18}$$

Substituting Equation (4.14) into the denominators of Equation (4.18) yields:

$$\mathbf{p} = \frac{\mathbf{Hp'}}{\mathbf{H_3p'}} - \frac{\mathbf{Hp'}\left(\frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{t}_z\right)}{\mathbf{H_3p'}\left(\frac{\mathbf{Z}}{\mathbf{Z'}}\right)} + \frac{\mathbf{H_3p'}\left(\frac{\mathbf{h}}{\mathbf{Z'd}_c}\mathbf{Kt}\right)}{\mathbf{H_3p'}\left(\frac{\mathbf{Z}}{\mathbf{Z'}}\right)} \tag{4.19}$$

$$= \frac{\mathbf{Hp'}}{\mathbf{H_3p'}} - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\frac{\mathbf{Hp'}}{\mathbf{H_3p'}} + \frac{\mathbf{h}}{\mathbf{Zd}_c}\mathbf{Kt} \tag{4.20}$$

The point $\mathbf{p}_w = [x_w, y_w, 1]^T = \frac{\mathbf{Hp'}}{\mathbf{H_3p'}}$ is the point $\mathbf{p'}$ transformed by the $\mathbf{H}$. Hence, we can simplify Equation (4.20) into:

$$\mathbf{p} = \mathbf{p}_w - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p}_w + \frac{\mathbf{h}}{\mathbf{Zd}_c}\mathbf{Kt} \tag{4.21}$$

Subtracting $\frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p}$ from both sides, we get:

$$\mathbf{p} - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p} = \mathbf{p}_w - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p}_w + \frac{\mathbf{h}}{\mathbf{Zd}_c}\mathbf{Kt} - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p} \tag{4.22}$$

$$\mathbf{p}\left(1 - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\right) = \mathbf{p}_w\left(1 - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\right) + \frac{\mathbf{h}}{\mathbf{Zd}_c}\mathbf{Kt} - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p} \tag{4.23}$$

By dividing both sides by $1 - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}$ we obtain:

$$\mathbf{p} = \mathbf{p}_w + \frac{\frac{\mathbf{h}}{\mathbf{Zd}_c}\mathbf{Kt} - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p}}{1 - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}} \tag{4.24}$$

Finally, by rearranging the terms and defining $\boldsymbol{\gamma} = \frac{\mathbf{h}}{\mathbf{Z}}$, we can derive Equation (4.2) as follows:

$$\mathbf{p}_w - \mathbf{p} = \frac{\frac{\mathbf{ht}_z}{\mathbf{Zd}_c}\mathbf{p} - \frac{\mathbf{h}}{\mathbf{Zd}_c}\mathbf{Kt}}{1 - \frac{\mathbf{ht}_z}{\mathbf{Zd}_c}} \tag{4.25}$$

$$= \frac{\frac{\mathbf{ht}_z}{\mathbf{Z}}\mathbf{p} - \frac{\mathbf{h}}{\mathbf{Z}}\mathbf{Kt}}{\mathbf{d}_c - \frac{\mathbf{ht}_z}{\mathbf{Z}}} \tag{4.26}$$

$$= \frac{\boldsymbol{\gamma}\mathbf{t}_z\mathbf{p} - \boldsymbol{\gamma}\mathbf{Kt}}{\mathbf{d}_c - \boldsymbol{\gamma}\mathbf{t}_z} \tag{4.27}$$

$$= \frac{\boldsymbol{\gamma}}{\mathbf{d}_c - \boldsymbol{\gamma}\mathbf{t}_z}\left(\mathbf{t}_z\mathbf{p} - \mathbf{Kt}\right) \tag{4.28}$$

4.1.3 Self-Supervised Training

Similar to MonoDepthSeg, our loss function is a combination of the L1 distance between the synthesised image and the target image. However, here, the synthesised image $\hat{\mathbf{I}}_w$ has been sampled from the aligned image $\mathbf{I}_w$ instead of the source image $\mathbf{I}_s$. Therefore, the loss function is:

$$\mathcal{L}_{\text{photo}}(\mathbf{p}) = \min_{w}\Big[(1-\alpha)\,|\mathbf{I}_t(\mathbf{p}) - \hat{\mathbf{I}}_w(\mathbf{p})| + \frac{\alpha}{2}\Big(1 - \text{SSIM}(\mathbf{I}_t, \hat{\mathbf{I}}_w)(\mathbf{p})\Big)\Big] \quad (4.29)$$

and the minimum reprojection loss is calculated over the reconstructed frames from the next and previous aligned frames. α is set to 0.85, and a depth smoothness loss $\mathcal{L}_{smooth}$ is defined similar to the MonoDepthSeg. With this definition of the photometric loss $\mathcal{L}_{\text{photo}}$, our overall loss function is defined exactly similar to Equation (3.9) by combining $\mathcal{L}_{photo}$ and $\mathcal{L}_{smooth}$.

4.1.4 Architecture of Networks

The depth network in DepthP+P takes as input a single target image $\mathbf{I}_t$ and outputs a metrically accurate depth map. This network is based on the U-Net architecture [Ronneberger et al., 2015]. A ResNet [He et al., 2016] pretrained on the ImageNet dataset [Russakovsky et al., 2015] is used as the depth network's encoder. Note that, unlike MonoDepthSeg, which uses ResNet-50, Our DepthP+P model uses ResNet-18 by default, unless otherwise specified. Another difference is that rather than estimating inverse depth values in an unknown scale as in MonoDepthSeg, our DepthP+P model multiples the output of the last sigmoid layer in the depth network by 250 and outputs the depth values directly.

Our second network which is responsible for estimating only the translation between the cameras, takes the aligned image $\mathbf{I}_w$ and the target image $\mathbf{I}_t$ and produces a vector in $\mathbb{R}^3$. Since our translation predictions are also in metric scale, instead of multiplying the outputs by 0.01 as in MonoDepthSeg, we multiply them by 10.

4.2 Experiments

4.2.1 Dataset

KITTI: For training and evaluating our model, we choose the Eigen split [Eigen et al., 2014] of the KITTI dataset [Geiger et al., 2013, Geiger et al., 2012], similar to MonoDepthSeg. For training, all of the frames in the Eigen split for which a homography matrix associated with aligning the road between the temporally consecutive frames could be computed has been used. Details of this step is discussed in detail in the next paragraph. As a result, we obtain 45000 and 1769 samples for training and validation, respectively. For our test set, we use the 697 test images in the Eigen split with original ground truth and 652 test images with improved ground truth [Uhrig et al., 2017], following the previous work. Using the reported value for the camera height in [Geiger et al., 2013], we fix the camera height to $\mathbf{d}_c = 1.65$. Unless noted otherwise, we assume that the roads in the scenes are perfectly flat, i.e. $\mathbf{N} = [0, 1, 0]^T$.

Preprocessing the Dataset: In order to apply the planar parallax method, the homographies between the temporally consecutive images need to be calculated so that the source images can be warped according to these matrices. As our work is focused on the KITTI dataset and driving scenarios, we select the "road" in the scene as the planar surface $\mathbf{\Pi}$. This is an ideal choice because roads are visible on most of the images in KITTI. Computing the homography matrix, requires a set of corresponding pairs of pixels on the road (at least 4 pairs) between two consecutive images, i.e. a source frame $\mathbf{I}_s$ and a target frame $\mathbf{I}_t$. In order to achieve this, we calculate the optical flow between these frames using [Teed and Deng, 2020] and find the corresponding pairs of pixels. Next, we use the semantic segmentation network [Zhu et al., 2019] to only select the pixels belonging to the "road" class. After finding a set of corresponding pairs of road pixels, we use OpenCV to compute the homography in a robust way using a RANSAC-based method. We repeat this process to calculate the homography $\mathbf{H}$ for all of the temporally consecutive pairs of images in the dataset. For any consecutive pair of images $\mathbf{I}_1, \mathbf{I}_2$ and the computed

homography $\mathbf{H}$ between them, we use $\mathbf{H}$ to warp $\mathbf{I}_1$ towards $\mathbf{I}_2$ and then we use inverse homography matrix $\mathbf{H}^{-1}$ for warping $\mathbf{I}_2$ towards $\mathbf{I}_1$.

4.2.2 Results

The depth estimation results of our DepthP+P model on the Eigen split of KITTI dataset is presented in Table 4.1. This is the first instance of training a deep neural network for self-supervised depth estimation that uses the P+P paradigm (Equation (4.2)) for view synthesis. We can see that as a DepthP+P performs significantly better than the initial models which were based on predicting depth and pose. After [Zhou et al., 2017] proposed the seminal SfMLearner approach, multiple improvements were proposed that resulted in better performing models. Thus, we believe that as future work, our approach can benefit from similar improvements to enable it to achieve better results than the state-of-the-arts models that are based on the previous approach. i.e., backprojecting the pixels, transforming using the rotation and translation and then projecting again.

Note that the DNet model proposed by [Xue et al., 2020] is not trained to estimate depth in metric scale. Instead, similar to other scale-unaware models, they generate depth outputs in an unknown scale. Then at test time, they rely on a visible ground plane to recover the scale of the network. As illustrated in Figure 4.3, their method completely fails to the target image does not contain a ground plane. As can be seen in this figure, a ground plane is not visible in this KITTI example and DNet [Xue et al., 2020] outputs completely wrong predictions because it cannot recover the scale and. Although DepthP+P relies on the existence of a ground plane (road) during training, it does not need a ground plane during to be available at test time. As a result it still performs well. For reference, DepthP+P achieves the absolute relative error of 0.252 on this sample, while the error for DNet [Xue et al., 2020] is 1.178. [Wagstaff and Kelly, 2021] achieve better results, but they rely on a pretrained depth network and a pretrained plane segmentation network, while our approach does not have these limitations.

Note that DepthP+P can also benefit from additional stereo supervision. In

Table 4.1: **Quantitative Results for Monocular Training.** We compare **DepthP+P** to previous methods that were trained only using monocular sequences on KITTI. The **scale** column indicates whether the model is capable of predicting depth in metric units. We provide results with both original ground truth and the improved ground truth. The results are shown for the input resolution 640×192. In each column, the best method is written in bold and the second best is underline.

	Method	**Scale**	Lower Better				Higher Better		
			Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original Ground Truth	[Zhou et al., 2017]	✗	0.183	1.595	6.709	0.270	0.734	0.902	0.959
	[Yang et al., 2018b]	✗	0.182	1.481	6.501	0.267	0.725	0.906	0.963
	[Mahjourian et al., 2018]	✗	0.163	1.240	6.220	0.250	0.762	0.916	0.968
	[Yin and Shi, 2018]	✗	0.149	1.060	5.567	2.226	0.796	0.935	0.975
	[Wang et al., 2018]	✗	0.151	1.257	5.583	0.228	0.810	0.936	0.974
	[Zou et al., 2018]	✗	0.150	1.124	5.507	0.223	0.806	0.933	0.973
	[Yang et al., 2018a]	✗	0.162	1.352	6.276	0.252	-	-	-
	[Ranjan et al., 2019]	✗	0.148	1.149	5.464	0.226	0.815	0.935	0.973
	[Luo et al., 2019]	✗	0.141	1.029	5.350	0.216	0.816	0.941	0.976
	[Chen et al., 2019]	✗	0.135	1.070	5.230	0.210	0.841	0.948	0.980
	[Godard et al., 2019]	✗	**0.110**	0.831	4.642	**0.187**	**0.883**	**0.962**	**0.982**
	[Guizilini et al., 2020a]	✗	0.111	**0.785**	**4.601**	0.189	0.878	0.960	**0.982**
	[Safadoust and Güney, 2021]	✗	**0.110**	0.792	4.700	0.189	0.881	0.960	**0.982**
	[Xue et al., 2020]	✓	0.118	0.925	4.918	0.199	0.862	0.953	0.979
	[Wagstaff and Kelly, 2021]	✓	0.123	0.996	5.253	0.213	0.840	0.947	0.978
	DepthP+P (Ours)	✓	0.152	1.322	6.185	0.239	0.781	0.920	0.970
Improved GT [Uhrig et al., 2017]	Zhou et al. [Zhou et al., 2017]	✗	0.176	1.532	6.129	0.244	0.758	0.921	0.971
	[Mahjourian et al., 2018]	✗	0.134	0.983	5.501	0.203	0.827	0.944	0.981
	[Yin and Shi, 2018]	✗	0.132	0.994	5.240	0.193	0.833	0.953	0.985
	Wang et al. [Wang et al., 2018]	✗	0.126	0.866	4.932	0.185	0.851	0.958	0.986
	[Ranjan et al., 2019]	✗	0.123	0.881	4.834	0.181	0.860	0.959	0.985
	[Luo et al., 2019]	✗	0.120	0.789	4.755	0.177	0.856	0.961	0.987
	[Godard et al., 2019]	✗	0.085	0.468	3.672	0.128	0.921	0.985	0.995
	[Safadoust and Güney, 2021]	✗	0.085	0.458	3.779	0.131	0.919	0.985	**0.996**
	[Guizilini et al., 2020a]	✗	**0.078**	**0.420**	**3.485**	**0.121**	**0.931**	**0.986**	**0.996**
	DepthP+P (Ours)	✓	0.134	1.042	5.566	0.199	0.820	0.946	0.983

the proposed approach, we train the model using the monocular supervision from the P+P paradigm. Additionally, the other frame in the stereo pair can be warped into the to the target frame, using the known camera baseline and the estimated depth, for additional supervision signal. In Table 4.2, we report the performances

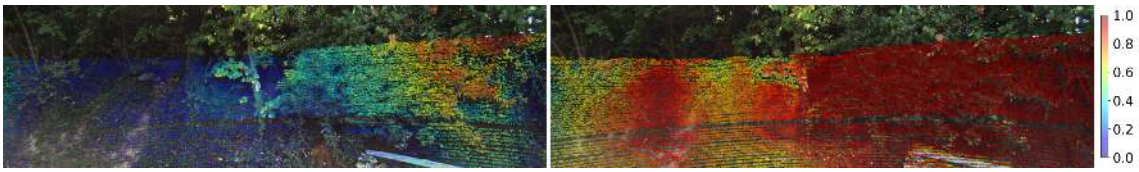

Figure 4.3: **Qualitative Comparison With Dnet.** We visualise the absolute relative error of our DepthP+P (**left**) with DNet [Xue et al., 2020] (**right**) on a sample from KITTI that does not contain a visible ground plane. The colourbar on the right represents the values of the absolute relative error metric. The maximum error is capped at the value of 1.0 for visualisation. DNet completely fails to estimate the depth due to the wrong scale recovery because the ground plane is not visible. Our model does not have this limitation and can perform well. For this sample, the absolute relative error of DepthP+P is 0.252, while it is 0.252 for DNet.

of our models that were trained with this additional stereo supervision. It can be seen that, by using stereo supervision our DepthP+P performs significantly well and outperforms all other models except for [Godard et al., 2019]. DepthP+P can obtain comparable results to [Godard et al., 2019] when using a ResNet-50 backbone instead of a ResNet-18 architecture [Godard et al., 2019].

We provide qualitative results of our models in Figure 4.4. We compare our monocular and stereo approaches with [Godard et al., 2019] and [Xue et al., 2020] on the KITTI dataset. Our stereo model generates sharp outputs, capturing the boundaries very well. Neither our monocular model, nor our stereo model suffers from artefacts such as the wrong depth prediction for the road lane line, as shown in the last row of Figure 4.4.

In our experiments so far, we have assumed that camera plane is perfectly perpendicular to the road plane. In other words, $\mathbf{N} = [0, 1, 0]^T$. However, the road planes can be tilted and are not always perfectly horizontal. e.g. one side of the road can be lower or higher than the other. Another example would be an downhill where the road slopes downwards. To investigate the effect of this assumption on the performance of our model, we perform an experiment in which we use ground truth depth data to fit a plane to the 3D road points and calculate its normal vector. Formally, during training, the road pixels are backprojected to 3D using their ground truth depth. Next a plane is fitted on these 3D points. Finally the normal vector

Table 4.2: **Quantitative Results with Additional Stereo Supervision.** We compare DepthP+P to previous approaches that use additional stereo supervision on the KITTI dataset. Additional stereo supervision improves the performance of DepthP+P significantly. Using ResNet-50, DepthP+P achieves comparable results with [Godard et al., 2019].

	Method	Lower Better				Higher Better		
		Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original GT	[Li et al., 2018]	0.183	1.730	6.570	0.268	-	-	-
	[Zhan et al., 2018]	0.135	1.132	5.585	0.229	0.820	0.933	0.971
	[Luo et al., 2019]	0.128	0.935	5.011	0.209	0.831	0.945	**0.979**
	[Godard et al., 2019]	**0.106**	**0.818**	**4.750**	**0.196**	**0.874**	**0.957**	**0.979**
	DepthP+P (ResNet-18)	0.110	0.907	4.888	0.199	0.867	0.954	**0.979**
	DepthP+P (ResNet50)	**0.106**	0.900	4.828	0.198	0.871	0.954	**0.979**
Improved GT	[Zhan et al., 2018]	0.130	1.520	5.184	0.205	0.859	0.955	0.981
	[Luo et al., 2019]	0.123	0.754	4.453	0.172	0.863	0.964	0.989
	[Godard et al., 2019]	**0.080**	**0.466**	**3.681**	**0.127**	**0.926**	**0.985**	**0.995**
	DepthP+P (ResNet-18)	0.088	0.572	3.905	0.138	0.911	0.981	0.994
	DepthP+P (ResNet-50)	0.084	0.543	3.784	0.134	0.916	0.982	**0.995**

of the fitted plane is used as the normal vector in Equation (4.7). In Table 4.3 we report the results for this experiment. We can see that calculating a more accurate normal vector and using it in our formulation results in improved performance. It is important to note that, we only used ground truth depth to compute the normal vector, not for supervising the depth outputs. As a result, it can be concluded that a more accurate prediction of the normal vector during training, can improve the results of this approach.

Table 4.3: **Effect of Calculating Normal Vector for Road.** This table analyses the effect of normal vector **N** for the road on the results. Fixed Normal assumes that the road is perfectly flat. GT Normal uses the ground truth depth values and produces the point clouds for the road region, fits a plane to these 3D points and calculates the normal vector. GT Normal achieves better results than Fixed Normal, demonstrating that a more accurate normal vector will result in improved performance.

Method	Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
DepthP+P (Fixed Normal)	0.152	1.322	6.185	0.239	**0.781**	0.920	0.970
DepthP+P (GT Normal)	**0.142**	**1.127**	**5.922**	**0.238**	0.780	**0.921**	**0.971**

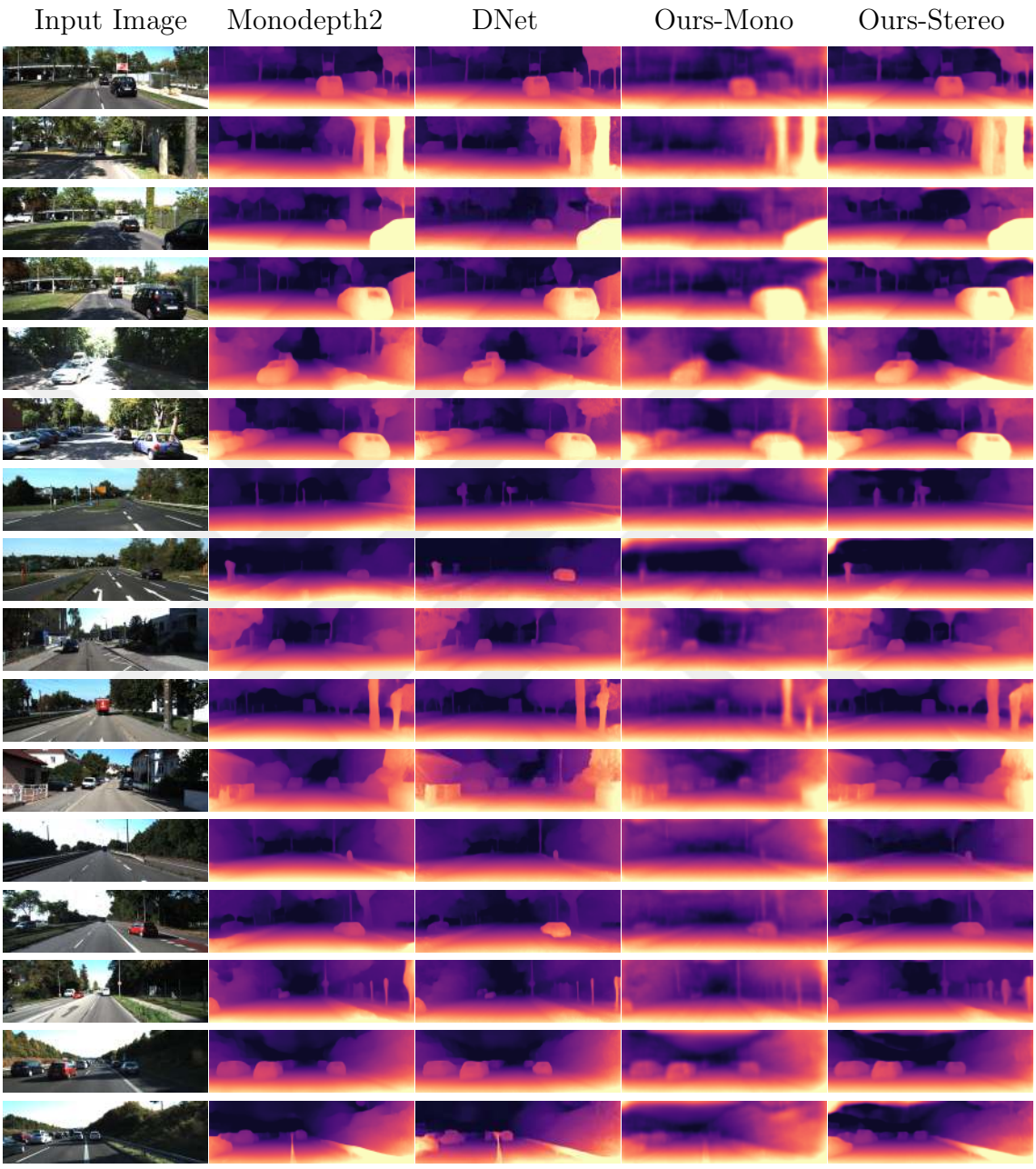

Figure 4.4: **Qualitative Results.** In each row, for an input image, we show the results of [Godard et al., 2019] and [Xue et al., 2020] and compare them with our models (the right-most two columns). *Ours-Mono* is the model that was trained only with monocular supervision using a ResNet-18 backbone, and *Ours-Stereo* is the model that was trained with additional stereo supervision using a ResNet-50 backbone.

Chapter 5

CONCLUSION

In this thesis, we proposed two methods to tackle two important problems of self-supervised monocular depth estimation methods. The MonoDepthSeg approach performs joint scene decomposition and depth prediction and addresses the problem of moving objects in the scene, which violate the static scene assumption. Without using any labels for semantics, our model can discover components on the image corresponding to independently dynamic regions. We showed that depth estimations can benefit from estimating moving objects and their motion. However, there is a gap between the quality of our estimated masks and using the state-of-the-art segmentation networks networks for producing them. Increasing the quality of our masks can be an exciting future direction. Our proposed method is generic and can be adopted by different architectures. At the risk of increasing complexity, a direct extension to our approach is to replace the ResNet with the state-of-the-art PackNet architecture [Guizilini et al., 2020a]

In our framework, the maximum number of components is a hyperparameter that needs to be fixed for the model. However, scenes are not equally complex. Previous monocular depth estimation models have shown that ego-motion alone can explain the structure of most scenes. We showed that by decomposing the scene and modeling several motions, the performance of these models, especially in the dynamic regions, can be improved. Predefining the maximum number of components is a limitation of MonoDepthSeg. Therefore, another research direction can be generalising this approach to handle a variable number of components. For example, iterative solutions can be used to decompose the scene layer by layer until there is nothing left to explain.

The other problem of current self-supervised depth estimation methods is scale ambiguity. We proposed DepthP+P, a method that uses the planar parallax paradigm

to estimate metrically accurate depths. We demonstrated that by only using the camera height information, our approach can predict depth in metric units. In contrast to previous methods, our method only needs to predict the camera translation, rather than the full rigid-body motion of the camera. However, a number of these previous models outperform our monocular model and produce much sharper depth estimates. Despite this, we believe that our approach can be a first step toward unlocking the potential of the plane and parallax paradigm for scale-aware depth prediction in an efficient way.

Assuming a completely horizontal road plane is a limitation of DepthP+P. As a future study, methods for estimating the normal vector of the plane more accurately can be used during training to improve performance. Another interesting research direction is detecting moving objects in the scene using the rigidity constraints based on the planar parallax approach [Irani and Anandan, 1996].

BIBLIOGRAPHY

[Bartoccioni et al., 2021] Bartoccioni, F., Zablocki, É., Pérez, P., Cord, M., and Alahari, K. (2021). Lidartouch: Monocular metric depth estimation with a few-beam lidar. *arXiv preprint arXiv:2109.03569*.

[Bian et al., 2019] Bian, J., Li, Z., Wang, N., Zhan, H., Shen, C., Cheng, M.-M., and Reid, I. (2019). Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 35–45.

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

[Byravan et al., 2018] Byravan, A., Leeb, F., Meier, F., and Fox, D. (2018). SE3-Pose-Nets: Structured deep dynamics models for visuomotor control. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 1–8.

[Cao et al., 2019] Cao, Z., Kar, A., Hane, C., and Malik, J. (2019). Learning independent object motion from unlabelled stereoscopic videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5594–5603.

[Casser et al., 2019] Casser, V., Pirk, S., Mahjourian, R., and Angelova, A. (2019). 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.

[Chaney et al., 2019] Chaney, K., Zhu, A. Z., and Daniilidis, K. (2019). Learning event-based height from plane and parallax. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 3690–3696.

[Chen et al., 2018] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Chen et al., 2019] Chen, Y., Schmid, C., and Sminchisescu, C. (2019). 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.

[Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

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

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

[Gadelha et al., 2019] Gadelha, M., Wang, R., and Maji, S. (2019). Shape reconstruction using differentiable projections and deep priors. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Garg et al., 2016] Garg, R., Bg, V. K., Carneiro, G., and Reid, I. (2016). Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 740–756.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*.

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

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

[Godard et al., 2019] Godard, C., Mac Aodha, O., Firman, M., and Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Guizilini et al., 2020a] Guizilini, V., Ambrus, R., Pillai, S., Raventos, A., and Gaidon, A. (2020a). 3D packing for self-supervised monocular depth estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Guizilini et al., 2020b] Guizilini, V., Hou, R., Li, J., Ambrus, R., and Gaidon, A. (2020b). Semantically-guided representation learning for self-supervised monocular depth. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 2961–2969.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

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

[Irani and Anandan, 1996] Irani, M. and Anandan, P. (1996). Parallax geometry of pairs of points for 3d scene analysis. In Buxton, B. and Cipolla, R., editors, *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 17–30, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Irani et al., 2002] Irani, M., Anandan, P., and Cohen, M. (2002). Direct recovery of planar-parallax from multiple frames. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 24(11):1528–1534.

[Irani et al., 1998] Irani, M., Anandan, P., and Weinshall, D. (1998). From reference frames to reference planes: Multi-view parallax geometry and applications. In Burkhardt, H. and Neumann, B., editors, *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 829–845, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Jaderberg et al., 2015] Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2017–2025.

[Jiao et al., 2018] Jiao, J., Cao, Y., Song, Y., and Lau, R. (2018). 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.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv.org*, 1412.6980.

[Li et al., 2020] Li, H., Gordon, A., Zhao, H., Casser, V., and Angelova, A. (2020). Unsupervised monocular depth learning in dynamic scenes. *arXiv.org*, 2010.16404.

[Li et al., 2018] Li, R., Wang, S., Long, Z., and Gu, D. (2018). Undeepvo: Monocular visual odometry through unsupervised deep learning. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 7286–7291. IEEE.

[Luo et al., 2019] Luo, C., Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R., and Yuille, A. (2019). 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.

[Mahjourian et al., 2018] Mahjourian, R., Wicke, M., and Angelova, A. (2018). 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.

[Mohamed et al., 2020] Mohamed, E., Ewaisha, M., Siam, M., Rashed, H., Yogamani, S., and El-Sallab, A. (2020). Instancemotseg: Real-time instance motion segmentation for autonomous driving. *arXiv preprint arXiv:2008.07008*.

[Ranjan et al., 2019] Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J., and Black, M. J. (2019). 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.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241.

[Roussel et al., 2019] Roussel, T., Eycken, L. V., and Tuytelaars, T. (2019). Monocular depth estimation in new environments with absolute scale. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 1735–1741.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[Sabour et al., 2021] Sabour, S., Tagliasacchi, A., Yazdani, S., Hinton, G. E., and

Fleet, D. J. (2021). Unsupervised part representation by flow capsules. *arXiv.org*, 2011.13920.

[Safadoust and Güney, 2021] Safadoust, S. and Güney, F. (2021). Self-supervised monocular scene decomposition and depth estimation. In *International Conference on 3D Vision (3DV)*, pages 627–636.

[Sawhney, 1994] Sawhney (1994). 3d geometry from planar parallax. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 929–934.

[Teed and Deng, 2020] Teed, Z. and Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 402–419. Springer.

[Uhrig et al., 2017] Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., and Geiger, A. (2017). Sparsity invariant CNNs. In *Proc. of the International Conf. on 3D Vision (3DV)*.

[Vertens et al., 2017] Vertens, J., Valada, A., and Burgard, W. (2017). Smsnet: Semantic motion segmentation using deep convolutional neural networks. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, Canada.

[Wagstaff and Kelly, 2021] Wagstaff, B. and Kelly, J. (2021). Self-supervised scale recovery for monocular depth and egomotion estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2620–2627. IEEE.

[Wang et al., 2018] Wang, C., Miguel Buenaposada, J., Zhu, R., and Lucey, S. (2018). Learning depth from monocular videos using direct methods. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2022–2030.

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing (TIP)*, 13(4):600–612.

[Wulff et al., 2017] Wulff, J., Sevilla-Lara, L., and Black, M. J. (2017). Optical flow in mostly rigid scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4671–4680.

[Xue et al., 2020] Xue, F., Zhuo, G., Huang, Z., Fu, W., Wu, Z., and Ang, M. H. (2020). Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 2330–2337. IEEE.

[Yang et al., 2018a] Yang, Z., Wang, P., Wang, Y., Xu, W., and Nevatia, R. (2018a). LEGO: Learning edge with geometry all at once by watching videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 225–234.

[Yang et al., 2018b] Yang, Z., Wang, P., Xu, W., Zhao, L., and Nevatia, R. (2018b). Unsupervised learning of geometry from videos with edge-aware depth-normal consistency. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*.

[Yin and Shi, 2018] Yin, Z. and Shi, J. (2018). 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.

[Zhan et al., 2018] Zhan, H., Garg, R., Saroj Weerasekera, C., Li, K., Agarwal, H., and Reid, I. (2018). 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.

[Zhou et al., 2017] Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1851–1858.

[Zhu et al., 2019] Zhu, Y., Sapra, K., Reda, F. A., Shih, K. J., Newsam, S., Tao, A., and Catanzaro, B. (2019). Improving semantic segmentation via video propagation and label relaxation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Zou et al., 2018] Zou, Y., Luo, Z., and Huang, J.-B. (2018). 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.