SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

MonoRec++: Scene Flow-guided Accurate Dense Reconstruction for 3D Object Detection

Wenliang Peng

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

MonoRec++: Scene Flow-guided Accurate Dense Reconstruction for 3D Object Detection

MonoRec++: Präzise und Dichte Rekonstruktion für 3D-Objekterkennung mittels Szenenfluss

Author:	Wenliang Peng
Supervisor:	Prof. Dr. Daniel Cremers
Advisor:	Mariia Gladkova
Submission Date:	15.04.2023

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.04.2023 Wenliang Peng

Acknowledgments

I would like to express my sincere gratitude to my family, whose unwavering love, support, and encouragement throughout my academic journey have been invaluable. Their belief in my abilities has been a constant source of motivation, and I could not have reached this milestone without them.

I extend my heartfelt appreciation to my advisor, Mariia Gladkova, whose guidance, expertise, and patience have been instrumental in shaping my research and refining my skills. Her insightful feedback and constructive criticism have helped me to grow as a researcher.

I would also like to acknowledge the invaluable support and resources provided by my supervisor, Prof. Dr. Daniel Cremers, who leads the institute. His vision, leadership, and dedication to excellence have created an inspiring academic environment that has allowed me to thrive.

Finally, I would like to thank all the individuals who have contributed to my academic journey. Your support, guidance, and inspiration have been integral to my success, and I am deeply grateful for your contributions.

Abstract

This paper discusses the problem of depth estimation in computer vision and proposes a new approach to improve the accuracy of depth estimation for moving objects. The proposed method, called MonoRec++, builds upon the MonoRec approach and incorporates a new scene flow module to provide additional information about the moving objects. The resulting cost volume is used to estimate the depth of both static and dynamic objects. A multi-stage training process is also proposed for better integration of the different modules. Experimental results on the KITTI tracking dataset show that MonoRec++ outperforms MonoRec in depth estimation of both dynamic and static objects, particularly for the depth estimation of dynamic objects. The proposed method offers a promising solution for the depth estimation problem in computer vision and has potential applications in various fields, such as autonomous driving and 3D reconstruction.

Zusammenfassung

In diesem Papier wird das Problem der Tiefenschätzung in der Computer Vision diskutiert und ein neuer Ansatz vorgestellt, um die Genauigkeit der Tiefenschätzung für sich bewegende Objekte zu verbessern. Die vorgeschlagene Methode, namens MonoRec++, baut auf dem MonoRec-Ansatz auf und integriert ein neues Szenenflussmodul, um zusätzliche Informationen über die bewegten Objekte bereitzustellen. Das resultierende Kosten-Volumen wird verwendet, um die Tiefe von sowohl statischen als auch dynamischen Objekten zu schätzen. Es wird auch ein mehrstufiger Schulungsprozess vorgeschlagen, um eine bessere Integration der verschiedenen Module zu erreichen. Experimentelle Ergebnisse auf dem KITTI-Tracking-Datensatz zeigen, dass MonoRec++ MonoRec in der Tiefenschätzung von sowohl dynamischen als auch statischen Objekten übertrifft, insbesondere bei der Tiefenschätzung von dynamischen Objekten. Die vorgeschlagene Methode bietet eine vielversprechende Lösung für das Tiefenschätzungsproblem in der Computer Vision und hat potenzielle Anwendungen in verschiedenen Bereichen wie autonomes Fahren und 3D-Rekonstruktion.

Contents

1 Introduction

Although the study of understanding 3D scenes from 2D images has a long history, it has always been a difficult research area for Computer Vision (CV) due to its ambiguity. With the rapid development of CV and Deep Learning (DL), understanding 3D scenes from 2D images is increasingly an important part of it. Among them, depth estimation is a very critical task, which is related to the downstream scene understanding tasks, 3D object detection tasks and segmentation tasks, as well as autonomous driving [GLU12], 3D reconstruction [Iza+11] and other application scenarios. Although the current LIDAR or depth sensors can measure depth directly by Time of Flight (ToF) and other principles. But they are limited by the higher price of other sensors or multi-sensor synchronization problems. How to maximize the scene understanding and estimate the depth is still a problem worth investigating.

1.1 Problem Statement

Depth estimation for dynamic scenes is a challenging problem in computer vision. The accurate estimation of depth in such scenes is crucial for a variety of applications, including autonomous driving, robotics, and augmented reality. In dynamic scenes, the presence of moving objects makes it difficult to estimate depth accurately since the scene's structure changes over time. This problem is compounded by the fact that traditional depth estimation algorithms are designed to work on static scenes and may not be able to handle the complexities of dynamic scenes. Therefore, the development of effective and efficient depth estimation methods that can handle dynamic scenes is a critical area of research in computer vision.

To solve the depth estimation problem, there are various approaches taken in research. The first one is simply monocular depth estimation, which estimates the depth of a given image. The other one is Multi-view Stereo (MVS), which estimates the depth from the perspective of reconstructing 3D scenes from images. In addition, depth estimation also appears as a subtask in Visual Odometry (VO) [Yan+18; Yan+20], scene flow estimation [HR20] or motion segmentation [YR21]. Although these methods are flourishing in their fields, they all have some limitations. Monocular depth estimation

methods are more accurate for depth perception of moving objects because they rely only on a single image to estimate depth. However, they do not contain temporal information, so the depth estimation for images is scene dependent and lacks generalization capability. And the VO, scene flow, etc. that treat depth estimation as a subtask are more concerned with their main task, that is, estimating camera pose and scene flow estimation, lacking further exploration of the depth estimation task. While the MVS method aggregates information from multiple images, which is beneficial to obtain higher accuracy, however, it adopts the static assumption and performs poorly for depth estimation of moving objects.

Due to the assumption of static environment and geometric consistencies, MVS-based depth methods struggle with dynamic scenes. And among them, MonoRec[Wim+21] filters the cost volume of moving objects by the mask module prediction to predict the depth of moving objects more accurately. But it lacks specific information about moving objects, such as the direction and distance of movement.

1.2 Contribution

To improve the depth estimation of MonoRec for dynamic scenes, we propose MonoRec++, an extension to MonoRec with the following additions:

- We add a new scene flow module. The object moving flow provided by the scene flow module is used as an additional input to build the cost volume together with the original images. This cost volume contains information about the moving direction and distance of the moving object.
- The original MonoRec mask module for estimating moving objects is also replaced with a simpler and more accessible semantic segmentation mask to eliminate incorrect object moving flows.
- A reasonable corresponding multi-stage training process is designed for better integration of multiple modules.
- We also evaluate the metrics of MonoRec++ and the current mainstream depth estimation models for the depth estimation and downstream 3D object detection task, giving our analysis and results. On the KITTI tracking dataset [GLU12], our MonoRec++ outperforms MonoRec in depth estimation of both dynamic and static objects and especially demonstrates superior performance estimating the depth of dynamic objects.

1.3 Outline

The structure of the thesis can be divided into four sections, as described below:
Chapter 2 describes work in related areas, including monocular depth estimation, MVS and scene flow estimation.
Chapter 3 explains our method, from the fundamentals to the structure of our proposed model, the loss function, the training process and the related settings.
Chapter 4 illustrates the dataset we used and the qualitative and quantitative results of our model in terms of depth estimation and downstream 3D object detection task, giving our analysis.
Chapter 5 concludes with a summary of our results and gives possible future directions for continued research.

2 Related Work

Understanding the 3D world from 2D images is a hot and challenging topic in computer vision. In this chapter, we will introduce the work related to depth estimation and scene flow estimation.

We will also introduce the work of 3D object detection based on point clouds as a downstream task of depth estimation.

2.1 Depth Estimation

2.1.1 Monocular Depth Estimation

Monocular depth estimation, also known as single-view depth estimation, has the goal of estimating the depth of a given individual image. Supervised monocular depth estimation requires the ground truth depth of the image, e.g., [Fu+18; EPF14]. [EPF14] is the first CNN-based depth estimation method that leverages a coarse-grained network for global depth prediction and a fine-grained network for local optimization of depth. [Fu+18] discretizes the continuous depth values as a classification problem of depth and solves the depth estimation problem from another perspective. But obtaining the ground truth depth of the image is very challenging and relies on manual annotation or LIDAR sensors.

Unsupervised monocular depth estimation, on the other hand, does not rely on the ground truth depth. Instead of learning the depth directly, [GMB17] learns the disparity of left and right images by Left-Right Consistency to get the depth. [PTM18] proposes a geometric constraint that exploits the trinocular consistency to train the neural network and estimate the disparity using stereo images. [Zho+17] proposes a self-supervised method for depth estimation and ego-motion estimation via sequential images from a monocular camera. And additionally, an explainability prediction network is trained for solving the cases of illegal reprojection errors in the view, such as non-Lambertian surfaces, occlusions, and moving objects. [God+19], on the other hand, proposes to use minimum reprojection loss to solve the occlusion problem, multi-scale depth prediction to reduce artifacts, and auto-masking loss to solve the reprojection error in

stationary camera scenes. [Gui+20] designs novel packing and unpacking blocks by 3D convolution, preserving the details of the predicted depth. And the scale inconsistency of monocular depth estimation is solved by the proposed velocity loss.

However, unsupervised methods also suffer from poor depth estimation for weakly textured, non-Lambertian objects. And the depth estimates obtained by the unsupervised method still have scale problems when compared to the ground truth depth when trained using only monocular images.

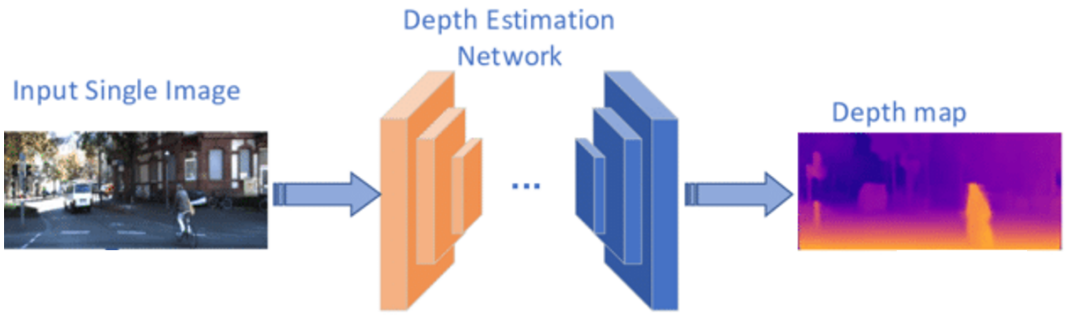


Figure 2.1: Monocular depth estimation pipeline[Li+18].

2.1.2 Multi-view Stereo

The original goal of the MVS method is to perform dense reconstruction of a static indoor 3D scene using the given disordered images and the corresponding poses, as in [SD99; Cam+08; LQ05; KS00], and Figure 2.2 shows a general pipeline of MVS. Traditional MVS-based work treats 3D reconstruction as an optimization problem, defining an energy function and optimizing that function to find the optimal solution.

However, more and more work is now using cost volume based Convolutional Neural Network (CNN), which has led to a substantial improvement in reconstruction accuracy. [Hua+18] proposes the first MVS method based on deep learning and cost volume, which computes a set of cost volumes from neighboring and reference images and uses CNN to predict the depth map. [Yao+18] extracts the feature maps of multiple temporal images by a shared feature extractor and constructs a single cost volume by variance, which can be applied to 3D convolution to obtain depth maps. [Yao+19] saves memory by using Gate Recurrent Unit (GRU) instead of 3D convolution on top of [Yao+18]. [Dai+19] enables self-supervised training of the cost volume based model through

the consistency between depth maps of multiple views and makes the model more robust by the learned occlusion maps. [Xue+19] achieves end-to-end depth estimation using Conditional Random Field (CRF) via Recurrent Neural Network (RNN) after cost volume regularization. [Gu+20] uses a multi-level pyramid to achieve cost volume and depth estimation from coarse to fine. While [ZUB18] leverages cost volume based fix band module and narrow band module to achieve coarse to fine depth estimation.

In addition to traditional indoor scene reconstruction work, there is now also work applying MVS theory to depth estimation of outdoor scenes. [Wim+21; Wat+21] uses cost volume to aggregate information from multiple frames to improve depth estimation of autonomous driving scenes.

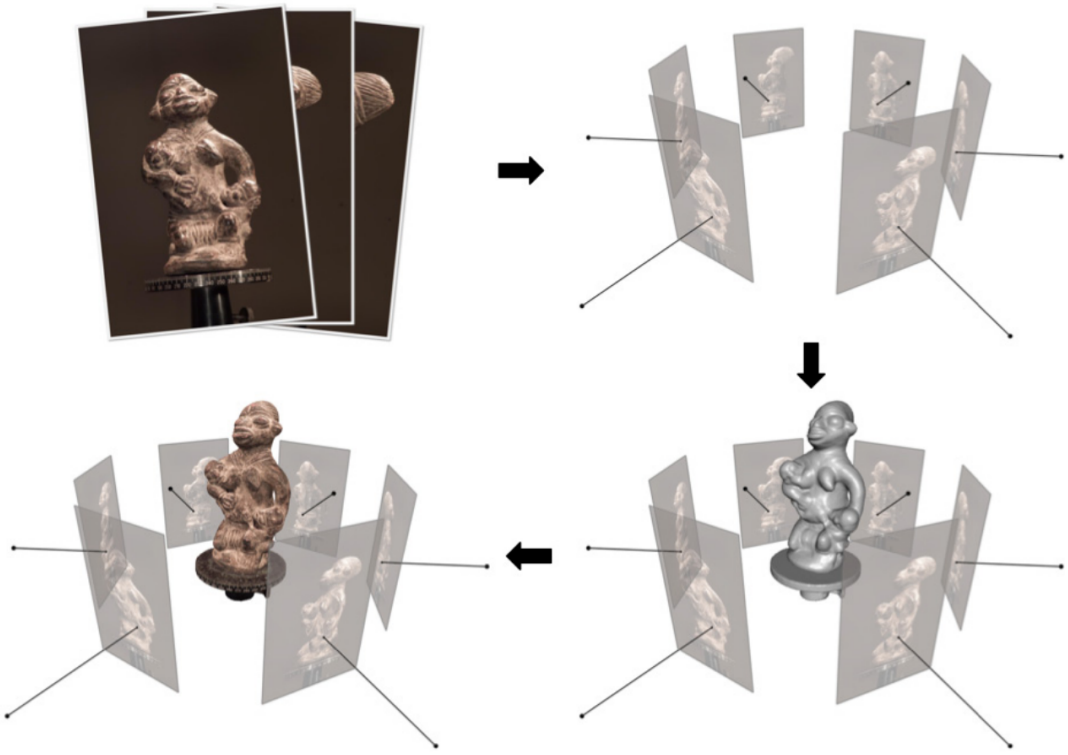


Figure 2.2: A general pipeline of MVS [FH+15].

2.1.3 Depth Estimation in Dynamic Environment

The handling of moving objects has always been a difficult problem in 3D vision. For example, many depth estimation methods do not explore information about moving

objects, and numerous MVS methods are based on static scene assumptions.

A lot of work has also been published on depth estimation for handling dynamic scenes. [Ran+19] uses the proposed competitive collaboration mechanism to introduce moving object segmentation and dynamic and static optical flow using a multi-task joint training approach. [Zha+21b] implements a test-time training framework using Multilayer Perceptron (MLP) to generate scene flow to model the movement of objects and to supervise the model with precomputed optical flow for temporal consistency of depth estimation. While [Kli+20] and [Lee+21] propose the use of semantic segmentation models to solve the problem of dynamic objects violating static scene assumptions. [Kli+20] addresses the dynamic object problem by introducing a new semantically-masked photometric loss and also proposes a method for identifying moving objects. With this method, the moving objects can be excluded from the training loss computation, while the non-moving objects continue to contribute to the overall loss. [Lee+21] proposes a method that incorporates instance-aware view synthesis and unified projection consistency into the training loss. The method first decomposes the image into background and object regions using a predicted instance mask and then warps each region to compute photometric consistency. For moving people, the model of [Li+19] learns the human pose and shape prior directly, bypassing the triangulation problem of moving objects. [Ran+16] and [RYA14] use motion segmentation to improve depth estimation. [Ran+16] proposes an approach that involves two stages. Firstly, the approach segments the dynamic scene into a set of motion models using a novel motion segmentation algorithm. Secondly, the approach assembles the scene by considering the different components and their location relative to the camera. [Li+21] decomposes the translation of the object in the image as the sum of its 3D translation and the translation of the camera. And the movement of the objects is constrained by regularization. Based on the monocular depth estimation framework, [Cas+19] proposes to estimate the movement of objects in images using separate models and adapt the models to unseen scenes by online refinement. [Wat+21] uses dynamic masks and a separate monocular depth estimation network to help learn the depth estimation of dynamic objects during the training phase.

(a) (b)

Figure 2.3: Dynamic environment from KITTI tracking dataset.

2.1.4 MonoRec

MonoRec is a depth estimation model based on MVS, as shown in Figure 2.4.

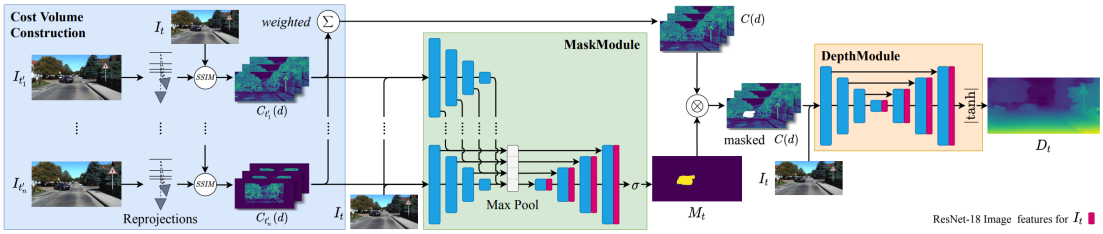


Figure 2.4: **The architecture of MonoRec**[Wim+21]: MonoRec aggregates multiple frames of images and generates a cost volume, and then uses the cost volume and image information as input to estimate the mask of moving objects in the mask module. The cost volume is filtered using the mask of moving objects, and the value of the cost volume with moving objects is set to 0. The masked cost volume and image information are used as input to estimate the depth of the image in the depth module. The masked cost volume can avoid estimating the depth error of the cost volume with static scene priors.

Nevertheless, MonoRec is able to filter the depth of moving objects in the reconstruction task to obtain more accurate static scene reconstruction results. However, for moving object depth estimation, MonoRec needs to use the trained mask module to detect moving objects and filter the cost volume of moving objects, forcing the value of cost volume with moving objects to 0, and only using image information and semantic segmentation information to estimate the depth of moving objects. It is not wise to introduce the moving object prior in this way, which changes the original calculation result of cost volume without introducing specific information about the moving object, such as the direction and distance of the object. In general, real-world data contains many dynamic objects that are not explicitly represented.

2.2 Scene Flow Estimation

Optical flow is a 2D moving field in the image plane, while scene flow is a 3D version of optical flow for describing the moving field of points in 3D space and was first proposed by [Ved+99].

2.2.1 Point Cloud Based Scene Flow Estimation

Scene flow estimation based on point clouds has attracted significant attention in recent years due to its potential applications in autonomous driving, robotics, and virtual reality. The point cloud-based scene flow estimation task is to estimate the relative motion vector of each point between different frames in the 3D point cloud, as shown in Figure 2.5. [LQG19] proposes a novel network structure FlowNet3D for estimating scene flows in two consecutive frames of point clouds. And two new learning layers are introduced on the point clouds: flow embedding layer is used to associate two point clouds to give flow embedding features. set upconv layer extends the feature vector from one set of points to another set of points. [Wan+20b] adds geometric feature supervision to FlowNet3D to improve the performance of scene flow estimation. Specifically, it utilizes point-to-plane distance and cosine distance to supervise the scene flow. Similarly, [Beh+19] exploits geometric relationships in the deep network and is able to jointly predict the 3D scene flow of the point cloud as well as the 3D bounding box and rigid body motion of objects in the scene through multiple decoders. [Wu+19a] proposed a new learnable cost volume layer that is capable of performing convolution on the cost volume without generating a dense 4D tensor. Utilizing this new layer, a new model called PointPWC-Net was introduced for estimating scene flow from two consecutive point clouds in a coarse-to-fine manner. Additionally, the paper introduced self-supervised losses which are capable of training PointPWC-Net without the requirement of ground truth labels. [Wei+21] presents a new approach called point-voxel correlation fields, which combines the benefits of point-based and voxel-based correlations for scene flow estimation of point clouds. To achieve this, they use K-Nearest Neighbor search to find neighboring points for point-based correlations and voxelization for multi-scale pyramid correlation voxels for voxel-based correlations. By leveraging the geometry of rigid scene flow, [Goj+21] introduces an inductive bias into the network to learn from weak supervision signals like background masks and ego-motion. The method breaks down the scene into objects that move rigidly, enabling reasoning on the object level rather than at the point level.

2.2.2 Image Based Scene Flow Estimation

Similar to MVS, the early scene flow estimation [HD07; VRS14; VSR13; VSR15] uses the traditional optimization paradigm. The energy function is defined and the scene flow is estimated using optimization methods to minimize the energy function, but only limited performance is achieved.

The current popular CNN breaks the original scene flow estimation paradigm, and

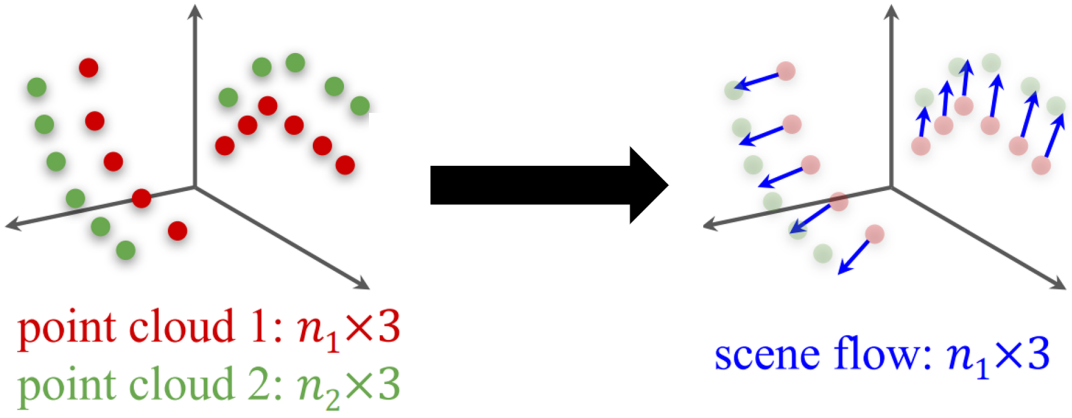
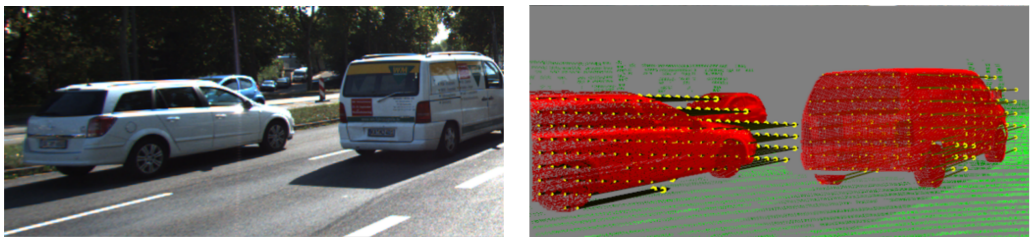


Figure 2.5: Point cloud based scene flow estimation [LQG19].

CNN-based scene flow estimation methods significantly improve the accuracy of estimation. [Ilg+18] focuses on the occlusion problem of scene flow. [Jia+19] proposes a shareable encoder for multi-task learning, which often occurs in scene flow, saving computational resources. The [Ma+19a] treats the scene flow as the motion of multiple actors, estimates the optical flow, disparity and instance segmentation separately minimizing the energy function with Gaussian Newton algorithm. The [Lv+18] divides the image into rigid and non-rigid regions and synthesizes the scene flow by ego motion in the rigid region and optical flow in the non-rigid region. [Luo+19] estimate the optical flow, depth and camera motion using three sub-networks respectively and send them to holistic 3D motion parser to calculate the motion of background and moving objects by geometric relationship to get the scene flow. [Liu+19] contains a depth estimation network and an optical flow estimation network to estimate scene flows by joint training and geometric relationships between multiple tasks. [Lee+19] contains an estimated self-motion network, a depth estimation network, and a residual flow network. The ego motion and object motions are decoupled by stereo-based depth estimation, and the optical flow is synthesized using the residual flow and rigid flow. [HR20] leverages only one encoder to learn the disparity and estimate the scene flow directly. Based on the method of solving optical flow [TD20], [TD21] uses Rigid-Motion Embeddings to group pixels as moving rigid objects and iteratively update the predicted scene flow.

3D scene flow estimation based on 2D images is essentially an ill-pose problem, therefore, much work has been done to estimate scene flow leveraging depth estimation, optical flow or motion and segmentation of moving objects.



(a) (b)

Figure 2.6: Image based scene flow estimation [Sch+18].

2.3 Point Cloud Based 3D Object Detection

2.3.1 LIDAR Based 3D Object Detection

Nowadays, the model of LIDAR-based point cloud has been a hot research topic in 3D object detection tasks because of its ability to obtain the ground truth depth of the scene and directly represent the 3D information in space. However, due to its sparse, irregular and other characteristics, several methods of processing point cloud data have emerged. [ZT18] represents the sparse point cloud as voxels in space and uses 3D convolution to extract point features. The main problem of [ZT18] is that the data representation is relatively inefficient and the 3D convolution in the middle layer is too computationally intensive, resulting in its slow running speed. [YML18] proposes sparse convolution, which avoids inefficient computation, improves the running speed, and reduces memory. [SWL19] is the first two-stage 3D object detection method using only the raw point cloud. [SWL19] utilizes the detection box of the foreground point to reduce the search scope of the detection box and proposes a refinement of the canonical coordinates and bin-based loss. [Lan+19] represents the sparse point cloud as pillars, and replaces the 3D convolution operation with 2D convolution, reducing the computational effort of the convolution operation. And [Qi+18] is based on [Qi+17a] and [Qi+17b], adding the image-based 2D object detection bounding boxes. [Che+17] projects the point cloud as the Bird-eye View (BEV) and foreground image, and fuse the 2D RGB image as well.

2.3.2 Pseudo-LIDAR Based 3D Object Detection

As an important sensor in self-driving cars and robots, LIDAR is able to directly sense 3D information of the environment. However, LIDAR systems can be expensive and may not always be suitable for certain scenarios due to their sensitivity to some weather

conditions. Pseudo-LIDAR offers a more cost-effective and powerful alternative as it requires only one or a few cheap cameras. The principle is to use a camera and a depth or disparity estimation algorithm to estimate the depth of the scene and create a 3D point cloud similar to a LIDAR scan and easily used with a collection of LIDAR-based algorithms for downstream tasks such as 3D object detection, localization and mapping. A general pseudo-LIDAR based object detection pipeline, as shown in Figure 2.7.

[Wan+19] first proposed to perform object detection using the pseudo-LIDAR of depth estimation model as an input to existing point cloud based object detection models. And [You+19] builds on [Wan+19] by estimating the depth using stereo images and correcting the pseudo point cloud using the sparse LIDAR point cloud. [Sun+20], on the other hand, applies stereo images to estimate the depth and obtains the pseudo point cloud of each instance using instance segmentation for object detection. [Ma+19b] selectively fuses depth information and RGB image information using attentional mechanisms. [WK19] performs end-to-end pseudo-LIDAR based object detection with a monocular depth estimation model and an instance segmentation model, proposing and attempting to solve the local misalignment and long tail problems of pseudo-point clouds. [Ma+20] mentions that coordinate transformations are more useful in object detection than pseudo point cloud representations and proposes a more generalized model PatchNet.

In contrast, [Par+21] proposes an end-to-end 3D object detection model based on deep pre-training and states that the generalization ability of pseudo LIDAR is weak and pseudo LIDAR requires in-domain deep fine tuning to generate satisfactory object detection results. [VAL19] points out that the point cloud density of pseudo-LIDAR is much higher than that of 64-line real LIDAR. In particular, more background point cloud data will cause more false positives and increase the computational effort. Therefore, the paper proposes a method to sparse the structure of point cloud data and points out that the performance gap between pseudo-LIDAR and ground truth LIDAR lies in its inaccurate depth estimation. [Wan+20a] argues that not all pixels are equal and that the depth of foreground objects is important in object detection, and proposes and leverages different optimization objectives and models to estimate foreground and background depths, respectively.

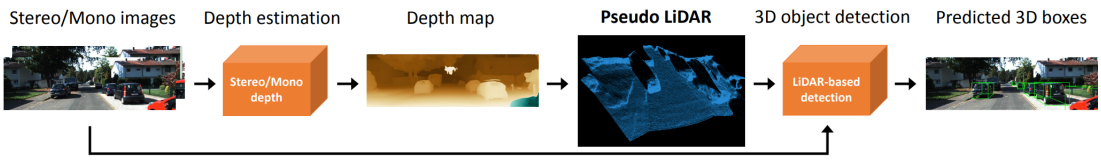


Figure 2.7: A general pseudo-LIDAR based object detection pipeline [Wan+19].

3 Method

Assuming we are given a set of consecutive frames and their corresponding camera poses, MonoRec++ is able to predict a dense depth map of the keyframe. MonoRec++ includes a depth module and a scene flow module. The scene flow module is used to provide the moving 3D vectors of moving objects. The depth module predicts the depth map from the cost volume guided by the scene flow. In this section, we first explain the relevant fundamentals and then describe the different modules of the architecture. Finally, the multi-stage training process and the implementation details are discussed.

3.1 Cost Volumes for Dynamic Scenes

3.1.1 Multi-view Geometry

In order to synthesize the current keyframe I_t from other frames $I_{t'}$, suppose we have the depth D_t of I_t, the intrinsics K of the camera, the relative camera pose $T_{t'\to t}$ from I_t to $I_{t'}$ and the moving flow $F_{t\to t'}$ of the moving object from I_t to $I_{t'}$. Then the synthesized frames from $I_{t'}$ to I_t can be represented as:

$$I_{t'\to t} = I_{t'} \left\langle \text{proj}\left(D_t, T_{t\to t'}, K, F_{t\to t'}\right)\right\rangle \tag{3.1}$$

Here $\langle\rangle$ is the differential sampling operation. With

$$\text{proj}\left(D_t, T_{t\to t'}, K, F_{t\to t'}\right) = \pi[K(R_{t\to t'} D_t K^{-1}\mathbf{x_t} + \mathbf{t}_{t\to t'} + F_{t\to t'})] \tag{3.2}$$

$$T_{t\to t'} = \begin{bmatrix} R_{t\to t'} & \mathbf{t}_{t\to t'} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{SE}_3 \qquad \mathbb{SE}_3 := \left\{R, \mathbf{t} \mid R \in \mathbb{SO}_3, \mathbf{t} \in \mathbb{R}^3\right\} \tag{3.3}$$

$\mathbf{x_t} \in \mathbb{R}^3 = (u, v, 1)^\top$ represents the homogeneous vector of 2D pixel coordinate of I_t, which corresponds to D_t. The function $\pi()$ is used to project and dehomogenise the points $\mathbf{p} \in \mathbb{R}^3 = (x, y, z)^\top$ in 3D space to the 2D plane $p \in \mathbb{R}^2 = (x/z, y/z)^\top$.

The proj() function we introduce contains a priori for object movement, allowing for better modeling of moving objects. When $F_{t\to t'} = 0$, the proj() function degenerates to a purely static scene of the proj() function as MonoRec.

$$I_{t'\to t,static} = I_{t'} \left\langle \text{proj}\left(D_t, T_{t\to t'}, K, 0\right)\right\rangle \tag{3.4}$$

3.1.2 Cost Volume

Cost volume is essentially a general term for a method of aggregating information from multiple frames. Like MonoRec, MonoRec++ uses Structural Similarity Index Measure (SSIM) and temporal images to build the cost volume. The temporal images $I_{t'}$ are projected to the current keyframe I_t through a set sequence of discrete depth values and relative poses, and the photometric error between the temporal images and the keyframe is calculated. But unlike MonoRec, MonoRec++ uses the moving flow $F_{t\to t'}$ to build the cost volume during the synthesis of $I_{t'\to t}$. The photometric error pe is defined by the following equation:

$$pe_{t'\to t}(\mathbf{x}, d) = \frac{1 - \text{SSIM}\left(I_{t'\to t}(\mathbf{x}, d), I_t(\mathbf{x})\right)}{2} \tag{3.5}$$

The patch size of SSIM is defined as 3×3. The error is limited from 0 to 1. While MonoRec uses the basic cost volume, that is, $I_{t'\to t,static}$ is used instead of $I_{t'\to t}$ as the synthesized frame. The basic cost volume is not limited to the number of temporal images $I_{t'}$ ($t' \in \{1, \cdots, N\}\backslash t$). However, since our method requires the moving flow $F_{t\to t'}$ for each temporal frame and keyframe by the scene flow module, we limit the number of temporal images to 2, $t' \in \{t-1, t+1\}$.

$$w_{t'}(\mathbf{x}) = 1 - \frac{1}{M-1} \sum_{d\neq d^*} \exp\left(-\alpha \left(pe_{t'\to t}(\mathbf{x}, d) - pe_{t'\to t}\left(\mathbf{x}, d^*_{t'}\right)\right)^2\right) \tag{3.6}$$

And we also define the matching confidence $w_{t'}(\mathbf{x})$ for the pe of a pixel and apply it to the pe with $d \in \left\{d_i \mid d_{\min} + \frac{i}{M} \cdot (d_{\min} - d_{\max})\right\}$. $w_{t'}(\mathbf{x})$ will approach 1 if there exists a clear and unique minimum pe with depth d, $w_{t'}(\mathbf{x})$ will approach 0 if there exists multiple depth values d such that $pe(d)$ is close to the minimum pe with $d^*_{t'} = \arg\min_d pe_{t'\to t}(\mathbf{x}, d)$.

Finally, the cost volume is aggregated by the multiple frames in the following equation:

$$C(\mathbf{x}, d) = 1 - 2 \cdot \frac{1}{\sum_{t'} \omega_{t'}} \sum_{t'} pe_{t'\to t}(\mathbf{x}, d)\omega_{t'}(\mathbf{x}) \tag{3.7}$$

The range of cost volume is from -1 to 1, representing the confidence that d is the correct depth from the perspective of photometric consistency for pixel p.

3.1.3 Moving Flow

For each pixel in image I_t, the goal of scene flow estimation is to estimate its 3D movement vector V to frame $I_{t'}$. If the estimated scene flow is projected onto the image plane, we can obtain the corresponding optical flow. Thus in order to reconstruct the moving flow we mentioned above from the scene flow, we need to obtain the relative pose $T_{t\to t'}$ and the depth map of the image I_t.

Assuming that we have the depth D_t of frame I_t, the relative pose $T_{t\to t'}$ and semantic segmentation mask $M_{seg,t}$, the moving flow $F_{t\to t'}$ can be obtained by the following equations. First,

$$F_{t\to t',real} = (D_t K^{-1}\mathbf{x_t} + S_{t\to t'}) - (R_{t\to t'} D_t K^{-1}\mathbf{x_t} + \mathbf{t}_{t\to t'}) \tag{3.8}$$

Ideally, the perfect moving flow can describe the movement of moving objects without any post-processing. But under realistic conditions, the moving flow provided by the scene flow module is not perfect. Therefore, we need masks that can classify moving objects to filter the moving flow of moving objects, while the original MonoRec needs to train the mask module, we can directly use the simple and accessible semantic segmentation mask of potential moving objects, such as pedestrians, cars, etc. Semantic segmentation masks of potentially moving objects were generated using Detectron2 pre-trained on the COCO dataset, which includes *person*, *bicycle*, *car*, *motorcycle*, *airplane*, *bus*, *train* and *truck*. Finally, the moving flow can be obtained by the following equation.

$$F_{t\to t'} = M_{seg,t} F_{t\to t',real} \tag{3.9}$$

3.2 Network Architecture

As shown in Figure 3.1, the Monorec++ network architecture consists of two submodules, the depth module and the scene flow module.

3.2.1 Scene Flow Module

The goal of the scene flow module is to provide information about moving objects for the input of the depth module, i.e., cost volume. Inspired by [HR20], we use its model

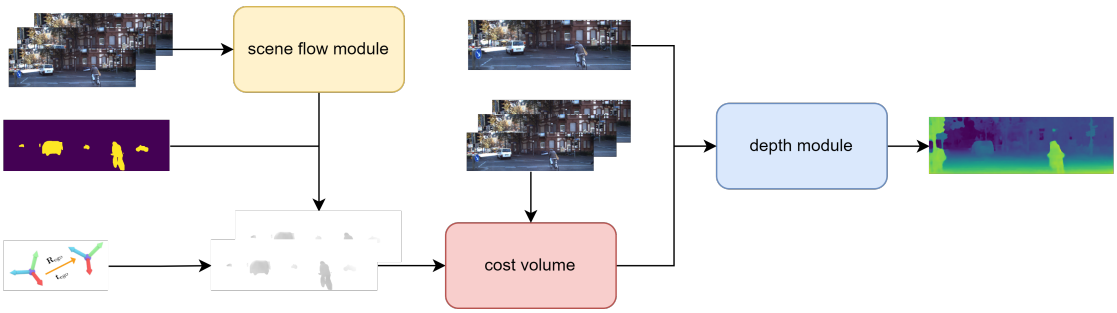


Figure 3.1: **The architecture of MonoRec++**: The scene flow module(Subsection 3.2.1) first predicts the scene flow and disparity between frames, then with the relative pose predicted by VO and the semantic segmentation mask, we can obtain the moving flow of potential moving objects. The depth module(Subsection 3.2.2) can use the moving flow and multiple frames to construct the cost volume(Section 3.1) to predict the depth map of the keyframe.

and training approach, which is able to obtain the scene flow of monocular videos and the disparity of stereo images in the inference phase by using only stereo images in the training phase in a self-supervised manner. Although we can estimate the depth through the disparity of the scene flow module, its depth estimation is not accurate. Therefore, we also need the depth module for more accurate depth estimation. And since the scene flow module only provides the scene flow with ego-motion, we also need to compensate the ego-motion with the given pose so that we can obtain the moving flow with only the information of moving objects.

3.2.2 Depth Module

We use the same structure as MonoRec for the depth module, whose purpose is to output the corresponding depth map given the cost volume and the image. However, unlike MonoRec, we do not use mask and cost volume to do pixel-level multiplication, eliminating the strong preference of cost volume for static scenes. Instead, we provide masked moving flow to the cost volume as a priori information about moving objects to assist the encoding of the cost volume.

3.3 Training Procedures

In this section, a multi-stage network training process is presented as shown in Figure 3.2. Specifically, the pretraining phase and the refinement phase are executed

sequentially. In the pretraining phase, all modules are trained separately. In the refinement phase, the scene flow module and depth module will be refined. The input image size for all modules is 256×832.

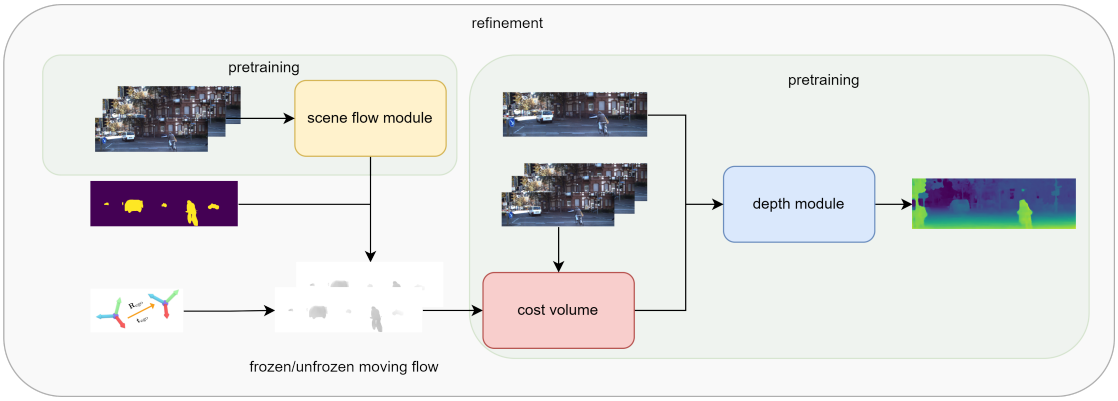


Figure 3.2: The multi-stage network training process.

3.3.1 Pretraining

Depth Module In the depth module, the input of the module is a simple concatenation of the image and cost volume. Despite the power of the neural network, we find that the depth module has difficulty in mining both image and cost volume information simultaneously. Inspired by the authors of [Wim+21], we propose a simple pretraining strategy. In the pretraining phase, we only use the image as input and do not use cost volume,i.e., set the cost volume to 0, as shown in Figure 3.3. Such a pretraining strategy allows the depth module to incrementally learn image data and cost volume. In the pretraining phase, the moving flow F is set to 0 and does not participate in the depth module pretraining. We use a simple self-supervised loss, and the depth module degenerates into a monocular depth estimation module. More specifically, the depth module takes cost volume $C(\mathbf{x}, d)$ without F as input and predicts the depth D_t. The supervision of the depth module is defined as a multi-scale self-supervised loss, similar to [Wim+21]. It combines a self-supervised photometric loss and an edge-aware smoothness term.

$$L_{depth} = \sum_{s=0}^{3} L_{self,s} + \alpha L_{smooth,s} \tag{3.10}$$

$$L_{self,s} = \min_{t^\star \in t' \cup \{t^S\}} \left(E_{self,photo} \left(I_{t^\star \to t}, I_t \right) \right) \tag{3.11}$$

$$E_{self,photo}\left(I_1, I_2\right) = \left(\lambda \frac{1 - \text{SSIM}\left(I_1, I_2\right)}{2} + (1-\lambda)\left\|I_1 - I_2\right\|_1\right) \tag{3.12}$$

with λ is 0.85, $\alpha = 10^{-3}/(2^s)$, $t' \in \{t-1, t+1\}$, $I_t{}^s$ is the stereo frame of I_t.

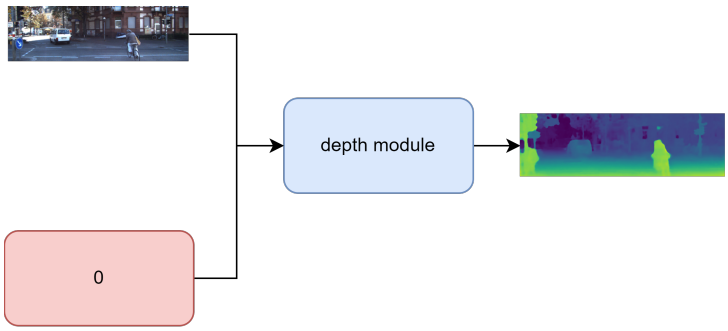


Figure 3.3: **The pretraining of the depth module**: We use 0 padding instead of cost volume as input to the depth module.

In the pretraining phase of the depth module, the depth module is pretrained for 70 epochs. The learning rate of the first 65 epochs is $1e^{-4}$, and the learning rate of the last 5 epochs is $1e^{-5}$.

Scene Flow Module We follow [HR20], the training of the scene flow module consists of two balanced loss functions, which are disparity loss $L_{d,t}$ and scene flow loss $L_{sf,t\to t'}$.

$$L_{sf,total,t\to t'} = \lambda_d L_{d,t} + \lambda_{sf} L_{sf,t\to t'} \tag{3.13}$$

with $L_{max} = \max(L_{d,t}, L_{sf,t\to t'})$, $\lambda_d = L_{d,t}/\lambda_{max}$, $\lambda_{sf} = L_{sf,t\to t'}/L_{max}$ and $t' \in \{t+1\}$.

Similar to [GMB17], the goal of disparity loss $L_{d,t}$ is to learn the disparity of the stereo image and thus to obtain the depth of the image and recover the 3D information of the image. It is composed of 2 parts, photometric loss $L_{d\_ph,t\to t'}$ and smoothness loss $L_{d\_sm,t}$.

$$L_{d,t} = L_{d\_ph,t} + \lambda_{d\_sm} L_{d\_sm,t} \tag{3.14}$$

where the value of λ_{d_sm} is 0.1. To supervise the disparity produced by the model, $L_{d\_ph,t}$ is used as a loss function to evaluate the photometric error between the left image I_t^l and the reconstructed left image $I_t^{l,d}$.

$$L_{d\_ph,t} = O_t^{l,disp} E_{self,photo}\left(I_t^l, I_t^{l,d}\right) \tag{3.15}$$

with $I_t^{l,d} = I_t^r(Disp_t^l)$ is the reconstructed left image from the right image and the disparity map of the left image. And this loss function penalizes only the unoccluded pixels. The occlusion map $O_t^{l,disp}$ is obtained from the disparity map of the right image $Disp_t^r$. And for local smoothness of disparity, as in [HR20], edge-aware 2nd-order smoothing loss $L_{d\_sm,t}$ is used.

In addition to the disparity loss $L_{d,t}$, the scene flow loss $L_{sf,t\to t'}$ is also needed. scene flow loss consists of 3 components, namely the photometric loss of the scene flow $L_{sf\_ph,t\to t'}$, the reconstruction loss of the 3D space $L_{sf\_pt,t\to t'}$, and the smoothing loss of the scene flow $L_{sf\_sm,t\to t'}$.

$$L_{sf,t\to t'} = L_{sf\_ph,t\to t'} + \lambda_{sf\_pt} L_{sf\_pt,t\to t'} + \lambda_{sf\_sm} L_{sf\_sm,t\to t'} \tag{3.16}$$

with regularization parameters λ_{sf_pt} is 0.2 and λ_{sf_sm} is 200.

In 3D space, we can reconstruct the spatial relationships by the disparity of the current frame and the next frame and the estimated scene flow.

$$D_t = b \cdot f_x / Disp_t \tag{3.17}$$

b is the baseline for stereo images, $fx = focal\_length * sx$, $focal\_length$ is the focal length of the camera and sx is the horizontal scale factor, $Disp_t$ is the disparity of the current frame.

$$\mathbf{p}_t = D_t K^{-1} \mathbf{x_t} \qquad \mathbf{p}_{t\to t'} = \mathbf{p}_t + S_{t\to t'} \qquad \mathbf{p}_{t'} = D_{t'} \cdot K^{-1} \pi[K \mathbf{p}_{t\to t'}] \tag{3.18}$$

Furthermore, we supervise the consistency between the 3D points $\mathbf{p}_{t\to t'}$ and $\mathbf{p}_{t'}$ between the current frame and the next frame. Again, this loss function penalizes only the pixels that are not occluded.

$$L_{sf\_pt,t\to t'} = O_t^{l,sf} \left\| \mathbf{p}_{t\to t'} - \mathbf{p}_{t'} \right\|_2 \tag{3.19}$$

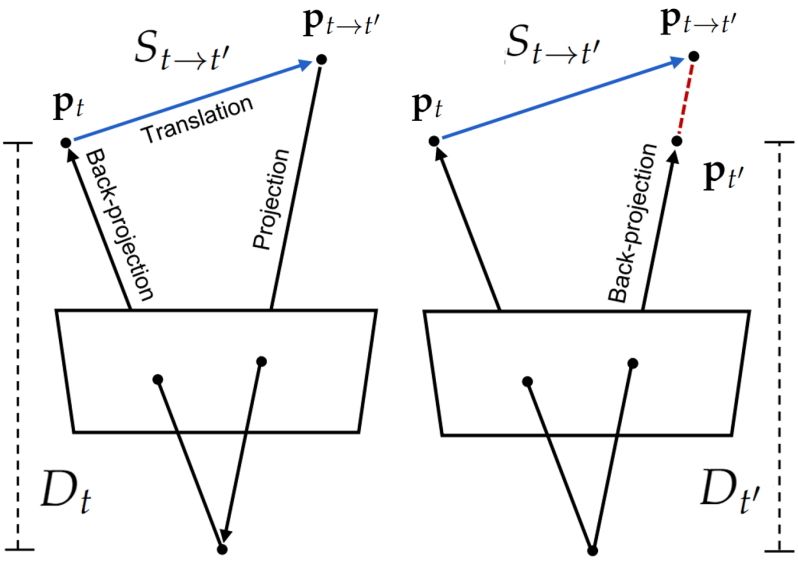
The purpose of $L_{sf\_ph,t}$ is to penalize the photometric error between the current frame I_t^l and the frame $I_t^{l,sf}$.

$$L_{sf\_ph,t\to t'} = O_t^{l,sf} E_{self,photo} \left(I_t^l, I_t^{l,sf} \right) \tag{3.20}$$

with

$$I_t^{l,sf} = I_{t'} \langle \pi[K\mathbf{p}_{t\to t'}] \rangle \tag{3.21}$$

Again, this loss function penalizes only the pixels that are not occluded. $O_t^{l,sf}$ is the occlusion map of the scene flow calculated from the backward scene flow $S_{t'\to t}$. And for local smoothness of scene flow, edge-aware 2nd-order smoothing loss $L_{sf\_sm,t\to t'}$ is used.



(a) Photometric loss (b) 3D point reconstruction loss

Figure 3.4: Visualization of scene flow loss [HR20].

The scene flow module is pretrained for 70 epochs, the initial learning rate is $2e^{-4}$, and then the learning rate is halved to $1e^{-4}$ at the 46th epoch.

3.3.2 Refinement

In order to optimize the moving flow generated by the scene flow, and also to make the cost volume contain information about moving objects, we propose the refinement phase. We first jointly train the depth module and the scene flow module for 10 epochs, the learning rate is $1e^{-4}$.

The loss function of the joint training is as follows.

$$L_{joint} = L_{depth} + \sum_{t'} \left(\lambda_{sf,total} L_{sf,total,t\to t'} + \lambda_{mf} L_{mf,t\to t'} \right) \tag{3.22}$$

with

$$L_{mf,t\to t'} = \left\| (1 - M_{seg,t}) F_{t\to t'} \right\|_2^2 \tag{3.23}$$

, $\lambda_{sf,total}$ is 0.01, λ_{mf} is 0.001 and $t' \in \{t-1, t+1\}$.

Here we optimize the scene flow by supervising the moving flow. The semantic segmentation mask $M_{seg,t}$ we used contains all potential moving objects. In other words, the pixels that are not masked are static, and the pixels that are masked may be static or moving. Therefore, we use square L2 loss to penalize the moving flow that is not masked. Because the moving flow of the static scene should be 0.

However, joint training makes the convergence speed of the depth module slow or just falls into a local optimum. Because the moving flow, which is part of the cost volume and the reprojection error, is constantly changing during training. Therefore, based on the joint training, we freeze the scene flow module and train the depth module for 20 epochs, the learning rate is $1e^{-4}$. The loss function is as L_{depth}.

4 Experiments and Results

4.1 Dataset

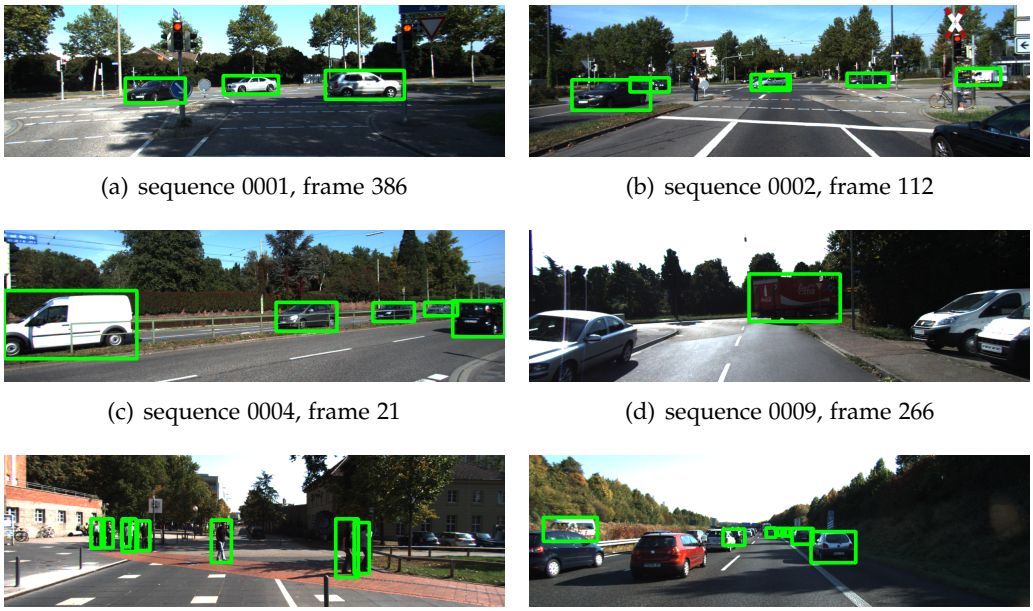
For depth estimation, Eigen split [EF15] of KITTI is one of the most commonly used approaches to divide the KITTI depth estimation dataset. However, Eigen split is only used for monocular depth estimation tasks. And Eigen split also does not provide the real mask of moving objects and cannot evaluate the depth estimation of moving objects. The KITTI tracking dataset is used because the KITTI tracking dataset can provide the track of the objects and the bounding box of the moving objects to help us identify the moving objects. The KITTI tracking dataset also provides LIDAR point cloud data for depth evaluation.

However, the track of moving objects and their bounding box from the KITTI tracking dataset are not sufficient for identifying specific pixels of moving objects. To determine the specific moving pixels, Detectron2 [Wu+19b] is also used to provide alternative moving masks. This is done as follows:

1. For all objects of the track, we calculate the $L2$ distance between their positions in 2 consecutive frames. If the class is *pedestrian* and the $L2$ distance is more than 0.1m, then we identify the object in these 2 frames as a moving object. If the category is *car*, *bicyclist*, *truck*, etc. and the $L2$ distance is more than 0.2m, then we consider the object in these 2 frames as a moving object.
2. Semantic segmentation masks of potentially moving objects were generated using Detectron2 pretrained on the COCO dataset, which includes *person*, *bicycle*, *car*, *motorcycle*, *airplane*, *bus*, *train* and *truck*.
3. We use the Hungarian algorithm [Kuh55] to assign the alternative mask for each bounding box of the moving object.

The generated moving object masks are shown in Figure 4.3.

The KITTI tracking dataset has a total of 21 sequences. We follow the [Voi+19] data split, using all 12 sequences of the [Voi+19] training dataset as our model training dataset, 4 sequences of the [Voi+19] validation dataset as our validation dataset, and 5 sequences



(a) sequence 0001, frame 386

(b) sequence 0002, frame 112

(c) sequence 0004, frame 21

(d) sequence 0009, frame 266

(e) sequence 0016, frame 12

(f) sequence 0020, frame 42

Figure 4.1: Examples of 2D tracking bounding boxes of moving objects.

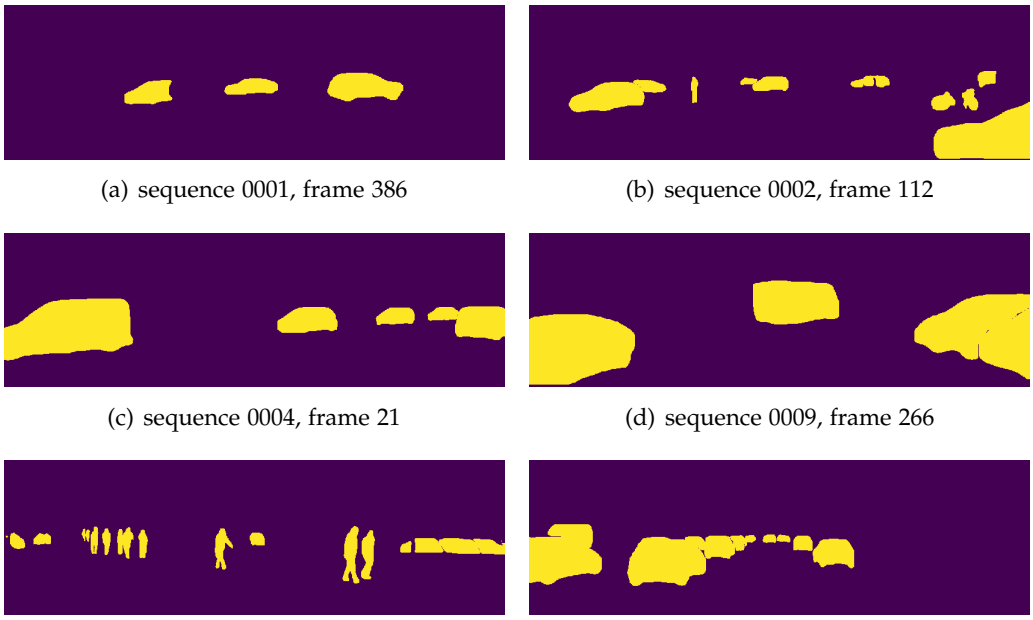
as our test dataset. And the sequences of the validation and test sets are balanced to ensure that the training and test sets contain similar distributions of moving objects. Finally, we choose sequences 0002, 0007, 0016, and 0018 as the validation set of the model and sequences 0006, 0008, 0010, 0013, and 0014 as the test set of the model.

For the pose, we use the transformations provided by the trained visual odometry system DF-VO [Zha+21a] on the KITTI tracking dataset split as described above. And to solve the scale ambiguity problem of the pose, DF-VO uses stereo images for training. In the inference phase, DF-VO only needs monocular images, which meets our requirements.

4.2 Depth Estimation

4.2.1 Metrics

For depth, we follow the previous work [EPF14]. Seven indicators were used to evaluate. They are:



(a) sequence 0001, frame 386

(b) sequence 0002, frame 112

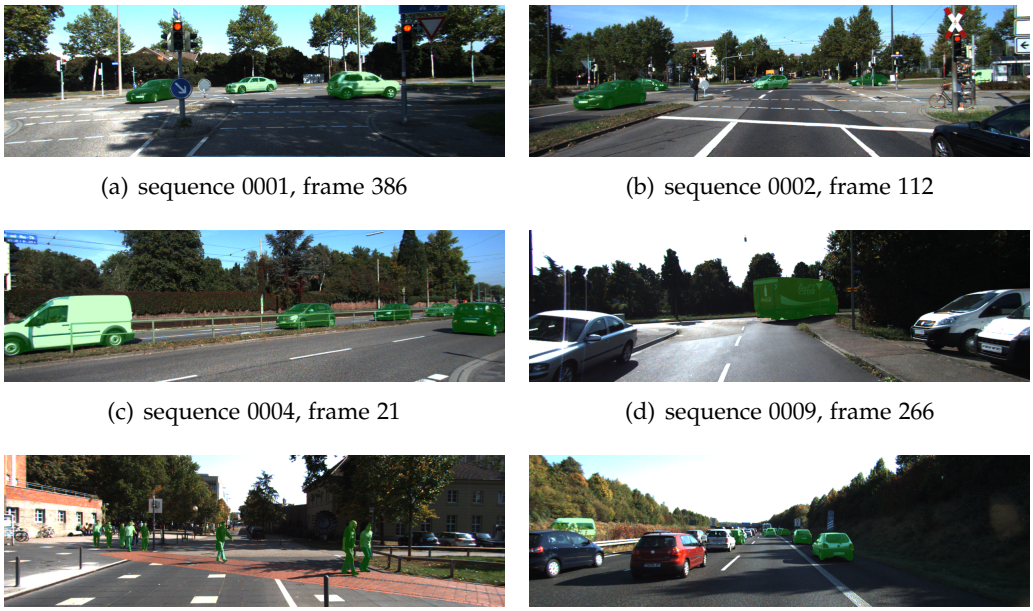
(c) sequence 0004, frame 21

(d) sequence 0009, frame 266

(e) sequence 0016, frame 12

(f) sequence 0020, frame 42

Figure 4.2: Examples of potential moving objects from Detectron2 semantic segmentation masks.



(a) sequence 0001, frame 386

(b) sequence 0002, frame 112

(c) sequence 0004, frame 21

(d) sequence 0009, frame 266

(e) sequence 0016, frame 12

(f) sequence 0020, frame 42

Figure 4.3: Examples of generated moving masks.

- Absolute Relative Difference (Abs Rel): $\frac{1}{n}\sum_p^n \frac{|d_p - \hat{d}_p|}{\hat{d}_p}$;
- Squared Relative difference (Sq Rel): $\frac{1}{n}\sum_p^n \frac{|d_p - \hat{d}_p|^2}{\hat{d}_p}$;
- Root Mean Squared Error (RMSE): $\sqrt{\frac{1}{n}\sum_p^n \left(d_p - \hat{d}_p\right)^2}$;
- RMSE$(\log_{10})$: $\frac{1}{n}\sum_p^n \left|\log_{10}\left(d_p\right) - \log_{10}\left(\hat{d}_p\right)\right|$;
- Threshold Accuracy (δ_i) : % of d_p s.t. $\max\left(\frac{d_p}{\hat{d}_p}, \frac{\hat{d}_p}{d_p}\right) = \delta < 1.25$
- Threshold Accuracy (δ_i) : % of d_p s.t. $\max\left(\frac{d_p}{\hat{d}_p}, \frac{\hat{d}_p}{d_p}\right) = \delta < 1.25^2$
- Threshold Accuracy (δ_i) : % of d_p s.t. $\max\left(\frac{d_p}{\hat{d}_p}, \frac{\hat{d}_p}{d_p}\right) = \delta < 1.25^3$

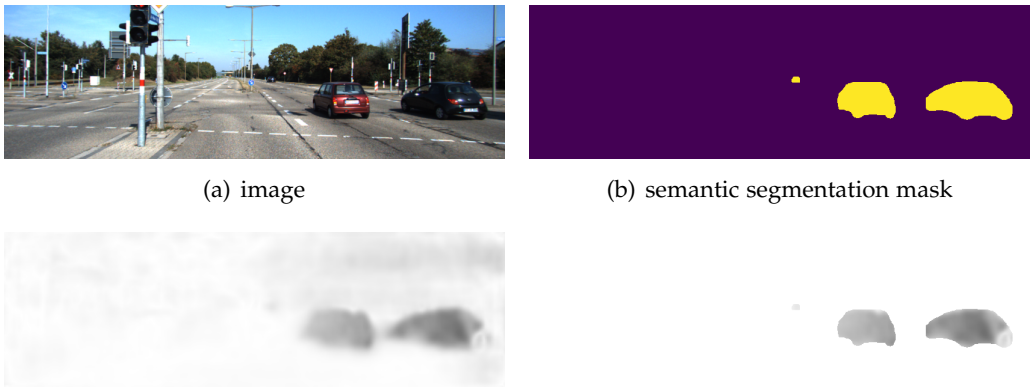
where d_p is the ground truth depth of a pixel in the depth image D, $\hat{d}_p$ is the predicted depth of the model for a pixel in the depth image $\hat{D}$. $p \in \{static, moving, all\}$ and n

is the total number of pixels in different masks. *static*, *moving*, and *all* mean pixels of non-moving objects, pixels of moving objects, and all pixels, respectively. Therefore, the depth estimates of different objects can be evaluated. We use the point cloud of the KITTI tracking dataset as the ground truth, i.e., the point cloud is reprojected onto the image to obtain the sparse ground truth depth D.

4.2.2 Results

Before showing the quantitative results of depth estimation, we show the visualization results of moving flow.

Figure 4.4, Figure 4.5 and Figure 4.6 shows the norm of the moving flow vector (The color scale of the norm is shown in Figure 4.7.), which shows that the moving flow vector without semantic segmentation mask is still noisy. Therefore, in the training and inference phases, we need to filter the moving flow with semantic segmentation mask to only use the moving flow of potential moving objects to aggregate information and generate cost volume. The main reasons for the inaccuracy of the moving flow are mainly in three aspects, the disparity and scene flow of the scene flow module, and the pose obtained by visual odometry. The accuracy of these information will affect the accuracy of the moving flow.



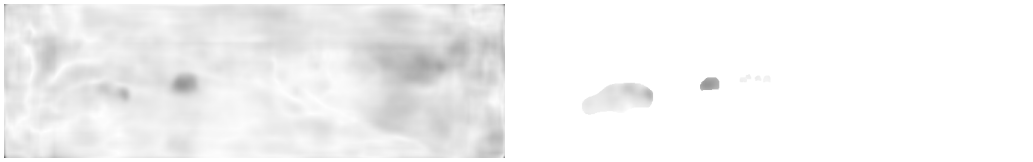
(a) image (b) semantic segmentation mask

(c) unmasked moving flow (d) masked moving flow

Figure 4.4: Moving flow of sequence 0006, frame 48.

In order to evaluate the proposed MonoRec++, we compare MonoRec++ with other depth estimation models [GMB17; God+19; Wat+21] in depth estimation task. All models use the same training dataset and validation dataset. Since MonoRec++ requires stereo images for training, all models are trained with stereo images for fairness of the

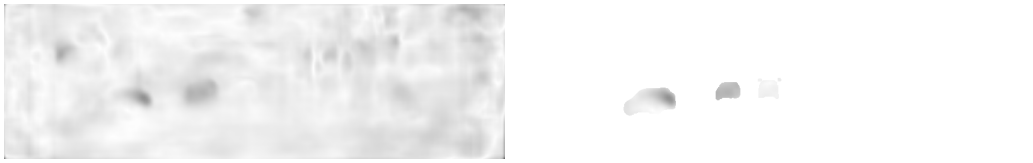
(a) image (b) semantic segmentation mask



(c) unmasked moving flow (d) masked moving flow

Figure 4.5: Moving flow of sequence 0008, frame 95.

(a) image (b) semantic segmentation mask



(c) unmasked moving flow (d) masked moving flow

Figure 4.6: Moving flow of sequence 0010, frame 82.

Figure 4.7: Color scale of moving flow norm.

comparison. In the testing phase, all the models use monocular images. The results are shown in Table 4.1, Table 4.2 and Table 4.3, The top-ranking and runner-up outcomes are highlighted in **bold** and underlined, respectively.

Table 4.1 shows the depth estimates of each model for moving objects MonoRec++ has outperformed MonoRec in terms of depth estimation of moving objects. For moving objects, moving flow provides more information about moving objects than simple zero padding to describe moving objects. However, there is still a gap in its depth estimation of moving objects compared to other depth estimation models, even compared to Manydepth which also utilizes cost volume.

Table 4.1: Depth estimation Results on KITTI tracking dataset of moving object.

Model	Training	Abs Rel	Sq Rel	RMSE	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth	MS	0.195	3.303	8.167	0.247	0.765	0.898	0.947
Monodepth2	MS	0.164	2.241	7.628	0.234	0.781	0.916	0.958
Manydepth	MS	**0.157**	**1.974**	**7.241**	**0.228**	**0.819**	**0.921**	**0.956**
MonoRec(pretrained)	MS	0.296	4.474	11.254	0.388	0.520	0.747	0.877
MonoRec	MS	0.226	3.316	9.788	0.330	0.667	0.822	0.893
MonoRec++	MS	0.170	2.241	7.739	0.264	0.772	0.895	0.949

We believe that this is due to imperfect scene flow estimation. Ideally, the accurate moving flow can perfectly construct the dynamic scene, including the direction and distance of object movement, transform the dynamic scene into a static scene after moving and construct a cost volume without noise, and the result of depth estimation of moving objects should be similar to that of static objects, as shown in Table 4.2. However, due to the noise of moving flow, even if we leverage the semantic segmentation mask to filter out potential moving objects, the representation of the 3D movement of potential moving objects by moving flow is still not accurate enough, as shown in Figure 4.5 and Figure 4.6. In theory, the moving flow of each semantic segmentation mask should be consistent for rigid objects, and its norm should be consistent. In addition, our moving flow also has a scale problem, the norm of the moving flow is too small. The color bar of moving flow is shown in Figure 4.7, the value ranges from 0 to 1. From [GLU12], we know that the camera in the KITTI dataset is set to work at 10 Hz. So the maximum value of the moving flow norm in examples is $1 * 3600/(0.1 * 1000) = 36km/hr$, which is less than the true moving flow norm in examples.

As shown in Table 4.2, for the estimation of static scenes, MonoRec++ follows MonoRec and its depth estimation outperforms all other models. From the strong performance of MonoRec++ and MonoRec for static scenes, it can also be observed that the cost volume based on temporal images has a very strong bias for static scenes. This also proves our conclusion in the depth estimation of moving objects.

Table 4.2: Depth estimation results on KITTI tracking dataset of static scene.

Model	Training	Abs Rel	Sq Rel	RMSE	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth	MS	0.157	2.139	7.812	0.263	0.782	0.904	0.960
Monodepth2	MS	0.126	1.432	6.457	0.222	0.834	0.941	0.974
Manydepth	MS	0.128	1.376	6.382	0.222	0.831	0.938	0.974
MonoRec(pretrained)	MS	0.118	1.311	5.879	0.220	0.868	0.935	0.966
MonoRec	MS	0.111	1.219	5.723	0.220	0.875	0.943	0.970
MonoRec++	MS	**0.098**	**1.103**	**5.518**	**0.204**	**0.891**	**0.953**	**0.975**

Table 4.3 reflects the overall depth estimate, without distinguishing between moving or static scenes. Comparing Table 4.1, Table 4.2 and Table 4.3, it can be seen that the metrics in Table 4.3 are closer to those in Table 4.2, indicating that even when evaluated on the KITTI tracking dataset, which contains a relatively large number of moving objects, the overall results are still closer to the static scene, with the moving objects accounting for a very small percentage of pixels in the overall scene.

Table 4.3: Depth estimation results on KITTI tracking dataset overall.

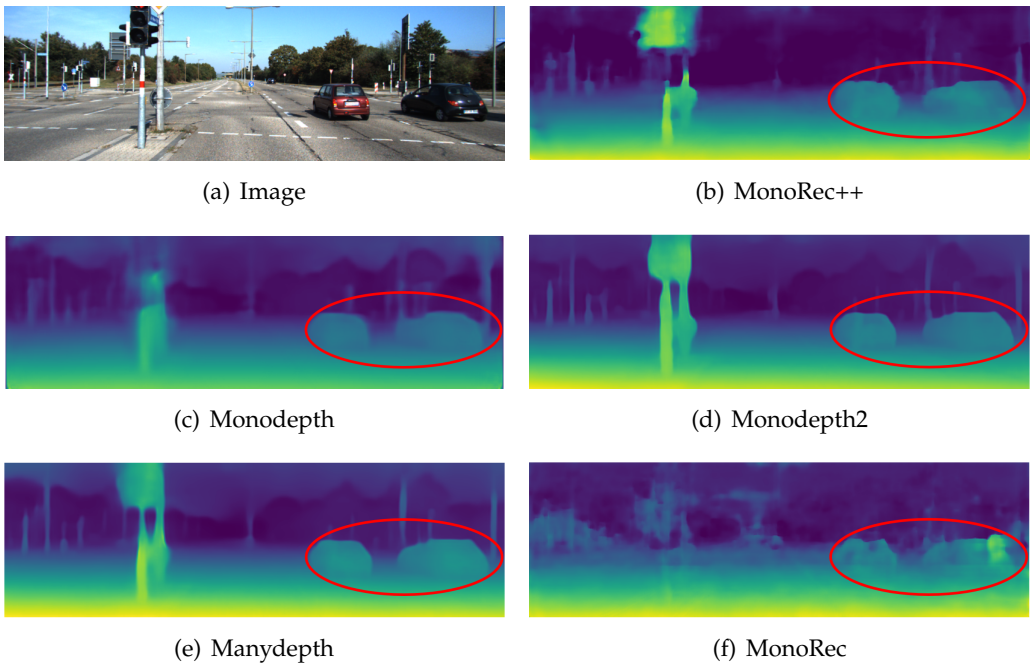
Model	Training	Abs Rel	Sq Rel	RMSE	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth	MS	0.160	2.211	7.856	0.267	0.780	0.902	0.959
Monodepth2	MS	0.129	1.458	6.499	0.227	0.831	0.939	0.973
Manydepth	MS	0.130	1.394	6.419	0.225	0.830	0.937	0.973
MonoRec(pretrained)	MS	0.124	1.396	6.016	0.230	0.859	0.929	0.962
MonoRec	MS	0.114	1.268	5.841	0.229	0.868	0.938	0.966
MonoRec++	MS	**0.101**	**1.133**	**5.601**	**0.210**	**0.887**	**0.950**	**0.973**

4.3 Scene Flow Estimation

In order to evaluate the impact of our method on the scene flow module, we need to quantitatively evaluate the scene flow module on KITTI dataset.

4.3.1 Datasets

KITTI tracking dataset does not provide the ground truth of scene flow, while the KITTI scene flow 2015 benchmark [MG15] provides the ground truth of scene flow. We will evaluate the scene flow module on this benchmark. Since the KITTI tracking dataset and KITTI scene flow 2015 have some identical samples, we removed all the KITTI scene flow 2015 samples that exist in the KITTI tracking training dataset to prevent data leakage.

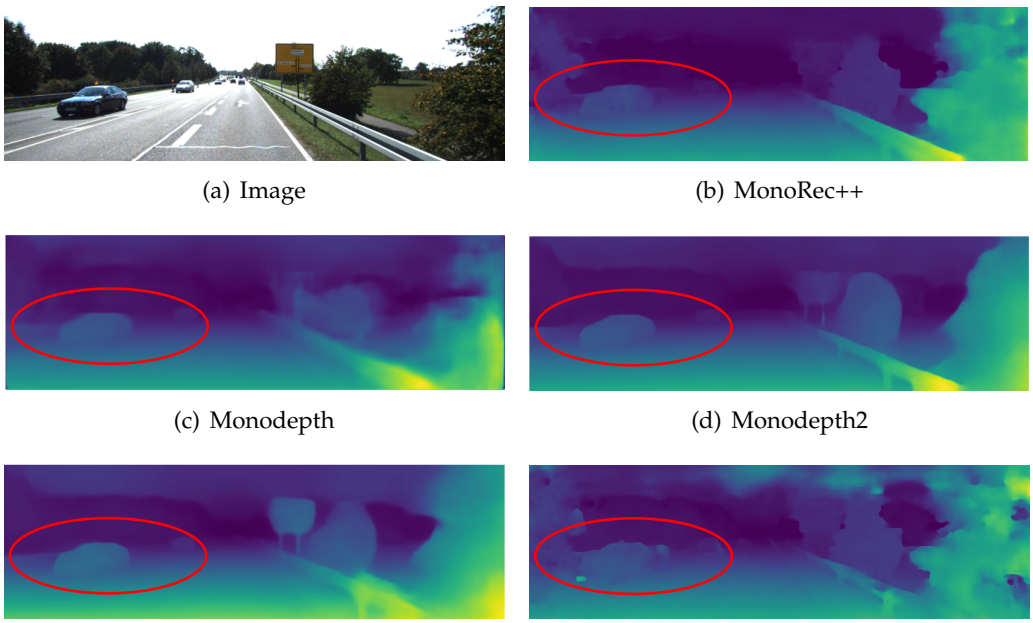


(a) Image (b) MonoRec++

(c) Monodepth (d) Monodepth2

(e) Manydepth (f) MonoRec

Figure 4.8: **Qualitative results on KITTI tracking dataset**(sequences 0006, frame 48): This is a static scene containing 2 moving cars. Compared with MonoRec, our depth estimation is more accurate on moving objects, but there is still the problem of unsmooth depth estimation for moving objects compared to other depth estimation models.

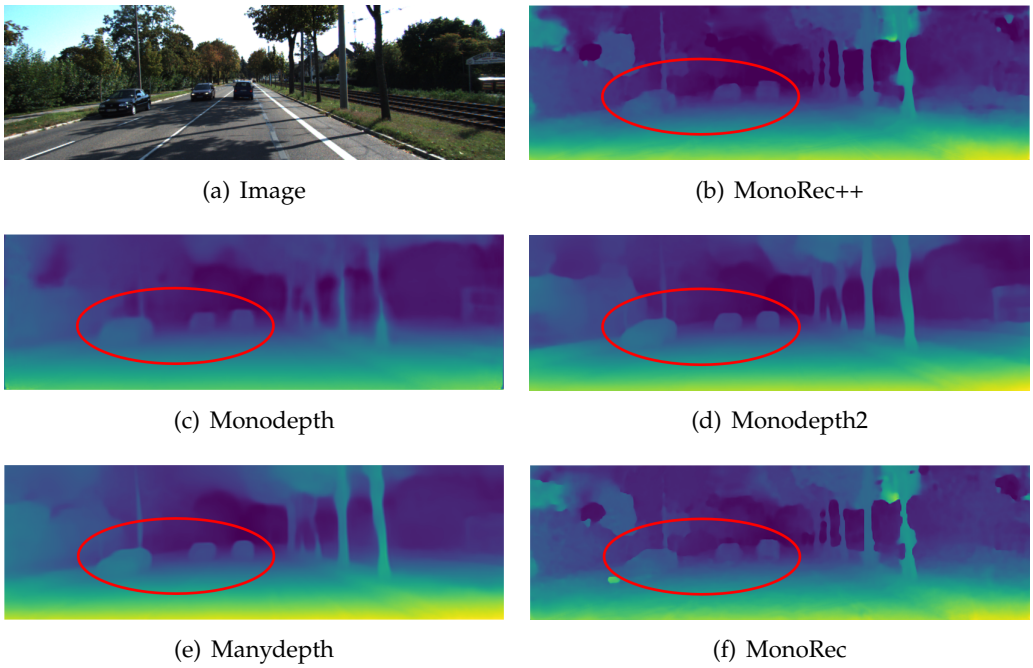


(a) Image (b) MonoRec++

(c) Monodepth (d) Monodepth2

(e) Manydepth (f) MonoRec

Figure 4.9: **Qualitative results on KITTI tracking dataset**(sequences 0008, frame 95): This is a highway scene, there are cars in opposite directions. Compared with MonoRec, our model still provides more accurate depth estimates in moving objects, but our results do not show an advantage with other depth estimation models. And there is still a gap compared to Manydepth, which is also consistent with the quantitative results.



(a) Image (b) MonoRec++

(c) Monodepth (d) Monodepth2

(e) Manydepth (f) MonoRec

Figure 4.10: **Qualitative results on KITTI tracking dataset**(sequences 0010, frame 82): This is an common driving scenario where there are cars in the same and opposite directions. In this scene, our model does not have an advantage for depth estimation of moving objects (except for MonoRec), and its results are poor compared to Manydepth.

4.3.2 Metrics

- D1-all: Percentage of stereo disparity outliers in the first frame.
- D2-all: Percentage of stereo disparity outliers in the second frame.
- EPE: Average end-point error of the optical flow.
- F1-all: Percentage of optical flow outliers.
- SF1-all: Percentage of scene flow outliers(outliers in either D1-all, D2-all or F1-all).

The outliers are defined as the pixels whose error is larger than 3 pixels or 5% of its true value[MG15].

4.3.3 Results

The results are shown in Table 4.4, The top-ranking and runner-up outcomes are highlighted in **bold** and underlined, respectively.

- **KITTI raw dataset pretrained (KITTI split)** The results of the scene flow module from the original paper and trained on the KITTI raw dataset with the KITTI split.
- **KITTI raw dataset pretrained (KITTI split) + ours** The result of the scene flow module pretrained on the KITTI raw datasetwith the KITTI split and our proposed method.
- **KITTI tracking dataset pretrained** The results of the scene flow module pretrained on the KITTI tracking dataset.
- **KITTI tracking dataset pretrained more epochs** The result of the scene flow module pretrained on the KITTI tracking dataset for 130 epochs.
- **KITTI tracking dataset pretrained + ours** The result of the scene flow module pretrained on the KITTI tracking dataset with our proposed method.

From Table 4.4, we can see that even though the scene flow module is trained with more epochs, its metrics do not improve significantly. In contrast, the scene flow module trained by our proposed method has improved on top of the original pretraining. Our method also improves the performance of the scene flow module pretrained on the raw dataset.

Table 4.4: Scene flow results on the KITTI scene flow dataset.

Method	D1-all	D2-all	F1-all	EPE	SF1-all
KITTI raw dataset pretrained (KITTI split)	33.97	38.35	**26.88**	**9.32**	51.71
KITTI raw dataset pretrained (KITTI split) + ours	**29.00**	**31.99**	28.61	10.19	**46.23**
KITTI tracking dataset pretrained	39.85	56.64	58.19	29.53	76.05
KITTI tracking dataset pretrained with more epochs	43.70	55.22	55.29	24.50	76.05
KITTI tracking dataset pretrained + ours	41.08	45.97	59.60	16.85	71.75

4.4 Ablation Study

In order to confirm the contribution of our model and the influence of each variable on the model, we performed an ablation study and the results are shown in Table 4.5, Table 4.6 and Table 4.7.

Table 4.5: Ablation study of depth estimation results on KITTI tracking dataset of moving object.

Model	Abs Rel	Sq Rel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
MonoRec++	0.170	2.241	7.739	0.264	0.772	0.895	0.949
GT moving mask	0.171	2.292	7.750	0.264	0.774	0.894	0.948
ORB-SLAM3 stereo mode pose	0.170	2.252	7.746	0.263	0.777	0.896	0.950
without depth module pretraining	0.246	4.342	9.725	0.328	0.659	0.821	0.911
without depth module	0.238	3.786	9.036	0.277	0.694	0.892	0.946
without scene flow module	0.231	3.409	9.063	0.335	0.652	0.814	0.904
MonoRec	0.226	3.316	9.788	0.330	0.667	0.822	0.893
MonoRec with depth module pretraining	0.196	2.931	8.814	0.292	0.726	0.856	0.917
pretrained depth module(cost volume = 0)	0.184	2.426	8.031	0.257	0.759	0.904	0.953
pretrained depth module(cost volume = 1)	0.190	2.595	8.695	0.276	0.754	0.892	0.942
pretrained depth module(cost volume = -1)	0.187	2.410	8.189	0.258	0.751	0.903	0.953
pretrained depth module(cost volume = uniform distribution)	0.194	2.673	8.498	0.264	0.736	0.900	0.953

Pose Since both MonoRec and MonoRec++ are MVS-based depth estimation models, the camera poses have a very important impact on the model as the key to constructing the cost volume. For this reason, we investigate the effect of different poses on the model. For the KITTI tracking dataset, we compared the poses obtained from the stereo mode of ORB-SLAM3 [Cam+21], and since the stereo mode uses stereo images, its estimation of the poses is more accurate compared to DF-VO. And indeed more accurate pose estimation leads to more accurate depth estimation, especially in the static scene part. We believe that more accurate pose estimation can produce a greater improvement in the accuracy of depth estimation for the static scene assumption of the basic cost volume, compared to moving objects.

Mask For the moving mask, we also compared the results of filtering moving flow using the ground truth moving mask. The results of filtering moving flow using the

Table 4.6: Ablation study of depth estimation results on KITTI tracking dataset of static scene.

Model	Abs Rel	Sq Rel	RMSE	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
MonoRec++	0.098	1.103	5.518	0.204	0.891	0.953	0.975
GT moving mask	0.097	1.100	5.522	0.204	0.892	0.953	0.975
ORB-SLAM3 stereo mode pose	0.095	0.964	5.068	0.199	0.901	0.956	0.976
without depth module pretraining	0.105	1.054	5.416	0.211	0.877	0.944	0.972
without depth module	0.166	2.215	7.224	0.250	0.797	0.924	0.965
without scene flow module	0.100	1.129	5.667	0.210	0.889	0.949	0.973
MonoRec	0.111	1.219	5.723	0.220	0.875	0.943	0.970
MonoRec with depth module pretraining	0.101	1.147	5.487	0.205	0.890	0.954	0.975
pretrained depth module(cost volume = 0)	0.169	2.134	7.717	0.279	0.765	0.895	0.952
pretrained depth module(cost volume = 1)	0.175	2.415	7.754	0.272	0.774	0.901	0.946
pretrained depth module(cost volume = -1)	0.168	2.097	7.862	0.287	0.761	0.890	0.949
pretrained depth module(cost volume = uniform distribution)	0.176	2.362	8.054	0.291	0.756	0.886	0.947

Table 4.7: Ablation study of depth estimation results on KITTI tracking dataset overall.

Model	Abs Rel	Sq Rel	RMSE	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
MonoRec++	0.101	1.133	5.601	0.210	0.887	0.950	0.973
GT moving mask	0.101	1.130	5.610	0.210	0.887	0.950	0.973
ORB-SLAM3 stereo mode pose	0.098	1.021	5.190	0.208	0.895	0.952	0.973
without depth module pretraining	0.110	1.138	5.553	0.220	0.871	0.939	0.969
without depth module	0.172	2.331	7.330	0.257	0.792	0.920	0.962
without scene flow module	0.105	1.205	5.818	0.222	0.880	0.942	0.968
MonoRec	0.114	1.268	5.841	0.229	0.868	0.938	0.966
MonoRec with depth module pretraining	0.105	1.200	5.601	0.214	0.884	0.949	0.972
pretrained depth module(cost volume = 0)	0.171	2.153	7.737	0.282	0.763	0.893	0.950
pretrained depth module(cost volume = 1)	0.177	2.445	7.762	0.275	0.773	0.900	0.945
pretrained depth module(cost volume = -1)	0.170	2.121	7.881	0.290	0.759	0.889	0.948
pretrained depth module(cost volume = uniform distribution)	0.178	2.382	8.080	0.294	0.754	0.884	0.946

ground truth moving mask are similar to the results of using the semantic segmentation mask for depth estimation.

Without depth module pretraining If there is no pretraining of the depth module, the depth estimation results of the model decrease. This proves that the pretraining of the depth module has a great improvement in the effect of feature extraction.

Without depth module If there is no depth module, we use the disparity of the scene flow module as the output of the pipeline to generate the depth map. As expected, the depth estimation results based only on the disparity of the scene flow module are not accurate. We still need the depth module to do a more accurate depth estimation.

Without scene flow module If there is no scene flow module, the moving flow does not participate in the calculation of the cost volume. Its depth estimation results have a clear decrease in moving objects, which proves that the scene flow module has a great improvement in the depth estimation of moving objects.

MonoRec with depth module pretraining Since the depth module pretraining is also applicable to MonoRec, we also compared the effect of pretraining the depth module in MonoRec. We found that the pretrained depth module also improved the depth estimation of MonoRec in all the metrics of moving objects and static scenes. The pretraining of depth estimation based on images can help the model extract better features of images and cost volumes.

Different depth module pretraining setups We compared the effect of different depth module pretraining setups. That is, set the cost volume to 0, 1, -1, and uniform distribution. We found that as long as the cost volume is set to a fixed value, the effect of the pre-trained depth modules is similar. Even if the cost volume is set to a uniform distribution, the effect of the pre-trained depth module is only slightly reduced. This shows that in the pretraining stage, the depth module has a certain robustness to meaningless cost volumes, and only learns meaningful image features.

4.5 3D Detection

In order to use a point cloud-based 3D object detection algorithm, similar to [You+19], we preprocess the obtained depth map for sparsification and obtain the pseudo point cloud simulating a 64-beams LIDAR. The pseudo point cloud is then used as the input to a downstream 3D object detection model based on point cloud estimation. We use [SWL19] as 3D object detection model.

4.5.1 Metrics

In the KITTI 3D object detection task and BEV detection task, the difficulty of object detection is classified into 3 classes, namely easy, moderate and hard:

Easy Min. bounding box height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15 %

Moderate Min. bounding box height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30 %

Hard Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50 %

We used Average Precision (AP) as a metric to evaluate the 3D object detection task and the BEV detection task. AP_{40} is used according to the suggestion of [Sim+19]. AP_{40} sampled 40 recall locations and is able to avoid bias, compared to AP_{11}, which originally samples only 11 points. And the threshold of Intersection over Union (IoU) is set to 0.5 and 0.7. Here we only show the results of the *Car* class in 3D object detection.

4.5.2 Results

As with the depth estimation, we compare the 3D object detection results of the pseudo point cloud generated by the depth estimation models with the results of the real velodyne point cloud, as shown in Table 4.8.

Table 4.8: 3D object detection results on the KITTI tracking dataset for class *Car*(BEV detection / 3D detection).

		IoU = 0.5			IoU = 0.7		
Detection algorithm	Input	Easy	Moderate	Hard	Easy	Moderate	Hard
Monodepth2	Mono	46.64/38.78	29.26/23.66	26.96/21.62	15.50/6.09	10.26/4.41	9.28/4.00
MonoRec++	Mono	41.75/34.74	25.48/20.51	22.49/18.70	12.62/5.69	8.70/3.95	7.99/3.74
LIDAR GT	LIDAR	99.83/99.82	92.74/92.45	92.38/92.05	92.42/75.22	84.25/57.90	82.13/57.53

In the 3D detection task, the 3D detection results of the pseudo LIDAR generated by all the depth estimation models we evaluated are worse than those of other state-of-the-art papers [You+19; Wan+19; Ma+20; Sun+20; Che+20] based on pseudo-LIDAR. Although these state-of-the-art papers use the KITTI 3D detection dataset, we use the KITTI tracking dataset, the difference between the datasets should not be large. But the results have a significant gap in terms of the numerical values of the metrics. We believe there are several reasons for this:

Dynamic scenes As the *Car* in the tested sequence is dominated by moving *Car*, we compare all the depth estimation models using monocular images, the depth estimation of moving objects is more difficult than static objects.

No data leakage Although pseudo point cloud based object detection performs brightly on the validation dataset proposed by [Che+15], it performs poorly on the benchmark of the 3D object detection test dataset of the KITTI tracking dataset. [Sim+21] claims that to generate the pseudo point cloud, [You+19; Wan+19; Ma+20] used Eigen split for the training of the depth estimation model. However, **1226**/**3769**(**32.5**%) images in the object detection validation dataset are included in the training dataset of Eigen split.

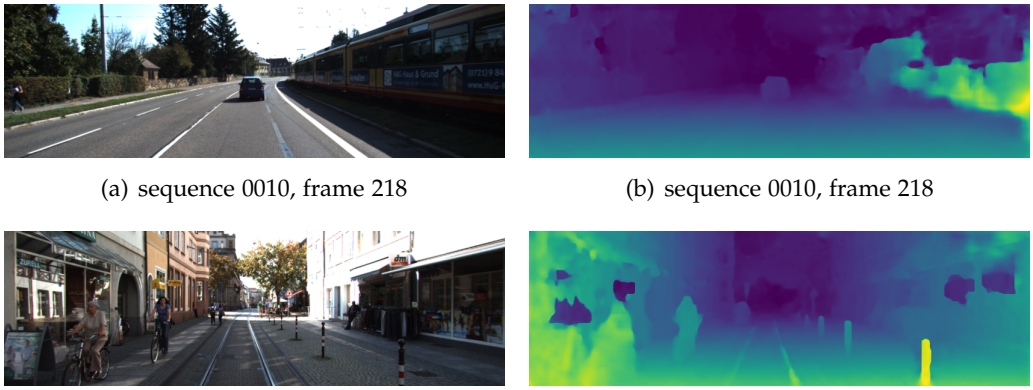
No optimization for 3D detection tasks Although the depth estimation of our MonoRec++ is improved compared to MonoRec. But compared with other pseudo point cloud based detection models such as [You+19; Ma+20], MonoRec++ is not optimized for the object detection task.

No stereo images at test phase Other detection models such as [Sun+20; Che+20] use stereo images in both training and testing phases in order to obtain more accurate depth estimates and pseudo point clouds.

And since the *Car* in the test dataset is dominated by moving *Car*, this explains why MonoRec++ performs worse than other monocular depth estimation models in the 3D target detection task, because of its poor performance in depth estimation of moving objects.

4.6 Limitation

1. **Slow training and inference** Because the cost volume depends on the output of the scene flow module, the two cannot be computed in parallel, so the speed is also slower than MonoRec in the training and inference stages.
2. **Non-Lambertian planes or poor texture scenes** For non-Lambertian planes or poor texture scenes, our proposed method cannot estimate the depth accurately, which is still a difficulty in depth estimation based on photometric errors for all related methods.
3. **Coarse moving objects** Although our method can partially improve the accuracy of the scene flow module, we still use a disparity-based self-supervised method to train the scene flow module, and we still cannot accurately model moving objects. as shown in Figure 4.4, Figure 4.5 and Figure 4.6



(a) sequence 0010, frame 218

(b) sequence 0010, frame 218

(c) sequence 0013, frame 307

(d) sequence 0013, frame 307

Figure 4.11: Failure case of non-Lambertian planes or poor texture scenes.

5 Conclusion and Outlook

5.1 Conclusion

Based on MonoRec, we propose a deep learning model that estimates the accurate scene depth using only consecutive images and the corresponding poses. In order to estimate the depth of moving objects more accurately, we propose to supplement the cost volume with scene flow information, which breaks the limitation of estimating depth based on MVS for static scenes. First, we use the scene flow module and the poses to estimate the moving flow. Then, we use semantic segmentation masks to remove the inaccurate moving flow. The cost volume based on the moving flow can obtain more information about moving objects. Moreover, we also propose a new joint learning process and loss function. The modules can be combined more reasonably.

On the KITTI tracking dataset, MonoRec++ is able to make significant progress in moving object depth estimation based on MonoRec. However, compared to other monocular depth estimation models, MonoRec++ still has a gap in the depth estimation of moving objects.

5.2 Future Work

Although MonoRec++ has been successful in depth estimation of moving objects compared to MonoRec. But we still haven't solved the non-Lambertian planes or poor texture scenes problem of self-supervised learning for depth estimation. Another point is the inference time, the cost volume needs to wait for the inference process of the scene flow module. It is worth exploring to improve the inference speed by solving the scene flow estimation and depth estimation based on MVS problems with the same model. Inspired by Manydepth, using the feature maps generated by the deep learning model and the feature scene flow to build the cost volume is also a possible direction. Finally, there is a gap between the 3D detection results of pseudo point clouds based on monocular depth estimation and point clouds. But how to better use the pseudo point cloud as an expression form needs more thinking from related researchers.

Abbreviations

VO Visual Odometry

BEV Bird-eye View

CNN Convolutional Neural Network

RNN Recurrent Neural Network

MVS Multi-view Stereo

SSIM Structural Similarity Index Measure

ToF Time of Flight

RMSE Root Mean Squared Error

Abs Rel Absolute Relative Difference

Sq Rel Squared Relative difference

AP Average Precision

IoU Intersection over Union

GRU Gate Recurrent Unit

CRF Conditional Random Field

MLP Multilayer Perceptron

CV Computer Vision

DL Deep Learning

List of Figures

List of Tables

Bibliography

[Beh+19] A. Behl, D. Paschalidou, S. Donné, and A. Geiger. "Pointflownet: Learning representations for rigid motion estimation from point clouds." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7962–7971.

[Cam+08] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. "Using multiple hypotheses to improve depth-maps for multi-view stereo." In: *European Conference on Computer Vision*. Springer. 2008, pp. 766–779.

[Cam+21] C. Campos, R. Elvira, J. J. G. Rodrıéguez, J. M. Montiel, and J. D. Tardós. "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam." In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[Cas+19] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 8001–8008.

[Che+15] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. "3d object proposals for accurate object class detection." In: *Advances in neural information processing systems* 28 (2015).

[Che+17] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. "Multi-view 3d object detection network for autonomous driving." In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 1907–1915.

[Che+20] Y. Chen, S. Liu, X. Shen, and J. Jia. "Dsgn: Deep stereo geometry network for 3d object detection." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12536–12545.

[Dai+19] Y. Dai, Z. Zhu, Z. Rao, and B. Li. "Mvs2: Deep unsupervised multi-view stereo with multi-view symmetry." In: *2019 International Conference on 3D Vision (3DV)*. Ieee. 2019, pp. 1–8.

[EF15] D. Eigen and R. Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2650–2658.

[EPF14] D. Eigen, C. Puhrsch, and R. Fergus. "Depth map prediction from a single image using a multi-scale deep network." In: *Advances in neural information processing systems* 27 (2014).

[FH+15] Y. Furukawa, C. Hernández, et al. "Multi-view stereo: A tutorial." In: *Foundations and Trends® in Computer Graphics and Vision* 9.1-2 (2015), pp. 1–148.

[Fu+18] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. "Deep ordinal regression network for monocular depth estimation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2002–2011.

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[GMB17] C. Godard, O. Mac Aodha, and G. J. Brostow. "Unsupervised monocular depth estimation with left-right consistency." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 270–279.

[God+19] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. "Digging into self-supervised monocular depth estimation." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3828–3838.

[Goj+21] Z. Gojcic, O. Litany, A. Wieser, L. J. Guibas, and T. Birdal. "Weakly supervised learning of rigid 3D scene flow." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 5692–5703.

[Gu+20] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan. "Cascade cost volume for high-resolution multi-view stereo and stereo matching." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2495–2504.

[Gui+20] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon. "3d packing for self-supervised monocular depth estimation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2485–2494.

[HD07] F. Huguet and F. Devernay. "A variational method for scene flow estimation from stereo sequences." In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–7.

[HR20] J. Hur and S. Roth. "Self-supervised monocular scene flow estimation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7396–7405.

[Hua+18] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang. "Deepmvs: Learning multi-view stereopsis." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2821–2830.

[Ilg+18] E. Ilg, T. Saikia, M. Keuper, and T. Brox. "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 614–630.

[Iza+11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera." In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pp. 559–568.

[Jia+19] H. Jiang, D. Sun, V. Jampani, Z. Lv, E. Learned-Miller, and J. Kautz. "Sense: A shared encoder network for scene-flow estimation." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3195–3204.

[Kli+20] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt. "Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance." In: *European Conference on Computer Vision*. Springer. 2020, pp. 582–600.

[KS00] K. N. Kutulakos and S. M. Seitz. "A theory of shape by space carving." In: *International journal of computer vision* 38.3 (2000), pp. 199–218.

[Kuh55] H. W. Kuhn. "The Hungarian method for the assignment problem." In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[Lan+19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. "Pointpillars: Fast encoders for object detection from point clouds." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12697–12705.

[Lee+19] S. Lee, S. Im, S. Lin, and I. S. Kweon. "Learning residual flow as dynamic motion from stereo videos." In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 1180–1186.

[Lee+21] S. Lee, S. Im, S. Lin, and I. S. Kweon. "Learning monocular depth in dynamic scenes via instance-aware projection consistency." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 3. 2021, pp. 1863–1872.

[Li+18] Y. Li, C. Xie, H. Lu, X. Chen, J. Xiao, and H. Zhang. "Scale-aware monocular SLAM based on convolutional neural network." In: *2018 IEEE International Conference on Information and Automation (ICIA)*. IEEE. 2018, pp. 51–56.

[Li+19] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman. "Learning the depths of moving people by watching frozen people." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4521–4530.

[Li+21] H. Li, A. Gordon, H. Zhao, V. Casser, and A. Angelova. "Unsupervised monocular depth learning in dynamic scenes." In: *Conference on Robot Learning*. PMLR. 2021, pp. 1908–1917.

[Liu+19] L. Liu, G. Zhai, W. Ye, and Y. Liu. "Unsupervised Learning of Scene Flow Estimation Fusing with Local Rigidity." In: *IJCAI*. 2019, pp. 876–882.

[LQ05] M. Lhuillier and L. Quan. "A quasi-dense approach to surface reconstruction from uncalibrated images." In: *IEEE transactions on pattern analysis and machine intelligence* 27.3 (2005), pp. 418–433.

[LQG19] X. Liu, C. R. Qi, and L. J. Guibas. "Flownet3d: Learning scene flow in 3d point clouds." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 529–537.

[Luo+19] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille. "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding." In: *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), pp. 2624–2641.

[Lv+18] Z. Lv, K. Kim, A. Troccoli, D. Sun, J. M. Rehg, and J. Kautz. "Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 468–484.

[Ma+19a] W.-C. Ma, S. Wang, R. Hu, Y. Xiong, and R. Urtasun. "Deep rigid instance scene flow." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3614–3622.

[Ma+19b] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan. "Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6851–6860.

[Ma+20] X. Ma, S. Liu, Z. Xia, H. Zhang, X. Zeng, and W. Ouyang. "Rethinking pseudo-lidar representation." In: *European Conference on Computer Vision*. Springer. 2020, pp. 311–327.

[MG15] M. Menze and A. Geiger. "Object scene flow for autonomous vehicles." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3061–3070.

[Par+21] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon. "Is pseudo-lidar needed for monocular 3d object detection?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3142–3152.

[PTM18] M. Poggi, F. Tosi, and S. Mattoccia. "Learning monocular depth estimation with unsupervised trinocular assumptions." In: *2018 International conference on 3d vision (3DV)*. IEEE. 2018, pp. 324–333.

[Qi+17a] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[Qi+17b] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." In: *Advances in neural information processing systems* 30 (2017).

[Qi+18] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. "Frustum pointnets for 3d object detection from rgb-d data." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 918–927.

[Ran+16] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun. "Dense monocular depth estimation in complex dynamic scenes." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4058–4066.

[Ran+19] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black. "Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12240–12249.

[RYA14] C. Russell, R. Yu, and L. Agapito. "Video pop-up: Monocular 3d reconstruction of dynamic scenes." In: *European conference on computer vision*. Springer. 2014, pp. 583–598.

[Sch+18] R. Schuster, C. Bailer, O. Wasenmüller, and D. Stricker. "Combining stereo disparity and optical flow for basic scene flow." In: *Commercial Vehicle Technology 2018: Proceedings of the 5th Commercial Vehicle Technology Symposium-CVT 2018*. Springer. 2018, pp. 90–101.

[SD99] S. M. Seitz and C. R. Dyer. "Photorealistic scene reconstruction by voxel coloring." In: *International Journal of Computer Vision* 35.2 (1999), pp. 151–173.

[Sim+19] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder. "Disentangling monocular 3d object detection." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1991–1999.

[Sim+21] A. Simonelli, S. R. Bulo, L. Porzi, P. Kontschieder, and E. Ricci. "Are we missing confidence in pseudo-lidar methods for monocular 3d object detection?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3225–3233.

[Sun+20] J. Sun, L. Chen, Y. Xie, S. Zhang, Q. Jiang, X. Zhou, and H. Bao. "Disp r-cnn: Stereo 3d object detection via shape prior guided instance disparity estimation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10548–10557.

[SWL19] S. Shi, X. Wang, and H. Li. "Pointrcnn: 3d object proposal generation and detection from point cloud." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 770–779.

[TD20] Z. Teed and J. Deng. "Raft: Recurrent all-pairs field transforms for optical flow." In: *European conference on computer vision*. Springer. 2020, pp. 402–419.

[TD21] Z. Teed and J. Deng. "RAFT-3D: Scene flow using rigid-motion embeddings." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8375–8384.

[VAL19] J. M. U. Vianney, S. Aich, and B. Liu. "Refinedmpl: Refined monocular pseudolidar for 3d object detection in autonomous driving." In: *arXiv preprint arXiv:1911.09712* (2019).

[Ved+99] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. "Three-dimensional scene flow." In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. IEEE. 1999, pp. 722–729.

[Voi+19] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe. "Mots: Multi-object tracking and segmentation." In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. 2019, pp. 7942–7951.

[VRS14] C. Vogel, S. Roth, and K. Schindler. "View-consistent 3D scene flow estimation over multiple frames." In: *European Conference on Computer Vision*. Springer. 2014, pp. 263–278.

[VSR13] C. Vogel, K. Schindler, and S. Roth. "Piecewise rigid scene flow." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1377–1384.

[VSR15] C. Vogel, K. Schindler, and S. Roth. "3d scene flow estimation with a piecewise rigid scene model." In: *International Journal of Computer Vision* 115.1 (2015), pp. 1–28.

[Wan+19] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8445–8453.

[Wan+20a] X. Wang, W. Yin, T. Kong, Y. Jiang, L. Li, and C. Shen. "Task-aware monocular depth estimation for 3d object detection." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 12257–12264.

[Wan+20b] Z. Wang, S. Li, H. Howard-Jenkins, V. Prisacariu, and M. Chen. "Flownet3d++: Geometric losses for deep scene flow estimation." In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2020, pp. 91–98.

[Wat+21] J. Watson, O. Mac Aodha, V. Prisacariu, G. Brostow, and M. Firman. "The temporal opportunist: Self-supervised multi-frame monocular depth." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1164–1174.

[Wei+21] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou. "Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 6954–6963.

[Wim+21] F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers. "MonoRec: Semi-Supervised Dense Reconstruction in Dynamic Environments from a Single Moving Camera." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[WK19] X. Weng and K. Kitani. "Monocular 3d object detection with pseudo-lidar point cloud." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[Wu+19a] W. Wu, Z. Wang, Z. Li, W. Liu, and L. Fuxin. "Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds." In: *arXiv preprint arXiv:1911.12408* (2019).

[Wu+19b] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[Xue+19] Y. Xue, J. Chen, W. Wan, Y. Huang, C. Yu, T. Li, and J. Bao. "Mvscrf: Learning multi-view stereo with conditional random fields." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4312–4321.

[Yan+18] N. Yang, R. Wang, J. Stuckler, and D. Cremers. "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 817–833.

[Yan+20] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers. "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1281–1292.

[Yao+18] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. "Mvsnet: Depth inference for unstructured multi-view stereo." In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 767–783.

[Yao+19] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan. "Recurrent mvsnet for high-resolution multi-view stereo depth inference." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5525–5534.

[YML18] Y. Yan, Y. Mao, and B. Li. "Second: Sparsely embedded convolutional detection." In: *Sensors* 18.10 (2018), p. 3337.

[You+19] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving." In: *arXiv preprint arXiv:1906.06310* (2019).

[YR21] G. Yang and D. Ramanan. "Learning to segment rigid motions from two frames." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1266–1275.

[Zha+21a] H. Zhan, C. S. Weerasekera, J.-W. Bian, R. Garg, and I. Reid. "DF-VO: What Should Be Learnt for Visual Odometry?" In: *arXiv preprint arXiv:2103.00933* (2021).

[Zha+21b] Z. Zhang, F. Cole, R. Tucker, W. T. Freeman, and T. Dekel. "Consistent depth of moving objects in video." In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–12.

[Zho+17] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. "Unsupervised learning of depth and ego-motion from video." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1851–1858.

[ZT18] Y. Zhou and O. Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499.

[ZUB18] H. Zhou, B. Ummenhofer, and T. Brox. "Deeptam: Deep tracking and mapping." In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 822–838.