A THOROUGH ANALYSIS OF UNSUPERVISED DEPTH AND EGO-MOTION ESTIMATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALP EREN SARI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2020

Approval of the thesis:

A THOROUGH ANALYSIS OF UNSUPERVISED DEPTH AND EGO-MOTION ESTIMATION

submitted by **ALP EREN SARI** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ______________

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** ______________

Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Engineering, METU** ______________

Assoc. Prof. Dr. Sinan Kalkan
Co-supervisor, **Computer Engineering, METU** ______________

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU ______________

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering, METU ______________

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Engineering, METU ______________

Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering, METU ______________

Assist. Prof. Dr. Osman Serdar Gedik
Computer Engineering, Ankara Yıldırım Beyazıt University ______________

Date: 17.08.2020

ABSTRACT

A THOROUGH ANALYSIS OF UNSUPERVISED DEPTH AND EGO-MOTION ESTIMATION

SARI, Alp Eren
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. A. Aydın Alatan
Co-Supervisor: Assoc. Prof. Dr. Sinan Kalkan

August 2020, 77 pages

Recent years have shown unprecedented success in depth estimation by jointly solving unsupervised depth estimation and pose estimation. In this study, we perform a thorough analysis for such an approach. Initially, pose estimation performances of classical techniques, such as COLMAP [1], are compared against recent unsupervised learning-based techniques. Simulation results indicate the superiority of Bundle Adjustment step in classical techniques. Next, the effect of the number of input frames to the pose estimator network is investigated in detail. The experiments performed at this step revealed that the state-of-the-art can be improved by providing extra frames to the pose estimator network. Finally, the semantic labels of objects in the scene are utilized individually during pose and depth estimation stages. For this purpose, pre-trained semantic segmentation networks are utilized. The effect of computing losses from different regions of the scene and averaging different pose estimations with learnable weights are investigated. The poses and losses corresponding to different semantic classes are summed with learnable weights yielding comparable results against state-of-the-art methods.

Keywords: unsupervised learning, depth estimation, pose estimation

ÖZ

GÜDÜMSÜZ DERİNLİK VE HAREKET KESTİRMİ ÜZERİNE DETAYLI BİR ANALİZ

SARI, Alp Eren
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan
Ortak Tez Yöneticisi: Doç. Dr. Sinan Kalkan

Ağustos 2020 , 77 sayfa

Derinlik kestirimi konusunda güdümsüz derinlik ve hareket kestirimi yöntemlerinin eş zamanlı eğitimi ile geçmiş yıllarda eşsiz bir başarı sağlanmıştır. Bu çalışmada ise böyle bir yaklaşımın detaylı bir analizi yapılmıştır. Öncelikle, COLMAP [1] gibi klasik yöntemler ile yeni güdümsüz öğrenme tabanlı yaklaşımların hareket kestirimi performansları karşılaştırılmıştır. Simülasyon sonuçları Demet Düzeltimi tabanlı yöntemlerin üstünlüğüne işaret etmektedir. Sonra, hareket kestirimi yapay sinir ağına girdi olarak verilen kare sayısının etkileri detaylıca incelenmiştir. Son teknoloji yaklaşımların fazladan kare sağlanarak iyileştirilebileceği bu aşamadaki deneyler ile gösterilmiştir. Son olarak, bir sahnedeki farklı semantik nesnelerden hareket ve derinlik kestirmi sırasında ayrı ayrı yararlanılmıştır. Bu amaçla ise önceden eğitilmiş bölütleme algoritmaları kullanılmıştır. Bir sahnenin farklı semantik sınıflarına ait farklı hareket kestirimlerinin öğrenilebilen katsayılar ile doğrusal kombinasyonunu almanın etkileri araştırılmıştır. Farklı semantik sınıflara ait olan hareket ve maliyetlerin öğrenilebilen katsayılar ile doğrusal kombinasyonunun alınması ile son teknoloji ile karşılaştırılabilir sonuçlar elde edilmiştir.

Anahtar Kelimeler: denetimsiz öğrenme, derinlik kestirimi, poz kestirimi

To my family and friends

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor and co-advisor Prof. Dr. Aydın Alatan and Assoc. Prof. Dr. Sinan Kalkan for all their effort and help to develop my knowledge and abilities in computer vision and research. This thesis and my works in the SISER project would not exist without their guidance and experience. I would like to express my gratitude to Prof. Dr. Aydın Alatan to give me the chance to be a part of OGAM and believe in me. I remember the first day I met Prof. Dr. Aydın Alatan when he first introduced me to computer vision and took me to OGAM to take a tour, although I was an undergraduate student at that time. I have learned a lot since that day.

I admit that I'm fortunate because I had the opportunity to work with a group of lovely people at OGAM. Above all, I want to express my gratitude to Esat Kalfaoğlu, my project partner and partner in crime in Ankara and UK, and a true friend. I'm fortunate to have an excellent colleague and a remarkable human-being like him. I want to thank Ece Selin Böncü for elevating my culture and her great friendship, Aybüke Erol for teaching me a tremendous amount of stuff about "Aşk-ı Memnu" and "Titanic", Oğul Can for letting me watch beIN Sports for free, İzlen Geneci for introducing me breakdance, Ayberk Aydın for all child stuff. I would like to thank Dr. Alper Koz for his guidance. I also would like to acknowledge Ufuk Efe, Ahmet Arslan, Mustafa Ergül, Mustafa Kütük, İhsan Emre Üstün, and Gamze Sever as great colleagues.

I would like to express my gratitude to Assoc. Prof. Dr. Ipek Gursel Dino for giving me a chance to work on a project that contributes to my thesis works. I also would like to all the other people who I worked within the SISER project, namely, Orçun Koral İşeri and Şahin Akın.

As Winnie the Pooh says, a day without a friend is like a pot without a single drop of honey left inside. Therefore, I am grateful to all of my friends, making me find honey drops in my pot. I would like to express my gratitude to Botan Yıldırım, a great friend, an excellent colleague who I seek his help whenever I need, and an awesome cat father of two cats Ekin and Suphi. I will never forget the time we have spent in Çatı and your home. I also would like to thank Ufuk Soylu for being a best friend and partner-in-crime in Germany, İsmail Hakkı Koçdemir for being an excellent friend, and Alperen Tüğen for teaching me so much about quantum physics.

I would never forget to express my gratitude to my lovely "Friends" friends Bahar Taşkesen, Başar Kütükçü, F. Volkan Mutlu, and Safa Özer for being the best senior design group members and being best friends even after "FREAK" failed in demo. I know that "Wheels & Horizons" will exist eternally even though the senior design project ended a long time ago. I am looking forward to seeing Bahar in Switzerland.

I can never express my gratitude to my mother, Mediha Sarı, enough to be the best mother and look after me. She spent all her life on the well-being of my brother and me. I also would like to thank my brother Buğra A. Sarı for being a great brother.

This work is supported by an Institutional Links grant under the Newton-Katip Celebi partnership, Grant No. 217M519, by the Scientific and Technological Research Council of Turkey (TUBITAK) and ID [352335596] by British Council, UK.

The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources)

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
BA	Bundle Adjustment
AR	Augmented Reality
VR	Virtual Reality
ATE	Absolute Trajectory Error
RPE	Relative Pose Error

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

One of the fundamental capabilities of humans is the 3D perception of the environment in which they reside. This situation has been vital for our ancestors to hunt, commute, and eliminate the potential dangers in the wild. Computer vision research on 3D perception has been ongoing for decades for this reason. Classical methods such as Structure from Motion (SfM) method Bundler [11], OpenMVG [12], and COLMAP [1], visual odometry methods Direct Sparse Odometry [13], D3VO [14], simultaneous localization and mapping methods (SLAM) ORB-SLAM2 [15], LSD-SLAM [16] are all outstanding attempts to solve 3D perception problem by approaching the problem from different aspects.

After deep neural networks (DNN) have become popular and best results in many different computer vision areas have been achieved with DNNs, computer vision researchers have begun an expedition to reveal what DNNs can offer to 3D computer vision problems. A considerable portion of this research focuses on monocular dense depth estimation, which is the problem of estimating each pixel's depth from just a monocular image. Since it was not possible to predict dense depth maps that can be utilized for practical applications before DNNs have become popular, the DNNs provide the computer vision researchers an excellent opportunity to build systems that were not possible in the past. Moreover, existing SLAM methods can yield higher performance with RGB-D inputs, which are both RGB image and the depth map, depth estimation is a fundamental problem for applications leveraging SLAM such as autonomous drones [17], augmented reality (AR) kits [18], virtual reality (VR)

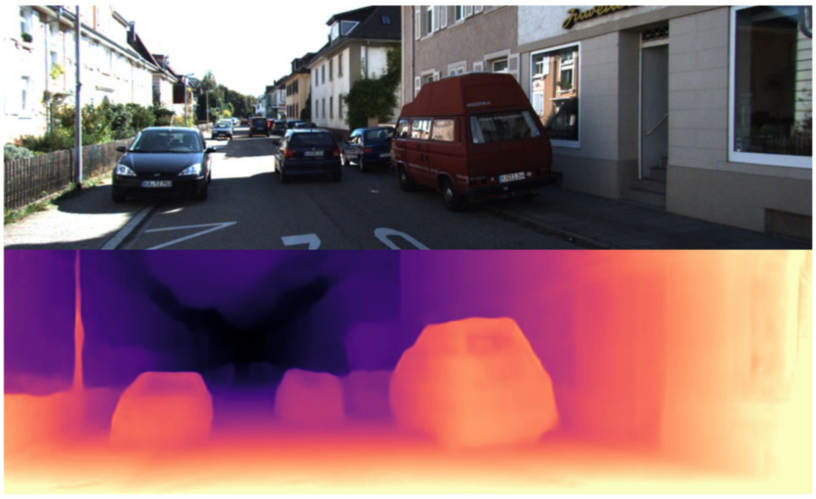
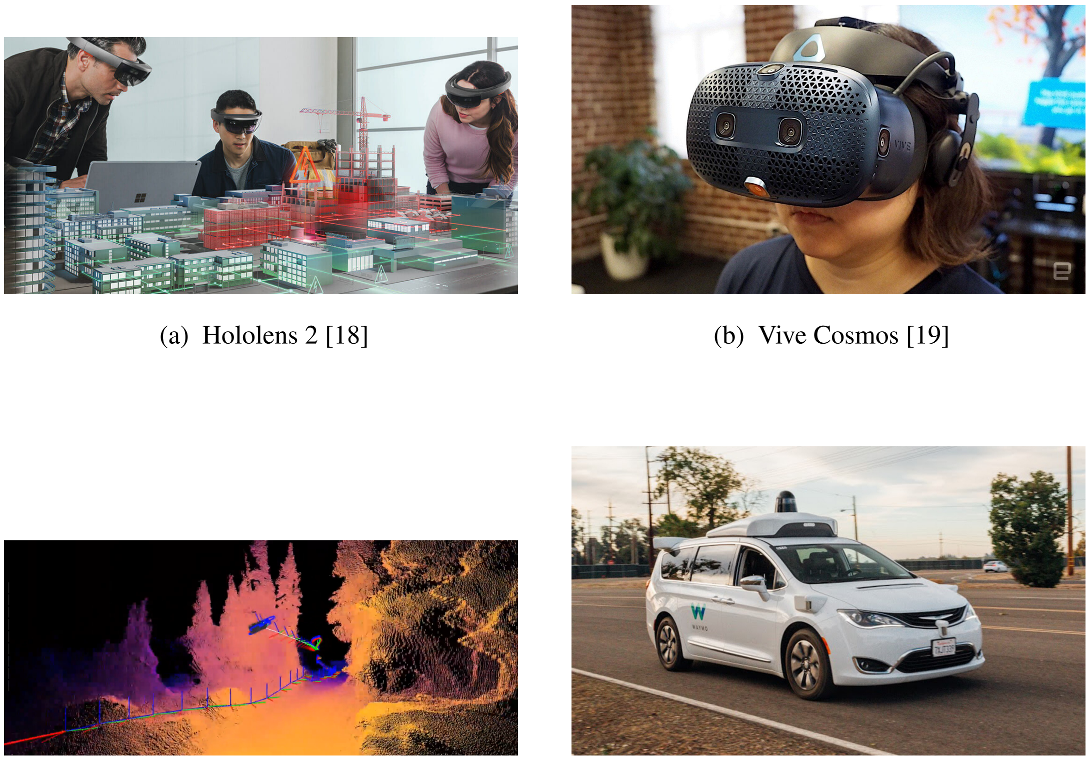


Figure 1.1: An illustration of the depth estimation concept. A depth value is predicted for each pixel in the monocular input image.

kits [19] and autonomous cars. Autonomous car applications have caught significant attention over the previous years, so a generous portion of dense depth estimation research mainly focuses on autonomous car applications. Therefore, dataset targeting autonomous car applications such as KITTI [2] and Cityscapes [20] has appeared, and most of the dense depth estimating studies utilize these datasets.

Some of the significant attempts to estimate a dense depth map from monocular images include [10, 22, 23]. However, these methods have a significant drawback; in fact, they require training with ground truth depth maps, which is costly to obtain. Therefore, unsupervised depth estimation methods attracted attention and some outstanding methods have been proposed, such as SfMLearner [24], Competitive Collaboration (CC) [8], Every Pixel Counts ++ (EPC++) [25], PackNet [5] and Monodepth2 [26]. This study focuses on unsupervised depth and pose estimation. These methods offer great opportunities to make reliable estimations for dense depth maps and relative pose without any need for costly ground truth.


(a) Hololens 2 [18]

(b) Vive Cosmos [19]

(c) Skydio 2 [17]

(d) Waymo One Service [21]

Figure 1.2: Dense depth estimation can open the way of many practical applications such as AR [18], VR [19], autonomous drones [17], and autonomous cars [21].

1.2 Scope of the Thesis

In this study, the performance of unsupervised depth and pose estimation methods is analyzed with different input data configurations. Almost all of the unsupervised depth and pose estimation methods in the literature use 3 input frames for the training process. Therefore, this study aims to investigate the possible benefits of using more than three frames and fill this gap in the literature. The effect of pose estimation performance on the depth estimation performance is investigated. Some novel approaches utilizing semantic segmentation for pose estimation and loss computation are proposed to improve the depth and pose estimation of the proposed network. The foresight of this contribution is that unsupervised methods are trained under the assumption that all scene is stable. However, this is not the case for all images in the datasets used. A way to tackle this issue with semantic segmentation is searched in our proposed method. All of the work in this thesis is based on Monodepth2 [26] since it is one of the best and up-to-date unsupervised methods for depth and pose estimation and since its source code is publicly available which is well-written and easy to manipulate. Although Monodepth2 achieves promising results, there is still a great place for possible improvements.

1.2.1 Methodological Contributions of the Thesis

To sum up the two main contribution of this study can be expressed as

- A thorough inspection to find an input frame combination to boost the performance of Monodepth2.
- A novel framework utilizes the semantic segmentation of the scenes and computes different poses and losses for each of the semantic classes. Then, the computed poses and losses are combined with learnable weights normalized by the softmax function.

1.3 The Outline of the Thesis

This thesis work consists of five chapters. The first chapter introduces our problem motivation and contributions. In Chapter 2, the fundamentals of various camera models and 3D geometry, which are vital for unsupervised depth and pose estimation methods, are presented. In Chapter 3, the current literature and approaches on the unsupervised depth and pose estimation are introduced. In Chapter 4, the tests and our proposed methods are introduced. In this part, every network trained is tested for its depth and pose estimation performance, and the relationship between the depth and pose estimation performance is investigated. The thesis finalizes by stating the conclusions based on experimental analysis.

CHAPTER 2

FUNDAMENTALS OF MULTI-VIEW GEOMETRY

The unsupervised depth and pose estimation methods discussed in Chapter 3 rely on a fundamental understanding of multi-view geometry, which is covered in this chapter.

2.1 Notation

A certain mathematical notation is used in this chapter to prevent any misunderstandings, which are

- Matrices are denoted with bold letters. (e.g. $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$)
- Vectors are denoted with bar. (e.g. $\bar{t} = [1\ 0\ 0]^T$)

2.2 Camera Models and Optical Distortion

A camera can be regarded as a device that maps the information of the 3D world into a 2D image plane. This section of the thesis discusses the underlying mechanisms of this mapping process. Camera models are divided into two categories: finite cameras and cameras at infinity [3]. Finite camera models are employed almost always in computer vision literature; therefore, this thesis and all of the literature discussed in this thesis assume finite camera models.

2.2.1 Pinhole Cameras

The basic pinhole model. This is the most specialized and simplest finite camera model. Suppose that a central projection is performed on plane $Z = f$, which is called the *image plane* or *focal plane* and that the center of the camera is also the origin of the corresponding Euclidean coordinate system. The projection of a 3D point $\bar{X} = (X, Y, Z)$ can be computed as the intersection of the line passes through both **X** and camera center and the image plane, which is illustrated in Figure 2.1.

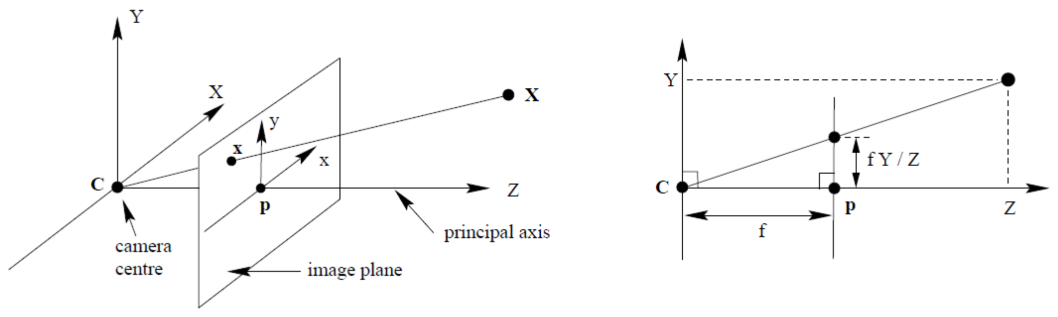


Figure 2.1: The pinhole camera projection model in [3]. The camera center is **C** and **p** is the principal point.

This projection mechanism of the pinhole camera model can be represented as follows:

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z). \tag{2.1}$$

The line which goes from the camera center to the image plane is called the *principal axis*, and the point where the principal axis intersects the image plane is called the *principal point*. The plane, which goes through the camera center and parallels the image plane, is called *principal plane* of the camera.

Central projection using homogeneous coordinates. Since the mapping in the Equation 2.1 is a non-linear transformation, this mapping cannot be expressed using matrices. Homogeneous coordinate system is employed to tackle this problem: An extra dimension with value 1 is added to the point coordinates defined in Euclidean coordinate system. Consider a 3D point $\mathbf{X} = (X, Y, Z)^T$ can be expressed as $(X, Y, Z, 1)^T$ in homogeneous coordinate systems. Similarly, a 2D point $\bar{x} = (u, v)^T$ can expressed as $(u, v, 1)^T$. By using homogeneous coordinate system the Equation

2.1 can be expressed as:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.2}$$

where the 3-by-4 matrix is called the *camera projection matrix*, which can be used to simplify Equation 2.2 as:

$$\bar{x} = \mathbf{P}\bar{X}, \tag{2.3}$$

where $\bar{x} = (fX, fY, Z)^T$ and $\bar{X} = (X, Y, Z, 1)^T$. Moreover, the matrix $\mathbf{P}$ can be decomposed as:

$$\mathbf{P} = diag(f, f, 1)[\mathbf{I}|0]. \tag{2.4}$$

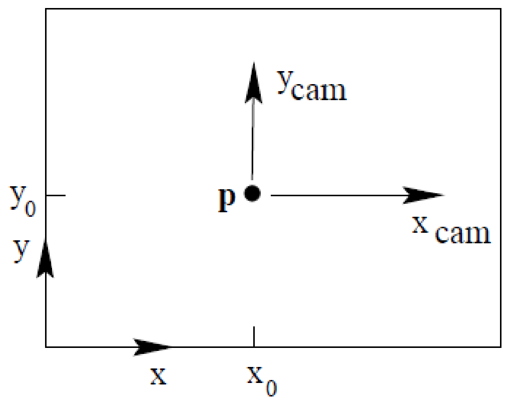


Figure 2.2: Image and camera coordinate systems are illustrated. Figure source: [3].

Principal point offset. The pinhole camera projection defined in Equation 2.1 assumes the origin of the image plane is at the principal point, which may not hold in practice. Therefore, the Equation 2.1 should be extended as:

$$(X, Y, Z)^T \mapsto (fX/Z + p_x, fY/Z + p_y)^T. \tag{2.5}$$

Then, the projection defined in Equation 2.2 can be rearranged as follows:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.6}$$

The 3x3 sub-matrix is called the *camera calibration matrix* and denoted by $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}. \tag{2.7}$$

Then, the projection equation defined in Equation 2.6 can be rewritten in a compact form as follows

$$\bar{x} = \mathbf{K}[I|0]\bar{X}_{cam}, \tag{2.8}$$

where the 3D point $(X, Y, Z, 1)^T$ to be projected is denoted as $\bar{X}_{cam}$

2.3 Fundamentals of Multi-view Geometry

In this section, the relationship between two or more cameras in a world frame is discussed. The relation expressed in Equation 2.3 can be also expressed as:

$$\lambda\bar{x} = \mathbf{P}\bar{X} \tag{2.9}$$

where $\bar{x}$ is the homogeneous 2D pixel coordinate, $\bar{X}$ is the homogeneous 3DD world coordinate, P is the projection matrix, and λ is the depth of the 3D world point $\bar{X}$. Previously, Equation 2.3 is decomposed as in Equation 2.8. A more general decomposition of the *projection matrix* can be expressed as

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\bar{t}] \tag{2.10}$$

where $\mathbf{K}$ is the intrinsic matrix of the camera, and $\mathbf{R}$ and $\bar{t}$ are the rotation matrix and translation vector from world coordinate system to camera frame coordinate system so that a 3D world coordinate $\bar{X}^{world}$ can be converted into a 3D camera coordinate $\bar{X}^{cam}$ as

$$\bar{X}^{cam} = \mathbf{R}\bar{X}^{world} + \bar{t}. \tag{2.11}$$

2.3.1 Epipolar Geometry

The relation between two cameras is described with epipolar geometry, which is defined by the observed 3D point and the camera centers. The plane which crosses the observed 3D point and the camera centers is called the epipolar plane, which is showed as the green triangle in Figure 2.3.

It is also possible to observe from Figure 2.3 that the projections of the 3D point have to lie in the *epipolar plane*. Then, if one of the right or left projections ($\bar{x}_L$ or $\bar{x}_R$) is known, it is possible to recover the epipolar plane since three points lie in it are known. The unknown projection on the other camera has to lie in this plane, and the intersection of this plane and camera projection plane is a line called *epipolar line*. That is, if one of the projected points is known, a line that consists of the other projection on the other camera can be computed. This relation can be expressed as follows [3] :

$$\bar{x}_L^T(\mathbf{R}[t_x])\bar{x}_R = 0 \tag{2.12}$$

where $\bar{x}_L$ and $\bar{x}_R$ are the left and right image projections in homogeneous coordinate system, $\mathbf{R}$ is the rotation matrix between left and right camera, and $[t_x]$ is the cross product matrix for translation between left and right image planes. The matrix in the center is called *essential matrix* and defined as:

$$\mathbf{E} = \mathbf{R}[\mathbf{t}_x]. \tag{2.13}$$

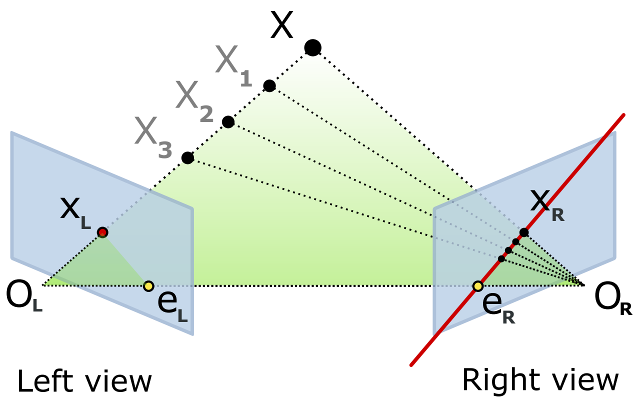


Figure 2.3: Epipolar geometry defines the nature of a two-camera system.

2.3.2 Triangulation

Estimating the 3D coordinate of a point from just one 2D image observation is not possible. On the other hand, if more than one 2D observation of this 3D point is available, an estimation of this 3D point can be made. This method is known as *Triangulation* and illustrated in Figure 2.4.

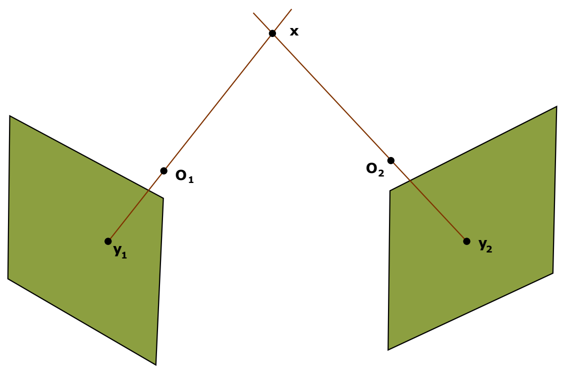


Figure 2.4: A simple triangulation process is basically intersecting two lines at one point.

2.4 Structure from Motion

The problem of estimating 3D points from multiple images of the same scene is known as "Structure from Motion (SfM)" in the computer vision literature. This problem has been studied for decades. Thus, several SfM pipelines were proposed, and new ones keep appearing every year. Bundler [11], VisualSFM [27], OpenMVG [12], and COLMAP [1] are some of the well-known and best SfM pipelines which are proposed. COLMAP [1] is one of the latest SfM pipelines, and the computer vision community widely uses it for different tasks such as ground truth generation for depth images [28, 29] or pose ground truth generation [30].

2.4.1 Bundle Adjustment

Bundle Adjustment (BA) is a non-linear optimization procedure commonly applied in classical computer vision methods such as SfM or SLAM. BA is known to significantly improve the pose and 3D point estimations of many classical applications, making it essential for almost all 3D computer vision applications. A classical BA formulation for estimating the camera poses and the 3D points in the scene is as follows:

$$\min_{\mathbf{a}_j,\,\mathbf{b}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij}\, d(\mathbf{Q}(\mathbf{a}_j,\, \mathbf{b}_i),\ \mathbf{x}_{ij})^2, \tag{2.14}$$

where v_{ij} denotes a binary variable indicating whether point i is visible from camera j, $\mathbf{x}_{ij}$ denotes the projection of point i onto camera j, $\mathbf{a}_j$ denotes each camera j, $\mathbf{b}_i$ denotes each 3D point i. $\mathbf{Q}(\mathbf{a}_j,\, \mathbf{b}_i)$ denotes the predicted projection of point i onto camera j, and d denotes a distance metric.

2.4.2 Pose Evaluation Metrics

The relative pose error (RPE) and absolute trajectory error (ATE), which are proposed by [31] are used in this study to investigate the pose estimation performance of various pose estimation networks. A detailed explanation about RPE and ATE are provided in the upcoming subsections.

Relative Pose Error (RPE). Relative pose error is a measure to assess the odometry systems by comparing the relative pose estimations with the ground truth values. In our experiments, the RPE is defined as

$$RMSE(\mathbf{E}_{1:n}) = \left(\frac{1}{m} \sum_{i=1}^{m} \|trans(\mathbf{E}_i)\|^2 \right)^{1/2} \tag{2.15}$$

where $trans(E_i)$ is the translational component of E_i. $\mathbf{E}_i$ is defined as

$$\mathbf{E}_i = \left(\mathbf{Q}_i^{-1}\mathbf{Q}_{i+1}\right)^{-1} \left(\mathbf{P}_i^{-1}\mathbf{P}_{i+1}\right) \tag{2.16}$$

where $\mathbf{P}_1, \ldots, \mathbf{P}_n \in SE(3)$ are estimated poses and $\mathbf{Q}_1, \ldots, \mathbf{Q}_n \in SE(3)$ are the ground poses for the same sequence.

Absolute Trajectory Error (ATE). Absolute trajectory error is also proposed in [31], and the same metric is also adapted for our tests. ATE is defined as

$$RMSE(\mathbf{F}_{1:n}) = \left(\frac{1}{m} \sum_{i=1}^{m} \|trans(\mathbf{F}_i)\|^2 \right)^{1/2} \tag{2.17}$$

where $trans(\mathbf{F}_i)$ is the translational component of $\mathbf{F}_i$. $\mathbf{F}_i$ is defined as

$$\mathbf{F}_i = \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{2.18}$$

where $\mathbf{P}_{1:n}$ are the estimated trajectory, $\mathbf{Q}_{1:n}$ are the ground truth trajectory, and $\mathbf{S} \in SE(3)$ is the optimal pose transformation which minimizes the error defined in Equation 2.17.

CHAPTER 3

RELATED WORK ON LEARNING BASED DEPTH ESTIMATION

3.1 Introduction

Deep learning-based methods are dominating the fundamental problems of the computer vision field for a while. Along with the many high-level recognition tasks, such as image classification [32, 33, 34], object detection [35, 36, 37, 38, 39], semantic segmentation [40, 41, 37, 42, 43], deep networks have proved themselves in low and mid-level vision task such as optical-flow estimation [44, 45, 46], interest point detection and description [47, 48, 49] and image denoising [50, 51]. Therefore, many researchers working on vision tried to apply deep learning techniques to visual odometry (VO) and simultaneous localization and mapping (SLAM) systems.

The most crucial part of these systems is to infer the 3D world observed by the camera. Estimating each pixel's depth value is one of the most straightforward solutions to this problem; hence, there are several studies aiming to estimate the depth from monocular images have aroused in the last few years. Some of the earlier supervised monocular depth estimation works can be listed as [52] and [53].

After deep CNNs have gained credibility and popularity, CNN-based supervised monocular depth estimation methods, such as [10, 22, 23] appeared. However, grountruth dense depth maps are required to train these networks, which avoids these networks' applicability, since to gather ground depth, LIDARs or RGB-D sensors have to be utilized, which is expansible and tedious.

In order to ease the strict requirement on supervised methods for gathering datasets with ground truth depth values, unsupervised (or self-supervised) monocular depth prediction methods are explored widely in the community. Godard et al. [4] proposed to exploit the stereo camera system used in gathering of KITTI dataset [2]; indeed, they trained an unsupervised depth estimation network to reconstruct the left image from the right image and vice versa. On the other hand, Zhou et al. [24] proposed to train a depth estimation network from just video sequences by calculating the relative pose transformation between consecutive frames and dense depth for each frame. Many relevant works extending this idea, also appeared in the following years, such as [8, 25, 26, 14].

The unsupervised depth and pose-estimation methods can be broadly divided into two main categories with the following subcategories:

1. Stereo Based Methods
2. Mono-view Based Methods
   (a) Intensity Based Methods
   (b) Optical Flow and Motion Segmentation Based Methods
   (c) Semantic Segmentation Based Methods

3.2 Stereo-based Methods

One of the first efforts [4] proposes training a CNN to regress the dense depth field in an unsupervised fashion. The authors propose to exploit the stereo camera setup that is already used in capturing process of the KITTI dataset [2]. The dense depth prediction network is trained by computing the photometric loss between the right stereo image and its prediction from the left stereo image. It is possible to reconstruct the right stereo image from the left stereo image, if the corresponding depth values for the each pixel of the left stereo image is estimated, since the relative pose transformation between left and right cameras is provided by KITTI dataset [2].

The authors [4] extend this idea by training a disparity estimation network which estimates the disparities for left-to-right image transformation d^r and right-to-left image transformation d^l from just left image. This method, namely Monodepth (see Fig.

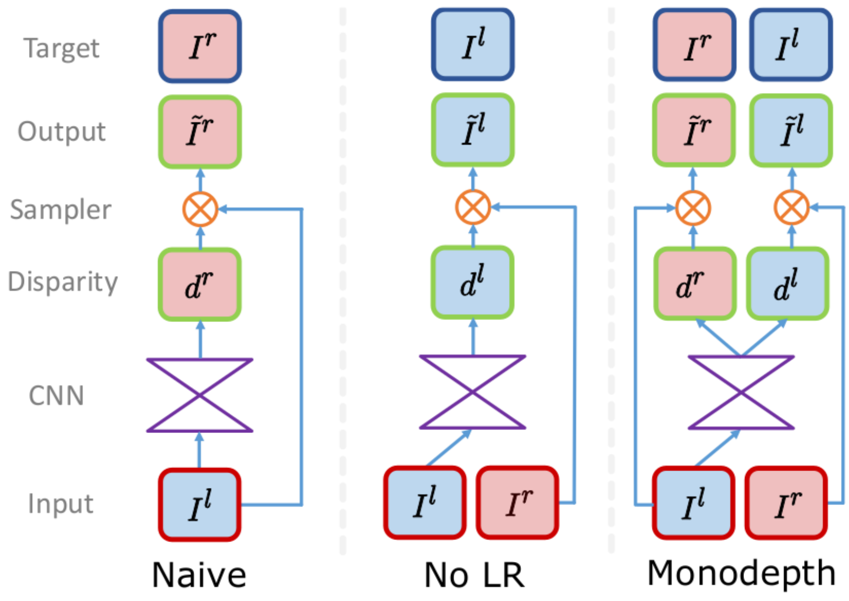


Figure 3.1: Different approaches for Monodepth to train the disparity network [4].

3.1) , enables the network to predict better disparities instead of just reconstructing the right image I^r from the left image I^l by using the disparity computed from left image d^l (illustrated as naive in Fig 3.1) or reconstructing the left image from right image using the disparity computed from left image (Fig. 3.1) .

If the reconstructed left and right images are denoted as $\tilde{I}^l$ and $\tilde{I}^r$, the corresponding disparity estimation network is trained to minimize the following loss after the predicting the disparities and the reconstructed left and right images,

$$C_s = \alpha_{ap}(C^l_{ap} + C^r_{ap}) + \alpha_{ds}(C^l_{ds} + C^r_{ds}) + \alpha_{lr}(C^l_{lr} + C^r_{lr}), \tag{3.1}$$

where C^l_{ap} and C^r_{ap} denotes the photometric loss between left (I^l) and right (I^r) images, and corresponding prediction of left ($\tilde{I}^l$) and right images ($\tilde{I}^r$) which is a combination of L1 loss and SSIM [54] metric shown as

$$C^l_{ap} = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM(I^r_{ij}, \tilde{I}^r_{ij})}{2} + (1-\alpha)|I^r_{ij} - \tilde{I}^r_{ij}|, \tag{3.2}$$

for only left loss. C^l_{ds} and C^r_{ds} losses forces the disparity estimation network to make smooth predictions on the textureless regions by punishing the abrupt changes in disparity and C^l_{ds} is expressed as

$$C^l_{ds} = \frac{1}{N}\sum_{i,j} |\partial_x d^l_{ij}| e^{-\|\partial_x I^l_{ij}\|} + |\partial_y d^l_{ij}| e^{-\|\partial_y I^l_{ij}\|}, \tag{3.3}$$

C^r_{ds} is also expressed in a similar way. C^l_{lr} term defined as

$$C^l_{lr} = \frac{1}{N}\sum_{i,j} |d^l_{ij} - d^r_{ij+d^l_{ij}}|, \tag{3.4}$$

and C^r_{lr} is also expressed in a similar way. These two losses force the network's left and right disparity predictions to be consistent with each other.

3.3 Mono-view Based Methods

Mono-view methods are more ambitious compared to their stereo counterparts, since the structure and the pose are estimated during the ego-motion of a single camera. As mentioned before, mono-view methods can broadly classified into 3 main subcategories as follows:

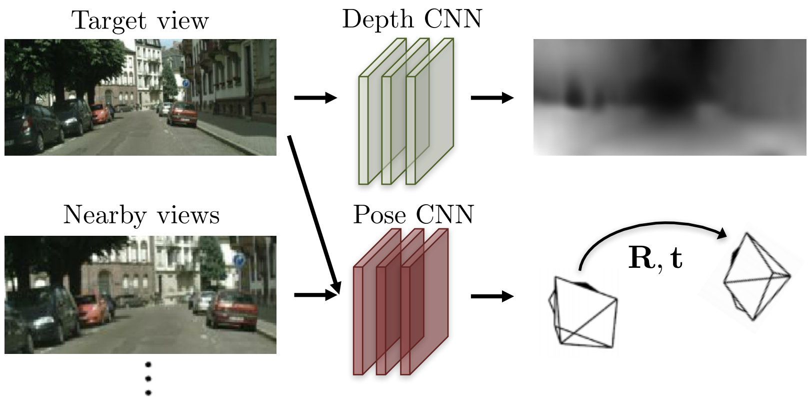
1. Intensity-based Methods
2. Optical Flow and Motion Segmentation-based Methods
3. Semantic Segmentation-based Methods

In the upcoming sections, the methods in these categories are examined in detail.

3.3.1 Intensity-based Methods

Zhang et al. [24] proposed a mono-view method for self-supervised learning of dense depth maps and ego-motion from unlabelled video sequences. The proposed architecture contains two different networks to estimate the dense depth map and the relative pose (rotation and translation) between two consecutive frames, which can be seen in Fig. 3.2. DispNet [55], which is based on an encoder-decoder structure with skip connections and utilizes multi-scale prediction, is adapted as the monocular depth estimation network. Later, the estimated dense depth map and relative poses are used to estimate a video frame from consecutive frames. The absolute difference between the target frame and the estimated frames are utilized as cost function.

(a) Training: unlabeled video clips.



(b) Testing: single-view depth and multi-view pose estimation.

Figure 3.2: A diagram of the proposed method of [4]. One network estimates the depth for every input pixel and the other network estimates the relative rotation and translation. Figure source: [4].

The main idea of this network is to reconstruct a target image from a nearby image taken from close time instants. To reconstruct each pixel of the target image from a nearby image, the 3D coordinate corresponding to each point and the transformation between the target image's camera coordinate system and the nearby camera's coordinate system, which can be represented as a rotation matrix and a translation vector. Therefore, the authors attempted to train two different networks simultaneously to reconstruct the target image with a small error. Suppose that p_t is a pixel coordinate in the target image. Then, the corresponding pixel coordinate p_s in the nearby image can be calculated as pixel coordinate p_s in the nearby image can be calculated as

$$p_s \sim K\hat{T}_{t\to s}\hat{D}_t(p_t)K^{-1}p_t, \tag{3.5}$$

where K is the intrinsic camera matrix, $\hat{T}_{t\to s}$ is the transformation from target camera to source camera, $\hat{D}_t(p_t)$ is the estimated depth value at p_t. Once calculating the p_s for every possible p_t, the target image is reconstructed from a nearby image using bilinear sampling, which is a differentiable operation. The reconstructed image is evaluated by calculating the absolute difference between the reconstructed and target images, which is available at Equation 3.6. Then, the network tries to produce results that minimize the following loss function:

$$\mathcal{L}_{vs} = \sum_{s}\sum_{p} |I_t(p) - \hat{I}_s(p)|, \tag{3.6}$$

where $I_1, ..., I_N$ are the target images using in the training procedure, and $I_s(1 \leq s \leq N, s \neq t)$ are the nearby or source images to be used for reconstructing the target images.

However, this model only covers the static regions in the image. The non-stationary objects in the images cannot be explained with this model. Therefore, the authors tried to tackle this issue by using a novel explainability mask to segment the pixels belonging to the stationary and dynamic objects. The explainability mask is got with up-scaling the network's output, which calculates the relative poses between target and source images. Hence, the proposed image reconstruction loss with explainability mask becomes

$$\mathcal{L}_{vs} = \sum_{s}\sum_{p} \hat{E}_s(p)|I_t(p) - \hat{I}_s(p)|. \tag{3.7}$$

On the other hand, training the network with the above loss will result in a solution where $\hat{E}_s(p) = 0$ for every p, since this solution drops $\mathcal{L}_{vs}$ for every image pair. The authors proposed to add a regularization loss $\mathcal{L}_{reg}(\hat{E}_s)$ to tackle this issue which forces $\hat{E}_s(p)$ to become 1 for every pixel location.

The gradients are calculated using the difference between the $I_t(p_t)$ and $I_s(p_s)$, and this fact could potentially curb the learning process, if p_t is in a low texture region. Therefore, the authors propose a smoothness loss, which forces the predicted depth image to be smooth; in other words, the depth image's average gradient magnitude is forced to be small. After that, the final loss function to be optimized during the training becomes the equation

$$\mathcal{L}_{final} = \sum_{l} \mathcal{L}^{l}_{vs} + \lambda_s \mathcal{L}^{l}_{smooth} + \lambda_e \sum_{s} \mathcal{L}_{reg}(\hat{E}^{l}_{s}), \tag{3.8}$$

where l denotes the different image scales, s denotes the different source images, and λ_s and λ_e are the weights for the depth smoothness loss and the regularization term for explainability mask.

The network is trained on KITTI dataset [2] with train and test splits proposed by Eigen et al. [10]

In a different approach, Guizilini et al. [5] improves [26] with 3 main modifications. The first main contribution is a novel architecture, called *PackNet*, for depth estimation utilizing not only 2D convolution layers but also 3D convolution layers. The second contribution is to adapting the measured velocity during the dataset shooting to gain the ability to estimate the relative poses on a real-world scale. The third contribution is a novel autonomous driving dataset, which consists of an assortment of driving sequences.

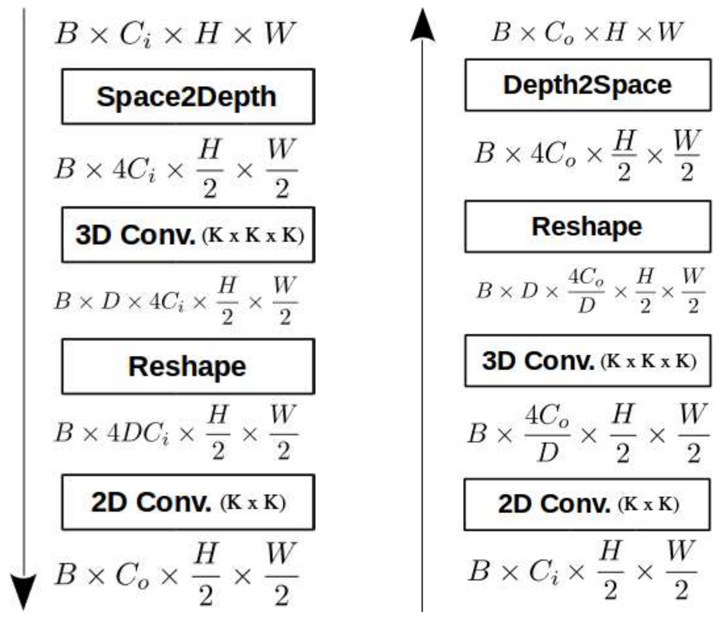
The authors [5] argue that this type of CNNs adapting 3D convolution layers is superior in estimating high-level information, such as depth values compared to the classical encoder-decoder architectures utilizing popular encoder structures ResNet [33] or VGG [34] networks. The proposed PackNet network is constructed with two types of blocks stages, called *packing block* and *unpacking block*, which are analogous to the convolution layer and transpose convolution layer of the classical depth estimation networks.

The packing block first applies *Space2Depth* [56] operation which transfers the information in the spatial dimensions to the channel dimension. After that step, a 3D convolution, a reshape and a 2D convolution operations follow. The unpacking blocks are just the opposite of packing blocks with a 2D convolution, a reshape, a 3D convolution, and *Space2Depth* [56]. The structure of packing and unpacking blocks are illustrated in Figure 3.3. Packing blocks are used in the encoder part, and the unpacking parts are used in the decoder part along with the Resnet [33] modules, which is illustrated in Figure 3.4.

The proposed depth and pose estimation networks are trained with the proposed loss function in

$$\mathcal{L}(I_t, \hat{I}_t) = \mathcal{L}_p(I_t, I_s) \odot \mathcal{M}_p \odot \mathcal{M}_t + \lambda_1 \mathcal{L}(\hat{D}_t), \tag{3.9}$$

where $\mathcal{L}(I_t, \hat{I}_t)$ is the photometric loss between target and predicted image defined in Equation 3.17, $\mathcal{M}_p$ and $\mathcal{M}_t$ are same the binary masks formulated in the Equation 3.20, and $\mathcal{L}(\hat{D}_t)$ is the depth smoothness loss defined in (3.19).


$B \times C_i \times H \times W$
Space2Depth
$B \times 4C_i \times \frac{H}{2} \times \frac{W}{2}$
3D Conv. (K x K x K)
$B \times D \times 4C_i \times \frac{H}{2} \times \frac{W}{2}$
Reshape
$B \times 4DC_i \times \frac{H}{2} \times \frac{W}{2}$
2D Conv. (K x K)
$B \times C_o \times \frac{H}{2} \times \frac{W}{2}$
$B \times C_o \times H \times W$
Depth2Space
$B \times 4C_o \times \frac{H}{2} \times \frac{W}{2}$
Reshape
$B \times D \times \frac{4C_o}{D} \times \frac{H}{2} \times \frac{W}{2}$
3D Conv. (K x K x K)
$B \times \frac{4C_o}{D} \times \frac{H}{2} \times \frac{W}{2}$
2D Conv. (K x K)
$B \times C_i \times \frac{H}{2} \times \frac{W}{2}$

(a) Packing (b) Unpacking

Figure 3.3: The inside architecture of the proposed packing and unpacking blocks. Figure source: [5].

One of the main drawbacks of the self-supervised depth estimation networks is that their estimation is not in real-world scales due to the well-known scaling ambiguity for mono-view sequences. A novel loss based on the measured velocity of the cars for which the KITTI dataset [2] were gathered, is proposed to solve this issue. This velocity loss basically forces the network to make relative pose estimations whose translation component to be close as possible as to the estimated translation computed by multiplying the velocity measured at that timestamp and the measured time difference between two consecutive frames which is formulated as

$$\mathcal{L}_v(\hat{t}_{t\to s}, v) = |\|\hat{t}_{t\to s}\| - |v|\Delta T_{t\to s}|, \tag{3.10}$$

where $\hat{t}_{t\to s}$ is the estimated translation vector between frame t and s, v is measured velocity, and $\Delta T_{t\to s}$ is the time difference between frames t and s. After calculating $\mathcal{L}_v(\hat{t}_{t\to s}, v)$, the final loss is calculated as

$$\mathcal{L}_{scale}(I_t, \hat{I}_t, v) = \mathcal{L}(I_t, \hat{I}_t) + \lambda_2 \mathcal{L}_v(\hat{t}_{t\to s}, v), \tag{3.11}$$

where λ_2 is a weight term to balance the losses $\mathcal{L}(I_t, \hat{I}_t)$ and $\mathcal{L}_v(\hat{t}_{t\to s}, v)$, to train scale-aware networks.

	Layer Description	**K**	**Output Tensor Dim.**
#0	Input RGB image		3×H×W
	Encoding Layers		
#1	Conv2d	5	64×H×W
#2	Conv2d → Packing	7	64×H/2×W/2
#3	ResidualBlock (x2) → Packing	3	64×H/4×W/4
#4	ResidualBlock (x2) → Packing	3	128×H/8×W/8
#5	ResidualBlock (x3) → Packing	3	256×H/16×W/16
#6	ResidualBlock (x3) → Packing	3	512×H/32×W/32
	Decoding Layers		
#7	Unpacking (#6) → Conv2d (⊕ #5)	3	512×H/16×W/16
#8	Unpacking (#7) → Conv2d (⊕ #4)	3	256×H/8×W/8
#9	InvDepth (#8)	3	1×H/8×W/8
#10	Unpacking (#8) → Conv2d (⊕ #3 ⊕ Upsample(#9))	3	128×H/4×W/4
#11	InvDepth (#10)	3	1×H/4×W/4
#12	Unpacking (#10) → Conv2d (⊕ #2 ⊕ Upsample(#11))	3	64×H/2×W/2
#13	InvDepth (#12)	3	1×H/2×W/2
#14	Unpacking (#12) → Conv2d (⊕ #1 ⊕ Upsample(#13))	3	64×H×W
#15	InvDepth (#14)	3	1×H×W

Figure 3.4: The complete architecture with depth estimation network of PackNet [5] which proposes to use packing and unpacking blocks inside the network. Figure source: Guizilini et al. [5].

Yang et al. [14] propose a similar method, namely D3VO, to [26] to train a depth and a pose estimation network. The proposed depth estimation network predicts an uncertainty map and the depth map, since the proposed network is part of the visual odometry (VO) system, and uncertainty is critical in VO and simultaneous localization and mapping (SLAM) systems. The depth estimation and pose estimation architecture proposed in [26] is utilized with slight modifications. Unlike the other unsupervised depth estimation methods, D3VO [14] does not directly compute the loss between the reconstructed image from nearby and the source image. Instead, an affine transformation is applied to the intensity values of the source image first. This modification aims to cope with the effect of the changing exposure and lighting conditions between two consecutive frames. The transformed image is computed as

$$I^{a,b} = aI + b. \tag{3.12}$$

The pose estimation network computes the affine transformation parameters a and b. Moreover, the depth estimation network also predict the uncertainty map along with the depth map which is used the loss formulation as

$$L_{self} = \frac{1}{|V|} \sum_{p \in V} \frac{\mathbf{min}_{t'} r(I_t^{a_{t'}, b_{t'}}, I_{t' \to t})}{\Sigma_t} + \mathbf{log}\Sigma_t \tag{3.13}$$

where Σ_t is the uncertainty map predicted by the depth estimator. A regularization loss also used to keep the affine transformation parameters a and b close to 1 and 0 just as

$$L_{ab} = \sum_{t'} (a_{t'} - 1)^2 + b_{t'}^2. \tag{3.14}$$

Then, the total loss is calculated as

$$L_{total} = \frac{1}{s} \sum_{s} (L_{self}^s + \lambda L_{reg}^s), \tag{3.15}$$

where s is equal to 4 and represents the number of image scales and $L_{reg} = L_{smooth} + \beta L_{ab}$.

Shi et al. [7] propose a relatively different unsupervised depth and pose estimation method by leveraging the bundle adjustment (BA) concept. The idea is to apply BA to the predicted depth and pose values so that the backpropagation is performed from update depth and pose values, which results in an improved depth and pose estimation network. A differentiable BA module [6] is adapted for this purpose. However, this BA module optimizes the loss computed from the computed feature pyramid instead of computing directly from the target and reconstructed images as

$$e_{i,j}^{feat}(S) = F_i(\pi(T_i, d_j \cdot q_j)) - F_1(q_j). \tag{3.16}$$

where F_i is the feature vector representing image I_i, S is the parameters to optimize (depth map and relative in this case), q_j is an image coordinate of image I_i, T_i is the relative pose between image I_i and the target image, d_j is the depth estimation for pixel q_j in image I_i, and π is the projection function which projects a 3D coordinate onto a 2D image plane. After updating the depth and pose values according to loss function defined in (3.16), the depth and pose networks are updated with the sum of same photometric, and smoothness losses defined in (3.29), and 3.3. A schematic of the proposed algorithm can examined on Figure 3.5.

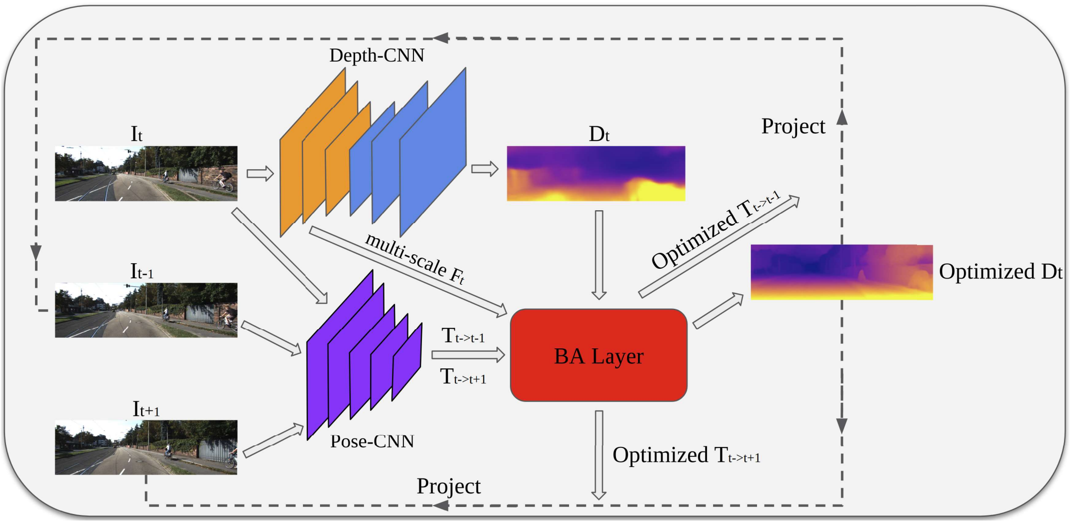


Figure 3.5: The proposed unsupervised depth and pose estimation network which adapts the differentiable BA layer [6]. Figure source: Shi et al. [7].

3.3.1.1 Monodepth2

This section is devoted to Monodepth2 [26] algorithm, since it is the main method utilized throughout the thesis. This method is an enhanced self-supervised depth and relative pose estimation method, which is an improved version of [24]. Since this method provides one of the most competing results in the literature and its source code is publicly available. it is preferred to be used in this study.

Monodepth2 contains two deep networks that predict the dense depth and ego-motion between two images simultaneously. The training procedure is similar to [24]. The target image is reconstructed by using the dense depth prediction of the target image and the relative pose between the target frame and the nearby frames using the equations 3.5 and bilinear interpolation. After that step, a loss function is calculated to assess the quality of the reconstructed image that is a weighted sum of L1 distance and the SSIM [54] metric, defined as

$$pe(I_a, I_b) = \alpha\frac{1 - SSIM(I_a, I_b)}{2} + (1-\alpha)\|(I_a - I_b)\|_1. \tag{3.17}$$

On contrary to [24], the authors preferred to construct the final loss to optimize by only taking one of the nearby image whose loss defined in (3.17) is minimum, instead of averaging them. The authors claim that this approach enables the network to discard the occluded pixels. Therefore, the network reconstructs the input image with minimum possible loss, which is defined as

$$L_p = \min_{t'}(pe(I_t, I_{t' \to t})) \tag{3.18}$$

where pe is photometric error defined Equation 3.17. The low gradient issue in the textureless regions considered by [24] is tackled by using an edge aware smoothness loss L_s shown in

$$L_s = |\partial_x d_t^*| e^{-|\partial_x I_t|} + |\partial_y d_t^*| e^{-|\partial_y I_t|}, \tag{3.19}$$

where $d_t^* = d_t / \overline{d_t}$ is the inverse depth value normalized by its mean, and ∂_x and ∂_y are partial derivative operators operate on x and y axes of the images. This loss forces the depth network to predict smooth depth values in the regions, where no edge or corner exist in the input image.

One of the assumptions for the reconstruction process of target image is that all the pixels in the image belong to stationary objects. However, this is a erroneous assumption in many available datasets. Therefore, the authors propose a way to mask the pixels belonging to stationary objects. For this purpose, the loss between the reconstructed image $I_{t' \to t}$ and the target image I_t is calculated. Then, the loss between corresponding nearby frame $I_{t'}$ and the target image I_t is calculated. Calculated these two losses are compared, and the pixels with a lower loss between the reconstructed and target images are assumed to belong to the stationary objects, using the following relation:

$$\mu = \left[\min_{t'}(pe(I_t, I_{t' \to t})) < \min_{t'}(pe(I_t, I_{t'})) \right]. \tag{3.20}$$

Multi-scale depth prediction and loss calculation on that scales is utilized in this approach to push to the network through global minimum value. The loss is calculated at each scale is combined, so that the total loss is calculated. The final training loss to be optimized is constructed by using a weighting factor λ as

$$L = \mu L_p + \lambda L_s. \tag{3.21}$$

The depth estimation network of Monodepth2 utilizes U-Net [57] architecture with a ResNet18 [33] based encoder initialized with weights trained on ImageNet dataset [58]. The decoder network is the same decoder network used by [24] with sigmoid function at the output and ELU nonlinearity [59] at the other layers. ResNet18 [33] is also utilized for pose estimation networks and modified to output 6 values which represent a 6 degree of freedom similarity transformation.

3.3.2 Optical Flow and Motion Segmentation Based Methods

Luo et al. [25] proposed a method which tackles the problem of learning the optical flow estimation, relative pose estimation and monocular depth estimation at the same from videos sequences. Their proposed network estimates explicitly the flow maps for forward and backward direction ($\mathbf{F}_{t\to s}$, $\mathbf{F}_{s\to t}$), relative rigid body motion of the camera ($\mathbf{T}_{t\to s}$), and the depth maps ($\mathbf{D}_t$, $\mathbf{D}_s$) given two consecutive frames (I_s and I_t). The predicted information is fed to a network block called holistic motion parser (HMP). The HMP is calculates a visibility mask ($\mathbf{V}$), a moving object segmentation mask ($\mathbf{S}$) predicting the pixel belonging to the moving objects in the scene, 3D motion of the each pixel belonging to background ($\mathbf{M}_b$), and moving objects ($\mathbf{M}_d$). Then, the corresponding matching pixels calculated from 3D rigid-body motion p_{st} and from optical flow p_{sf} for each pixel p_t in the target image can be calculated as

$$h(p_{st}) = \pi(\mathbf{K}[\mathbf{T}_{t\to s}\mathbf{D}_t(p_t)\mathbf{K}^{-1}h(p_t) + \mathbf{M}_d^*(p_t)]) \tag{3.22}$$

$$p_{sf} = p_t + \mathbf{F}_{t\to s}(p_t), \tag{3.23}$$

and

$$\mathbf{M}_b(p_t) = \mathbf{T}_{t\to s}\phi(p_t|\mathbf{D}_t) - \phi(p_t|\mathbf{D}_t) \tag{3.24}$$

$$\mathbf{M}_d(p_t) = \mathbf{V}(p_t)[\phi(p_t + \mathbf{F}_{t\to s}|\mathbf{D}_s) - \phi(p_t|\mathbf{D}_t) - \mathbf{M}_b(p_t)] \tag{3.25}$$

$$\mathbf{V}(p_t) = \mathbb{1}(\sum_{p_s}(1 - |p_t - (p_s + \mathbf{F}_{s\to t})|) > 0) \tag{3.26}$$

$$\mathbf{S}_{p_t} = 1 - exp\{-\alpha_s(\|\mathbf{M}_d(p_t)\|_2)\}. \tag{3.27}$$

Therefore, it is possible to reconstruct the source image's target image since matching pixel coordinate for each pixel in the source image is known. The final photometric reconstruction loss is a combination of the absolute difference and SSIM as

$$\mathcal{L}_{vs}(\mathbf{O}) = \sum_{p_t}\mathbf{V}_*(p_t, \mathbf{O})s(I_t(p_t), \hat{I}_t(p_t)), \tag{3.28}$$

$$s(I_t(p_t), \hat{I}_t(p_t)) = (1-\beta)|I_t(p_t) - \hat{I}_t(p_t)| + \beta\frac{1 - SSIM(I_t(p_t), \hat{I}_t(p_t))}{2}. \tag{3.29}$$

Ranjan et al. [8] extends [24] by adding two networks to predict the optical flow between the frames and segment the target image into two segments as moving objects and stationary objects. The depth network's training procedure and the pose network are similar to [24] with one difference. The input images of the depth and pose networks are masked by the motion segmentation network so that only the pixels belonging to the stationary objects are taking account.

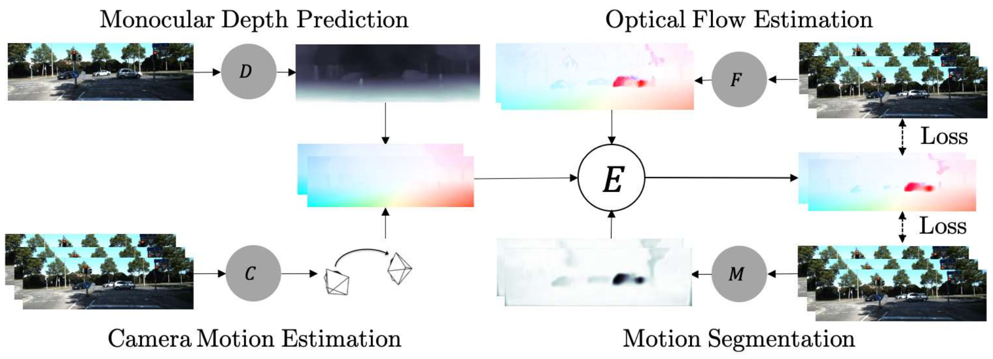


Figure 3.6: A schematic diagram of the Ranjan et al. [8]. Four different network performs dense depth estimation, relative pose estimation, optical flow estimation, and motion segmentation. Figure source: Ranjan et al. [8].

This work's main idea is to segment the scene into two segments as moving and static objects. Then, the static part of the target image is reconstructed from nearby images similar to [24], and the moving part of the target image is reconstructed by using the optical flow estimation. The training of these four different deep networks is performed in two main steps. Assume that an unlabeled dataset $\mathcal{D} = \{\mathcal{D}_i : i \in \mathbb{N}\}$. In the first step, the loss defined as

$$E_1 = \sum_i \sum_\Omega m \cdot L_R(R(\mathcal{D}_i)) + (1 - m) \cdot L_F(F(\mathcal{D}_i)), \tag{3.30}$$

is optimized while keeping the motion segmentation network fixed, where m represents the motion segmentation mask, $\cdot$ represents the element-wise multiplication, $L_R(R(\mathcal{D}_i))$ represents the loss of the static scene reconstruction with depth and relative pose estimation and $L_F(F(\mathcal{D}_i))$ denotes the loss of dynamic scene reconstruction with optical flow estimation. After the loss E_1 is optimized the second loss E_2 denoted in

$$E_2 = E_1 + \sum_i \sum_\Omega L_M(\mathcal{D}_i, R, F), \tag{3.31}$$

is optimized while keeping the depth, pose and optical flow networks fixed, which mean that only the motion segmentation network is optimized in the second step, where L_M denotes the consensus between two network R and F where R denotes the depth and pose networks together, and F denotes the optical flow network. The total loss is reconstructed as follows

$$E = \lambda_R E_R + \lambda_F E_F + \lambda_M E_M + \lambda_C E_C + \lambda_S E_S. \tag{3.32}$$

The losses E_1 and E_2 denoted in the equations 3.30 and 3.31 are derived from E by adjusting the weights $\lambda_R, \lambda_F, \lambda_M, \lambda_C, \lambda_S$. The loss E_R which calculates loss for dynamic scene reconstruction is calculated as follows

$$E_R = \sum_{s \in \{+,-\}} \sum_\Omega \rho(I, \omega_c(I_s, e_s, d)) \cdot m_s, \tag{3.33}$$

where I_-, I, I_+ denotes the image sequence used for training; in fact, I is the target image. Ω is the spatial pixel domain, ρ is the error function to compare the original and reconstructed images, and ω_c warps the reconstructed image from nearby images I_- and I_+ using the depth d and camera motion estimates e_- and e_+. The networks used for estimate the depth, pose, camera motion, optical flow and motion segmentation are denoted as $\{D_\theta, C_\phi, F_\psi, M_\chi\}$. The depth estimation d, the camera motion estimation e_s, and motion segmentation masks m_- and m_+ are estimated as

$$d = D_\theta(I) \tag{3.34}$$

$$e_-, e_+ = C_\phi(I_-, I, I_+) \tag{3.35}$$

$$m_-, m_+ = M_\chi(I_-, I, I_+). \tag{3.36}$$

The reconstruction loss for the dynamic scenes using the image sequence I_-, I, I_+ and optical flow estimates are calculated as

$$E_F = \sum_{s \in \{+,-\}} \sum_\Omega \rho(I, \omega_f(I_s, u_s)) \cdot (1 - m_s), \tag{3.37}$$

where ω_f reconstructs the target image I from I_- and I_+ using the optical flow estimates u_- and u_+. The optical flow estimates are calculated as

$$u_- = F_\psi(I, I_-),\ u_+ = F_\psi(I, I_+). \tag{3.38}$$

The error function $\rho(x, y)$ is similar to Eq. 3.29 with small differences as

$$\rho(x, y) = \lambda_p \sqrt{(x - y)^2 + \epsilon^2} + (1 - \lambda_p)(1 - SSIM(x, y)). \tag{3.39}$$

The loss E_M is the cross entropy between the motion segmentation mask and a unit tensor to prevent a solution which always predicts every pixel in every input image as dynamic object pixels, and calculated as

$$E_M = \sum_{s \in \{+,-\}} \sum_{\Omega} H(\mathbf{1}, m_s). \tag{3.40}$$

The consensus loss E_C ensures the results of the motion segmentation, depth, pose and optical flow estimation networks are consistent. It is computed as

$$E_C = \sum_{s \in \{+,-\}} \sum_{\Omega} H(\mathbb{I}_{\rho_R < \rho_F} \vee \mathbb{I}_{\|v(e_s, d) - u_s\|_2 < \lambda_c}, m_s), \tag{3.41}$$

where $\mathbb{I} \in \{0, 1\}$ is an indicator function whose value is equal to 1 if the statement in subscript is true, $v(e_s, d)$ denotes the optical flow of the dynamic scene which is calculated using the depth and relative pose provided by the depth and pose networks D_θ, and C_ϕ, $\rho_R = \rho(I, \omega_c(I_s, e_s, d))$ and $\rho_F = \rho(I, \omega_f(I_s, u_s))$ which denotes the photometric reconstruction loss for static and dynamic regions of the target image.

Finally, a smoothness loss is introduced to force the depth, optical flow and motion segmentation results to be smooth as

$$E_S = \sum_{\Omega} \|\lambda_e \nabla d\|^2 + \|\lambda_e \nabla u_-\|^2 + \|\lambda_e \nabla u_+\|^2 + \|\lambda_e \nabla m_-\|^2 + \|\lambda_e \nabla m_+\|^2. \tag{3.42}$$

3.3.3 Semantic Segmentation Based Methods

Guizilini et al. [9] proposed a method to train an unsupervised depth and an ego-motion estimation network by leveraging a pre-trained semantic segmentation network and applying a novel two-stage training method to eliminate the faulty depth estimates placed at the infinity. The architecture proposed in [5] adapted for depth estimation and pose estimation networks. The loss formulations in the Equations 3.17, 3.18, 3.19 and the auto-masking method defined in Equation 3.20 proposed by [26] utilized to train the networks. Apart from the other existing methods, the depth

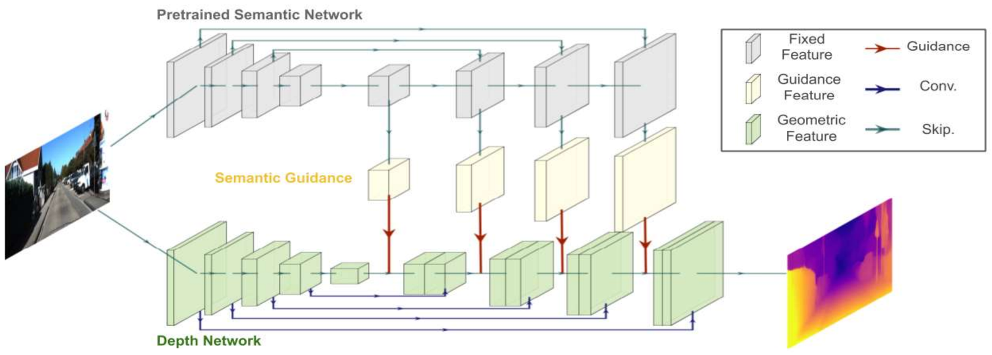


Figure 3.7: A diagram of the semantically guided depth estimation network proposed by [9]. Figure source: Guizilini et al. [9].

estimation network in [9] is trained with the guidance of the pre-trained semantic segmentation network [60] by utilizing pixel-adaptive convolutions [61]. This guidance process is illustrated in Figure 3.7.

A 3×3 and a 1×1 convolutions, group normalization [62] and ELU non-linearity [59] are applied to each fixed feature map of the guidance network. Thus, the guidance features in the Figure 3.7 are constructed. These guidance features are convolved with the geometric features as formulated as

$$\mathbf{v}'_i = \sum_{j\in\Omega(i)} K(\mathbf{f}_i, \mathbf{f}_j)\mathbf{W}[\mathbf{p}_i - \mathbf{p}_j]\mathbf{v}_j + \mathbf{b}, \tag{3.43}$$

where $\mathbf{f} \in \mathcal{R}^D$ denotes the vectors of the guidance features, $\mathbf{p} = (x, y)^T$ denotes the pixel coordinates, $[\mathbf{p}_i - \mathbf{p}_j]$ denotes the 2D offset between pixels i and j, $W_{k\ timesk}$ is the weight matrix to used in the window $\Omega(i)$, $\mathbf{b} \in \mathcal{R}^1$ denotes the bias term, and K denotes the kernel function used. In this case, the authors prefer to use a standard Gaussian kernel as formulated

$$K(\mathbf{f}_i, \mathbf{f}_j) = exp(-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^T\Sigma_{ij}^{-1}(\mathbf{f}_i - \mathbf{f}_j)), \tag{3.44}$$

where $\Sigma_{ij}^{-1} = \sigma I_D$, I_D is an identity matrix with shape of $D \times D$ and σ is a variable to learned during the training.

Other than the semantic guidance, [9] proposed a novel method to filter out the data producing erroneous results. The ground plane is detected using RANSAC [63], and the points estimated below this ground plane are detected. If more than 10% of all points lie under the plane corresponding image is filtered out and not used in the upcoming epochs for training.

3.4 Comparison of Methods in the Literature

A comparison of the depth estimation performance of all methods in this chapter is to be provided. For this purpose, the definitions of the depth estimation metrics that are used throughout this study are also given next.

3.4.1 Error Metrics for Depth Estimation

All the unsupervised depth estimation methods discussed in Chapter 3 utilizes the metrics proposed in [10] for comparing depth estimation results. These metrics are defined in this section for completeness.

It should be noted that before computing the metrics between the estimated depth maps and the ground truth depths, the estimated depth maps are scaled by using median scaling, as proposed in [24]. In this technique, each estimated depth map's depth values are multiplied with a scale $\hat{s}$, which is computed by using the ratio of the median of the depths in estimated depth map and the ground truth depths. Hence, the following relation is obtained $\hat{s} = Median(D_{gt})/Median(D_{pred})$.

The error metric relations defined below use y as the depth estimate, whereas $\hat{y}$ as the ground truth depth.

- Absolute Relative Difference: $\frac{1}{|T|}\sum_{y\in T}|y-\hat{y}|/\hat{y}$
- Squared Relative Difference: $\frac{1}{|T|}\sum_{y\in T}||y-\hat{y}||^2/\hat{y}$
- Root Mean Square Error (RMSE) Linear: $\sqrt{\frac{1}{|T|}\sum_{y\in T}||y-\hat{y}||^2}$
- RMSE Log: $\sqrt{\frac{1}{|T|}\sum_{y\in T}||log(y)-log(\hat{y})||^2}$
- Threshold: $\%$ of y s.t. $max(\frac{y}{\hat{y}},\frac{\hat{y}}{y}) = \delta < thr$

The lower values of absolute relative difference, squared relative difference, root means square error (RMSE), RMSE Log, and the higher values of threshold ($\delta < thr$) indicate the superior performance and vice versa.

3.4.2 Depth Estimation Performance Comparison

The results of the leading depth estimation algorithms are tabulated in Table 3.1 based on their own reports in the original papers.

Table 3.1: Experimental results of the aforementioned unsupervised depth estimation network. In the trains column "S" denotes the stereo supervision and "M" denotes the monocular video sequence supervision (The best metric in each column is denoted with bold characters, while the lower is better for Abs. Rel., Sq. Rel., RMSE, and RMSE Log, whereas the higher is bette forr $\delta < 1.25$, $\delta < 1.25^2$, and $\delta < 1.25^3$)

Method	Train	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth [4]	S	0.133	1.142	5.533	0.230	0.830	0.936	0.970
SfMLearner [24]	M	0.183	1.595	6.709	0.270	0.734	0.902	0.959
CC [8]	M	0.148	1.149	5.464	0.226	0.815	0.935	0.973
EPC++ [25]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Monodepth2 [26]	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
PackNet [5]	M	0.111	0.785	4.601	0.189	0.878	0.960	0.982
Guizilini et al. [9]	M + Sem	0.102	**0.698**	**4.381**	**0.178**	**0.896**	**0.964**	**0.984**
D3VO [14]	MS	**0.099**	0.763	4.485	0.185	0.885	0.958	0.979
Shi et al. [7]	M	0.113	1.079	4.931	0.206	0.853	0.947	0.979

Among all these methods, Guizilini et al. [9] achieves the best results on 6 metrics out of the provided 7 metrics, whereas D3VO [14] returns the best results only on the absolute relative error metric. Although D3VO utilizes both mono and stereo images during its training, it is surpassed by Guizilini et al. [9] on 6 metrics of all 7, that might indicate that semantic guidance is promising for unsupervised depth and pose estimation. A better guidance network and a way to guide the depth estimation could be proposed to extend this study as well.

CHAPTER 4

EFFECTS OF MULTIPLE FRAMES AND SEMANTIC OBJECTS FOR DEPTH AND POSE ESTIMATION

In this chapter, we perform three crucial analyses for unsupervised depth and ego-motion estimation:

- Analysis 1: The comparison of the pose estimation performance for classical and DNN-based methods,
- Analysis 2: The effects of using various frame combinations input to DNN-based method,
- Analysis 3: The idea of leveraging the unsupervised depth and ego-motion by utilizing semantic segmentation.

4.1 Experimental Settings

4.1.1 Selected Architecture

Monodepth2 [26] is used to conduct all the experiments and propose extension methods that are discussed in this chapter. As explained in the literature work, Monodepth2 is composed of two different deep networks, which are depth estimation and pose estimation networks, which consist of around 15M and 13M parameters, respectively.

4.1.2 Optimization

The default training settings of Monodepth2 is kept the same for all of our experiments. Adam [64] optimizer used with initial learning rate of 1.0×10^{-4}, step size 15 with ratio 0.1 and it is trained for 20 epochs.

4.1.3 Dataset

The training, validation, and test splits of KITTI Dataset [2] that are proposed in [10] is utilized during all of our depth estimation experiments, since the Monodepth2 is also trained on this split as the other unsupervised depth and pose estimation methods discussed in Chapter 3. KITTI Odometry Dataset [2] is utilized for all experiments where pose estimation performance is computed, since this dataset contains the ground pose data for all frames. KITTI Odometry Dataset is also utilized for our depth estimation experiments in Section 4.2.1.

4.2 Analysis 1: Comparison of Classical and DNN-based Methods for Pose Estimation

The unsupervised depth estimation networks discussed in Chapter 3 requires a reliable pose estimation between two images. Therefore, the pose estimation performance of the various pose estimation networks and the classical SfM pipelines are compared. For this comparison, absolute trajectory error (ATE) and relative pose error (RPE) are typically used in the literature [31].

All the available sequences (i.e. Sequence-0,-1,-3,-4,-5,-6,-7,-9,-10) of KITTI 2012 Odometry Dataset [2] are utilized in this section. However, this section mainly focuses on the results of Sequence-9, since the unsupervised depth and pose estimation methods in the literature mostly report the pose estimation performance for Sequence-9. It should also be noted that our experimental observations on the performances of all the sequences are consistent with that of Sequence-9.

In the first test, Sequence-9 is processed as 3 frames, since most of the pose estimation networks in the literature take 3 frames as input. For example, Sequence-9 has 1591 frames, so the first frames 0, 1, 2 are fed to networks, then 1,2,3, and so on. Therefore, the pose estimation networks estimate the relative pose between the center to nearby frames. It is possible to compute the estimation errors, since ground truth pose data is available on KITTI 2012 Odometry Dataset [2]. The ATE and RPE values are calculated for each 3 frame snippet; then, the mean and standard deviation of these errors of every snippet are computed and provided in Table 4.1. The best performing method for all metrics in this table is COLMAP.

The unsupervised pose estimation networks proposed in SfMLearner [24], Competitive Collaboration (CC) [8] and Monodepth2 [26] are used during these test. On the other hand, the pose of each frame are also computed in global manner using BA with COLMAP [1] and OpenMVG [12], but then 3 pose snippets from these results are gathered from all poses to compare these SfM pipelines against the deep learning-based methods.

According to Table 4.1, the relative pose errors are obtained better for all learning-based techniques, possibly due to utilizing the image pixel intesity for the whole image, while calculating relative poses, whereas the absolute trajectory errors are smaller for conventional pose estimators, which optimizes the results in an holistic way including 3D feature coordinates and poses by using BA.

Table 4.1: Mean and standard deviation of ATEs and RPEs of the chosen deep learning based and classical SfM methods calculated on Sequence-9 of KITTI Odometry Dataset are provided in the table. The upper part is DNN based methods and the lower part is classical SfM methods. (The best results are shown in bold letters)

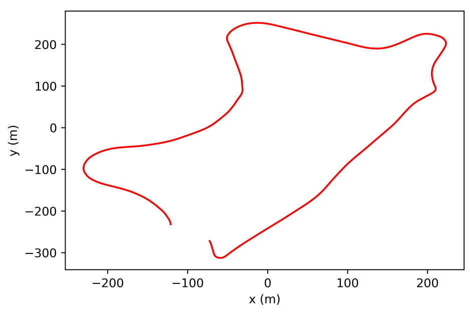
Method	ATE-Mean (m)	ATE-Std (m)	RPE-Mean (m)	RPE-Std (m)
SfMLearner [24]	1.1776	0.6059	1.5890	0.7410
CC [8]	1.0970	0.600	1.5026	0.7553
Monodepth2 [26]	1.1083	0.5881	1.5154	0.7449
COLMAP [1]	**0.0036**	**0.0025**	**0.0055**	**0.0040**
OpenMVG [12]	0.0079	0.0795	0.0430	0.1138

After computing the ATE and RPE from 3-frame length snippets from sequences, the full trajectory errors with metrics ATE, and RPE as before are computed by using the whole Sequence-9 from KITTI 2012 Odometry Dataset [2]. For that purpose, the relative pose between each consecutive frame in the Sequence-9 is computed with SfMLearner [24], CC [8] and Monodepth2. After this step, by setting the pose at $t = 0$ to identity, each frame's absolute pose is computed by multiplying the relative poses with each other. Conversely, the poses of each frame are directly computed by COLMAP and OpenMVG [12]. The comparisons are provided in Table 4.2. Moreover, the estimated trajectories and ground truth trajectory of Sequence-09 can be examined in Figure 4.1.

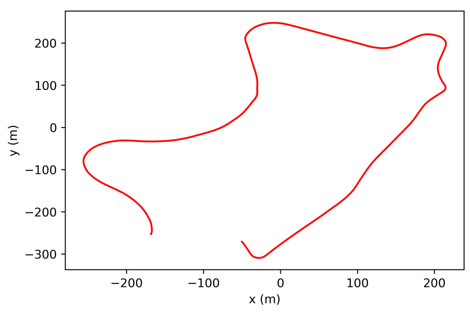
Table 4.2: Full trajectory pose estimation results for Sequence-09 from KITTI 2012 Odometry Dataset [2] are provided in the table. The upper part is DNN based methods and the lower part is classical SfM methods. (The best results are shown in bold letters)

Method	ATE (m)	RPE (m)
SfMLearner [24]	46.6107	1.4586
CC [8]	31.5470	**1.2828**
Monodepth2 [26]	37.3393	1.2868
COLMAP [1]	**4.4360**	1.7961
OpenMVG [12]	6.6950	1.8216

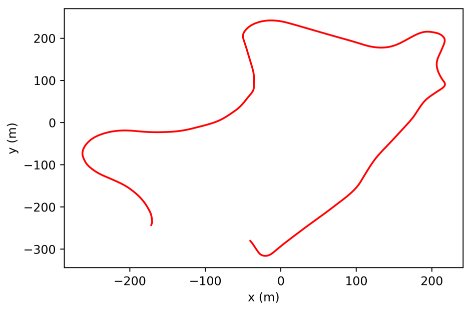
The full trajectory experiments are also performed on Sequence-0, -1, -3, -4, -5, -6, -7, -8, and -10 of the KITTI Odometry Dataset to compare Monodepth2 and COLMAP to generalize the results computed only on Sequence-9 of the KITTI Odometry Dataset and provided in Appendix-A as detailed results, plots of the estimated trajectories and ground truth trajectories . These results also confirm our initial conclusion that COLMAP is superior in absolute pose estimation; especially, with trajectories with loop closures. Based on the full results provided in Table A.1 and the resultant mean of the ATEs of RPEs of these sequences in Table 4.3, it is possible to observe that COLMAP is much superior during pose estimation with trajectories with several loop closures. On the other hand, if the trajectory does not include any loops, it should be emphasized that the pose estimation network of Monodepth2 provides a quite competitive performance with COLMAP.



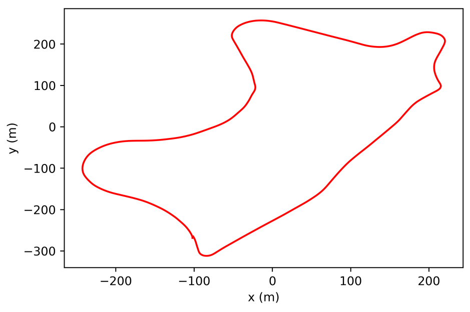
(a) Trajectory Estimate of SfmLearner [24]



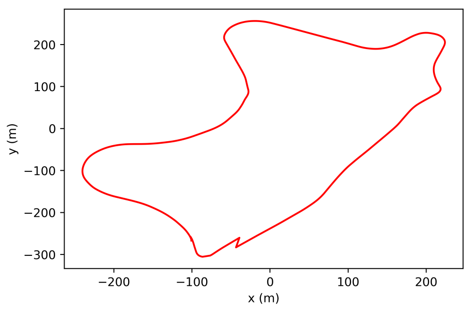
(b) Trajectory Estimate of CC [8]



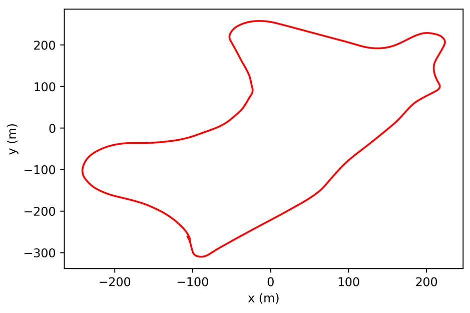
(c) Trajectory Estimate of Monodepth2



(d) Trajectory Estimate of COLMAP



(e) Trajectory Estimate of OpenMVG [12]



(f) Ground Truth Trajectory

Figure 4.1: The illustration of the estimated trajectory of Sequence-9 from KITTI 2012 Odometry Dataset [2] and the ground truth trajectory. For 2D illustration purposes only x and z component of the estimated poses are used which define the car's 2D movement in KITTI dataset [2] on Earth surface.

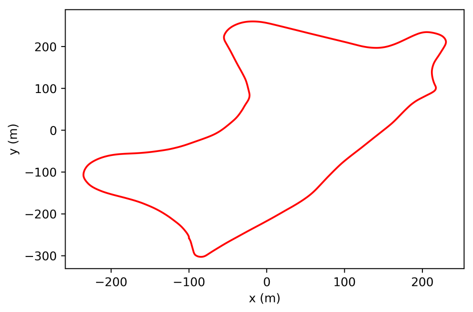
Table 4.3: Mean pose estimation errors for all sequences from KITTI 2012 Odometry Dataset [2]. The best metrics are shown in bold letters.

Method	ATE (m)	RPE (m)
Monodepth2 [26]	25.1558	1.1459
COLMAP [1]	**12.7905**	**0.8257**

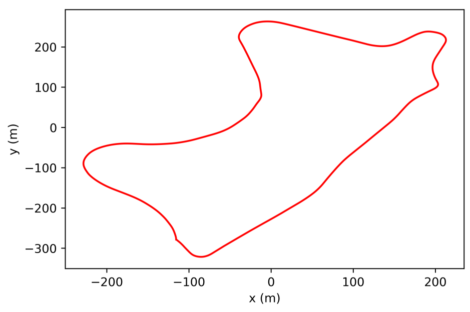
For improving the pose estimation errors of the unsupervised neural networks that are provided in Table 4.2, bundle adjustment (BA) is tested. Since Sequence-9 of the KITTI 2012 Odometry Dataset [2] is circular, the first frame and the last frame are coinciding frames. By exploiting this fact, the relative pose between the first and the last frame is also computed using the unsupervised pose estimation networks, then using all the computed relative poses between each consecutive frames and minimizing the mean square error between the pose difference computed from absolute poses of the frames and the estimated relative poses the absolute poses are computed using Nelder-Mead method [65]. The optimized results of the unsupervised pose estimation networks can be seen in Table 4.4. The results of the SfM methods COLMAP and OpenMVG [12] are kept the same, since these methods apply BA within their pipelines. as it can be observed in Table 4.4, ATE of Monodepth2 is improved from 37.3m to 12.5m. The improved trajectories of learned methods with BA can be examined in Figure 4.2, although such a loop closure is manually enforced to the solution; i.e. it is impractical.

Table 4.4: The resultant table for improved pose estimation on Sequence-9 of KITTI Odometry Dataset with bundle adjustment is provided where the upper part is DNN based methods and the lower part is classical SfM methods. The best metrics are shown in bold letters.

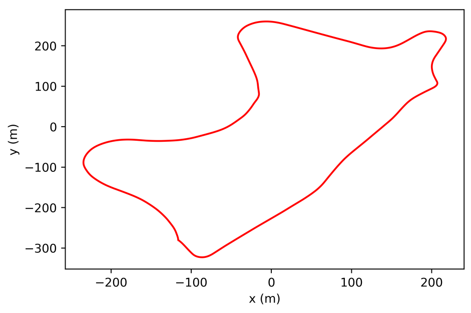
Method	ATE (m)	RPE (m)
SfMLearner [24]	10.0869	1.1510
CC [8]	14.7501	**1.0750**
Monodepth2 [26]	12.4692	1.0817
COLMAP [1]	**4.4360**	1.7961
OpenMVG [12]	6.6950	1.8216



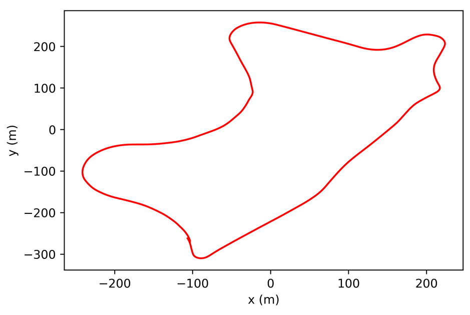
(a) Improved Trajectory Estimate of Sfm-Learner [24] with BA



(b) Improved Trajectory Estimate of CC [8] with BA



(c) Improved Trajectory Estimate of Monodepth2 with BA



(d) Ground Truth Trajectory

Figure 4.2: The improved trajectory estimates by applying the BA with closing the loop by estimating the relative pose between the first and the last frame of the Sequence-9 from KITTI 2012 Odometry Dataset [2].

4.2.1 Providing Relative Poses Estimated with Classical SfM to Monodepth2

The relative poses are computed with a classical SfM pipeline to assess the relative pose estimation effect on the quality of the depth estimation. COLMAP is selected for this task, since it is one of the most recent and best SfM pipelines available in the literature. Hence, it is widely used by the computer vision community also for a variety of tasks. Initially, each frame's pose belongs to KITTI Dataset [2] is attempted to compute by COLMAP. The frames whose pose can be estimated by COLMAP are listed and used during training of Monodepth2. In this configuration, the relative pose between two different frames can be easily obtained, since pose of each frame in the dataset is available; i.e. the pose estimation network of Monodepth2 is neither trained nor used. The relative pose calculation is obtained from the absolute pose, $\mathbf{P}_i$ as

$$\mathbf{P}_{rel} = \mathbf{P}_{i+\delta}^{-1}\mathbf{P}_i. \tag{4.1}$$

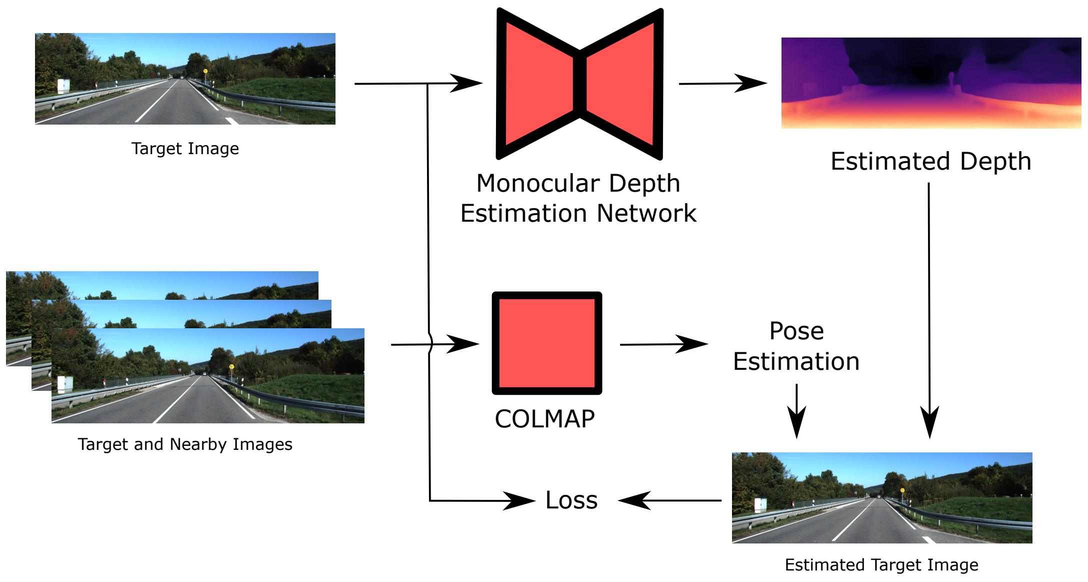


Figure 4.3: Schematic diagram of the training methodology. During training, relative poses are computed by using COLMAP instead of a pose estimator DNN.

This approach has a problem due to scaling ambiguity. KITTI Dataset [2] consists of different shots taken from different places and under varying circumstances, and COLMAP computes the 3D point cloud of these different shots independently. This situation results in inconsistent scales between different shots, since COLMAP can only reconstruct the scene up to an unknown scale. In order to tackle this issue, IMU measurements that are available in the KITTI Dataset [2] are utilized. The idea is to estimate the pose of N number of frames from each shot and compute the scale value to convert COLMAP's reconstructions to the real-world scale. For this purpose, the first 3 frames pose from each shot is estimated by multiplying the car's velocity with the time difference between frames, and the scale between this 3 estimated pose and COLMAP's estimated pose is calculated.

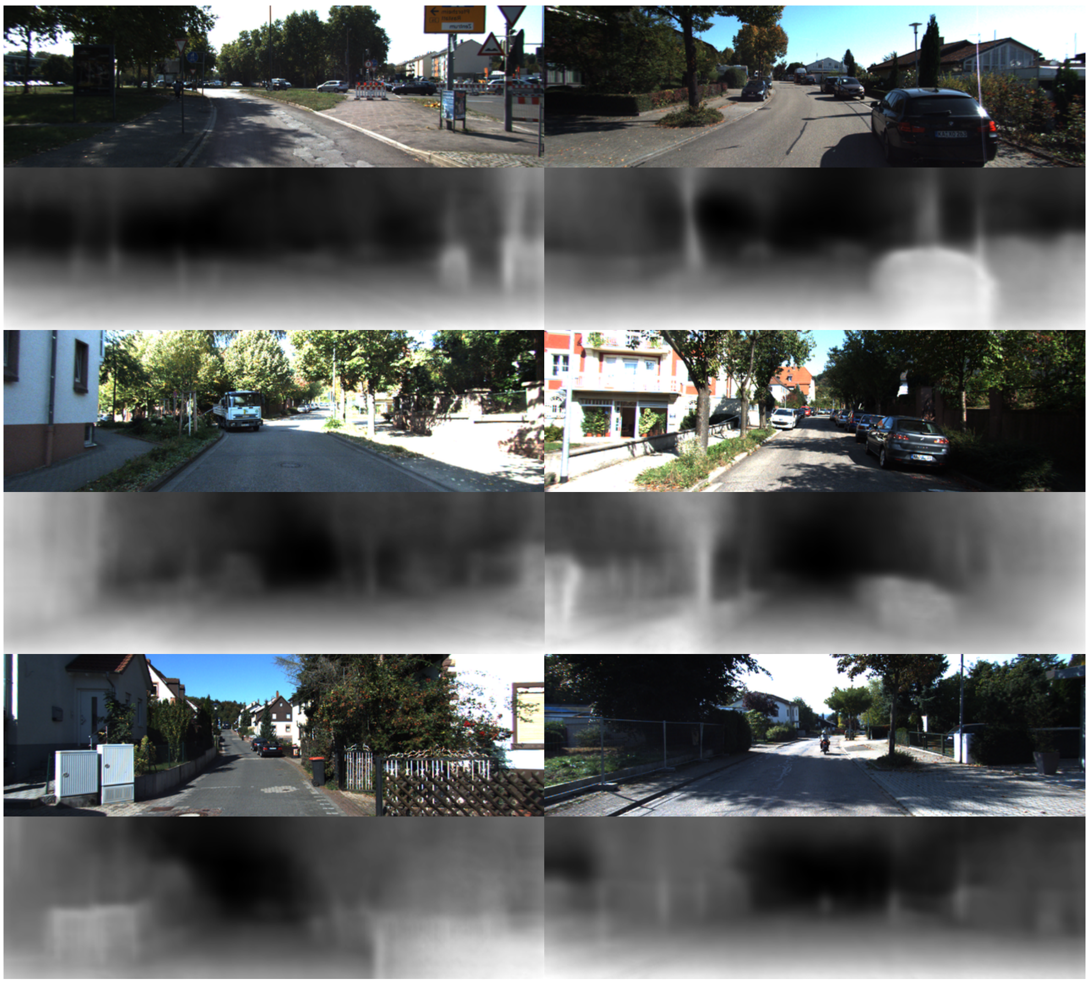


Figure 4.4: Disparity estimation results for Monodepth2 whose network is trained by using COLMAP pose estimates. (Bright regions indicate closer points)

A similar experiment with IMU data available in the KITTI Dataset is also performed. In this experiment, the relative pose between target and nearby views are estimated by using the absolute orientation and velocity information measured IMU. The relative translation is computed by multiplying the velocity with the time difference between frames. The depth estimation results are presented in Table 4.5, in which COLMAP

Pose and IMU Pose are behind the baseline. Various justifications could be possible for this case, which are tried to be explained in the next two paragraphs.

Since COLMAP matches the frames with classical feature descriptors, such as SIFT [66], computing poses between all frames in KITTI Dataset [2] is impossible, since sufficient number of matches cannot be obtained between every image pair. Therefore, only a portion of the dataset is reconstructed in terms of pose. Monodepth2 uses Eigen et al. [10] split for training and validation, which is also a portion of the KITTI Dataset [2]. The number of frames in the training split of Eigen et al. [10], which has a COLMAP pose, is 19,962, while the total number of frames in this dataset is 45,200. Using only these frames might lead to lose training data, so losing performance. On the other hand, the number frames in KITTI Dataset [2] which has a COLMAP pose is 50,010, which is to Eigen et al. [10] The network is trained with both dataset options and the best performed reported in Table 4.5 is achieved with the second dataset with 50,010 frames.

The architecture or hyperparameters used for training are incompetent for such an experiment. However, the network is trained with a different dataset from the rest of the network configurations in Table 4.5. Hence, comparing this network's performance with the other ones may not be a fair comparison since they are not trained with the same data. Some sample depth estimations are provided in Figure 4.4. Moreover, the same ground truth depth estimation samples are provided in Figure 4.5.

On the other hand, possible explanation about IMU Pose case is that IMU measurement could be noisy or erroneous.

Table 4.5: Experimental results of the depth estimation with SfM instead of a DNN in Monodepth2 trained and tested on KITTI Dataset are provided in the table. The best metrics are shown in bold letters.

Method	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	**0.1165**	**0.923**	**4.853**	**0.1932**	**0.8735**	**0.9583**	**0.9807**
COLMAP Pose	0.1382	1.029	5.416	0.2126	0.8232	0.9437	0.9780
IMU Pose	0.1638	1.134	5.811	0.2418	0.7580	0.9267	0.9730

Additional experiments are performed on KITTI Odometry Dataset [2] to verify the results found above. Monodepth2 is trained on this dataset with the default setting, the estimated poses of COLMAP, and its ground truth poses. Our findings are consistent with the last part, which uses external pose data such as COLMAP pose or ground truth pose does not provide any improvements. In these experiments, it is observed that some poses of specific sequences could be estimated very poorly by COLMAP if a suitable parameter setting is not performed. Therefore, our experiments in which Monodepth2 is trained with COLMAP poses could be distorted by the false pose estimation of the COLMAP. The depth estimation results of these experiments can be seen in Table 4.6.

Table 4.6: Experimental results of the depth estimation with SfM instead of a DNN in Monodepth2 trained on on KITTI Odometry Dataset and tested on KITTI Dataset. The best metrics are shown in bold letters.

Method	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	**0.1165**	**0.923**	**4.853**	**0.1932**	**0.8735**	**0.9583**	**0.9807**
Using GT Pose	0.1300	0.946	5.256	0.2097	0.8359	0.9474	0.9774
Using COLMAP Pose	0.1534	1.187	5.918	0.2350	0.7878	0.9307	0.9730

4.2.2 Discussions on Analysis 1

The following conclusions can be stated based on the pose estimation performances of both classical and DNN based methods:

- Classical pose estimation methods are more suitable for estimating the whole trajectory and their trajectory estimates have much smaller drift, since typical classical SfM pipelines use BA. This situation is especially apparent for the trajectories with loop closures. If the trajectory does not include any loops, DNN-based methods could also provide quite competitive performance against the classical methods.
- The pose estimation performance of DNN-based methods could also be improved by incorporating relevant constraints, such as loop closure.
- It is possible to further improve the trajectories estimated for DNN-based methods by applying BA that ends up with competitive trajectory estimates with SfM

methods.

- DNN-based methods provide a higher pose estimation performance based on the RPE metric. However, the inferior performance of ATE indicates that at specific frame instants, possibly due to unlearned motion of the camera, DNN-based methods make crucial mistakes, resulting in an inferior ATE.
- The best DNN-based pose estimation performance is provided CC method [8]. Motion segmentation and optical flow utilized in this method could be the reason behind this fact.
- Finally, it is interesting to observe the ineffectiveness of the groundtruth or COLMAP relative poses which are provided during training of the depth estimator network (Table 4.6). Although, pose estimator network burden is eliminated in such a scenario, the resulting depth estimator is unable to yield better results compared to the baseline Monodepth2, possibly due to quite accurate relative poses of Monodepth2 for intensity matching.

4.3 Analysis 2: The effects of input frame selection for Monodepth2

Every unsupervised depth estimation method discussed in Chapter 3 utilizes 3 consecutive frames from KITTI dataset [2] which are captured sequentially at time instants t_i, and $t_{i\pm1}$. However, direct odometry methods, such as D3VO [14] or DSO [13], and direct odometry based simultaneous localization and mapping (SLAM) systems, such as LSD-SLAM [16], perform a BA step over the selected frames from the whole trajectory, which are called "keypoint" frames. In this section, the opportunities for utilizing different frame combinations other the default input frame settings in the methods in the literature. The experiments performed in this section are

- Training Monodepth2 with wider baseline by using frames within a larger temporal neighborhood. In other words, Monodepth2 is trained with nearby views taken at $t_{i\pm\delta}$ where $\delta > 1$,
- Training Monodepth2 with two extra frames taken at $t_{i\pm\delta}$ where $\delta > 1$, while keeping the consecutive frames,

- Training Monodepth2 with four extra frames taken at $t_{i\pm5}$ and $t_{i\pm7}$.

A frame taken at $t_{i+\delta}$ will be denotes as I_δ in following parts of this section.

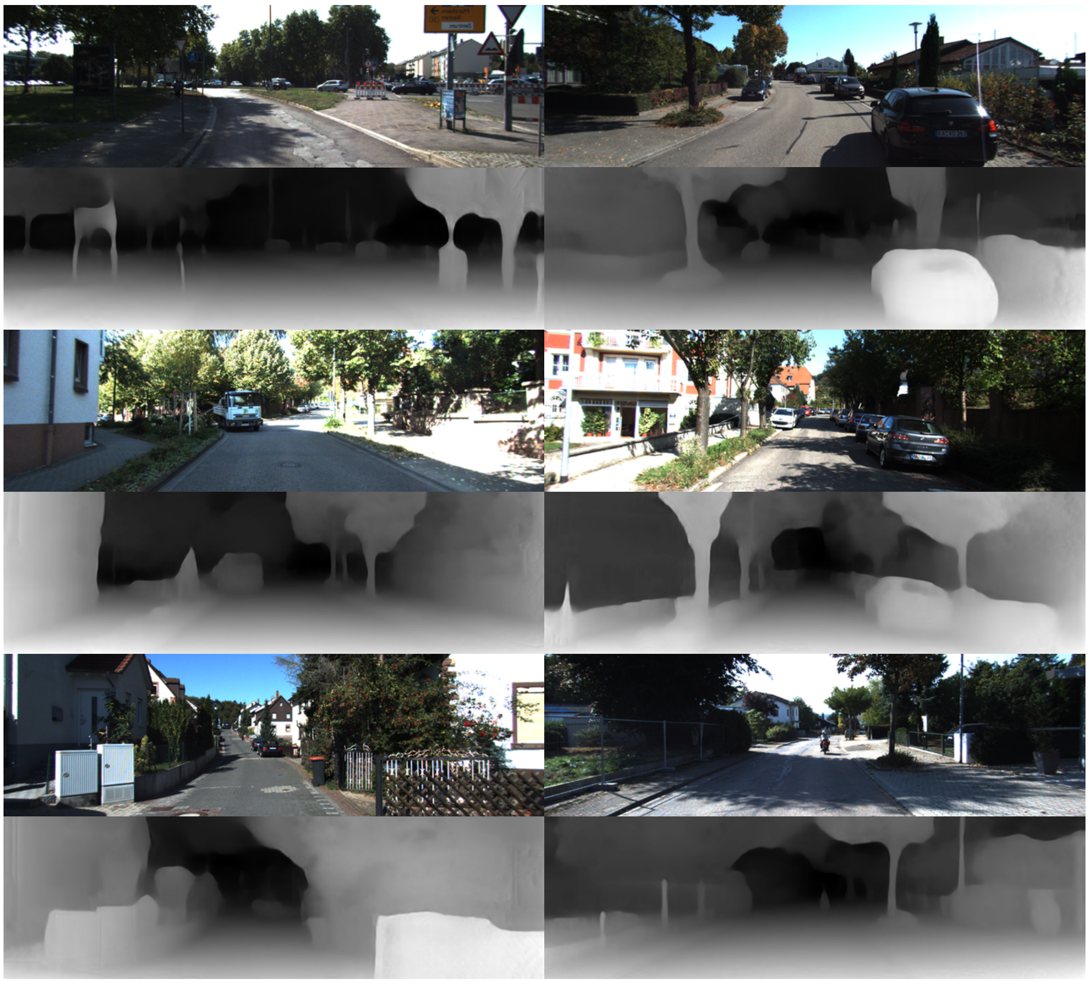


Figure 4.5: Some images and their depth estimation results, when Monodepth2 is trained with its default settings.

4.3.1 Training with Wider Baseline

The learning-based unsupervised depth and pose estimation methods in the literature utilizes one target and two consecutive frames during their training process. However, none of these methods utilizes wider baseline for training procedure and it is not clear, whether it is possible to train these networks with wider baseline between consecutive frames (e.g. captured from a faster car) compared to the one from KITTI Dataset.

Therefore, Monodepth2 is trained with a wider baseline in this section by simply skipping frames. Frame combinations from the temporal neighborhood, such as $I_{\pm 2}$, $I_{\pm 3}$, $I_{\pm 5}$, and $I_{\pm 7}$ are utilized in these experiments. The results are presented in Table 4.7, indicating a poor performance for depth estimation.

In order to improve the results, another approach is tested by training with spatially decimated frames to suppress the effect of large pixel displacement between target and nearby images. However, the resultant performance does not also improve significantly. This situation might be due to the fact that pose estimation can not produce "good" pose estimates, when the scene changes significantly compared to the target image, which is two frames different in about experiments with Monodepth2 on KITTI 2012 Dataset [2] which is taken with a 30 frame per second (FPS) stereo camera system mounted on a car. These experiments clearly show that removing the consecutive frames $I_{\pm 1}$ from the training process degrades the depth and pose estimation performance.

Pose estimation performance of the trained networks on sequnce 9 of KITTI Odometry Dataset is provided in Table 4.8. The pose estimation performance of the networks directly trained with $I_{\pm 2}$, $I_{\pm 3}$, $I_{\pm 5}$, and $I_{\pm 7}$ are much more poorer than the baseline as expected. However, the experiments in which pose estimator takes decimated input images can make competitive pose estimations compared to the baseline. In fact, the experiment with decimated $I_{\pm 2}$ frames surpasses the baseline in the ATE metric although baselie is still superior in RPE. The interesting point here is that this networks performs significantly worse in depth estimation than the baseline. The reason of poor depth estimation despite the competitive pose estimation could be the auto-masking procedure of Monodepth2. Higher errors compared to the low baseline could result in filtering a great portion of image which significantly harms the training of depth estimator. Occlusions in the images, brightness change between nearby and target views could be another reason.

4.3.2 Training with Two Extra Frames

A vast majority of unsupervised depth and pose estimation methods in the literature utilizes one target frame and two consecutive nearby frames. It is not unclear

Table 4.7: Depth estimation performance of wider baseline Monodepth2 is provided in the table. "dec." means that the input of the pose estimator is decimated by a ratio of frame difference. The best metrics are shown in bold letters.

Method	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	**0.1165**	**0.923**	**4.853**	**0.1932**	**0.8735**	**0.9583**	**0.9807**
Monodepth2 w/ only $I_{\pm 2}$	0.2080	3.852	6.229	0.2723	0.8075	0.9222	0.9580
Monodepth2 w/ only dec. $I_{\pm 2}$	0.1878	2.815	5.815	0.2574	0.8126	0.9291	0.9629
Monodepth2 w/ only $I_{\pm 3}$	0.3451	6.017	7.705	0.3837	0.6537	0.8309	0.9001
Monodepth2 w/ only dec. $I_{\pm 3}$	0.3161	4.998	7.308	0.3647	0.6696	0.8434	0.9097
Monodepth2 w/ only $I_{\pm 5}$	0.7180	18.217	15.377	0.7023	0.2570	0.4947	0.6832
Monodepth2 w/ only $I_{\pm 7}$	0.5462	13.168	14.160	0.6166	0.3663	0.6215	0.7692

Table 4.8: Pose estimation performance on the Sequence-9 of KITTI Odomery Dataset of wide baseline Monodepth2 experiments are shown in the table. The best metrics are shown in bold letters.

Method	ATE (m)	RPE (m)
Baseline	37.3393	**1.2868**
Monodepth2 w/ only $I_{\pm 2}$	244.621	2.25086
Monodepth2 w/ only dec. $I_{\pm 2}$	**26.379**	2.6732
Monodepth2 w/ only $I_{\pm 3}$	244.357	4.0996
Monodepth2 w/ only dec. $I_{\pm 3}$	44.727	3.801
Monodepth2 w/ only $I_{\pm 5}$	169.941	12.161
Monodepth2 w/ only $I_{\pm 7}$	252.515	26.6737

the effect of utilizing more than 3 frames in such methods. However, it should be remembered that providing five frames increases the size of the network; hence training becomes relatively more difficult.

This test aims to investigate the depth estimation performance of Monodepth2 trained with two more extra nearby views together with t_i, $t_{i\pm1}$ frames. Table 4.9 presents the results for training of Monodepth2 whose inputs are taken at t_i, $t_{i\pm1}$ and $t_{i\pm\delta}$, where $\delta = \{2, 5, 6, 7, 8, 9, 10\}$ is utilized at each row in Table 4.9. The best depth estimation results are attained with Monodepth2, whose 5 input frames belong to the time instants t_i, $t_{i\pm1}$ and $t_{i\pm7}$.

Table 4.9: Depth estimation performance of Monodepth2 trained with two extra frames are provided in the table. The best metrics are shown in bold letters.

Method	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	0.1165	0.923	4.853	0.1932	0.8735	0.9583	0.9807
Monodepth2 w/ $I_{\pm 2}$	0.1197	0.988	4.878	0.1973	0.8703	0.9570	0.9797
Monodepth2 w/ $I_{\pm 5}$	0.1140	0.884	4.794	0.1912	0.8784	0.9597	0.9812
Monodepth2 w/ $I_{\pm 6}$	0.1138	0.908	4.822	0.1911	0.8770	0.9599	0.9810
Monodepth2 w/ $I_{\pm 7}$	**0.1135**	**0.881**	**4.762**	**0.1902**	**0.8787**	**0.9604**	**0.9817**
Monodepth2 w/ $I_{\pm 8}$	0.1155	0.931	4.871	0.1921	0.8772	0.9592	0.9811
Monodepth2 w/ $I_{\pm 9}$	0.1146	0.939	4.857	0.1920	0.8779	0.9589	0.9810
Monodepth2 w/ $I_{\pm 10}$	0.1169	0.943	4.895	0.1931	0.8751	0.9592	0.9808

It can be argued that the underlying reason behind this interesting result is due to the auto-masking procedure in Monodepth2. The auto-masking is designed to filter out the pixels which fail during depth estimation, so that the noise on the weight updates of the depth and pose estimation networks is suppressed to a some extent. As a result of this masking strategy, some pixels of the target image is not utilized, while computing the loss and its gradient, which are used to update the depth and pose estimation networks.

During the experiments, it is observed that between 60 to 70% of all image pixels are filtered out by auto-masking, which means loss computation, i.e. training, is performed by only the remaining 30-40% of the all pixels of an image. After examining the training process of Monodepth2 with the extra frame taken at t_i, and $t_{i\pm\delta}$, this approach increases the percentage of the pixels used during the training process. This scenario is illustrated in Figure 4.8. Sample disparity estimations for $\delta = 5$ and $\delta = 7$ can be observed in Figure 4.6 and 4.7.

Regarding the pose estimation quality, based on Table 4.10, the best results are obtained with Monodepth2 baseline for ATE and RPE metrics rather than a modified version of Monodepth2 with extra frames, which raises an important question: How depth estimation performance can be improved by using using extra frames, while pose estimation becomes inferior with respect to baseline? One possible explanation could lie in the RPE results. Although, ATE results are significantly worse than the baseline, RPE results of the tested networks are quite close to the baseline performance. Since RPE is the metric for a relative pose estimator, these pose estimators could provide poses with sufficient accuracy to train depth estimator.

Table 4.10: Different frame training full trajectory pose estimation results on Seqeunce-9 of KITTI Odometry Dataset can be seen in the table. The best metrics are shown in bold letters.

Method	ATE (m)	RPE (m)
Monodepth2	**37.3393**	1.2868
Monodepth2 w/ extra $I_{\pm 5}$	137.983	1.1232
Monodepth2 w/ extra $I_{\pm 7}$	195.873	**1.1176**
Monodepth2 w/ extra $I_{\pm 10}$	56.193	1.1302

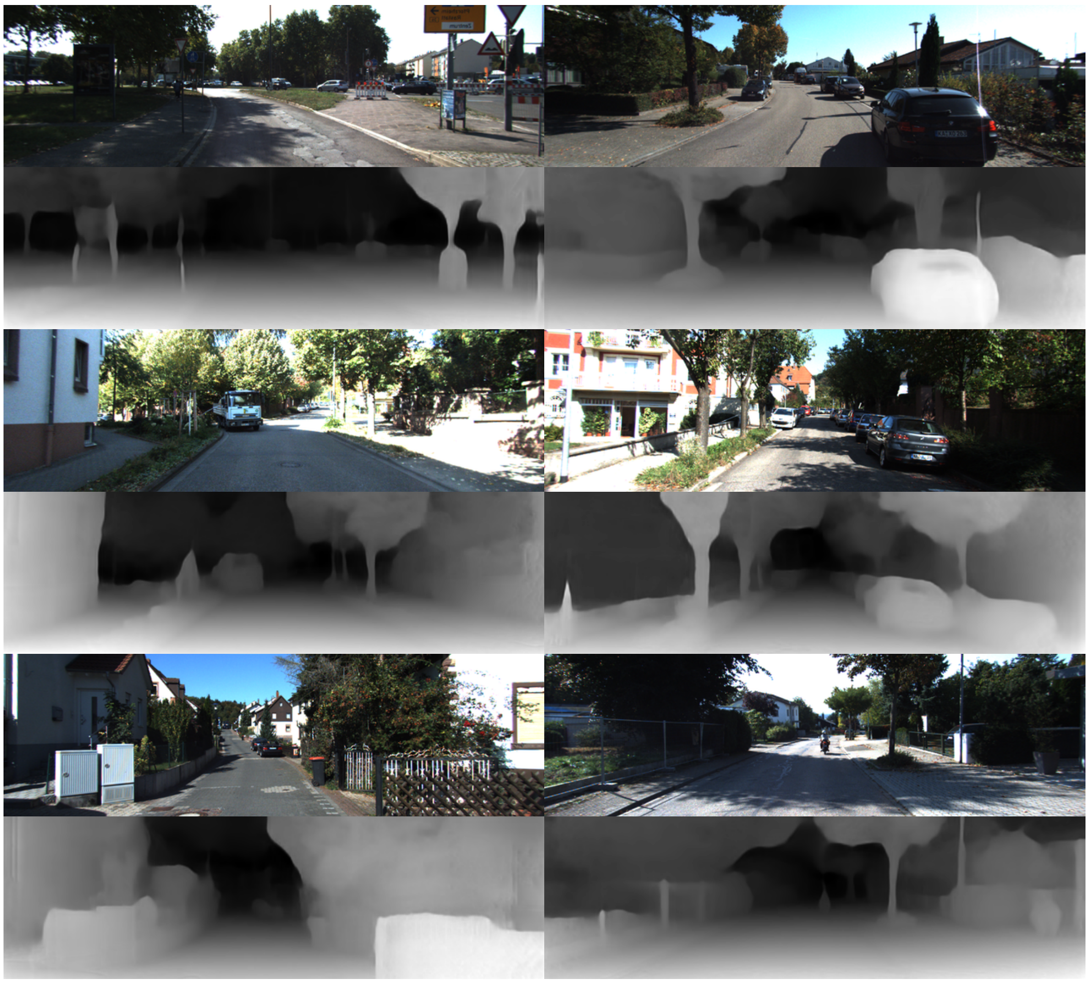


Figure 4.6: Some images and their depth estimation when Monodepth2 is trained with frames $I_{\pm 5}$ as extra frames.

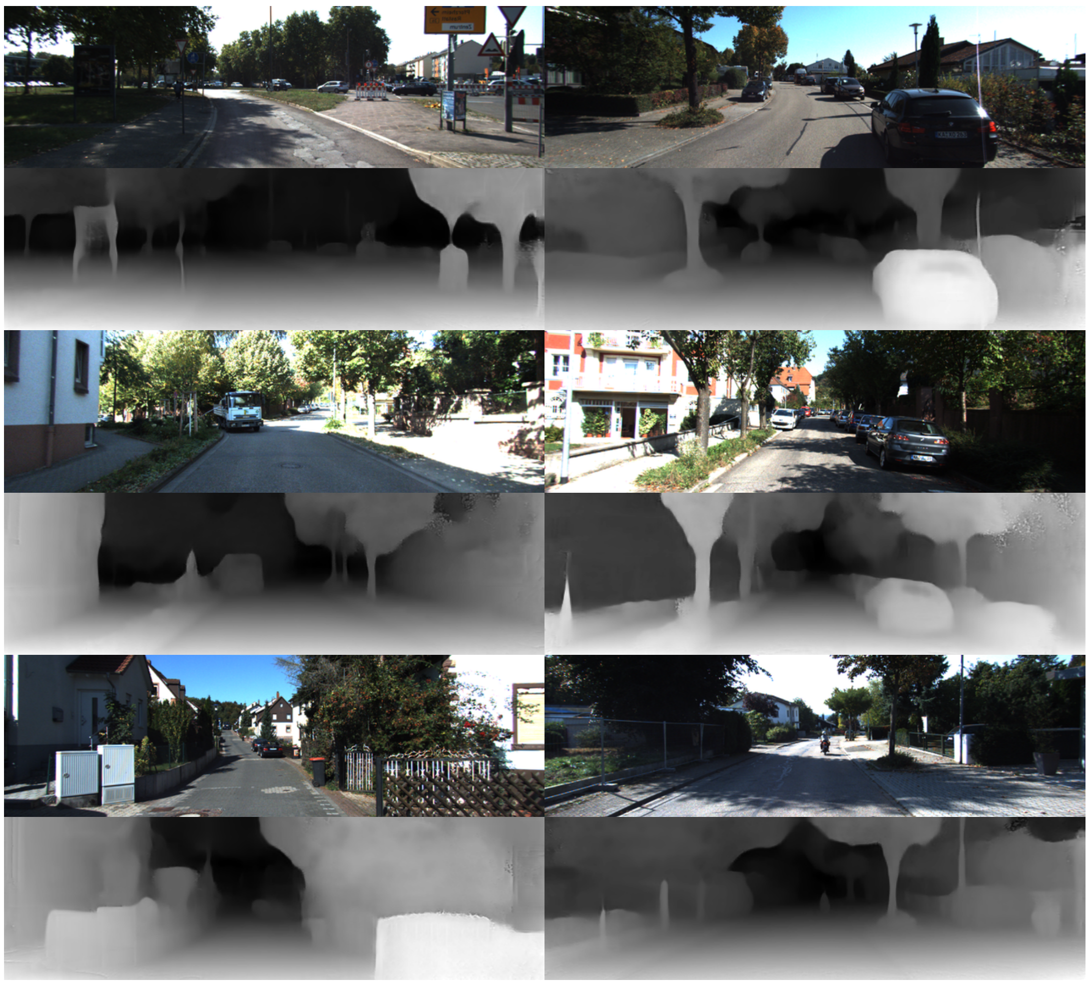


Figure 4.7: Some images and their depth estimation when Monodepth2 is trained with frames $I_{\pm 7}$ as extra frames.

After inspecting the predicted images from images $I_{\pm 1}$, and $I_{\pm \delta}$, it is also observed that the reconstructions from the earliest and latest images, i.e. $I_{\pm \delta}$, of the target image look quite similar to the original $I_{\pm \delta}$ images, whereas reconstructions from $I_{\pm 1}$ look very similar to target image as seen in Figure 4.9.

4.3.3 Training with Four Extra Frames

After observing the improvement in Monodepth2, depth estimation training due to two extra frames, the question remains whether adding more frames could further improve depth estimation results. In Table 4.11, the results of this test with extra frames $I_{\pm 5}$, and $I_{\pm 7}$ are provided. Moreover, the depth estimation results with extra

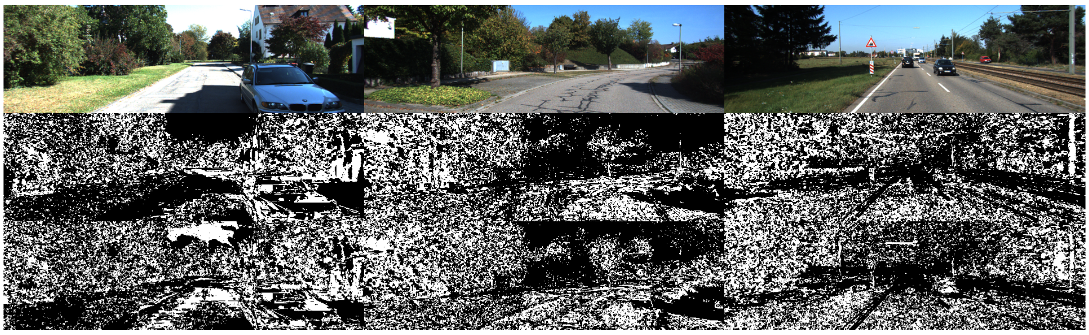


Figure 4.8: Illustration of the filtered pixels for the input images from training split proposed by Eigen et al. [10] in KITTI 2012 Dataset [2]. The input images are located in the row, the selected pixels with default setting of Monodepth2 are shown in the second row, and the selected pixels when frames taken at $t_{i\pm7}$ are added to training process are shown in the third row. The white pixels are selected for loss calculation and black pixels are filtered out and not being used for training.

frame $I_{\pm5}$, and $I_{\pm7}$ from Section 4.3.2 is also included in this table for completeness. In these tests $I_{\pm5}$, and $I_{\pm7}$ frames are included during training, since the best depth estimation results are attained with these frames while adding only two extra frames to the training procedure. Even though depth estimator performs significantly better in this case compared to the baseline, depth estimation is still behind the two extra frame cases as shown in Table 4.11.

Table 4.11: Depth estimation performance of Monodepth2 trained with four extra frames are provided in the table. The best metrics are shown in bold letters.

Method	Ab. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	0.1165	0.923	4.853	0.1932	0.8735	0.9583	0.9807
Monodepth2 w/ $I_{\pm5}$ & $I_{\pm7}$	0.1151	0.912	4.815	0.1929	0.8749	0.9585	0.9805
Monodepth2 w/ $I_{\pm5}$	0.1140	0.884	4.794	0.1912	0.8784	0.9597	0.9812
Monodepth2 w/ $I_{\pm7}$	**0.1135**	**0.881**	**4.762**	**0.1902**	**0.8787**	**0.9604**	**0.9817**

The reason for four extra frame method being outperformed by two extra frame method could be erroneous pose estimation of the pose estimator. The pose estimators' pose predictions for $I_{\pm5}$, and $I_{\pm7}$ is observed to be very close to the identity after a brief examination. In other words, $\mathbf{R} \approx \mathbf{I}$ and $\bar{t} \approx \bar{0}$ where $\mathbf{R}$, and $\bar{t}$ are the relative rotation matrix and translation vector between a target and a nearby image.

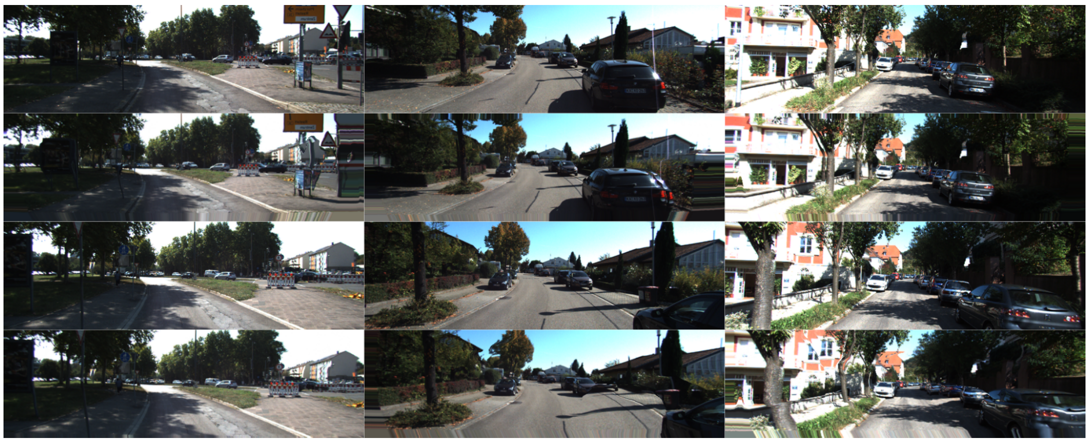


Figure 4.9: The first row shows the images I_0, the second row show I_0's prediction from I_{+1}, the third row shows the I_{+5} and the fourth row shows the prediction of I_0 from I_{+5}

This situation degrades the training of depth estimator network, since the network takes different nearby views estimated at the same place. This situation is less problematic in two extra frame case since the pixels chosen by auto-masking process of Monodepth2 is usually sky pixels which are far away from camera and it is not a bad assumption for these pixels the camera movement is close to zero.

4.3.4 Mimicking Bundle Adjustment by Adding Random Frames

Direct odometry methods, such as D3VO [14] or DSO [13] apply BA by selecting keyframes and computing the BA loss from all these selected keyframes. Almost all unsupervised depth and pose estimation methods discussed in Chapter 3 utilizes 3 frames from the KITTI Dataset [2] sequences, which are the target, immediate next and previous frames. With the intuition that D3VO [14] or DSO [13] uses multiple frames to improve the depth and pose estimations, and the results presented in Section 4.2 which indicates that BA can significantly improve the pose estimation, a network based on Monodepth2, which uses more than 3 frames, are implemented. The difference of this network from the previously discussed in Section 4.3 is that the index difference between the target and the extra frames are selected randomly between 10 and 30. Moreover, to estimate the pose between these extra frames and the target frame, a different approach is developed. In the first epoch of the training, relative

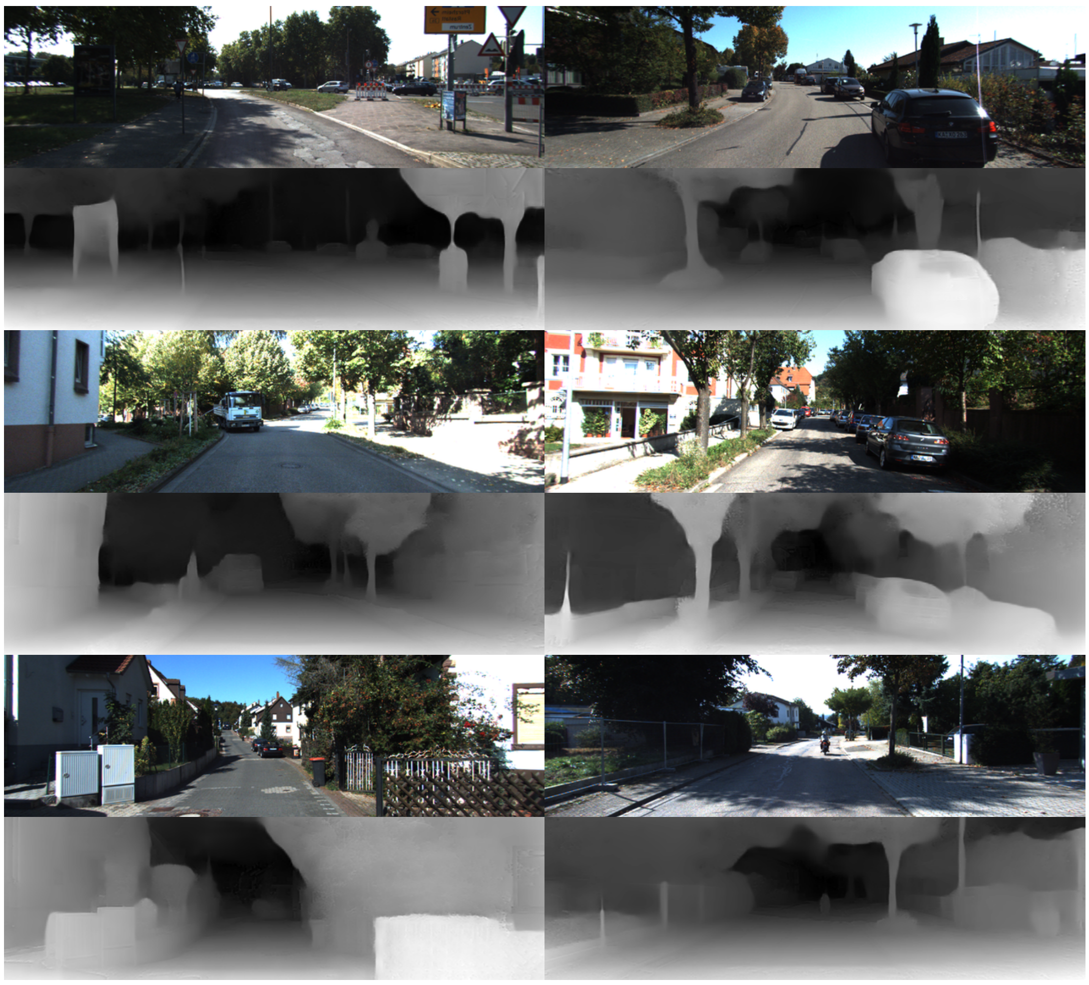


Figure 4.10: Some images and their depth estimation when Monodepth2 is trained BA mimicking method described in Section 4.3.4

poses between all frames are estimated and saved. In this first epoch, the extra frames are not used during training. In the second and other epochs extra frames are involved in training, and their pose is estimated by concatenating all the poses between these extra frames and the target frames, which could save the network from the distortion when the pose of the $I_{\pm 1}$ and $I_{\pm \delta}$ are estimated by the same network. The results can be found in Table 4.12 denoted as "BA mimicking."

Our experiments with "BA mimicking" reached a slightly worse depth estimation performance than the baseline model, which indicates that the extra information provided by frames further than ten does not boost the performance of Monodepth2, since the scene in these frames and the target frame are quite different. Moreover, the cameras' automatic exposure system for recording the KITTI Dataset [2] is another issue. The fundamental assumption for calculating the training loss of Monodepth2 is that the same 3D points' intensities on different images are the same, which is a legitimate hypothesis if the input used for training is I_{-1}, I_0, and I_1. However, this assumption breaks down, if frames $I_{\pm\delta}$ where $|\delta| > 2$ are involved in training, since the automatic exposure system of the camera adjusts the exposure on every new frame considering the lighting conditions of the scene.

Table 4.12: Depth estimation performance when Monodepth2 is trained with BA mimicking method and baseline are showed in the table. The best metrics are shown in bold letters.

Method	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	0.1165	0.923	4.853	**0.1932**	**0.8735**	0.958	**0.9807**
BA Mimicking	0.1176	**0.893**	**4.850**	0.1949	0.8714	**0.9588**	**0.9809**

4.3.5 Discussions on Analysis 2

The simulation results in Section 4.3 indicate adding certain frame combinations to Monodepth2 during training might improve depth estimation performance.

- Using the immediate temporal neighbors of a reference frame as input yields the best depth estimation for Monodepth2. Such learning-based systems seem to prefer small displacements between frames, in case of corresponding pixels satisfying constant intensity assumption.
- Adding two extra frames generally improves the depth estimation performance, as in Table 4.9 with the optimal result being achieved when the frame difference between the target and the extra view is equal to 7. Using wider baseline, size of displacements and brightness change in the images become more significant or the structure of the scene changes dramatically, preventing the extra information to become useful.
- Training Monodepth2 with four extra frames improves the depth estimation; nevertheless, the performance improvement is below the improvement of adding

two extra frames, possibly due to training inefficiencies of such a larger network.

- Noting the observed improvements for pose estimation due to BA utilization, a naive approach to mimic BA is also proposed that takes multiple frames with random time differences into a learned system. Depth estimation performance is competitive with the baseline method, although a clear improvement is not observed.

4.4 Analysis 3: Improving Depth and Ego-Motion Estimation with Semantic Segmentation

The fundamental idea of the Monodepth2 and the other unsupervised depth and pose estimation methods discussed in Chapter 3 is the assumption that the scene observed is static. Although this assumption is valid for most of the pixels of an image in the Eigen et al. [10] split of KITTI Dataset [2], it is not valid. The possible opportunities for utilizing semantic segmentation is investigated to tackle this issue. Experiments are performed with 4 different methods which are

- Method 1: Estimating pose and computing loss from only certain semantic classes.
- Method 2: Separating the losses of each semantic class and weighting them with learnable weights as illustrated in Figure 4.11,
- Method 3: Separating the poses of each semantic class, and weighting them with learnable weights, reconstructing the target image with the average pose, compute losses of each semantic class and weighting them with the same learnable weights as illustrated in Figure 4.12.

All images in the KITTI Dataset [2] are segmented with HRNets [67] trained on Cityscapes Dataset [20]. This method has been chosen since the it is oone of the best performing semantic segmentation methods in literature trained on Cityscapes Dataset. Moreover, Cityscapes Dataset is also similar to KITTI Dataset which could increase the accuracy of the final segmentation.

4.4.1 Method 1

The nature of objects found in an urban environment differs dramatically. Having the segmentation masks of all images of KITTI Dataset empowers us to compare the depth estimation performance on different semantic classes. These test are performed in two ways in this section. First, the errors of each semantic classes is computed separately in testing phase. In other words, different depth error metrics are computed for each semantic class during the testing process. This experiment is performed with pretrained Monodepth2 firstly and the resultant depth performance for some of the semantic classes in Table 4.13 with Training class "Baseline". Semantic classes with static and textured objects such as "Road", "Pavement", or "Building" classes are the best performing classes as expected. This experiment is repeated with different networks trained on the second part.

Second, the depth and pose estimation networks are trained by computing the loss from only certain semantic classes. Monodepth2 is trained only computing losses from "Road", "Building, and "Person" classes. The networks trained by only computing the loss of X class will be denoted as "X" network in the rest of this section. The resultant depth estimation performance for the networks trained on these classes can be seen in Table 4.13 with different Training classes other than "Baseline". Best depth estimation performance is achieved with the "Road" network amongst "Road", "Building, and "Person" networks. "Building" networks provides a similar performance to "Road" network; however, "Person" network performs dramatically poorer than the others. These results are expected since "Road" and "Building" networks are trained on static object pixels; however, "Person" network is trained pixels of moving objects. Performance decrease in "Road" and "Building" networks compared to baseline shows that computing loss from the only a portion of the image decreases the information fed to the networks during training. Although, the pixels providing loss during the training is known to be static, the lack of information compared to the baseline decreases depth estimation performance. The depth estimation performances for these networks are also inspected for single classes that they are trained on. The results also show that these networks are not superior than the baseline even in the classes that they are trained on, which can be seen in Table 4.13.

Monodepth2 is also trained by computing the loss from different combinations of different semantic classes as the results of the previous part indicate that one class is not enough for beating baseline. First, Monodepth2 is trained with losses computed on road, building, pavement, sky, outer wall, pole classes. road, building, pavement, outer wall, and pole classes the best performing classes in baseline networks as indicated in Table 4.13 with baseline training and different testing classes. Sky class is added to these classes since a great portion of images in the KITTI Dataset is belong to sky and it is also a static object. This experiment is denoted as "Best 6 Classes" in Table 4.13 Next, Monodepth2 is trained by computing the loss only from static classes, which are road, pavement, building, outer wall, fence, pole, traffic lamp, traffic sign, vegetation, terrain, and sky classes. This experiment is denoted as "Static Classes" in Table 4.13. Finally, Monodepth2 is trained by computing loss on a combination of static and dynamic classes, where the loss computed from static classes are weighted with 1.0 and the loss of dynamic classes are weighted with 0.3. The results of this experiment is denoted as "Static & Dynamic W." in Table 4.13.

Table 4.13: Depth estimation for results of method 1 is showed in the table. Training row indicates the semantic class combination that Monodepth2 is trained and the "Class" row indicates which semantic class on which the network is trained. The best metrics are shown in bold letters.

Training	Class	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Baseline	All	0.1165	0.923	4.853	0.1932	0.8735	0.9583	0.9807
Baseline	Road	**0.0701**	**0.204**	**1.991**	**0.0962**	**0.9629**	**0.9932**	**0.9977**
Baseline	Pavement	0.1026	0.468	3.032	0.1379	0.8947	0.9806	0.9943
Baseline	Building	0.1681	1.731	6.723	0.2314	0.7430	0.9204	0.9744
Baseline	Outer Wall	0.1745	1.882	6.940	0.2434	0.7546	0.9178	0.9658
Baseline	Sky	0.2533	4.336	9.489	0.3161	0.6397	0.8450	0.9261
Baseline	People	0.3001	6.584	10.544	0.3530	0.6157	0.8095	0.9000
Road	Road	0.0811	0.252	2.167	0.1097	0.9460	0.9892	0.9958
Building	Building	0.2213	2.367	7.935	0.2882	0.6216	0.8676	0.9533
People	People	0.4775	9.703	18.482	0.7785	0.2134	0.4090	0.5606
Road	All	0.1394	1.145	5.401	0.2161	0.8304	0.9412	0.9747
Building	All	0.1744	1.396	5.797	0.2358	0.7604	0.9330	0.9756
Person	All	0.4429	4.757	12.083	0.5876	0.3033	0.5608	0.7662
Best 6 Classes	All	0.1249	1.017	4.911	0.1967	0.8628	0.9548	0.9798
Static Classes	All	0.1259	0.946	4.818	0.1975	0.8582	0.9546	0.9795
Static & Dynamic W.	All	0.1350	1.227	4.982	0.2066	0.8519	0.9511	0.9774

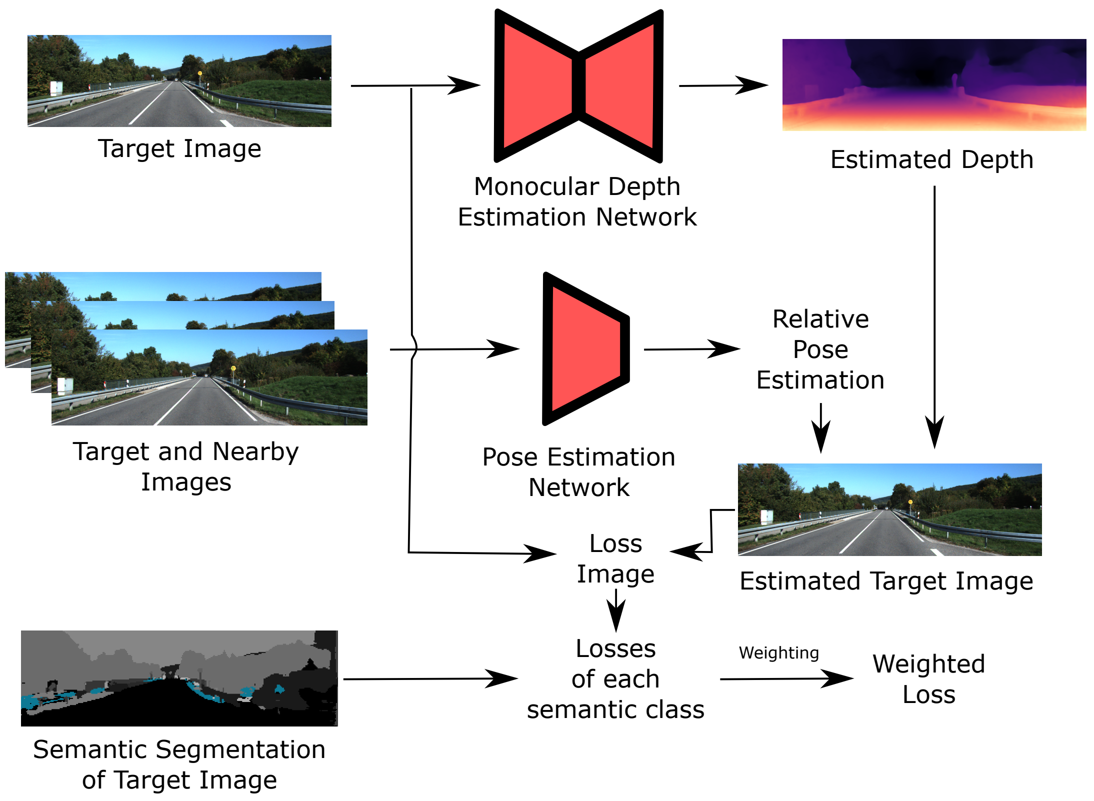


Figure 4.11: A schematic of semantic weighting process of losses of different semantic classes.

4.4.2 Method 2

A pretrained version High-resolution networks (HRNets) for Semantic Segmentation [67] trained on Cityscapes Dataset [20] is adapted for semantic segmentation since Cityscapes [20] is a similar dataset to KITTI 2012 Dataset [2], which are both autonomous driving datasets.

In this method, only semantic loss weighting is implemented, and the Monodepth2 is trained with this loss configuration. In detail, the average losses corresponding to each semantic class is computed. Then, these losses are averaged with learnable weights. This method is illustrated in Figure 4.11

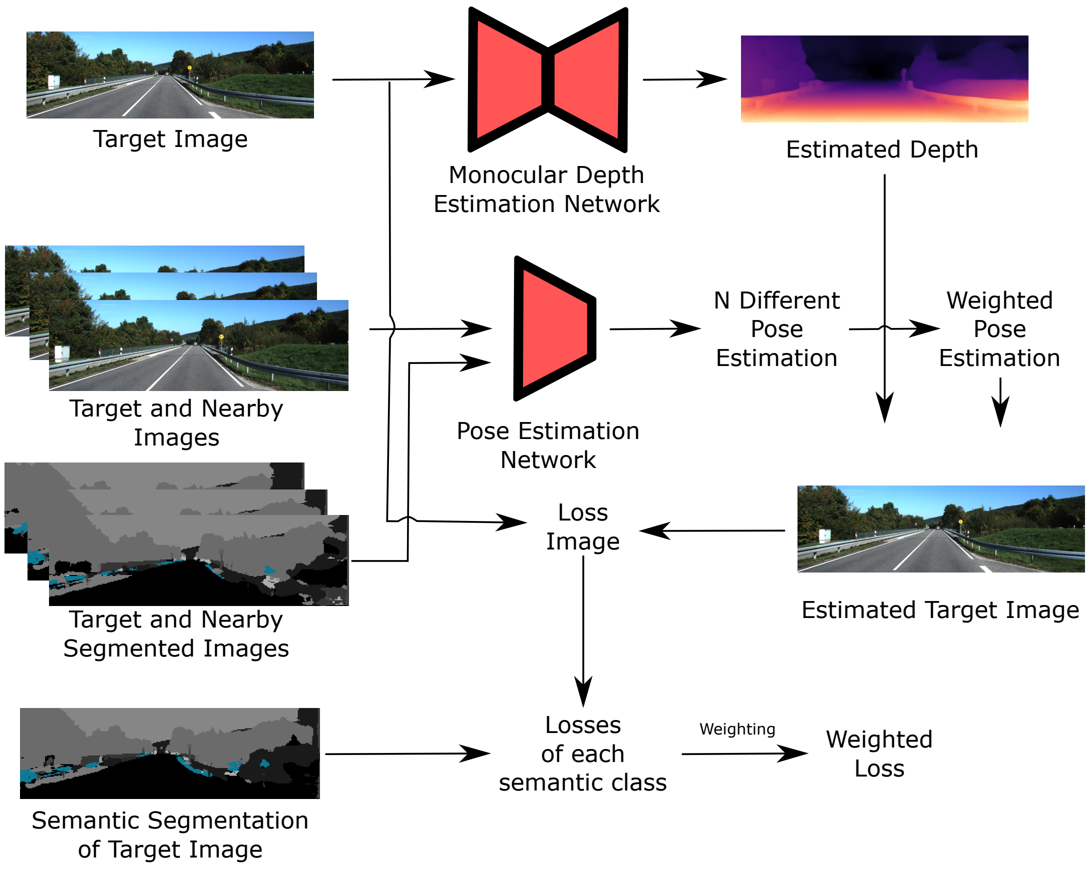

Target Image
Monocular Depth Estimation Network
Estimated Depth
Target and Nearby Images
Pose Estimation Network
N Different Pose Estimation
Weighted Pose Estimation
Target and Nearby Segmented Images
Loss Image
Estimated Target Image
Semantic Segmentation of Target Image
Losses of each semantic class
Weighting
Weighted Loss

Figure 4.12: A schematic of semantic weighting process of poses of different semantic classes.

4.4.3 Method 3

Our idea here is to compute different poses for each semantic class included in the training of HRNets [67], so the semantic masks, which include 19 different semantic classes, are also fed to the pose estimation network of Monodepth2, and 19 different poses are computed at the output of the pose estimation network. The final translation is computed by a weighted sum of the available translation vectors with weights produced by a Softmax operation to ensure that all weights are positive and their sum is equal to 1. The same is also used for averaging the rotations; however, averaging rotations is not trivial as in the translation case. For that purpose, quaternion representation of rotation is chosen since weighted averaging or interpolation of rotation is more reliable when using quaternions rather than using Euler angles or axis angle (Lie-algebra) representations. The method proposed in [68] is adapted to compute the average of the quaternions. First, N weighted quaternion vectors are stacked in a $4 \times N$ $\mathbf{Q}$ matrix as

$$\mathbf{Q} = \begin{bmatrix} w_1\bar{\mathbf{q}}_1 & w_2\bar{\mathbf{q}}_2 & \dots & w_N\bar{\mathbf{q}}_N \end{bmatrix}. \tag{4.2}$$

Then, the optimal average rotation expressed as quaternion is the eigenvector of $\mathbf{Q}\mathbf{Q}^T$ corresponding to its largest eigenvalue. The weighted average translation vector can be found as the linear combination of the estimated N translation vectors as

$$t_{avg} = w_1 t_1 + w_2 t_2 + \cdots + w_N t_N. \tag{4.3}$$

After computing the average pose out of the pose network's estimations corresponding to the different classes, the losses corresponding to the different semantic classes are also computed. Their weighted average is then computed using the softmax output of the same weights used for computing the average pose. $L_1, L_2, \dots, L_N$ denotes the mean losses corresponding to each semantic class. Then, the final loss becomes

$$L = w_1 L_1 + w_2 L_2 + \cdots + w_N L_N. \tag{4.4}$$

The process of weighted averaging the different poses estimated for other different classes is illustrated in Figure 4.12. The results of methods 2 and 3 can be seen in Table 4.14

Table 4.14: Depth estimation performance of Monodepth2 trained with method 2 and 3 discussed in Section 4.4. The best metrics are shown in bold letters.

Method	Loss W.	Pose W.	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2			0.1165	0.923	4.853	0.1932	0.8735	0.9583	0.9807
Method 2	✓		**0.1152**	**0.890**	**4.800**	**0.1911**	**0.8754**	**0.9597**	**0.9812**
Method 3	✓	✓	0.1219	0.942	4.897	0.1965	0.8649	0.9566	0.9804

4.4.4 Discussion of Analysis 3

To conclude, a potential is observed in our methods, leveraging semantic segmentation. Semantic segmentation is critical in computing losses and training the networks with back-propagation since it can discriminate the static and dynamic parts of a scene. Our main conclusion is as follows

Table 4.15: Depth estimation performance of all methods discussed in this chapter trained on KITTI Dataset. The best metrics are shown in bold letters.

Method	Abs. Rel.	Sq. Rel.	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	0.1165	0.923	4.853	0.1932	0.8735	0.9583	0.9807
Extra $I_{\pm 2}$	0.1197	0.988	4.878	0.1973	0.8703	0.9570	0.9797
Extra $I_{\pm 5}$	0.1140	0.884	4.794	0.1912	0.8784	0.9597	0.9812
Extra $I_{\pm 6}$	0.1138	0.908	4.822	0.1911	0.8770	0.9599	0.9810
Extra $I_{\pm 7}$	**0.1135**	**0.881**	**4.762**	**0.1902**	**0.8787**	**0.9604**	**0.9817**
Extra $I_{\pm 8}$	0.1155	0.931	4.871	0.1921	0.8772	0.9592	0.9811
Extra $I_{\pm 9}$	0.1146	0.939	4.857	0.1920	0.8779	0.9589	0.9810
Extra $I_{\pm 10}$	0.1169	0.943	4.895	0.1931	0.8751	0.9592	0.9808
Extra $I_{\pm 5}$ and $I_{\pm 7}$	0.1151	0.912	4.815	0.1929	0.8749	0.9585	0.9805
Method 2	0.1152	0.890	4.800	0.1911	0.8754	0.9597	0.9812
Method 3	0.1219	0.942	4.897	0.1965	0.8649	0.9566	0.9804
BA Mimicking	0.1176	0.893	4.850	0.1949	0.8714	0.9588	0.9809
COLMAP Pose	0.1382	1.029	5.416	0.2126	0.8232	0.9437	0.9780
IMU Pose	0.1638	1.134	5.811	0.2418	0.7580	0.9267	0.9730

- Our method 1 experiments showed that depth of static classes could be more successfully estimated than the dynamic classes as expected.
- Training Monodepth2 on single classes does not improve the depth estimation performance since the information fed to the network significantly decreases in these cases.
- Training Monodepth2 with different combinations of the semantic classes performs significantly better than the single class training methods. A careful weighting scheme has the potential to improve the depth estimation.
- Separating the loss according to the semantic class pixels belonging and weighting them achieved comparable results with the state-of-the-art method in depth estimation methods,
- Estimating different poses, averaging them, and reconstructing the target image does not improve depth estimation performance. This situation can be observed in method 3. The depth estimation performances of all depth estimation methods discussed in this Chapter are provided in Table 4.15. The best performing method in this table is Monodepth2 trained with two extra frames $I_{\pm 7}$.

4.5 Conclusions on Chapter 4

In this chapter, a number of experiments and methods are proposed to develop a greater understanding in unsupervised depth and pose estimation. Monodepth2 is chosen as the baseline in all of our experiments since it is one of the best performing and common unsupervised depth and pose estimation method in the literature. The findings of this chapter can be summarized as

- Classical methods such as COLMAP can provide a better pose estimation performance compared to the unsupervised DNN-based pose estimation methods. This situation is more apparent with trajectories with loops; however, unsupervised DNN-based pose estimation methods can provide a competitive performance with trajectories without any loops.
- Training Monodepth2 with different input frame combinations reveals important information. First training the Monodepth2 with wider baseline than the default setting (or without frames $I_{\pm 1}$) significantly degrades the depth estimation. Next, training the Monodepth2 with two extra nearby views along the default consecutive frames improves depth estimation since extra information is provided by these pixels. Finally, training Monodepth2 with four extra frames also improves the depth estimation; however, this improvement is behind the improvement of adding two extra frames.
- An image contains different semantic classes with different properties and it is observed that the depth of some classes can be estimated more easily such as classes with static objects and texture such as "Road" or "Building" classes. Next, training Monodepth2 with different semantic class and weighting combinations has the potential to improve depth estimation with a carefully design. However, estimating different poses for each semantic class is not a promising approach and it also increases the model complexity.

CHAPTER 5

CONCLUSIONS

In this study, different aspects in training of unsupervised depth and pose estimation networks are investigated. Initially, the pose estimation of classical SfM methods utilizing BA and unsupervised learning based method are compared. The results showed that classical SfM methods are superior in pose estimation, especially in the trajectories with loops. Next, the fact that the pose estimation performance of unsupervised learning based can be improved by incorporating meaningful constraints.

Second, the effect of frame selection is investigated thoroughly and it is revealed that the frame selection in this networks are crucial for the performance of the networks. Increasing the baseline between the target and the nearby views, significantly distort the depth and pose estimation. Sufficiently small baseline should be used for a successful training of depth and pose estimators. Next, adding extra frames to the training enhances the depth estimation performance in most cases since extra information is provided to the networks with this frames. Especially, it is observed that the pixels of the extra frames corresponding to the distant parts of the scene such as sky contribute to the training process since the filtering method used in Monodepth2 filters the pixels with unsuccessful depth estimation and the extra frames decreases the number of filtered pixels. In addition, our experiments with excluding the previous and next frames of the target frame from the training process showed that these frame are vital for depth and pose estimation. The performance of both depth and pose estimation degraded dramatically when the frame difference between the target and the nearby frames are increased.

Final experiments discussed in Chapter 4 are based on leveraging semantic segmentation to enhance the unsupervised networks by utilizing the semantic information.

These experiments can be divided into three categories as training the networks from only certain semantic class combinations, training the networks from a weighted combination of the losses of each semantic class, and training the networks from a weighted combination of the losses of each semantic and pose estimation of each semantic class. These experiments are denoted as method 1, method 2, method 3 in Section 4.4.

Method 1 shows that training Monodepth2 by using the loss of certain semantic classes does not improve depth estimation compared to baseline. The approaches are based on grouping the semantic classes regarding their characteristics and weighing them to compute loss and train the networks to seem promising. They provide competitive depth estimation compared to the baseline. The reason behind this result could be some classes contain moving objects by their nature (e.g., car, truck, person), which violates the rigid scene assumption utilized in the unsupervised depth and pose estimation methods in the literature.

Method 2 extends the approach of the weighting of certain semantic classes to another level. The average loss corresponding to each semantic class is computed in Method 2. Next, a weighted average of each loss is computed to come up with the final loss, where the weight of each semantic class is learnable. Similar to the results attained in method 1, competitive results compared to the baseline are attained with method 2, and it is promising.

Method 3 extends the method 2 further. A different pose is estimated for each semantic class, and the weighted average of these poses are used to reconstruct the target image. The loss is computed by averaging the loss of each semantic class with weights. The weights used for averaging the pose and the loss are the same, and they are learnable. Method 3 performs poorly in depth estimation compared to the baseline and other methods proposed in this study. The complexity of the networks is also increased significantly in this case. Depth estimation performance of method 1 can be seen in Table 4.13 and the depth estimation performance of method 2 and method 3 can be seen in Table 4.14. Method 1 and method 2 seem promising for possible future works.

REFERENCES

[1] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104–4113, 2016.

[2] 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*, pp. 3354–3361, IEEE, 2012.

[3] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[4] 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*, pp. 270–279, 2017.

[5] V. Guizilini, R. Ambrus, S. Pillai, and A. Gaidon, "Packnet-sfm: 3d packing for self-supervised monocular depth estimation," *arXiv preprint arXiv:1905.02693*, 2019.

[6] C. Tang and P. Tan, "Ba-net: Dense bundle adjustment network," *arXiv preprint arXiv:1806.04807*, 2018.

[7] Y. Shi, J. Zhu, Y. Fang, K. Lien, and J. Gu, "Self-supervised learning of depth and ego-motion with differentiable bundle adjustment," *arXiv preprint arXiv:1909.13163*, 2019.

[8] 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 Conference on Computer Vision and Pattern Recognition*, pp. 12240–12249, 2019.

[9] V. Guizilini, R. Hou, J. Li, R. Ambrus, and A. Gaidon, "Semantically-guided

representation learning for self-supervised monocular depth," *arXiv preprint arXiv:2002.12319*, 2020.

[10] 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*, pp. 2366–2374, 2014.

[11] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International journal of computer vision*, vol. 80, no. 2, pp. 189–210, 2008.

[12] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, "Openmvg: Open multiple view geometry," in *International Workshop on Reproducible Research in Pattern Recognition*, pp. 60–74, Springer, 2016.

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

[14] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," *arXiv preprint arXiv:2003.01060*, 2020.

[15] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[16] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*, pp. 834–849, Springer, 2014.

[17] "Skydio 2™ and x2™." Skydio Inc.

[18] "Microsoft hololens 2." Microsoft Corp.

[19] "Vive cosmos." Vive Enterprise.

[20] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic ur-

ban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[21] "Waymo one." Waymo LLC.

[22] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*, pp. 239–248, IEEE, 2016.

[23] 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*, pp. 2002–2011, 2018.

[24] 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*, pp. 1851–1858, 2017.

[25] 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," *arXiv preprint arXiv:1810.06125*, 2018.

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

[27] C. Wu *et al.*, "Visualsfm: A visual structure from motion system," 2011.

[28] 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 Conference on Computer Vision and Pattern Recognition*, pp. 4521–4530, 2019.

[29] Z. Li and N. Snavely, "Megadepth: Learning single-view depth prediction from internet photos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2041–2050, 2018.

[30] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe, "Understanding the limitations of cnn-based absolute camera pose regression," in *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pp. 3302–3312, 2019.

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

[32] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy: Fixefficientnet," *arXiv preprint arXiv:2003.08237*, 2020.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[35] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *arXiv preprint arXiv:1911.09070*, 2019.

[36] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[37] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[38] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[39] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[40] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-scnn: Gated shape cnns for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5229–5238, 2019.

[41] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[42] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

[43] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[44] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3703–3712, 2019.

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

[46] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8934–8943, 2018.

[47] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 224–236, 2018.

[48] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint detection and description of local features," *arXiv preprint arXiv:1905.03561*, 2019.

[49] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *Advances in Neural Information Processing Systems*, pp. 4826–4837, 2017.

[50] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, "Unprocessing images for learned raw denoising," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11036–11045, 2019.

[51] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9446–9454, 2018.

[52] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH 2005 Papers*, pp. 577–584, 2005.

[53] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 824–840, 2008.

[54] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[55] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4040–4048, 2016.

[56] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883, 2016.

[57] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[59] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[60] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[61] H. Su, V. Jampani, D. Sun, O. Gallo, E. Learned-Miller, and J. Kautz, "Pixel-adaptive convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11166–11175, 2019.

[62] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

[63] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[65] F. Gao and L. Han, "Implementing the nelder-mead simplex algorithm with adaptive parameters," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 259–277, 2012.

[66] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[67] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

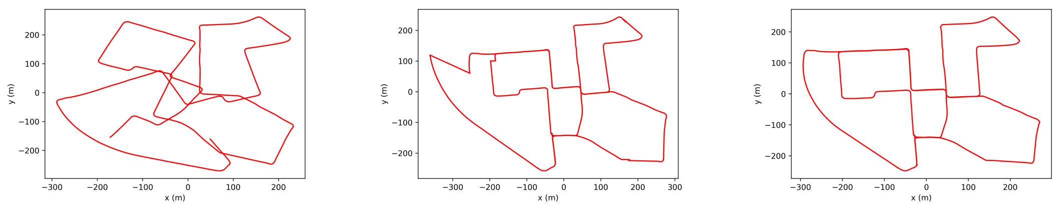
[68] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

APPENDIX A

COMPREHENSIVE RESULTS FOR COMPARISON BETWEEN CLASSICAL AND DNN-BASED METHODS FOR POSE ESTIMATION (ANALYSIS 1)

In this part of the thesis, a detailed comparison of Monodepth2 [26] and COLMAP [1] with respect to their pose estimation performance is provided. In Section 4.2, the comparison is presented only on Sequence-9 of the KITTI Odometry Dataset [2]. In this section, the pose performance of Monodepth2 [26] and COLMAP [1] is compared on Sequence-1,-3,-4,-5,-6,-7,-8,-9 and -10. Sequence-2 and -8 are excluded, since COLMAP [1] cannot estimate these trajectories.

It can be observed that COLMAP [1] is superior in pose estimation in almost all sequences, except sequence 7. However, a careful parameter setting for COLMAP should be performed for each sequence by regarding the trajectories' properties. The resultant pose estimation performance for COLMAP and Monodepth for these sequences can be observed in Table A.1. Moreover, the plots of the trajectories can be examined in the following figures.

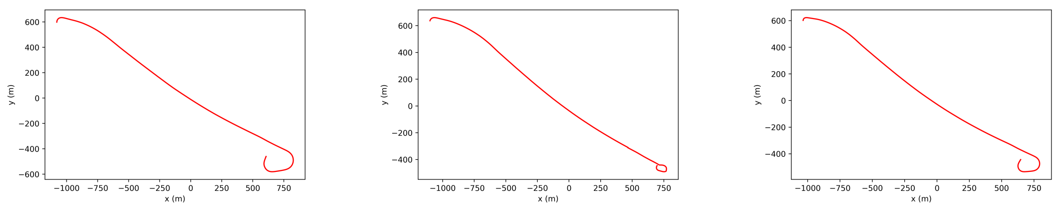


(a) Monodepth2 Estimation (b) COLMAP Estimation (c) Ground Truth

Figure A.1: Trajectory estimations with Monodepth2 and COLMAP on Sequence-0 of KITTI Odometry Dataset

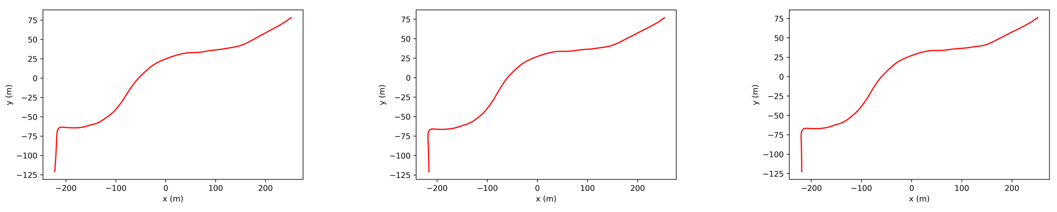
Table A.1: Pose estimation performance on different sequences of KITTI Odometry Dataset [2]. The best metrics are shown in bold letters.

Sequences	Method	ATE (m)	RPE (m)
00	Monodepth2	85.9410	0.8944
00	COLMAP	18.1722	0.8484
01	Monodepth2	33.4207	2.3344
01	COLMAP	37.7809	0.5647
03	Monodepth2	3.1318	0.7626
03	COLMAP	1.0713	0.02922
04	Monodepth2	2.0076	1.5035
04	COLMAP	0.3687	1.4125
05	Monodepth2	26.8637	0.8815
05	COLMAP	10.8126	0.8327
06	Monodepth2	9.9668	1.1964
06	COLMAP	3.4880	1.1277
07	Monodepth2	11.3271	0.7339
07	COLMAP	35.625	0.7096
09	Monodepth2	37.3393	1.1405
09	COLMAP	4.4360	1.0857
10	Monodepth2	16.4042	0.8658
10	COLMAP	3.3588	0.8209



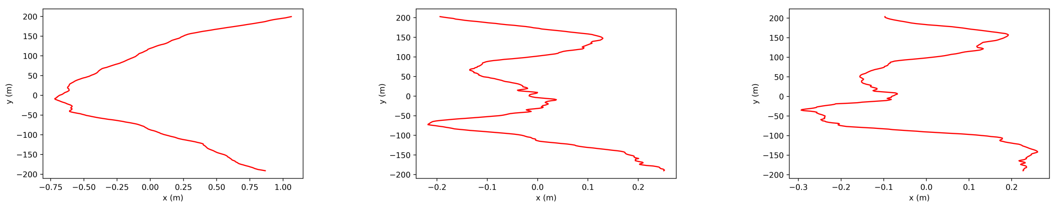
(a) Monodepth2 Estimation (b) COLMAP Estimation (c) Ground Truth

Figure A.2: Trajectory estimations with Monodepth2 and COLMAP on Sequence-1 of KITTI Odometry Dataset



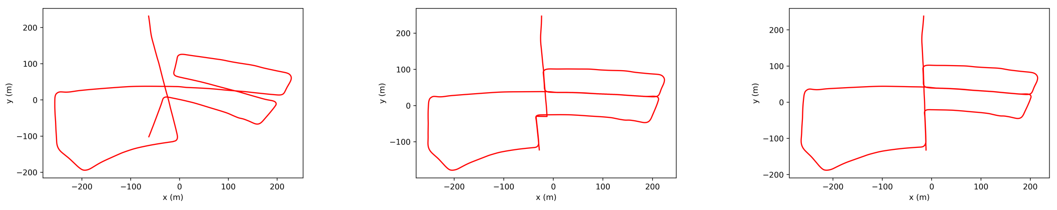
(a) Monodepth2 Estimation (b) COLMAP Estimation (c) Ground Truth

Figure A.3: Trajectory estimations with Monodepth2 and COLMAP on Sequence-3 of KITTI Odometry Dataset



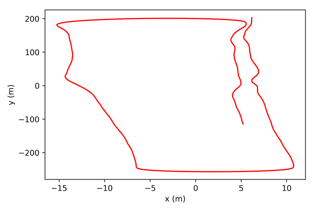
(a) Monodepth2 Estimation (b) COLMAP Estimation (c) Ground Truth

Figure A.4: Trajectory estimations with Monodepth2 and COLMAP on Sequence-4 of KITTI Odometry Dataset

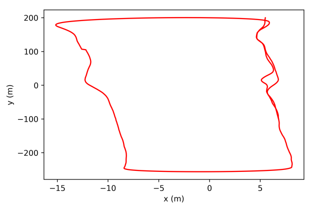


(a) Monodepth2 Estimation (b) COLMAP Estimation (c) Ground Truth

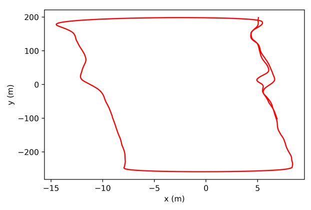
Figure A.5: Trajectory estimations with Monodepth2 and COLMAP on Sequence-5 of KITTI Odometry Dataset



(a) Monodepth2 Estimation

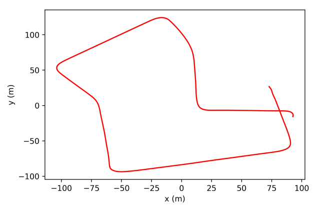


(b) COLMAP Estimation

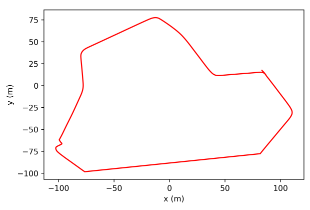


(c) Ground Truth

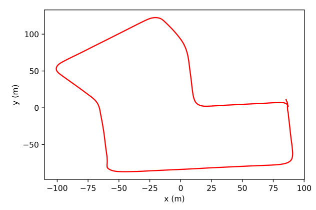
Figure A.6: Trajectory estimations with Monodepth2 and COLMAP on Sequence-6 of KITTI Odometry Dataset



(a) Monodepth2 Estimation

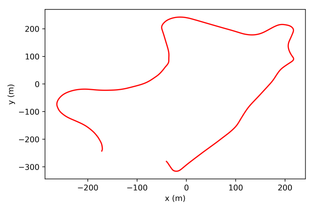


(b) COLMAP Estimation

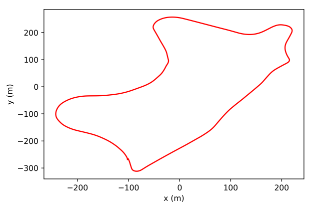


(c) Ground Truth

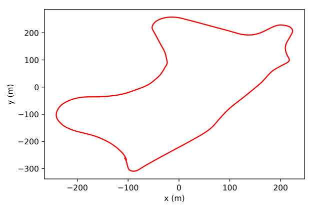
Figure A.7: Trajectory estimations with Monodepth2 and COLMAP on Sequence-7 of KITTI Odometry Dataset



(a) Monodepth2 Estimation

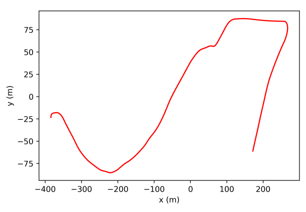


(b) COLMAP Estimation

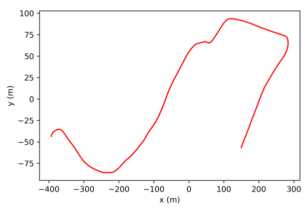


(c) Ground Truth

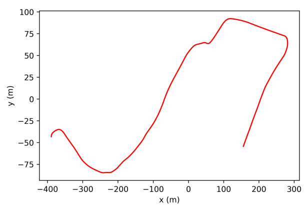
Figure A.8: Trajectory estimations with Monodepth2 and COLMAP on Sequence-9 of KITTI Odometry Dataset



(a) Monodepth2 Estimation



(b) COLMAP Estimation



(c) Ground Truth

Figure A.9: Trajectory estimations with Monodepth2 and COLMAP on Sequence-10 of KITTI Odometry Dataset