

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



Master in Deep Learning for
Audio and Video Signal Processing

MASTER THESIS

**SELF-SUPERVISED MONOCULAR DEPTH
ESTIMATION AND VISUAL ODOMETRY ON
UNSEEN SYNTHETIC CAMERAS**

Cecilia Diana Albelda
Advisor: Juan Ignacio Bravo Pérez-Villar
Lecturer: Álvaro García Martín

June 2023

SELF-SUPERVISED MONOCULAR DEPTH ESTIMATION AND VISUAL ODOMETRY ON UNSEEN SYNTHETIC CAMERAS

Cecilia Diana Albelda
Advisor: Juan Ignacio Bravo Pérez-Villar
Lecturer: Álvaro García Martín

Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
June 2023

This work is supported by Comunidad Autónoma de Madrid (Spain) under the Grant IND2020/TIC-17515 and the HVD (PID2021-125051OB-I00) projects funded by the Ministerio de Ciencia e Innovación of the Spanish Government.



Resumen

Los modelos de odometría visual y estimación de profundidad monocular han ido tomando más presencia en el campo de la visión por computador recientemente. Existen muchos métodos de aprendizaje profundo autosupervisado dedicados a estas tareas que consiguen resultados competitivos con modelos supervisados. Sin embargo, dichos métodos tienden a rendir por debajo de lo esperado cuando se trata de generalizar a secuencias tomadas por nuevas cámaras.

Por un lado, uno de los principales retos para la generalización de cámaras no vistas es la ausencia de un conjunto de datos adecuado que permita realizar comparaciones justas entre secuencias idénticas tomadas por diferentes cámaras. Para abordar este problema, hemos creado un conjunto de datos personalizado utilizando el simulador CARLA [1]. Este conjunto de datos consta de tres secuencias de vídeo diferentes, cada una capturada por 5 cámaras con distancias focales distintas.

Por otro lado, proponemos una arquitectura que utiliza aprendizaje adversario para que el modelo sea invariante a los cambios en los parámetros internos de la cámara. De este modo, podemos estimar la profundidad y la pose independientemente de la cámara utilizada para grabar la secuencia.

Los resultados obtenidos con la arquitectura propuesta se comparan con los proporcionados por una de referencia. Además, también se ha realizado un proceso de evaluación del efecto de aprendizaje adversario, demostrando así las ganancias potenciales de nuestro enfoque.

Específicamente, este proyecto explora la mejora de modelos actuales de estimación monocular autosupervisada de profundidad y odometría visual incrementando su robustez ante cambios en los parámetros de calibración de la cámara mediante aprendizaje adversario.

Palabras clave

Odometría visual, Estimación monocular de profundidad, Visión por computador, Autosupervisado, Aprendizaje profundo, Simulador CARLA, Entrenamiento adversario, Generalización.

Abstract

Visual odometry and monocular depth estimation models have been taking more presence in the computer vision field recently. There are many self-supervised deep learning methods devoted to these tasks that achieve state-of-the-art results. However, they tend to underperform when it comes to generalize to sequences taken by new cameras.

On the one hand, one of the major challenges for unseen camera generalization is the absence of a suitable dataset that enables fair comparisons between identical sequences taken by different cameras. To address this problem, we have created a custom dataset using the CARLA simulator [1]. This dataset comprises three different video sequences, each one captured by 5 specific cameras of diverse focal distances.

On the other hand, we propose an approach that makes use of adversarial training in order to make the model invariant to changes within the internal camera parameters. Hence, we enable depth and pose estimation independently from the camera used to record the sequence.

The results obtained using the proposed architecture are compared to those achieved by a baseline one. Moreover, it has also been performed an evaluation process of the adversarial learning effect, therefore demonstrating the potential gains of our approach.

Specifically, this project explores the improvement of actual self-supervised monocular depth estimation and visual odometry models by increasing their robustness under changes in the camera calibration parameters through adversarial training.

Keywords

Visual odometry, Monocular depth estimation, Computer vision, Self-Supervised, Deep Learning, CARLA Simulator, Adversarial training, Generalization

Acknowledgements

First of all, I would like to thank the Autonomous University of Madrid for giving me the opportunity to study the master's degree in "Deep Learning for Audio and Visual Signal Processing". This experience has allowed me to broaden my knowledge and fundamentals in this exciting area.

I would like to thank specifically the Video Processing and Understanding Lab (VPU) for welcoming me and accepting me as a member of their team, as well as for allowing me to develop my master thesis with them. The people in this group have always shown initiative to help, encouraging me to face every challenge with enthusiasm and hard work.

Special thanks to my tutor, Juan Ignacio Bravo Pérez-Villar, and ponent, Álvaro García Martín, whose guidance and knowledge have been of great help in introducing me to this area of research.

I would also like to mention my Master's classmates, who have shared this academic journey with me and have been a source of support and motivation. Furthermore, I would like to thank my friends in Valencia and my partner for their constant support and for being by my side in the most difficult moments throughout this experience.

Finally, I want to express my gratitude to my family for their unconditional support in every step I have taken in my academic career, as it made possible for me to carry out my Master's studies at this university in the Community of Madrid.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Goals	2
1.3	Report Structure	2
2	Theoretical Background	5
2.1	Self-supervised Monocular Depth estimation and Visual Odometry	6
2.1.1	Self-supervised Monocular Depth estimation	6
2.1.2	Visual Odometry	7
2.1.3	Overall Method	7
2.2	Restrictions of existing methods	8
2.2.1	Reconstruction of the image	8
2.2.2	Scale assumptions	9
2.2.3	Camera assumptions	9
2.3	State of the art addressing camera assumptions	10
2.3.1	Adding Camera Information Explicitly	10
2.3.2	Continuous Learning	12
2.3.3	Fusion of Multi-View Geometry and Deep Learning	12
2.3.4	Mixture of Datasets	13
3	Design	17
3.1	Baseline architecture	17
3.1.1	Description	17
3.1.2	Loss functions	18
3.1.3	Auto-Masking Stationary Pixels	21
3.2	Baseline architecture + Adversarial learning	22
3.2.1	Description	22
3.2.2	Adversarial Learning	22
3.3	Neural Networks	24
3.3.1	ResNet-18	24
3.3.2	U-Net	24
4	Dataset	27
4.1	Dataset Description	27
4.2	Complexity of test sets	29
4.3	CARLA Simulator	30
4.4	Custom Dataset vs KITTI	31

5 Experiments and Results	33
5.1 Evaluation Metrics	33
5.1.1 Absolute Relative Error	34
5.1.2 RSE	34
5.1.3 RMSE	34
5.1.4 Log Scale Invariant RMSE	35
5.1.5 Accuracy under a threshold	35
5.2 Experimental procedure	35
5.3 Analysis of the results	38
5.3.1 Depth Estimation	39
5.3.2 Visual odometry	43
6 Conclusions	49
6.1 Review of objectives	49
6.2 Future work	50
6.3 Conclusion	50
Bibliography	51

List of Figures

1.1	Self-supervised Monocular Depth Estimation and Visual Odometry. . .	2
2.1	Structure of Chapter 2 flowchart. Adversarial training is the proposed architecture of the thesis.	5
3.1	Baseline method of self-supervised visual odometry and depth estimation: a) pose and depth estimation networks; b) novel view-synthesis via differentiable image warping; c) computation of the difference between the target frame, I_t , and the estimated reconstruction of it, \hat{I}_t	17
3.2	Modified block a) from Figure 3.1 by adding another decoder, K , as a camera classifier.	22
4.1	Visual examples of the dataset.	28
4.2	Aerial view of both test sequence maps.	29
5.1	Calculation of the K matrix	36
5.2	Visual results of depth estimation over Test Set 4	40
5.3	Visual results of depth estimation over Test Set 5	42
5.4	Graphical results of visual odometry over Test Set 4	44
5.5	Graphical results of visual odometry over Test Set 5	46

List of Tables

4.1	Dataset description. It contains 5 sequences; 8400 images. Each of them has been captured by 5 different cameras, therefore we have a total of 42000 images.	27
5.1	Training hyperparameters configuration	38
5.2	Results of depth estimation over Test Set 4	39
5.3	Results of depth estimation over Test Set 5	41
5.4	Results of visual odometry over Test Set 4	43
5.5	Results of visual odometry over Test Set 5	45

Chapter 1

Introduction

In this chapter, the problem addressed in this project will be presented. In addition, it will be justified its relevance and motivation, establishing main and secondary objectives. Finally, the structure of the thesis report will be explained.

1.1 Motivation

Three-dimensional reconstruction of scenes from two-dimensional images is a fundamental task in the computer vision field. It is useful in many areas, such as autonomous driving, aerospace and computational medicine. Traditional methods address this task fundamentally using multiview techniques through epipolar geometry or feature matching. However, these approaches have a major limitation in that they assume that the objects in the scene are rigid, which does not take into account the possibility of elastic or deformable objects. In addition, these methods require careful tuning and cannot recover dense depth maps.

Deep learning is considered as a promising solution to overcome these limitations. However, it requires training on very large datasets. To ease this, self-supervised deep learning can be used, as it is able to generate a supervisory signal from unlabeled data by exploiting the underlying structure on it. This methods either accept multiple views of a scene (multi-view), stereo or a single view (monocular).

In addition, these methods can be trained solely with monocular RGB unlabeled sequences, as shown in Figure 1.1. When available, reference depth maps acquired with LIDAR or other depth sensors might be used to evaluate monocular estimation performance. On the other hand, odometry performance is assessed by comparison with GNSS/IMU measurements.

Nevertheless, self-supervised deep learning models for depth estimation show often dependency to the intrinsic calibration parameters of the camera used during training, leading to a degradation of the results when applied to sequences recorded with different cameras.

There is a need to make these methods invariant to camera changes. Therefore, we propose to investigate the use of adversarial learning to increase its generalisation ability to different cameras. In this way, we enable more robust and accurate 3D

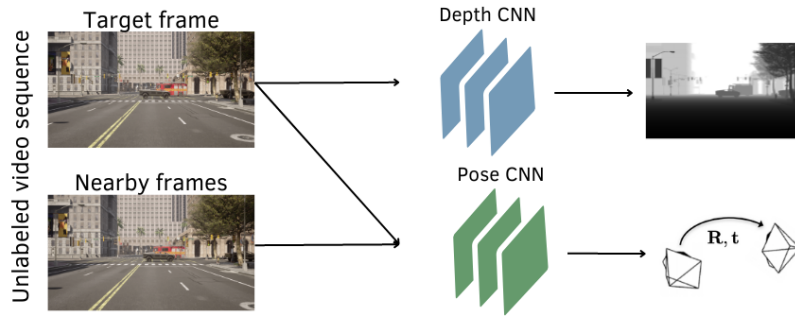


Figure 1.1: Self-supervised Monocular Depth Estimation and Visual Odometry.

reconstruction in diverse scenarios, thus improving one of the main limitations of self-supervised deep learning models in this task.

This research line, focused on the inclusion of adversarial training in self-supervised depth estimation models began in the 2021-2022 academic year in the VPU Lab [2]. However, in this project, a continuation of that study is carried out by also introducing visual odometry extraction and further evaluation. Furthermore, this time more experiments have been carried out and a larger dataset has been generated with increased realism.

1.2 Goals

The main objective of this project is to design and implement a self-supervised deep learning model capable of estimating both the depth of each image belonging to a video sequence and its trajectory, showing robustness to changes in the intrinsic calibration parameters of the camera through adversarial learning.

The secondary objectives are the following:

- Develop knowledge in the area by studying and analyzing the limitations of the state-of-the-art.
- Create a custom dataset that contains different realistic sequences and enables fair comparisons of the same sequence captured by various cameras.
- Demonstrate the effect that changes in the Field Of View (FOV) have on the performance of a baseline model.
- Evaluate the proposed architecture in terms of depth and odometry estimation and compare the results with the ones obtained by a baseline architecture.

1.3 Report Structure

This report is structured in 6 chapters.

Chapter 1 introduces the motivation of the thesis and its context. It also includes the main objective and the secondary objectives.

On the other hand, Chapter 2 consists of a detailed explanation of the theoretical background that forms the basis of the thesis. Thus, this chapter explains in detail the problem we are dealing with, the different solutions that are currently available to solve it and the main limitations of each one. Moreover, it includes the mathematical fundamentals on which these methods are based.

In Chapter 3, we explain the architecture we are going to use as a reference one (Baseline) and the difference with the one that we have proposed by introducing adversarial learning to it.

Chapter 4 describes the structure of the created dataset, providing examples of images contained in it. In addition, we include a brief explanation of the simulator used for its generation.

Chapter 5 includes the explanation of the metrics used to evaluate the proposed architecture, the experimental procedure that will be carried out throughout the project and the results obtained in each of the experiments.

Finally, Chapter 6 is based on the conclusions obtained after finishing the thesis and in view of the results obtained, as well as showing possible future work in this area.

Chapter 2

Theoretical Background

Throughout this chapter we will explain the concept of self-supervised monocular depth estimation and how visual odometry is obtained from it. Moreover, we will expose the main assumptions of this type of methods and the limitations implied by each of them. Then, a theoretical background will be provided in which we will describe multiple approaches to solve the described limitations. Finally, it will be expanded the framework in which this project is focused.

We are going to follow the flowchart showed in Figure 2.1.

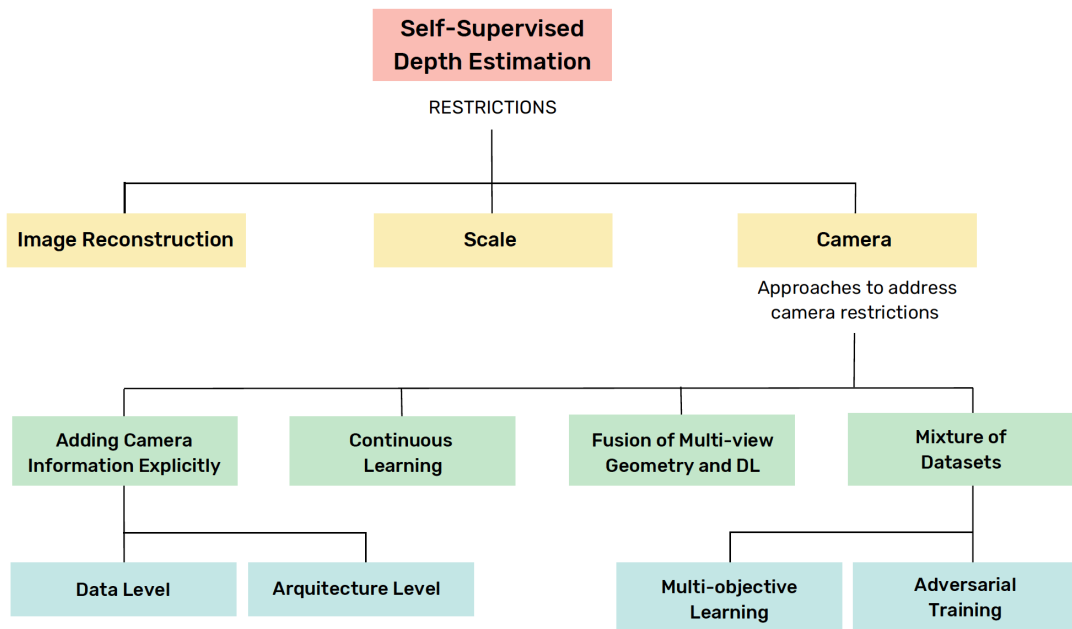


Figure 2.1: Structure of Chapter 2 flowchart. Adversarial training is the proposed architecture of the thesis.

2.1 Self-supervised Monocular Depth estimation and Visual Odometry

2.1.1 Self-supervised Monocular Depth estimation

To correctly understand what self-supervised monocular depth estimation is, it is necessary to understand each of the terms that make up its name. The word ‘monocular’ refers to the fact that the images that are available for us have been captured by a single camera. On the other hand, the term ‘self-supervised’ implies that we are using a learning approach where we obtain supervisory signals from the data itself. Finally, the concept of ‘depth estimation’ defines the process of generating depth maps from an input image. Thus, self-supervised monocular depth estimation is an advanced technique in the field of computer vision and deep learning that allows depth inference at the pixel level in images captured by a single camera without the need of annotations.

Traditional methods primarily use epipolar geometry for depth estimation. This is based on the geometry of the cameras and establishes the relationship between images captured by different cameras or at different times. Thus, they use the fundamental matrix and the projection matrix in order to calculate the stereo disparity between the cameras and estimate the depth. In addition, the epipolar constraint defines point correspondences between the images, which helps in accurate depth estimation.

Nevertheless, an important limitation of traditional approaches is that they require additional information such as multiple stereo views or depth sensors. Alternatively, self-supervised monocular depth estimation uses deep learning techniques to infer depth from single images without the need of explicit annotations. To this end, deep convolutional neural networks (CNN) are employed to learn the relationship between the input images and their corresponding depth. These models are trained in a self-supervised manner, comparing each image with its reconstruction and taking that difference as the supervisory signal. It is important to mention that the reconstruction of each frame is computed by combining the parameters of the camera model together with the depth and pose estimation.

Camera models are described by intrinsic (focal length, optical centre) and extrinsic (camera position and orientation) characteristics. These models allow points in three-dimensional space to be related to their projections on images, which is crucial for accurate depth estimation.

In this context, different loss functions are used to evaluate the discrepancy between the reconstructed image and the original one. These loss functions play a crucial role in the training of the CNN. Some of the common loss functions include L1 loss [3], which measures the absolute difference between the estimate and the actual depth value, and structural similarity loss (SSIM) [4], which evaluates the structural similarity between the reconstructed image and the original image. In addition, custom loss functions can be designed to suit the specific characteristics of the depth estimation problem.

2.1.2 Visual Odometry

The term ‘visual odometry’ refers to the process of incrementally estimating the position and orientation of a vehicle from images captured with an on-board camera. This information is essential in autonomous navigation applications and motion tracking systems. There are multiple methods to solve this task, which can be based on visual features [5], point of interest tracking [6] or pixel matching across frames [7].

In the context of self-supervised monocular depth estimation, visual odometry can be extracted as a part of the process [8]. This is achieved by incorporating into the deep learning model a network devoted to pose prediction, that is responsible of estimating changes in camera position and orientation between successive frames. By jointly training the depth estimation network and the visual odometry one, cross-feedback between the two tasks is achieved.

The monocular depth estimation process provides information about the scene structure and the three-dimensional geometry of the objects present in the image. Depth estimation is performed on the image points. Concurrently, the pose estimation network yields estimations of the camera’s relative position and orientation in relation to those points. These pose estimations are subsequently used to continually update the vehicle’s trajectory as new images are acquired.

2.1.3 Overall Method

As explained in Section 2.1.2, self-supervised monocular depth estimation methods include a network dedicated to pose estimation, from which the visual odometry task is performed.

Although the relationship between both networks is explained in detail in Chapter 3, in order to make this chapter self-contained, it should be noted that the learning process in these methods is based on the estimation of a target frame I_t from a source frame I_s (adjacent to the target one), by the following equation [8]:

$$p_s = K T_{t \rightarrow s} D_t(p_t) K^{-1} p_t, \quad (2.1)$$

where p_s and p_t are a pixel of I_s and I_t respectively, D_t is the estimated depth for I_t , $T_{t \rightarrow s}$ is the pose change between both frames and K is the calibration matrix of the camera used to capture the sequence.

From Equation 2.1, p_t is projected in the 3D world with K^{-1} and scaled through D_t . Then, we use $T_{t \rightarrow s}$ to transform the pixel to the source position. Finally, we use K to project it back to the camera, thus obtaining the source pixel, p_s .

Therefore, since self-supervised monocular depth estimation models use K in their learning algorithm, they tend to generate dependence on these intrinsic parameters of the camera that has recorded the sequence, hence hindering their applicability in environments with different cameras.

2.2 Restrictions of existing methods

Self-supervised monocular depth estimation improves over limitations of traditional methods. However, it is important to note that they also present certain restrictions that can be classified into three main categories: scale, image reconstruction and camera model.

Firstly, as for the image reconstruction constraints, the self-supervised monocular depth estimations assume that the scene is static, i.e. no moving objects are present. Thus, the reconstruction relies on all pixels moving relative to the ego-motion of the camera device. This assumption simplifies the depth estimation process, but can be limiting in dynamic scenes or scenes with multiple moving objects. In addition, these methods may have difficulties dealing with occlusion, as the synthetic monitoring information may not accurately reflect the occluded regions in the real image.

In addition, consistency of scale between the predicted depth and the camera pose is assumed. This is necessary to avoid having infinite combinations of initial depth and translation vector resulting in the exact solution for the relative distances. This constraint allows for a consistent relationship between depth scale and camera movement, which is essential for accurate results. However, scale is inherently ambiguous in a single image, which implies a major challenge.

Finally, it is assumed that the intrinsic parameters of the camera remain constant both during training and test phases. This constraint implies that, among other parameters, the calibration matrix of the camera will remain unchanged. It is important to note that this constraint is based on the assumption that assumes a fixed camera. However, in real situations, it may be difficult to keep the intrinsic parameters constant, which limits the applicability of these methods in practical contexts.

2.2.1 Reconstruction of the image

The problem of image reconstruction in monocular depth estimation presents additional challenges when faced with real-world scenes. In these scenes, it is difficult to find perfect scenes where there are no occlusions and the assumptions of static scenes are met, especially in urban environments where many existing objects exhibit different movements relative to the camera movement.

A common approach is to exclude incomprehensible regions for camera ego-motion. Therefore, a possible solution remains on the idea of using an architecture that has a combined pose-explainability network retrieving both the camera pose between frames and a mask that has information on unexplainable pixels [9].

Another solution for addressing this is based on the use of optical flow to mask areas of the image that break the ego-motion assumption through the measurement of the consistency of forward and backward optical flow between two frames [10, 11, 12].

Alternatively, a loss term can be included that takes occlusions into account [13]. This loss term uses the measurement of depth differences for corresponding points.

This can be combined with the classification of objects as "possibly moving" or "static" through semantic information [14], subsequently masking the moving area in the loss calculation.

Other approaches focus on calculating a mask based on the distribution of the image reconstruction loss along with a weak supervision based on the estimation of the epipolar geometry [15]. Finally, it is possible to use depth to generate three-dimensional clouds and then construct depth alignments for supervision [8].

In general, these approaches seek to address the image reconstruction problem in monocular depth estimation by accounting for factors such as occlusions and motion consistency. By using masks, adaptive loss and techniques based on epipolar geometry, they attempt to improve the quality of image reconstruction and obtain more accurate depth estimates in real-world scenarios.

2.2.2 Scale assumptions

The problem of scale assumptions in depth estimation is a major challenge in this field. There is a scale ambiguity between the predicted depth and the camera pose. For example, some detected objects may be small and close to the camera, or conversely, they may be large objects and far away from the camera.

The problem lies in the infinite possible combinations between the initial depth and the translation vector that provide the exact solution for the endpoint. During training, the pose and depth estimation networks reach a scale factor that helps to provide meaningful results. Thus, several attempts have been made to ensure scaling consistency during training.

One of the proposed solutions is based on minimising the loss defined as the difference between the target depth and the source depth warped to the target frame [16]. Through this technique, an attempt is made to guarantee geometric consistency. Alternatively, it can be addressed by using inverse depth terms [17] or even by directly aligning depth estimations with sparse depth points [18].

These approaches seek to address the problem of scale ambiguity and improve the accuracy of depth estimation. However, it is important to note that these approaches may introduce additional limitations and dependencies on the input data, such as the need for sparse depth points or source depth information.

2.2.3 Camera assumptions

In the field of depth estimation, one of the most common restrictions is the assumption that the intrinsic camera parameters remain constant across sequences. While this is usually true in controlled environments, it often leads to a decrease in the generalization capability of the model in the test phase when using a different camera model.

This restriction can be addressed by adding an up-to-scale loss function and a new layer in the learning architecture that explicitly adds intrinsic camera information to

the images [19]. Hence, information such as the calibration matrix is incorporated directly into the learning model. This allows the model to better learn and generalise to different camera configurations.

Epipolar geometry can also be used to estimate camera rotation and translation from point correspondences [16] [18]. Epipolar geometry provides a relationship between images captured by two different cameras and is used to estimate the camera pose. These methods use specific algorithms, such as the fundamental matrix decomposition, to solve the pose estimation.

It has been explored the design of a neural network combining the perspective-n-point problem (PnP) and the fundamental matrix decomposition [20]. By combining different techniques, it aims to improve the accuracy of camera pose estimation and overcome the limitations of traditional methods. This approach is especially useful in cases where significant camera translations result in degenerate epipolar solutions, i.e. solutions with poor generation of the lines formed by the correspondences between points in two different views.

The focus of this experiment is specifically on this particular constraint. An attempt is made to solve the problem by proposing an invariant architecture to the camera model. This means that the model is able to learn and generalise to different camera configurations without the need to know the specific intrinsic parameters. In doing so, the aim is to improve the model’s ability to adapt to various situations and to achieve more accurate and generalisable depth estimates.

2.3 State of the art addressing camera assumptions

In the field of monocular depth estimation, there are several state-of-the-art strategies for dealing with camera assumptions.

One option is to explicitly incorporate camera information into the learning architecture, considering intrinsic parameters such as the calibration matrix and distortion coefficients. Alternatively, continuous learning is being investigated to adapt the model to new conditions and changes in image capture. Furthermore, multi-view geometry fusion and deep learning are also used to improve the accuracy of depth estimates by exploiting the spatial structure of the scene and the correspondence between images. Finally, combining diverse datasets allows addressing the limitations of camera assumptions by training and evaluating the model under different conditions to obtain more accurate and robust estimates.

These strategies seek to improve the quality and reliability of depth estimates by more fully considering camera characteristics and scene geometry.

2.3.1 Adding Camera Information Explicitly

One of the strategies used to address camera assumptions in self-supervised monocular depth estimation methods is by explicitly incorporating camera information. There are two levels where it can be done: data and architecture level.

Data Level

At the data level, there is an important approach to address the limitations of visual odometry models: TartanVO [19], which is a learning-based model that seeks to generalise across multiple datasets. Visual odometry models often face two common problems: lack of training with diverse data and the omission of fundamental theories of geometry-based visual odometry, such as scale ambiguity. To overcome these challenges, TartanVO uses a synthetic and diverse dataset called TartanAir [21] during the training process. By using synthetic data, the model can face challenging scenes and learn from a wide range of situations.

However, since TartanAir only provides a set of intrinsic camera parameters, additional parameters are simulated using data augmentation techniques such as resizing and random clipping. The model uses a pre-trained network called PWC-Net [22] and a modified version of ResNet50 [23]. In addition, TartanVO proposes a loss function that explicitly addresses scale ambiguity and adds information about the intrinsic data. Through extensive experiments, the model demonstrates its ability to generalise to previously unseen data. Although it outperforms geometry-based methods, it is important to note that this model still requires the inclusion of intrinsic data to work.

Architecture Level

Architectural changes to neural networks have been proposed to explicitly incorporate camera information. A relevant example is the Camera-Aware Multi-scale Convolutions (CAMConvs) method [24]. It proposes a new type of convolution that considers camera parameters such as the calibration matrix and distortion coefficients.

CAMConvs allows the network to learn patterns sensitive to the camera calibration, which improves the consistency and accuracy of depth estimations. The main idea behind CAMConvs is to recognise that different regions of an image may require different filters due to the particular geometry. Instead of applying standard convolutions to all regions of the input image, CAMConvs adapts the convolution filters according to the camera parameters. This allows the network to more effectively capture camera-specific spatial and structural relationships, resulting in more accurate and consistent depth estimations.

In addition to multi-scale convolutions, CAMConvs can incorporate camera information into other layers of the neural network. For example, attention layers can be added that weight the importance of different features based on camera calibration. These layers allow the network to focus on relevant regions, thus improving the quality of depth estimates.

They mainly experiment using the 2D-3D Semantic dataset [25], which allowed them to generate images with the same content but with different camera intrinsics. The results confirmed that CAMConvs outperforms the state of the art and demonstrates robust generalisation in terms of camera intrinsics, unseen sensor sizes and unseen cameras. However, this method has only been applied in supervised depth estimation.

2.3.2 Continuous Learning

Continuous learning is based on the idea of constantly updating and adapting the model as new data is collected, allowing the system to adjust to changing capture conditions and overcome the limitations of static camera assumptions.

A relevant study in this context is the online adaptation framework for deep visual odometry proposed in [26]. This approach describes a model that incorporates Bayesian inference and scene-independent geometric computations. The model consists of two main branches: FlowNet and DepthNet.

The FlowNet branch is responsible for estimating the optical flow between two consecutive frames. It studies how objects and features in the scene move between frames, which provides information about the relative motion of the camera and objects in the scene. This optical flow estimation is fundamental to understanding the change in camera perspective and allows the model to be dynamically updated as new frames are captured.

On the other hand, the DepthNet branch is responsible for estimating the depth of the scene from the input frames. It uses deep learning techniques to infer depth information based on the relationship between the image pixels and the features learned by the neural network. The triangulated depth patches improve the initial depth during an online adaptation.

Finally, the pose and optimized depth are the pseudo ground-truth for self-supervision during online learning of DepthNet and FlowNet. Even though it enables fast adaptation of visual odometry networks to unseen scenes, it requires an online optimization and hardware to support that.

In summary, the continuous learning approach is based on the idea that the model updates and adapts as new data is captured. This allows the system to adjust to changes in capture conditions, such as variations in lighting, the appearance of moving objects or changes in camera settings. By incorporating new data and periodically updating the model, the system becomes more robust and able to cope with static camera assumptions. The incorporation of Bayesian inference, geometric calculations and the use of branches such as FlowNet and DepthNet improves the system's ability to overcome the limitations of static camera assumptions and obtain more accurate and reliable depth estimates.

2.3.3 Fusion of Multi-View Geometry and Deep Learning

This approach combines the spatial structure of the scene obtained from multiple images with the deep learning capabilities of neural networks to improve the robustness and accuracy of depth estimates.

The main idea behind the fusion of multi-view geometry and deep learning is to leverage information obtained from different viewpoints of the scene to obtain more accurate and consistent depth estimates independently of the camera. By using multiple images, a more complete view of the scene can be captured, which helps overcome the

limitations of single image information and static camera assumptions. In this way, it is possible to determine correspondences between points in different views, estimate the fundamental matrix that describes the relationship between images, and calculate the projection matrix that relates 3D points in the real world to their projection in the images.

In [27], the challenge of camera variability in estimating 3D geometry from images is comprehensively addressed. In this context, the proposed approach combines deep learning techniques with classical multi-view geometry methods to address the inconsistency that can arise due to differences in camera configurations.

One of the main achievements of the paper is the elimination of the dependence of the learning model on camera-specific parameters. To this end, the authors propose to use traditional multi-view geometry methods to resolve this issue. Specifically, a two-view triangulation module is employed which, by combining information from different images, achieves a three-dimensional structure with an appropriate scale.

A key feature of this approach is that the entire system can be trained in an end-to-end manner, allowing end-to-end learning to take place. This means that the model learns to infer the 3D geometry and fuse it with the input images together, thus improving the consistency and accuracy of the estimates.

Experimental results obtained by evaluating the proposed approach using the KITTI dataset [28] demonstrate significant improvements in the generalisation capability of the model. Furthermore, its outstanding performance in self-supervised models is highlighted, achieving state-of-the-art results in this type of approaches.

However, it is important to mention that this method may present certain limitations and be prone to failure in certain situations. One of the possible sources of error lies in the estimation of the fundamental matrix, which can be affected by noise in the data. It should also be noted that this approach is not applicable in situations of pure rotations and requires the availability of the camera matrix for its correct operation.

2.3.4 Mixture of Datasets

Mixture of Datasets is a strategy used in monocular depth estimation that seeks to address camera assumptions by combining diverse datasets. This technique is primarily based on two approaches: multi-objective learning and adversarial training.

Multi-objective Learning

First, multi-objective learning is a technique that involves training the model using multiple targets or training criteria. In the context of depth estimation, this involves using different datasets that exhibit variations in capture conditions, such as illumination, object motion and scene geometry. By combining these datasets, the model benefits from greater diversity and generalisation, which helps to overcome camera assumptions and obtain more accurate and robust depth estimates.

A relevant approach to address the challenge of monocular depth estimation in diverse scenarios through multi-objective learning is [29]. In this method, data from different sources are combined, even when they have incompatible annotations. By combining these data in an intelligent way, the model can learn more robust patterns and features, thus improving the quality and accuracy of depth estimates.

One of the key features of this model is its invariance to depth range and scale, which means that it can generalise and estimate depth accurately across different ranges and scales of scenes. To achieve this, it is used a training dataset consisting of six diverse datasets, each of which has unique and challenging characteristics. In addition, a comprehensive evaluation is performed on a previously unseen dataset, allowing the model’s generalisability and performance in new situations to be measured.

Furthermore, the importance of pre-training encoders on complementary tasks before applying multi-objective learning is emphasised. This pre-training approach allows capturing general features and useful representations that benefit the subsequent depth estimation process.

Experimental results show that, after being trained with five different datasets and evaluated with a different one, the model is able to perform more accurate and reliable depth estimation compared to other approaches. However, it is important to note that this model has been evaluated and applied exclusively in the context of supervised depth estimation, which means that its performance in other scenarios or with unsupervised data may require further investigation.

Adversarial Training

In order to achieve robust depth estimation in the face of changes in the intrinsic parameters of the camera, the use of adversarial training has been proposed.

In the context of adversarial training, a discriminator is introduced that aims to estimate which camera has recorded each video frame or sequence from a set of possible cameras. The discriminator is trained to distinguish between the frames generated by the model and the actual frames recorded by different cameras. Its function is to learn to identify the specific characteristics and patterns of each camera.

On the other hand, a CNN acts as a generator, taking care of extracting features from which it is possible to generate depth estimates that are robust and generalisable across different cameras. During training, the generator is confronted with the discriminator, and its goal is to trick the discriminator so that it cannot distinguish between the generated estimates and the actual depths recorded by different cameras. As the generator improves its ability to generate indistinguishable depth estimates, greater robustness to changes in intrinsic camera parameters is achieved.

The adversarial training process involves the joint optimisation of the generator and the discriminator. The loss of the discriminator is defined to maximise its ability to correctly identify the source chamber, while the loss of the generator is defined adversarially, seeking to minimise the discriminator’s ability to distinguish between generated estimates and true depths.

This adversarial training approach allows the generator to learn relevant features and patterns that are invariant to the intrinsic parameters of the camera. By introducing competition between the generator and the discriminator, the generator becomes more aware of variations in camera parameters and learns to generate more robust and consistent depth estimates.

The limitation of solving the problem of camera generalization is the existence of the proper dataset. For example, to train a model invariant to camera models, we need a dataset containing images of the same scene captured by different cameras.

Primarily everything was developed using the KITTI dataset [28]. This dataset served as a benchmark dataset for depth estimation problems. However, this will not be the proper dataset for camera generalization as it only has one camera. Therefore, we generated a custom dataset using CARLA Simulator. It contains different sequences recorded by diverse camera models, as explained in Chapter 4.

Chapter 3

Design

In this chapter we will detail the architecture taken as a reference model, as well as the modifications we have made to it. Thus, we will explain the inclusion of adversarial learning to provide the model with robustness to changes in the intrinsic camera calibration parameters.

3.1 Baseline architecture

3.1.1 Description

First of all, we are going to show a schematic figure in which we can visually appreciate the different parts of the baseline architecture.

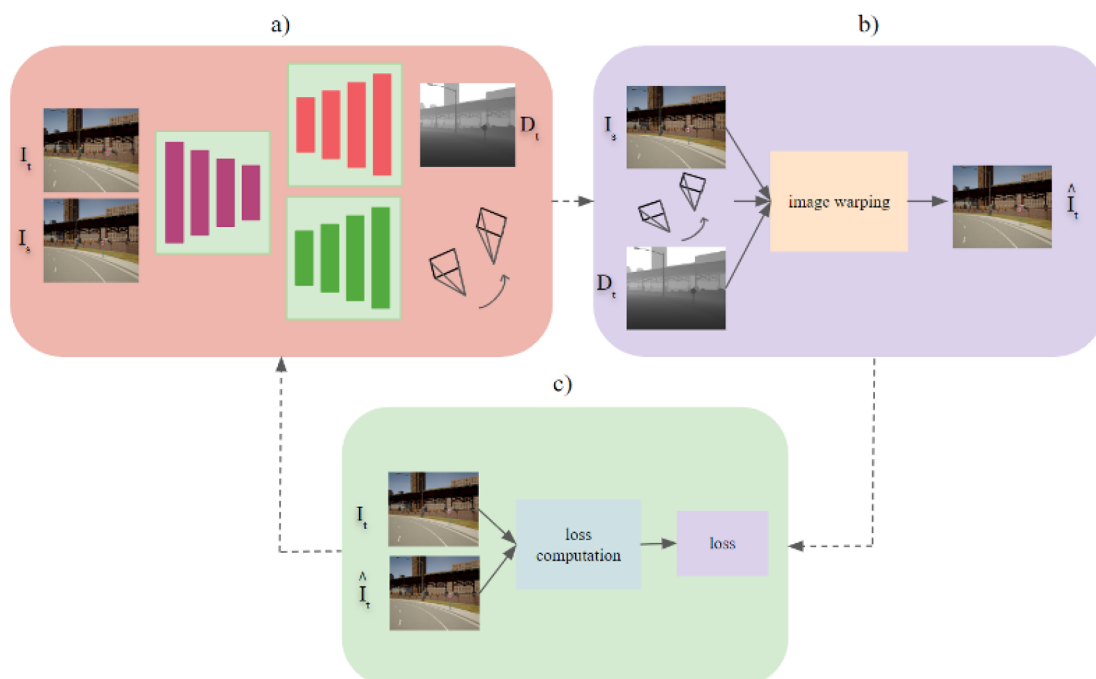


Figure 3.1: Baseline method of self-supervised visual odometry and depth estimation: a) pose and depth estimation networks; b) novel view-synthesis via differentiable image warping; c) computation of the difference between the target frame, I_t , and the estimated reconstruction of it, \hat{I}_t .

The basic pipeline of self-supervised visual odometry and depth estimation has three main parts, as we can see in Figure 3.1. The first part (a) consists of an encoder which, given two consecutive frames I_s and I_t ($I_s = I_{t-1}$), is in charge of feature extraction. Subsequently, these features are used as input in two different decoders; one of them estimates the depth of I_t , D_t , and the other one predicts the pose changes, $T_{t \rightarrow s} = [R|t]$, which contains the estimated translation vector t and rotation matrix R from I_t to I_s .

The second part of the architecture (b) uses I_s , D_t and $T_{t \rightarrow s}$ to reconstruct the I_t frame, hence generating \hat{I}_t .

Finally, the third part (c) measures the difference, defined by a specific loss, between the I_t frame and the reconstruction of it, \hat{I}_t , thus providing positive or negative feedback to the network.

As we have seen in Chapter 2, self-supervised learning methods for depth prediction and visual odometry are based on the reconstruction of a target frame from a sequence of nearby views.

Nevertheless, in this case \hat{I}_t is obtained from two adjacent frames, I_s and I_t , the camera calibration matrix, K , $T_{t \rightarrow s}$ and finally D_t . We obtain the projection of each target pixel, p_t , on each source pixel, p_s , by:

$$p_s = KT_{t \rightarrow s}D_t(p_t)K^{-1}p_t. \quad (3.1)$$

Intuitively, we first project p_t in the 3D world with K^{-1} . However, we have no information on the depth, as K projects into the 3D world up to a scale factor. Therefore, we obtain this pixel in the 3D world through D_t . Then, we use the transformation matrix $T_{t \rightarrow s}$ to transform the pixel to the source position. Finally, we use K to project it back to the camera, thus obtaining the source pixel, p_s .

As we can see, it is possible to express p_s in terms of p_t using the previous equation. Therefore, the model makes use of that expression in order to reconstruct the target frame and, after that, it computes the difference between the target frame and the reconstructed one. This difference measure acts as the supervision signal to improve predictions.

3.1.2 Loss functions

The loss function is what determines the learning goal, hence reflecting the performance of the learning process based on the provided data. A standard loss function used in self-supervised models for depth estimation is the absolute difference between the reconstructed image against the real one (L1 loss [3]). However, we are going to use also other two losses: structural similarity index measure (SSIM) [4] and smoothness loss [30]. We follow other works [31] by combining the mentioned loss functions as each of them gives us a different perspective of the error: L1 is merely the per-pixel difference; SSIM computes the similarity between two images by evaluating their luminance,

contrast and structure; smoothnes measures how smooth/abrupt the depth transitions are in the depth maps. Thus, by using these three losses together we ensure a better result both numerically and visually.

L1

As described above, L1 loss is computed as the mean of the absolute value difference between the reconstructed image \hat{I}_t and the target one I_t at the pixel level.

$$L_1(I_t, \hat{I}_t) = \frac{1}{N} \sum_{p=0}^N |I_t(p) - \hat{I}_t(p)|, \quad (3.2)$$

being p a pixel coordinate and N the total number of pixels in the frame.

SSIM

The Structural Similarity Index Measure (SSIM) loss [4] is a metric used to quantify the similarity between two images. It measures the structural similarity and perceptual quality by comparing the luminance, contrast, and structure of the images.

The SSIM loss provides a value between 0 and 1, where a higher value indicates a greater similarity between the images, meaning lower error. It is designed to address some of the limitations of traditional pixel-wise loss functions, such as mean squared error (MSE), by considering human visual perception and image structures.

The luminance is expressed as:

$$l(a, b) = \frac{2\mu_a\mu_b + C}{\mu_a^2 + \mu_b^2 + C}, \quad (3.3)$$

where μ_a and μ_b are the mean intensity values of a and b images, respectively. Moreover, C is added to avoid numerical instability.

The contrast measure is computed as:

$$c(a, b) = \frac{2\sigma_a\sigma_b + C}{\sigma_a^2 + \sigma_b^2 + C}, \quad (3.4)$$

where σ represents the standard deviation of the mean centered image.

Finally, the structure index is defined as:

$$s(a, b) = \frac{\sigma_{ab} + C}{\sigma_a\sigma_b + C}. \quad (3.5)$$

Then, SSIM is expressed as:

$$SSIM(a, b) = [l(a, b)]^\alpha [c(a, b)]^\beta [s(a, b)]^\lambda, \quad (3.6)$$

being α , β and λ adjustable parameters, strictly greater than 0, which allow us to establish the influence given to each of the three components in the SSIM computation. For simplicity, we have established $\alpha = \beta = \lambda = 1$. Hence, taking this into account and from the combination of the above equations, we can express the SSIM as follows:

$$SSIM(a, b) = \frac{(2\mu_a\mu_b + C)(2\sigma_{ab} + C)}{(\mu_a^2 + \mu_b^2 + C)(\sigma_a^2 + \sigma_b^2 + C)}. \quad (3.7)$$

Given that the SSIM provides a measure of similarity between two frames, it means that a higher SSIM value corresponds to a lower error. To formulate this measure as a loss function, we express it as follows:

$$L_{SSIM}(I_t, \hat{I}_t) = \frac{1 - SSIM(I_t, \hat{I}_t)}{2}. \quad (3.8)$$

Smoothness

Smoothness loss encourages smooth and continuous depth or disparity maps [30]. Therefore, it penalizes abrupt or inconsistent depth transitions between neighboring pixels in the estimated depth map. Hence, it achieves helping to produce more coherent depth maps and visually plausible depth variations.

The smoothness loss is computed as:

$$L_{smooth} = |\partial_x \partial_t^*| e^{-|\partial_x I_t|} + |\partial_y \partial_t^*| e^{-|\partial_y I_t|}, \quad (3.9)$$

where ∂_x and ∂_y are the derivatives in the horizontal and vertical direction. Moreover, the normalised mean inverse depth, ∂_t^* , is also being considered. This term takes the average of the estimated inverse depths and normalises them, so that they are in a suitable range. Thus, this helps to penalise the decrease of the estimated depth towards smaller values.

Final loss function

The final loss function is obtained by combining the three previously detailed loss functions: L1, SSIM, and smoothness. Thus, it is expressed as follows:

$$Loss = (1 - \alpha)L_1 + \alpha L_{ssim} + \beta L_{smooth}, \quad (3.10)$$

being α and β adjustable parameters, both strictly greater than 0, that enables to determine the relative importance assigned to each of the three components.

Complementary weighting between L1 and SSIM is used to leverage the strengths of both loss functions. L1 loss is sensitive to the absolute differences between the actual and predicted values, which makes it effective for capturing large errors and maintaining structural detail. On the other hand, SSIM loss considers the structural and perceptual similarities between images, which makes it more suitable for preserving fine details and textures.

By introducing the parameter α , it is possible to adjust the balance between these two components. A higher value of α will give more weight to the SSIM loss, which will encourage the preservation of fine details and textures, while a lower value will give more weight to the L1 loss, which will emphasise the capture of large errors.

In addition, the parameter β is added to adjust the influence of smoothness loss on the final loss function. This allows to control the importance of smoothness and regularity in the depth maps predicted, which helps to avoid inconsistent or noisy solutions.

We have established $\alpha = 0.85$ and $\beta = 0.01$, as done in the literature [32].

3.1.3 Auto-Masking Stationary Pixels

Self-supervised monocular training operates under the assumptions of a moving camera and a static scene, as we have seen in Chapter 2. When these assumptions break down, performance can suffer greatly, even causing ‘holes’ of infinite depth to appear in the predicted depth maps [33]. To mitigate this problem, we use a simple auto-masking method [32] that filters out pixels which do not change appearance from one frame to the next one. This has the effect of letting the network ignore pixels that break the reconstruction assumptions (see Section 2.2.1), such as objects which move at different velocity as the camera, occlusions, and even to ignore whole frames when the camera stops moving.

We perform this auto-masking method through the following equation:

$$Loss^{i,j} = \min(pe(I_t, I_{s \rightarrow t})^{i,j}, pe(I_t, I_s)^{i,j}), \quad (3.11)$$

where pe is the projection error defined by the specified final loss, and the terms i, j represent the pixel coordinates of the image.

Intuitively, we are comparing for each pixel of I_t what gives us the smallest error: using its corresponding value in I_s or using the warped value $I_{s \rightarrow t}$. As this loss is a tensor of the same shape of the images, i.e. provides an error for each RGB pixel, we first compute the mean per channel and, then, global mean. Hence, a single error value per image is obtained.

3.2 Baseline architecture + Adversarial learning

3.2.1 Description

The proposed architecture is almost identical to the baseline one seen in Section 3.1.1. However, we have performed a modification in block a), where a new decoder has been added whose purpose is to determine the camera with which each image was captured.

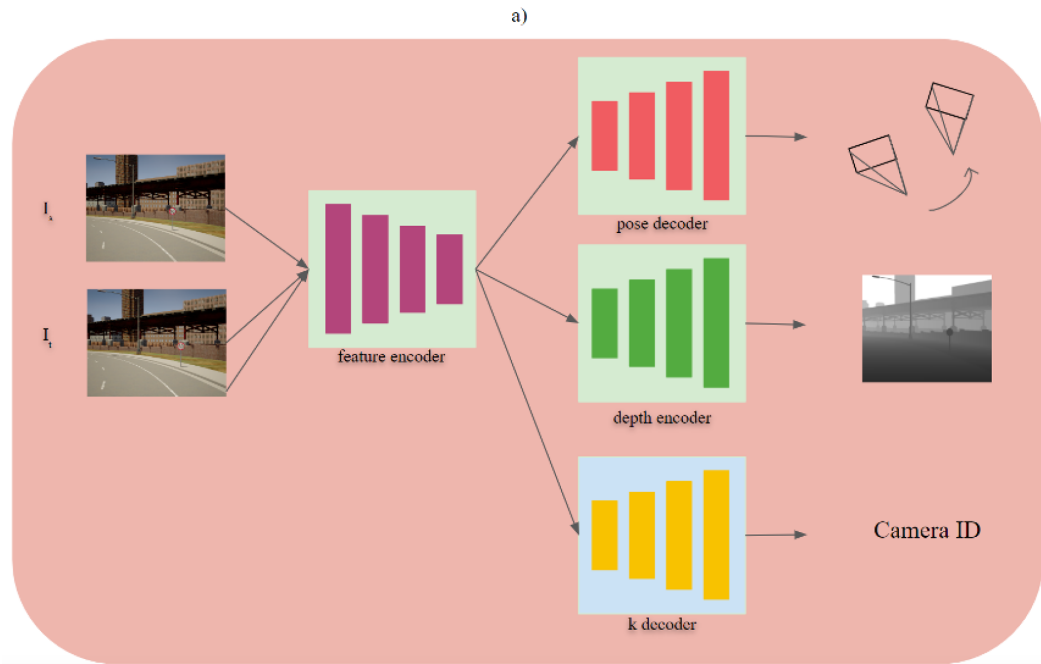


Figure 3.2: Modified block a) from Figure 3.1 by adding another decoder, K, as a camera classifier.

This pipeline, as that of the baseline, accepts two images: a source image I_s and a target one I_t . Then, both images are passed to the encoder, which is in charge of feature extraction, and, after that, these features are the input of the pose decoder, depth decoder, and the additional camera K decoder.

This K decoder primarily handles the task of classifying the image's corresponding camera. In this approach, the proposed method involves training the networks in an adversarial manner to achieve a feature representation that remains independent of the camera model used.

3.2.2 Adversarial Learning

Adversarial learning is included in the proposed architecture to generate camera-invariant features in the context of monocular depth estimation and visual odometry. The core idea behind adversarial learning is to train the encoder network to generate features that are independent of the specific camera used to record the training sequence.

The adversarial learning framework consists of two main components: the encoder and the discriminator (K decoder in this case). The encoder’s goal is to learn representations that are robust to camera variations, while the discriminator aims to distinguish between features originated from different cameras.

During training, the encoder generates features from the input images, and these features are fed into both the depth estimation and visual odometry networks. Simultaneously, K decoder receives the encoded features and attempts to predict the camera that has captured the images to which these features correspond.

The training process involves a competition between the encoder and the discriminator. The encoder aims to generate camera-invariant features that make it challenging for the discriminator to correctly identify the camera source. On the other hand, the discriminator strives to improve its accuracy in camera classification by differentiating the features.

By optimizing the encoder and discriminator simultaneously, the encoder learns to transform the features in such a way that they become invariant to camera parameters. In other words, the encoder attempts to modify the features to fool the discriminator into making incorrect camera predictions. This adversarial training encourages the encoder to generate features that capture the underlying scene geometry and dynamics rather than camera-specific intrinsics.

As including adversarial learning in the pipeline, there are two different optimizers. The first optimizer, represented by the networks in the green boxes of Figure 3.2, aims to obtain accurate depth and pose values while simultaneously maximizing the K decoder loss. By maximizing this loss, the features extracted become indistinguishable between different cameras.

This is done by the following equation:

$$L_{opt1} = (1 - \alpha)L_1 + \alpha L_{ssim} + \beta L_{smooth} - \gamma L_k, \quad (3.12)$$

where $\alpha = 0.85$, $\beta = 0.01$, $\gamma = 0.01$ and L_k is the Cross-Entropy loss function.

On the other hand, the second optimizer, represented by the network in the blue box (see Figure 3.2), focuses solely on improving the discriminator’s ability to correctly identify the camera associated with the input image. Its objective is to enhance the network’s proficiency in camera classification by:

$$L_{opt2} = L_k. \quad (3.13)$$

By following this procedure, ideally the network would obtain the same feature representation for a given scene captured by different cameras.

3.3 Neural Networks

As we have to provide a depth value for every input pixel, encoder-decoder architectures will be used. In this case, the encoder is in charge of creating a rich feature representation of the image, while decoders are used to reconstruct the depth, estimate the pose and classify the camera model from the features given by the encoder.

The encoders shown in Figure 3.1 and Figure 3.2 are built using a standard Residual Neural Network (ResNet18). Depth decoder is a fully convolutional U-Net like network, while pose decoder is a modified ResNet18 to be able to accept six channels (rotation and translation from 2 images concatenated). Finally, K decoder of Figure 3.2 is designed to be small to encourage changes in the features. It consists of convolutional layers followed by LeakyReLU activation functions, and finally a linear layer is applied to generate the classifications.

3.3.1 ResNet-18

ResNet-18 (Residual Network-18) is a deep convolutional neural network architecture [34] that is a variant of the ResNet family of models designed to address the problem of vanishing gradients and improve training deep neural networks.

The architecture of ResNet-18 consists of 18 layers, including convolutional layers, batch normalization layers, ReLU activation functions, and a global average pooling layer followed by a fully connected layer for classification. The core idea behind ResNet-18 is the use of residual connections, or skip connections, that allow the network to learn residual mappings rather than directly mapping inputs to outputs. These skip connections enable the gradient flow and alleviate the degradation problem that can occur with deeper networks.

The basic building block of ResNet-18 is the residual block. Each residual block consists of two convolutional layers with a shortcut connection that bypasses one or more convolutional layers. The shortcut connection allows the input to be added element-wise to the output of the residual block. This helps propagate gradients through the network, enabling the network to learn more effectively. The residual blocks are stacked together to form the overall architecture of ResNet-18.

3.3.2 U-Net

U-Net is a popular and widely used convolutional neural network architecture designed for image segmentation tasks [35].

The U-Net architecture consists of an encoder (contracting path) and a decoder (expanding path). The encoder gradually reduces the spatial dimensions of the input image while increasing the number of feature channels through convolutional and pooling layers. This allows the network to capture high-level semantic information from the input image.

The decoder takes the encoded feature maps from the encoder path and progressively upsamples them while reducing the number of feature channels. Each upsampling step is achieved through transposed convolutions or upsampling operations, which recover the spatial resolution lost during the encoding process. The decoder also incorporates skip connections that connect corresponding feature maps from the encoder to the decoder path at different spatial resolutions. These skip connections help retain fine-grained details and aid in the localization of segmented regions.

The architecture's contracting path and expanding path work together to form a U-shaped structure.

In this case, we only use the expanding path of U-Net, which receives features at different resolution levels that are the output of the ResNet-18 encoder.

Chapter 4

Dataset

In this chapter, the dataset used to carry out this project, as well as the simulator used to generate it, will be explained in detail. Furthermore, we will compare our custom dataset with the KITTI dataset, which is the most used in similar benchmarks.

4.1 Dataset Description

The project is based on monocular depth and pose estimation through self-supervised deep learning and adversarial learning to provide increased robustness to camera intrinsics. CARLA simulator [1], further described in Section 4.3, has been used to generate synthetic data with different realistic scenes. We have generated a total of 8400 images, distributed in 5 video sequences (sampling rate = 10 fps), all of them with a resolution of 1980 x 1080. In order to develop an algorithm that is robust to camera changes, it is necessary to use data with varied camera calibration parameters. Thus, we have considered 5 different cameras for each video sequence by using different focal distances (40, 60, 80, 100 and 120). Consequently, our final dataset has a total of 42000 images (8400 images x 5 cameras), available both in RGB and at depth level.

Sequence	# Images	Image's Size		Camera ID	FOV
1	2000	1980 x 1080		1	40
2	2000	1980 x 1080		2	60
3	2000	1980 x 1080	⊗	3	80
4	1200	1980 x 1080		4	100
5	1200	1980 x 1080		5	120

Table 4.1: Dataset description. It contains 5 sequences; 8400 images. Each of them has been captured by 5 different cameras, therefore we have a total of 42000 images.

It is important to mention from Table 4.1 that sequences 1, 2 and 3 will be used to train the model, while sequences 4 and 5 will only be used for the evaluation phase. Moreover, each of the training sequences belongs to a different map of the simulator. On the other hand, Test Set 4 consists of an unseen trajectory captured in one of the maps seen during training; sequence 5 has been generated in an unseen map.

Below, Figure 4.1 shows 5 different pairs of RGB-Depth images available in our dataset, thus we can visually appreciate the quality of the generated images and some of the different scenarios represented in our data.



(a) Sequence 1 - RGB



(b) Sequence 1 - Depth



(c) Sequence 2 - RGB



(d) Sequence 2 - Depth



(e) Sequence 3 - RGB



(f) Sequence 3 - Depth



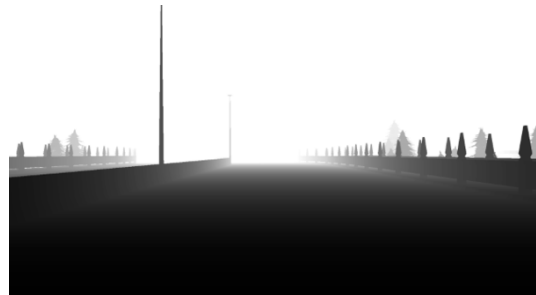
(g) Sequence 4 - RGB



(h) Sequence 4 - Depth



(i) Sequence 5 - RGB



(j) Sequence 5 - Depth

Figure 4.1: Visual examples of the dataset.

4.2 Complexity of test sets

As explained in Section 4.1, we have two test sets: Test Set 4 and Test Set 5. Both use the same parameters for the initial configuration of the simulation environment, except for the map where the sequence has been captured. These parameters consist on the insertion of objects in the map, with the following quantities:

- Motorbikes, bikes, cars = 10 each
- Trucks = 6
- People = 15

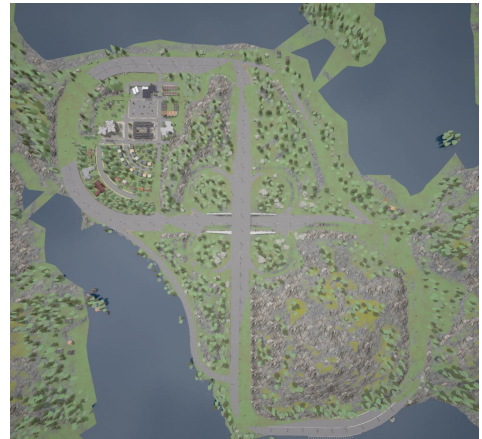
It is important to note that these numbers do not determine the exact amount of objects that will appear in the sequence. Alternatively, these values represent the number of objects that have been inserted into the map for the simulation, so the actual appearance of objects in the sequence will depend on the recorded trajectory and the stochastic movement of those objects.

On the other hand, there are significant differences in the complexity of the two test sets due to the characteristics of the map used. As we can see in Figure 4.2, Test Set 4 is generated in an urban environment with a large presence of buildings, while Test Set 5 takes place in a rural environment, almost without buildings.

Consequently, it can be considered that Test Set 4 has a higher complexity compared to Test Set 5. Hence, it is expected that this difference will be reflected in the results of the experiments, i.e. the errors obtained in Test Set 4 should be slightly higher than in Test Set 5.



(a) Test Set 4



(b) Test Set 5

Figure 4.2: Aerial view of both test sequence maps.

4.3 CARLA Simulator

The CARLA simulator [1] is a popular and highly adaptable open-source tool with a wide range of resources for, among other things, the creation of synthetic autonomous driving data in urban or rural environments. This simulator is used by many researchers, specially for the purpose of developing and testing machine learning algorithms for tasks such as pose and depth estimation, semantic segmentation and autonomous driving systems. Therefore, CARLA provides an accurate simulation of vehicles, traffic, weather conditions and many other factors.

On the one hand, the use of the CARLA simulator for the creation of a synthetic dataset has multiple benefits. Firstly, it allows us to test algorithms in a safe and controlled environment, avoiding risks of material damage or injury in certain simulations. Furthermore, it enables easier generation of a much larger amount of high quality data with a wide variety of conditions, which can improve the model’s ability to generalize to new and unseen situations. In addition, CARLA is very versatile in generating data in a realistic way, as it allows us to select different times of day, different weather conditions, set the number and variety of objects in the scene, configure their movement and speed, and modify many other parameters such as road geometry to create different scenarios.

Another point to highlight from CARLA simulator is the large number of sensors available on it to obtain annotations, such as RGB cameras, depth cameras, navigation satellite system (GNSS), among others. Moreover, it is also important to mention that the use of synthetic data allows the creation of perfect labels or ground truth, which is specially useful in supervised, semi-supervised or, as in the case of this project, self-supervised learning tasks, as it enables accurate evaluation of the models.

For all the above reasoning, we can assume that CARLA is a very valuable tool for the task of creating high quality synthetic datasets and to evaluate deep learning algorithms in the framework of this project.

Nevertheless, even though the simulator offers a high degree of customization and flexibility, as it allows us to create a wide variety of accurate and varied synthetic data, it is also necessary to take into account as a limitation that there is always a gap between this data and the real one. Thus, the results obtained may not capture all the complexity and variability of real-world data, which may limit the model’s ability to generalize to unseen situations, thus hindering its applicability to real-world situations.

Concerning depth and pose annotations provided by CARLA, it is remarkable that they are automatically generated during the simulation and are available for use. Specifically, the annotations from the depth camera are raw data of a particular scene, codifying the distance of each pixel relative to the camera, therefore creating a map of depth with millimetric precision. On the other hand, pose annotations can be obtained from its position in the global reference coordinate system of the map. As these are synthetic data, we certainly know that the annotations are accurate, detailed and free of human error.

4.4 Custom Dataset vs KITTI

As KITTI is the standard benchmark for state-of-the-art depth estimation, we introduce it and compare it with our custom dataset.

Significant differences exist in terms of origin and characteristics between the KITTI dataset [28] and the custom dataset generated with CARLA simulator.

The KITTI Vision dataset has been created specifically for the development of challenging benchmark tests in the field of autonomous driving. An autonomous driving platform equipped with high-resolution cameras, a Velodyne laser scanner and an advanced localisation system has been used. This dataset covers several tasks, such as stereo, optical flow, visual odometry/SLAM and 3D object detection. It contains 389 stereo and optical flow image pairs, 39.2 km long stereo visual odometry sequences and more than 200,000 3D object annotations. In addition, the stereo and optical flow images have a resolution of 1240×376 pixels after a rectification process.

On the other hand, the customised dataset was generated using the CARLA simulator, which provides an accurate simulation of vehicles, traffic, weather conditions and other relevant factors. In this case, a total of 8400 images of 1980×1080 resolution were generated, distributed in 5 different video sequences, each captured with 5 cameras with diverse focal distances. This results in a dataset of 42000 images in total.

In order to develop depth estimation and visual odometry algorithms that are robust to changes in intrinsic camera parameters, the custom dataset generated with CARLA is more suitable. This is because the CARLA simulator allows precise control and customisation of intrinsic camera parameters such as focal length, field of view and distortion.

By having the ability to adjust these parameters in the custom dataset, different camera configurations can be simulated in a systematic and controlled way. This results in a synthetic dataset that reflects a wide variety of conditions that might be encountered in the real world. By training depth estimation and visual odometry models with data that encompass changes in intrinsic camera parameters, the model's ability to adapt to variations in these parameters is improved, which is one of the main motivations for the thesis.

On the other hand, the KITTI dataset provides real-world data that may contain limitations in camera calibration and errors in the intrinsic parameters. These errors can affect the accuracy of depth estimation and visual odometry methods, especially when looking for robustness to changes in intrinsic camera parameters. In contrast, the custom dataset ensures accurate calibration and error-free depth and pose annotations, as they are automatically generated during simulation.

From the above, we can say that the custom dataset generated with the CARLA simulator is more appropriate for developing depth estimation and visual odometry methods that are robust to changes in intrinsic camera parameters.

Chapter 5

Experiments and Results

Throughout this chapter we will explain the different metrics used to evaluate the models, as well as the experimental procedure carried out during the thesis. Moreover, we will show the results obtained in each of the experiments and interpret them for further analysis.

5.1 Evaluation Metrics

In this section we will detail the different metrics that will be used to evaluate both depth and pose estimation performance.

For depth evaluation, we are going to use the following metrics: Absolute Relative Error, RSE, RMSE, Log Scale Invariant RMSE and Accuracy under a threshold [36]. It is important to note that, due to the monocular nature of the predicted depth, determining the correct scale becomes ambiguous. As a result, it is necessary to scale the predicted depth to match the ground truth. To achieve this, we adopt a commonly used approach in the literature, where the predicted depth is scaled by the median value of the ground truth depth. This scaling process helps align the predicted depth with the true depth values, enabling more accurate comparisons and evaluations through the mentioned metrics.

In the case of odometry evaluation, we are just going to use the RMSE metric. In addition, we are going to apply a 3D alignment technique called “Least-Squares Fitting of Two 3-D Point Sets” [37] prior to the calculation of the RMSE.

This alignment consists of, given two sets of 3D points and their correspondence, returning a least square optimal rigid transform (also known as Euclidean) between them. Therefore, the process consists of finding the transformation parameters (translation, rotation and, optionally, scaling) that minimise the sum of the squared distances between the transformed points in one set and their corresponding points in the other set. This has been achieved through Singular Value Decomposition (SVD).

As follows, the mentioned metrics are going to be detailed. We will denote \hat{d}_p as the depth prediction at pixel p and d_p as the actual depth value at pixel p .

5.1.1 Absolute Relative Error

The Absolute Relative Error provides an indication of how accurate is model at predicting depth compared to the actual values. It serves as a measure of precision, where a lower value suggests a higher level of accuracy/precision.

$$Abs.Rel.Error = \frac{1}{N} \cdot \sum_{p=0}^N \frac{|d_p - \hat{d}_p|}{d_p} \quad (5.1)$$

As we can see in the above equation, the Absolute Relative Error is calculated by dividing the absolute magnitude of the difference between the predicted depth values and the actual depth values by the actual depth value. This relative error is then averaged over the total number of pixels in the frame, N .

5.1.2 RSE

The Relative Squared Error, RSE, is a widely used metric to measure the discrepancy between predicted and actual depth values in a relative manner, according to the following formula:

$$RSE = \frac{1}{N} \cdot \sum_{p=0}^N \frac{(d_p - \hat{d}_p)^2}{d_p} \quad (5.2)$$

The RSE is calculated by taking the difference between the predicted depth values and the actual depth values, squaring them and dividing by the actual depth value. This relative squared error is then averaged over all pixels in the frame. By squaring the errors and normalising them to the actual depth values, the RSE penalises larger errors more. Therefore, a lower RSE value indicates higher accuracy.

5.1.3 RMSE

The Root Mean Squared Error, RMSE, is calculated by taking the square root of the mean squared errors between the predicted depth values and the actual depth values. The mean is used to ensure that the error is averaged over all pixels in the frame. The equation for its computation is:

$$RMSE = \sqrt{\frac{1}{N} \cdot \sum_{p=0}^N (x_p - \hat{x}_p)^2}, \quad (5.3)$$

where, in the case of depth evaluation, $x_p = d_p$ and $\hat{x}_p = \hat{d}_p$.

On the other hand, in odometry evaluation x_p is the p^{th} value (out of T) of the ground-truth pose in XYZ coordinates and \hat{x}_p is its correspondent predicted value.

5.1.4 Log Scale Invariant RMSE

The Log Scale Invariant RMSE is a widely used depth estimation metric that takes into account the invariance to the logarithmic scale. This type of scale is used because of the non-linear nature of depths in the scene. Instead of working directly with the actual depths, we take the logarithm of the depths before performing the calculations. In this way, we avoid the inherent ambiguity of monocular self-supervised depth estimation regarding to the large source of uncertainty coming from the overall scale, by using a scale-invariant error [38]. This metric is computed as:

$$\text{Log Scale Invariant RMSE} = \sqrt{\frac{1}{N} \cdot \sum_{p=0}^N (\log(d_p) - \log(\hat{d}_p))^2} \quad (5.4)$$

5.1.5 Accuracy under a threshold

Unlike the rest of the traditional metrics explained above, which assess the magnitude of errors without taking into account a specific acceptance range, the accuracy under a threshold metric considers a set threshold or limit, thus allowing to assess whether the depth predictions fit within an acceptable range of error.

The formula used for the calculation of this metric is:

$$\text{Acc. under Threshold} = \max\left(\frac{d_p}{\hat{d}_p}, \frac{\hat{d}_p}{d_p}\right) < \delta, \quad (5.5)$$

being δ the specific threshold established. In this case, we have decided to evaluate the performance under three different thresholds: $\delta_1 = 1.25$, $\delta_1 = 1.25^2$ and $\delta_1 = 1.25^3$.

5.2 Experimental procedure

In this section, we are going to explain the experimental procedure followed in the implementation of the algorithm. First of all, it should be noted that, in order to decrease the computational cost, a reduced version of the original dataset has been used. As explained in Chapter 4, the dataset consists of 3 sequences \times 5 different cameras \times 2000 images. However, for our experiments a subset consisting of 3 sequences \times 5 cameras \times 500 images has been selected. Hence, we have a total of 7500 training images instead of 30000.

In the same way, in order to mitigate the computational cost, we have applied resizing to each of the images from an initial resolution of [1980, 1080] to [576, 960], both in train and test phases. This change in the scale of the images has significantly accelerated the training during the experimental process, as well as the GPU memory consumption.

The experiments carried out in this study are based on the training and evaluation (in terms of both depth estimation and visual odometry) of three different models:

- **Baseline.** This model has been trained only with images captured by camera 1 (detailed architecture in Section 3.1). Thus, it serves as a reference to compare the performance of the other two models. ~ 20 h of training.
- **Multiseq.** In this case, the model has the same architecture as the baseline one, but the training has been performed using images captured by cameras 2, 3 and 4. This strategy takes advantage of visual information from multiple image “perspectives”, which is expected to result in improved results comparing to the baseline model. ~ 3 days of training.
- **Multiseq-adv.** The architecture of this model is described in Section 3.2. The training has been performed using images captured by cameras 2, 3 and 4, as Multiseq model. However, the adversarial training technique has been included in order to achieve greater robustness to changes in the intrinsic parameters of the camera. Therefore, the model is expected to learn to generalise better in situations where these parameters may vary. ~ 5 days of training.

In each experiment, the first step consists of selecting the cameras with which the model will be trained and loading the data associated to them. Then, the calibration matrix of each chosen camera, K , is extracted as follows:

```

image_w = 1980
image_h = 1080
resized_w = 576
resized_h = 960
focal = image_w / (2.0 * tan(FOV * pi / 360.0))
K = identity(3)
K[0, 0] = K[1, 1] = focal
K[0, 2] = image_w / 2.0
K[1, 2] = image_h / 2.0
K[0, :] *= (resized_w/image_w)

```

Figure 5.1: Calculation of the K matrix

In Figure 5.1, it is worth remembering that 1980×1080 are the original dimensions of the images, while 576×960 are the dimensions of the resized images. On the other hand, the term ‘FOV’ is one of the intrinsic parameters of the camera that refers to the Field Of View value.

In each experiment, once the data has been loaded and the corresponding K matrices have been extracted, we proceed to run the iterative learning algorithm, which is explained below.

1. First, a random image is selected and loaded from the dataset, as well as its previous and next frame, using the DataLoader. During this step, the ID of the camera that captured these images is also obtained.
2. Gaussian noise is then introduced into the three loaded images. This noise is added with a probability of 1, mean of 0.0 and a standard deviation of 0.01, in order to introduce variability in the images.
3. Then, the encoder is used to extract the features from the images. Thus, we obtain high quality representations that capture the relevant information of each image.
4. (a) Next, the pose decoder is used to receive as input the features extracted from the previous image and the target image, both concatenated. From these features, the pose decoder produces the axisangle and the translation matrix, which are used to construct a rotation-translation matrix representing the estimated pose change between the two frames. The same procedure is performed using the concatenated target and next images.

(b) In turn, the depth decoder receives as input the features extracted from the target image by the encoder, thus generating a disparity value for each pixel of the image. From the inverse of these values, the depth is obtained. Then, a scaling is performed using predefined minimum and maximum depth values.
5. In the case of Multiseq-adv model, the features of the target image are also introduced in a discriminator, whose aim is to predict the camera that captured the image. This step is performed in order to apply adversarial training to improve the robustness of the model to changes in the intrinsic parameters of the camera.
6. Then, we warp the previous frame to the target one, taking into account the estimated rotation-translation matrix, the K-matrix and the predicted depth. This warping consists of projecting the previous image to the target image space using bilinear interpolation [9]. In the same way, a warping of the next image to the target one is also performed.
7. (a) Subsequently, the loss associated to the warping process is calculated. This is done by taking into account both the warping of previous-target and next-target. A linear combination of both losses is used, each calculated using equation 3.10, weighting both by 0.5.

- (b) The same loss formula is computed on the previous and next frames with the target image without warping. This technique is known as automasking (see Section 3.1.3), which avoids penalising pixels whose movement cannot be explained by camera movement alone. To do this, the minimum value between the two losses (the one that takes into account warping and the one that does not) is selected for each pixel.

These steps are repeated in each iteration of the algorithm, allowing to gradually improve the quality of the pose and depth estimations and performing for each batch the calculation of the gradients and the update of the model parameters.

On the other hand, Table 5.1 shows the most relevant hyperparameters chosen for training. It should be noted that all three models share the same hyperparameters, which ensures a fair and consistent comparison between them.

Hyperparameter	Value
Batch Size	5
Learning Rate	1e-5
Optimizer	Adam
Min depth	0.1
Max depth	100
# Epochs	300
# Workers	6

Table 5.1: Training hyperparameters configuration

Note that the parameters ‘Min depth’ and ‘Max depth’ are the ones that, as we have explained above, have been used to scale the depth values. These values have been limited in this range to avoid that the existence of pixels very far away from all the others (for example, those belonging to the sky), generate depth maps in which the differences between the rest of the objects are not appreciated.

5.3 Analysis of the results

As detailed in Section 4.1, the evaluations of the three models will be carried out on two test sets: Test Set 4 and Test Set 5. It should be noted that there is a difference in complexity between these two test sets, as justified in Section 4.2, so we have different expectations regarding the results obtained. Consequently, we expect to obtain worse results in Test Set 4 compared to Test Set 5, due to the fact that the first one was recorded in a more complex environment than the one used in the second test set.

We will now proceed to analyse and discuss these results in detail, both numerically and visually.

5.3.1 Depth Estimation

From Table 5.2 we can analyse the numerical results obtained by the different models for each of the cameras. Note that the lowest error value for each case is in bold.

Camera	Approach	abs_rel	sq_rel	rmse	rmse_log	a1	a2	a3
1	Baseline	0.22	3.44	10.45	0.27	0.69	0.91	0.97
	Multiseq	0.28	5.71	12.3	0.33	0.63	0.87	0.94
	Multiseq-adv	0.25	4.75	11.82	0.3	0.66	0.88	0.95
2	Baseline	0.26	3.18	10.26	0.3	0.6	0.89	0.96
	Multiseq	0.28	5.67	11.0	0.33	0.66	0.88	0.94
	Multiseq-adv	0.24	3.9	10.48	0.29	0.67	0.89	0.95
3	Baseline	0.3	3.35	10.42	0.35	0.51	0.83	0.95
	Multiseq	0.26	3.86	9.47	0.31	0.69	0.88	0.94
	Multiseq-adv	0.26	3.9	9.72	0.31	0.67	0.89	0.95
4	Baseline	0.37	3.79	10.62	0.41	0.43	0.74	0.89
	Multiseq	0.26	2.96	8.46	0.32	0.68	0.87	0.94
	Multiseq-adv	0.24	2.83	8.5	0.31	0.68	0.87	0.94
5	Baseline	0.39	3.64	11.45	0.48	0.33	0.66	0.84
	Multi-seq.	0.25	2.42	8.71	0.34	0.63	0.84	0.93
	Multiseq-adv	0.23	2.32	8.63	0.32	0.65	0.86	0.94

Table 5.2: Results of depth estimation over Test Set 4

As we can see in Table 5.2, the Baseline model shows better performance in the test sequence of camera 1 compared to the other models. This supports the idea that the Baseline model benefits from the consistency between the training and test images in terms of the intrinsic camera parameters.

Furthermore, we observe a worsening effect of the Baseline model as the FOV increases, since the error values increase progressively as the camera is changed. For example, the absolute error starts at 0.22 in camera 1 and increases to 0.39 in camera 5, which evidences the dependence of the models on the intrinsic parameters of the training camera.

Finally, the Multiseq-adv model provides better results on most of the error measures compared to the other models: Baseline and Multiseq. This model has proven to be more robust in terms of generalisation, as it does not appear to show dependence on the FOV value. This is reflected in the fact that the error values obtained are very similar to each other regardless of the camera on which they are evaluated. Hence, this result confirms the expected effect with the inclusion of adversarial training, showing how this change in the architecture improves the model’s robustness in the presence of unseen cameras.

Alternatively, Figure 5.2 shows an image from the Test Set 4 in RGB format captured by the 5 different cameras. In turn, we show in each case the depth predictions generated by the 3 models, which allows us to compare and analyse their performance in each of the cameras.







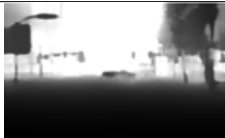
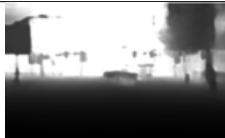












Camera	Original Image	Baseline	Multiseq	Multiseq-adv
1				
2				
3				
4				
5				

Figure 5.2: Visual results of depth estimation over Test Set 4

After a careful examination of the depth predictions in Figure 5.2, certain relevant patterns and effects have been identified.

Firstly, when analysing the predictions of the Baseline model, it is clear that it generates a relatively accurate estimate when the image has been captured by camera 1, which makes sense, since the model has been trained only with images from this camera. However, as the camera is changed and the FOV value increases, the predictions deteriorate progressively. This deterioration can be seen as a blurring effect both in areas close to the camera and in the background. This effect confirms the dependence of the depth estimation models on the intrinsic calibration parameters of the camera that recorded the training sequence, which supports the main motivation of the thesis.

On the other hand, when considering the predictions provided by the Multiseq model, it can be observed an improvement compared to the Baseline model. The reason of this improvement is that Multiseq model has been trained with images captured by different cameras (2, 3 and 4), which allows it to generalise better. However, the same deterioration effect is observed in cameras 4 and 5, although less pronounced. In addition, an inferior performance is evident in camera 1 compared to the Baseline model, as in this case it fails to correctly capture the shapes of nearby objects, such as the car in front of the camera.

Regarding the predictions of the Multiseq-adv model, it can be seen how the blurring effects as the FOV increases are much less present, which significantly improves the predictions compared to the other two models. Although, once again, the Baseline

model seems to perform better on the images captured by camera 1, Multiseq-adv model manages to correctly identify the structure of most of the nearby objects also in this camera, such as the car in front. In fact, this model is the only one that achieves at identifying this car in camera 2. Hence, this model provides better results both on the cameras used during training (2, 3 and 4) and on cameras not seen during training, such as camera 5. Furthermore, it also improves the predictions on camera 1 compared to the Multiseq model.

Now, we are going to evaluate the numerical and visual results for depth estimation over Test Set 5. Table 5.3 shows the error obtained by each of the models in all cases over this test set.

Camera	Approach	abs_rel	sq_rel	rmse	rmse_log	a1	a2	a3
1	Baseline	0.16	2.26	9.7	0.22	0.74	0.92	0.98
	Multiseq	0.21	4.15	11.69	0.26	0.75	0.89	0.96
	Multiseq-adv	0.18	3.19	10.95	0.23	0.79	0.92	0.98
2	Baseline	0.18	2.15	10.02	0.26	0.72	0.89	0.96
	Multiseq	0.18	3.47	10.44	0.26	0.78	0.89	0.96
	Multiseq-adv	0.16	2.98	9.86	0.23	0.8	0.92	0.97
3	Baseline	0.2	2.57	10.81	0.34	0.7	0.85	0.91
	Multiseq	0.17	2.53	8.69	0.25	0.79	0.91	0.97
	Multiseq-adv	0.16	2.45	8.6	0.24	0.8	0.92	0.97
4	Baseline	0.22	2.93	11.56	0.42	0.67	0.81	0.88
	Multiseq	0.2	2.46	8.44	0.3	0.74	0.86	0.92
	Multiseq-adv	0.16	2.03	8.22	0.26	0.79	0.91	0.96
5	Baseline	0.25	3.9	13.47	0.54	0.62	0.77	0.83
	Multiseq	0.21	2.36	9.29	0.34	0.71	0.84	0.9
	Multiseq-adv	0.17	2.06	9.11	0.3	0.75	0.88	0.93

Table 5.3: Results of depth estimation over Test Set 5

Despite the fact that this sequence has been captured in a map not seen during training, it has been observed to present a lower complexity compared to Test Set 4, as explained in Section 4.2. Due to this, we can appreciate in Table 5.5 a significant overall reduction in all the numerical errors obtained.

On the other hand, similar patterns than those obtained with Test Set 4 have been identified in this case in the error differences between the models. Baseline still performs the best on camera 1, while the Multiseq-adv model leads (overall) the rest of the evaluations, showing lower errors on both seen and unseen cameras. This suggests that the inclusion of adversarial training improves the robustness of the model to new cameras and appears to reduce its dependence on intrinsic camera parameters, both in known and unknown maps.

In Figure 5.3, we show an RGB image from Test Set 5 captured by each camera and the predictions obtained by the models.





















Camera	Original Image	Baseline	Multiseq	Multiseq-adv
1				
2				
3				
4				
5				

Figure 5.3: Visual results of depth estimation over Test Set 5

After examining the images of Figure 5.3, it has been observed that there is a deterioration in the performance of the Baseline model as the FOV increases, as well as in Test Set 4. However, this model provides the best visual results for images recorded with camera 1.

On the other hand, as in the previous case, Multiseq model shows an improvement compared to the Baseline model. However, it still faces difficulties in generalising to unseen cameras, especially with a high FOV (camera 5).

Finally, Multiseq-adv model seems to offer the best results in terms of depth predictions, mostly for the scene background. However, in this case some artifacts are observed at the top of the images generated with cameras 3, 4 and 5, apparently caused by the presence of clouds.

5.3.2 Visual odometry

As follows, we will evaluate numerically and graphically the results obtained by the three models with respect to visual odometry.

Table 5.4 shows the exact error values obtained by comparing the ground-truth pose with the predicted ones over Test Set 4 for each of the cameras and models.

Camera	Approach	RMSE
1	Baseline	37.2176687
	Multiseq	42.9560020
	Multiseq-adv	44.0374543
2	Baseline	33.0748976
	Multiseq	39.6795019
	Multiseq-adv	39.7457542
3	Baseline	38.1689396
	Multiseq	34.4765797
	Multiseq-adv	35.2289832
4	Baseline	45.2183431
	Multiseq	37.9663446
	Multiseq-adv	38.4187968
5	Baseline	44.9504420
	Multiseq	42.5429585
	Multiseq-adv	41.6491539

Table 5.4: Results of visual odometry over Test Set 4

Firstly, it stands out from Table 5.4 that the baseline model obtains the best results both for cameras 1 and 2. This is consistent, as this model has been trained with the smallest FOV, which allows it to perform better on cameras with the same FOV value or a very similar one.

On the other hand, Multiseq model provides the best results in cameras 3 and 4, in contrast to the conclusions drawn from depth estimation. However, the error values are practically identical between Multiseq and Multiseq-adv models. Furthermore, Multiseq-adv shows the best performance in camera 5, which has not been seen by any of the models. Thus, although the differences between the results are small and do not allow conclusive information to be extracted, it can be assumed that Multiseq-adv has poor results but demonstrates a slightly higher generalisability on unseen cameras.

Figure 5.4 shows the ground-truth (blue) and predicted trajectories (orange) for each model and camera over Test Set 4. This way, the figure allows us to evaluate the visual differences between the models, as well as to analyse the effect of the FOV on the pose estimation task.

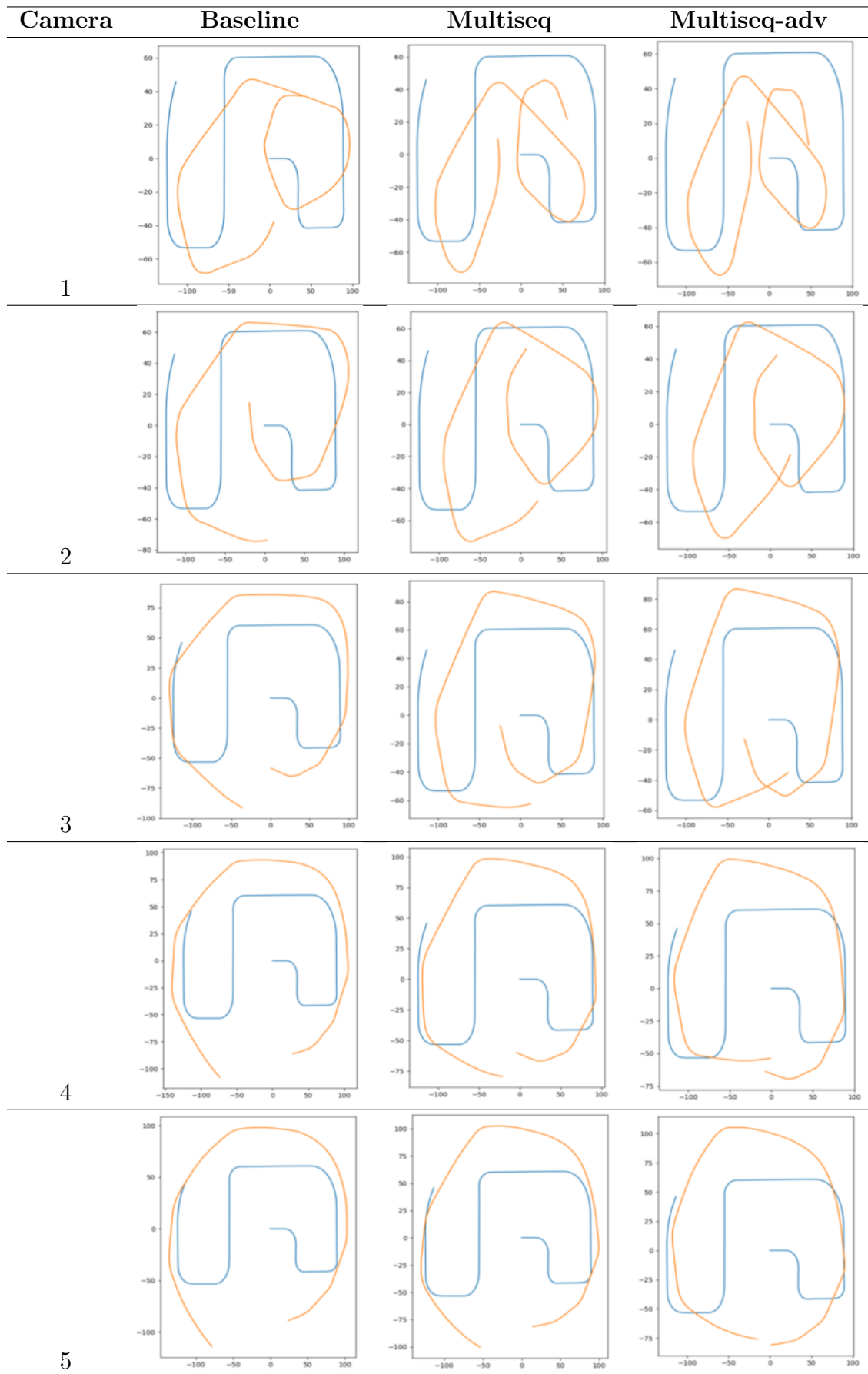


Figure 5.4: Graphical results of visual odometry over Test Set 4

We observe in Figure 5.4 that the ground truth trajectory remains constant as the FOV increases. This is logical, since the intrinsic parameters of the camera have no influence on the trajectory followed by the on-board camera. However, there is a deterioration in the pose predictions as this value increases, which is evidence of the dependence of the models on this variable.

In general, we can appreciate that estimated poses are very poor, as none of them manages to correctly model the real trajectory. This may be due to the complexity of both the environment and the generated random trajectory, which represents a challenge for the evaluated models. It is possible that more data could improve the performance in this case. However, based on the graphical results obtained, we can only draw the above-mentioned conclusion about the FOV effect on visual odometry.

On the other hand, Table 5.5 shows the numerical results on Test Set 5.

Camera	Approach	RMSE
1	Baseline	64.2253016
	Multiseq	94.2972567
	Multiseq-adv	70.3132984
2	Baseline	24.6336585
	Multiseq	16.6271762
	Multiseq-adv	11.2942427
3	Baseline	33.8584413
	Multiseq	21.7097867
	Multiseq-adv	14.5281325
4	Baseline	60.0052456
	Multiseq	36.1814413
	Multiseq-adv	17.1696794
5	Baseline	95.5537893
	Multiseq	60.4466667
	Multiseq-adv	24.1619515

Table 5.5: Results of visual odometry over Test Set 5

After examining the results of Table 5.5, we can see more consistency compared to those obtained with Test Set 4. This may be because this ground-truth trajectory is simpler, therefore, it could be modelled more accurately.

Moreover, the numerical results suggests that all models have poor performance when evaluated with camera 1. In this case, the Baseline model provides the lowest error compared to the other two models.

Both the Baseline and Multiseq models show an increase in error values as they are evaluated with higher FOV cameras, from camera 2 to camera 5. However, the Multiseq-adv model manages to provide considerably lower error values. Although the latter model also experiences an increase in its RMSE with increasing FOV, it manages to mitigate this effect to a large extent compared to the other two models thanks to the inclusion of adversarial training in its architecture.

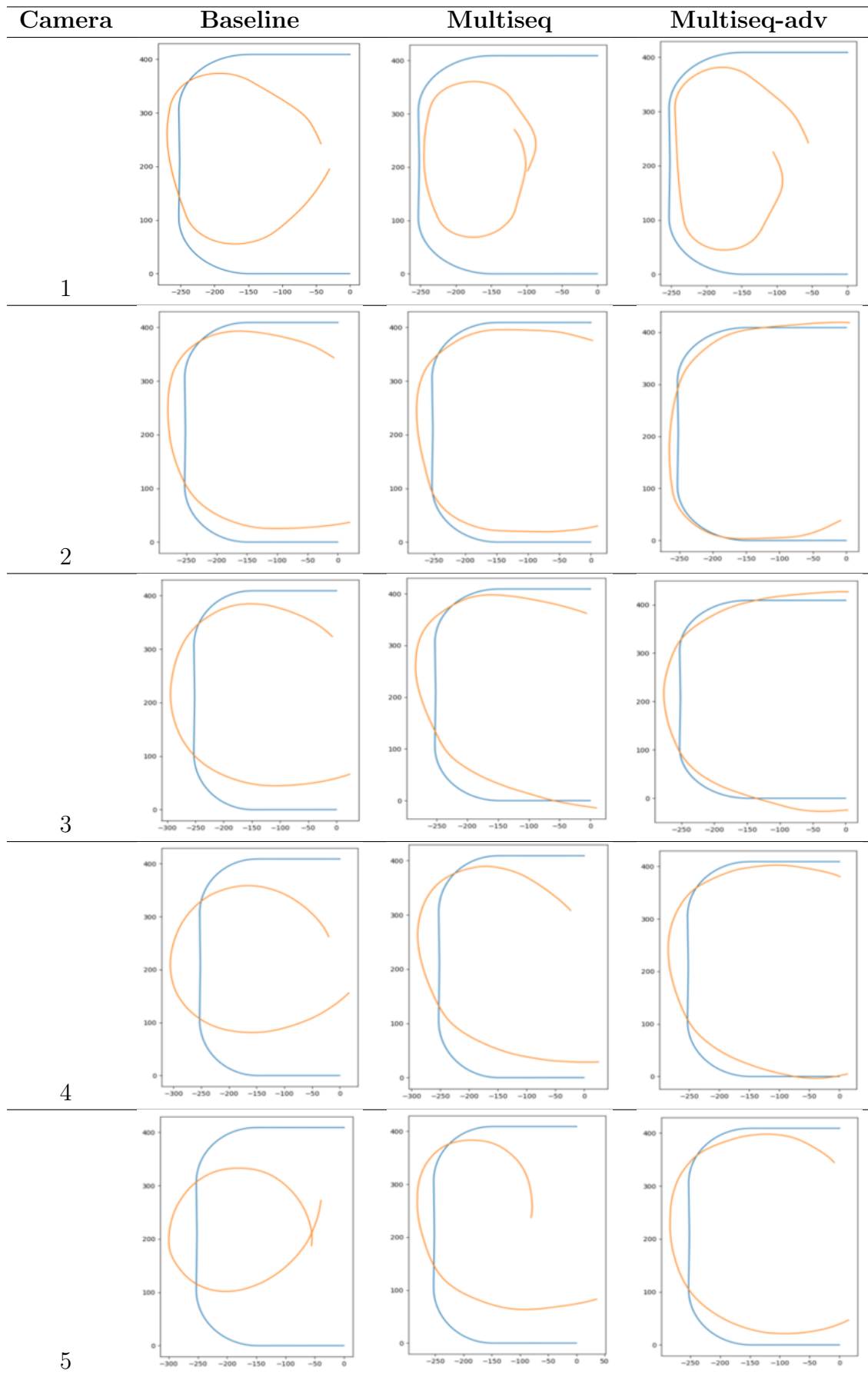


Figure 5.5: Graphical results of visual odometry over Test Set 5

We can observe from Figure 5.5 that, in general, all models seem to provide poor pose estimates for camera 1, according to Table 5.5. This can be attributed to the fact that the smallest FOV is used in camera 1, which can result in images with excessive “zoom”. The presence of multiple objects close to the camera could be hindering accurate trajectory estimation.

On the other hand, it can be observed that the baseline model shows a progressive deterioration in performance as we move from camera 2 to camera 5, again highlighting its dependence on intrinsic camera parameters.

Both the Multiseq and Multiseq-adv models perform well in cameras 2, 3 and 4, although there is a slight improvement in the performance of Multiseq-adv model. Furthermore, Multiseq-adv shows significantly better results compared to the other models when evaluated in camera 5, demonstrating a higher robustness to unseen cameras.

Chapter 6

Conclusions

In this chapter we will review the fulfilment of the objectives set for the development of the thesis. Furthermore, we will present some ideas for future work to continue with this research line. Finally, the conclusions obtained throughout the thesis will be presented.

6.1 Review of objectives

This project has been successful in meeting the objectives set out in Section 1.2.

Firstly, a self-supervised deep learning model capable of estimating both the depth of each image in a video sequence and its trajectory has been designed and implemented, demonstrating robustness to changes in the intrinsic calibration parameters of the camera through adversarial learning.

In terms of secondary objectives, extensive knowledge in the area has been acquired by studying and analysing the limitations of the state of the art in depth estimation and visual odometry. In addition, a customised dataset has been created that contains several realistic sequences and allows for fair comparisons between sequences captured by different cameras. This has provided a suitable evaluation environment for studying the effects of the FOV value on the performance of a baseline model.

The proposed architecture has also been evaluated in terms of depth estimation and visual odometry, comparing the results obtained with those of the baseline architecture.

Finally, in general, the experiments have shown improvements with the inclusion of adversarial learning in the proposed model.

In summary, this project has successfully met all the objectives set, advancing the knowledge and application of depth estimation and visual odometry using a self-supervised model that is robust to changes in the intrinsic parameters of the camera.

6.2 Future work

Although this research has shown the potential of adversarial learning in adding robustness to camera variations, there are still areas that could be explored in future research. The following ideas are proposed:

- Training the model on the full dataset. As it has more examples, it would potentially allow modelling more complex environments, such as Test Set 4. This could help to improve the generalisability of the model and its performance in more challenging scenarios.
- Evaluation on the KITTI dataset [28], as it is widely used in depth estimation tasks. This would allow demonstrating in this dataset, that is recognised in the scientific community, the improvement in the results obtained by incorporating adversarial training .
- Independent analysis of depth estimation and visual odometry. For a deeper understanding of the results and targeted improvement, the individual effect of depth estimation and visual odometry on final error could be investigated.

These research lines of future work may expand knowledge and improve results in the field of depth estimation and visual odometry.

6.3 Conclusion

In this Master thesis, the challenge of estimating depth in conjunction with visual odometry in video sequences using self-supervised deep learning models has been addressed. Clear objectives were set and comprehensive experiments were conducted to evaluate the performance of three proposed models: Baseline, Multiseq and Multiseq-adv. Through analysis of the results, significant conclusions have been drawn.

Firstly, we have shown the dependence of the models on the intrinsic camera parameters, especially the Baseline one in Depth estimation and the three of them in terms of visual odometry estimation. As the FOV increased, a deterioration in the accuracy of the results was observed.

Furthermore, it was found that the inclusion of adversarial training in the Multiseq-adv model significantly improved the robustness to unseen cameras and reduced the dependence on intrinsic parameters. This model demonstrated superior performance in terms of depth estimation and visual odometry.

Despite the progress made, there are open lines of research to continue this work, as explained in Section 6.2.

In conclusion, this work has achieved the proposed objectives, demonstrated improved performance by incorporating adversarial training, and identified challenges and areas for improvement. Thus, these results open up new opportunities for the application of self-supervised deep learning models and stimulate future research in this line.

Bibliography

- [1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, pages 1–16, 2017. [i](#), [iii](#), [27](#), [30](#)
- [2] Mary Chris Roperos Go. *Self-supervised monocular depth estimation and visual odometry on unseen cameras*. TFM-UAM, 2022. [2](#)
- [3] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016. [6](#), [18](#)
- [4] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. [6](#), [18](#), [19](#)
- [5] Huan Yu, Ling Xie, Jiabin Chen, Chunlei Song, and Fei Guo. Visual odometry based on improved feature matching and unscented kalman filter. In *2016 35th Chinese Control Conference (CCC)*, pages 5446–5450. IEEE, 2016. [7](#)
- [6] Volker Nannen and Gabriel Oliver. Grid-based spatial keypoint selection for real time visual odometry. In *ICPRAM*, pages 586–589, 2013. [7](#)
- [7] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017. [7](#)
- [8] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5667–5675, 2018. [7](#), [9](#)
- [9] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017. [8](#), [37](#)
- [10] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1983–1992, 2018. [8](#)
- [11] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings*

- of the *IEEE/CVF conference on computer vision and pattern recognition*, pages 12240–12249, 2019. 8
- [12] Hanhan Li, Ariel Gordon, Hang Zhao, Vincent Casser, and Anelia Angelova. Un-supervised monocular depth learning in dynamic scenes. In *Conference on Robot Learning*, pages 1908–1917. PMLR, 2021. 8
- [13] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019. 8
- [14] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 582–600. Springer, 2020. 9
- [15] Tianwei Shen, Zixin Luo, Lei Zhou, Hanyu Deng, Runze Zhang, Tian Fang, and Long Quan. Beyond photometric loss for self-supervised ego-motion estimation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6359–6365. IEEE, 2019. 9
- [16] Jiexiong Tang, Rares Ambrus, Vitor Guizilini, Sudeep Pillai, Hanme Kim, Patric Jensfelt, and Adrien Gaidon. Self-supervised 3d keypoint learning for ego-motion estimation. In *Conference on Robot Learning*, pages 2085–2103. PMLR, 2021. 9, 10
- [17] Huangying Zhan, Chamara Saroj Weerasekera, Ravi Garg, and Ian Reid. Self-supervised learning for single view depth and surface normal estimation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4811–4817. IEEE, 2019. 9
- [18] Wang Zhao, Shaohui Liu, Yezhi Shu, and Yong-Jin Liu. Towards better generalization: Joint depth-pose learning without posenet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9151–9161, 2020. 9, 10
- [19] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. In *Conference on Robot Learning*, pages 1761–1772. PMLR, 2021. 10, 11
- [20] Huangying Zhan, Chamara Saroj Weerasekera, Jia-Wang Bian, and Ian Reid. Visual odometry revisited: What should be learnt? In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 4203–4210. IEEE, 2020. 10
- [21] Wenshan Wang, DeLong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020. 11

- [22] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan 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*, pages 8934–8943, 2018. 11
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 11
- [24] Jose M Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Cam-convs: Camera-aware multi-scale convolutions for single-view depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11826–11835, 2019. 11
- [25] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 11
- [26] Shunkai Li, Xin Wu, Yingdian Cao, and Hongbin Zha. Generalizing to the open world: Deep visual odometry with online adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13184–13193, 2021. 12
- [27] Yuesong Wang, Keyang Luo, Zhuo Chen, Lili Ju, and Tao Guan. Deepfusion: A simple way to improve traditional multi-view stereo methods using deep learning. *Knowledge-Based Systems*, 221:106968, 2021. 13
- [28] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 13, 15, 31, 50
- [29] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 14
- [30] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2022–2030, 2018. 18, 20
- [31] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 18
- [32] Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3828–3838, 2019. 21
- [33] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2624–2641, 2019. 21

- [34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 24
- [35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 24
- [36] Cesar Cadena, Yasir Latif, and Ian D Reid. Measuring the performance of single image depth estimation methods. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4150–4157. IEEE, 2016. 33
- [37] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, 5:698–700, 1987. 33
- [38] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014. 35