Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Thiago Cutrim dos Santos

Improving Visual SLAM by Combining Depth Estimation, Semantic Segmentation, and Dynamic Object Removal Using Visual Foundation Models

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática, do Departamento de Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Sérgio Colcher

Rio de Janeiro
April 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Thiago Cutrim dos Santos

Improving Visual SLAM by Combining Depth Estimation, Semantic Segmentation, and Dynamic Object Removal Using Visual Foundation Models

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Sérgio Colcher
Advisor
Departamento de Informática – PUC-Rio

Prof. Julio Cesar Duarte
Instituto Militar de Engenharia - IME

Prof. Edward Hermann Haeusler
Departamento de Informática - PUC-Rio

Prof. Antonio José Grandson Busson
BTGPactual

Rio de Janeiro, April 19th, 2024

Pedro Thiago Cutrim dos Santos

Graduated in Computer Science by the Universidade Federal do Maranhão.

Bibliographic data

Santos, Pedro Thiago Cutrim dos.

Improving Visual SLAM by Combining Depth Estimation, Semantic Segmentation, and Dynamic Object Removal Using Visual Foundation Models / Pedro Thiago Cutrim dos Santos; advisor: Sérgio Colcher. – 2024.

60 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2024.

Inclui bibliografia

1. Informática – Teses. 2. SLAM. 3. Estimação de Profundidade. 4. Modelos Fundacionais Visuais. 5. Yolov8. I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

I would like to express my deepest gratitude to my family. Especially to my aunts, thank you for your unwavering support, encouragement, and love throughout my academic journey. Your belief in my abilities has been my greatest source of motivation.

To my colleagues and my advisor, thank you for your collaboration, support, and camaraderie throughout this journey. Your insights, feedback, and encouragement have been crucial in shaping this work. The countless hours spent together in discussions and shared challenges have made this experience both enriching and enjoyable.

To my dear friends, I extend my heartfelt gratitude. Your companionship, support, and understanding have been invaluable throughout this journey. Thank you for the countless moments of laughter, your patience, and for always believing in me. Whether it was through late-night gaming sessions, encouraging words, or simply being there to listen, you have all played a significant role in helping me reach this milestone. I am incredibly fortunate to have such amazing friends in my life, and this accomplishment is as much yours as it is mine. Thank you for everything.

Finally, I dedicate this work to Ednara, my love whom I lost during the course of this master's program. Thank you for all the moments of happiness we shared together. Even though I feel a constant sadness and longing for you, it was through your love that I found the strength to continue with this master's program and with my life. I will love you forever.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and by the Air Force Office of Scientific Research under award number FA9550-22-1-0475.

Abstract

Santos, Pedro Thiago Cutrim dos.; Colcher, Sérgio (Advisor). **Improving Visual SLAM by Combining Depth Estimation, Semantic Segmentation, and Dynamic Object Removal Using Visual Foundation Models**. Rio de Janeiro, 2024. 60p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The goal of a SLAM (Simultaneous Localization and Mapping) system is to estimate the camera's trajectory in space while reconstructing an accurate map of the surrounding environment. Its definition can be explained in two parts: the first one, mapping an unknown environment, and the second, performing agent localization in this environment through available sensors. Among the different types of sensors, cameras have lower operating costs while providing a rich amount of environmental information that allows for more precise mapping. Because of this, solutions where only the use of the camera is employed as the main sensor, called Visual SLAM Systems, are of great interest. This work proposes an adaptation of a Visual SLAM System that uses Visual Foundation Models to generate depth images that assist in the robustness of mapping and localization in the environment. Additionally, such a system should also be capable of identifying dynamic elements in the environment and removing them from the map, through the use of computer vision models. Finally, this should be viable for real-time applications.

Keywords

SLAM; Depth Estimation; Visual Foundation Models; Yolov8.

Resumo

Santos, Pedro Thiago Cutrim dos.; Colcher, Sérgio. **Aperfeiçoando Modelos de SLAM Visuais pela Combinação da Estimação de Profundidade, Segmentação Semântica e Remoção de Objetos Dinâmicos Usando Modelos Fundacionais Visuais**. Rio de Janeiro, 2024. 60p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O objetivo de um sistema SLAM (Localização e Mapeamento Simultâneos) é estimar a trajetória da câmera no espaço enquanto reconstrói um mapa preciso do ambiente ao redor. Sua definição pode ser explicada em duas partes: a primeira, mapear um ambiente não conhecido, e a segunda, realizar a localização do agente neste ambiente através dos sensores disponíveis. Dentre os diferentes tipos de sensores, câmeras possuem um custo menor de operação ao mesmo tempo que fornecem uma quantidade rica de informações do ambiente que permitem um reconhecimento e mapeamento mais preciso. Devido a isso, soluções onde apenas o uso da câmera é utilizado, chamado de Sistemas SLAM Visuais, são de grande interesse. Este trabalho propõe a adaptação de um Sistema SLAM que necessite apenas de uma câmera como sensor principal e que use Visual Foundation Models para gerar imagens de profundidade que auxiliem na robustez do mapeamento e localização no ambiente. Além disso, tal sistema também deve ser capaz de identificar elementos dinâmicos no ambiente e removê-los do mapa, através do uso de modelos de visão computacional. E por fim, deve ser viável para aplicações em tempo real.

Palavras-chave

SLAM; Estimação de Profundidade; Modelos Fundacionais Visuais; Yolov8.

Table of contents

List of figures

List of Abreviations

RGB – Red Green Blue

RGB-D – Red Green Blue and Depth

SLAM – Simultaneous Localization and Mapping

V-SLAM – Visual Simultaneous Localization and Mapping

FM – Foundation Models

VFM – Visual Foundation Models

YOLO – You Only Look Once

LIDAR – Light Detection and Ranging

VAE - Variational Autoencoders

"The light of the mind alone cannot burn away all darkness."

Joshua Graham, *Fallout: New Vegas.*

1
Introduction

Simultaneous Localization and Mapping (SLAM) systems have played a fundamental role in various applications, from autonomous vehicles to robotics. These systems aim to determine the camera's trajectory in space while simultaneously reconstructing a precise map of the surrounding environment (KAZEROUNI et al., 2022).

SLAM can be utilized to cover a wide range of applications, such as domestic robots, autonomous vehicles, drones, video games and virtual/augmented reality devices (BARROS et al., 2022). Given the expanding use of robotics, this research field has garnered significant interest from both industry and research community members lately (TOURANI et al., 2022).

Among the different sensors used in SLAM systems, cameras have a lower operating cost while providing a rich amount of environmental information that allows for more precise recognition and mapping (DONG et al., 2022). Due to this, solutions where only the use of the camera is employed, called visual SLAM systems, are of great interest.

When there's no pre-existing map of the environment or the agent's position is uncertain, SLAM becomes essential for various applications. Visual SLAM systems have the advantages of lower operational costs, simplicity in sensor setup, and miniaturized size (BARROS et al., 2022). Leveraging the camera as the primary sensor opens up the possibility of employing various algorithms from the fields of computer vision and image processing within a visual SLAM system. Despite these advantages, these systems continue to face challenges, such as difficulty in handling dynamic environments and computational constraints when needed for real-time applications (DONG et al., 2022). This latter issue may escalate, particularly if there's a necessity to integrate deep learning methodologies into the SLAM system, such as semantic segmentation or object detection.

Despite advancements in the field, SLAM systems continue to face challenges, such as difficulty handling dynamic environments and computational constraints when needed for real-time applications (BARROS et al., 2022).

The aim of this study is to employ computer vision and Visual Foundation Models within Visual SLAM (Simultaneous Localization and Mapping) systems to address two issues: the handling of dynamic agents in the environment and the generation of depth information from images captured by the system's main camera. These types of models, named Visual Foundation

Models, are capable of learning complex information at both image and pixel levels, which could be overlooked in a supervised approach (OQUAB et al., 2023). Many of these supervised tasks have direct applicability to problems involving the field of SLAM, such as object detection and depth estimation in images (TOURANI et al., 2022).

1.1 Overview

This work is organized as follows: Chapter 2 reviews the related literature that forms the foundation for this dissertation. Chapter 3 provides the theoretical basis of this study, explaining all the concepts and techniques necessary to understand the developed methodology. Chapter 4 describes the methodology developed in this work, detailing each stage from the data acquisition to the execution of the experiments. Chapter 5 presents the results obtained with the developed methodology, discussing and interpreting them. Lastly, Chapter 6 provides an overview of the work undertaken, assessing its efficacy, potential enhancements, contributions of the study, and proposing suggestions for future research.

2
Related Work

This chapter begins by surveying important studies in the field of SLAM, examining both recently published and more established works.

In 2015, Mur-Artal, Montiel e Tardos (2015) proposed the first version of ORB-SLAM, a SLAM system based on ORB Features capable of operating in real-time, both for open and closed environments. Its operation consists of generating a graph in which its nodes are the keyframes, and the edges are the amounts of interest points shared between these frames.

A second version, called ORB-SLAM2, was published in 2017 by Mur-Artal e Tardós (2017). The main contribution was the addition of Stereo and RGB-D cameras that compute trajectory and a sparse 3D reconstruction with true scales while also maintaining real-time performance.

Campos et al. (2021) published in 2021 a third version called ORB-SLAM3, capable of performing Visual, Visual-Inertial, and Multi-Map SLAM with monocular, stereo, and RGB-D cameras. Its results demonstrated robustness, ranking among the best systems available in the literature, and notably achieving higher accuracy. As of the writing of this paper, this version of ORB-SLAM is still considered state-of-the-art in many real-time performance-demanding applications (KAZEROUNI et al., 2022).

Webb, Brown e Luján (2019) proposed the construction of a semantic map using the mapping generated by ORB-SLAM and a neural network, combining the interest points of the SLAM system with the semantic segmentation mask generated by a Convolutional Neural Network (CNN). The proposed architecture is capable of operating in real-time with the use of a Graphics Processing Unit (GPU); however, the method is still subject to the problems faced by ORB-SLAM, such as the need for inertial sensors to increase localization accuracy (BARROS et al., 2022). This work does not offer a solution to address the adaptation to dynamic elements within the environment.

Another approach that uses neural networks to detect and segment objects in the environment is presented by (RUI et al., 2021). Their methodology consists of combining YOLOv3 and ORB-SLAM to create a multisensorial SLAM, aiming to serve as a guided orientation system for people with some level of visual impairment. However, there may still be inaccuracies in sensor localization due to the absence of handling moving objects in the environment within the ORB-SLAM algorithm.

Wu et al. (2022) proposed YOLO-SLAM, a robust SLAM system for

dealing with dynamic environments, which, in some cases, achieved superior performance when compared to versions of ORB-SLAM. Despite the system's ability to filter these dynamic environment characteristics, the proposed method does not benefit from additional information from other sensors, such as depth sensors, which allow for a more precise 3D mapping estimation of the scene. Such mapping is essential for tasks involving robotics and other autonomous systems, including obstacle detection, trajectory estimation, localization, and scene understanding (DONG et al., 2022). Furthermore, this methodology has not been tested for open environments, only providing results for closed-environment datasets.

In this context, this work presents the use of depth images generated by a visual foundation model to eliminate the need for additional sensors in the SLAM system, simplifying its cost and complexity. Our approach also addresses the need for object detection in the environment, which is essential for effective mapping in dynamic settings.

3
Theoretical Foundation

This chapter presents a theoretical foundation with the core concepts for understanding this work. It begins by explaining the basis of SLAM, followed by a discussion of the techniques applied in the depth estimation and dynamic object removal steps of the proposed method. Lastly, it introduces the datasets used in the experiments.

3.1
Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) systems are capable of constructing a map of an environment around them while simultaneously calculating their location using that same map (WHYTE, 2006). Its definition can be explained in two parts: the first one being to map an unknown environment, and the second one to perform the localization of the agent within this environment through available sensors (KAZEROUNI et al., 2022). These sensors can include cameras, sonar, LIDAR, lasers, among others. An SLAM system can be utilized in various applications, such as robots, drones, autonomous vehicles, video games, and virtual/augmented reality devices.

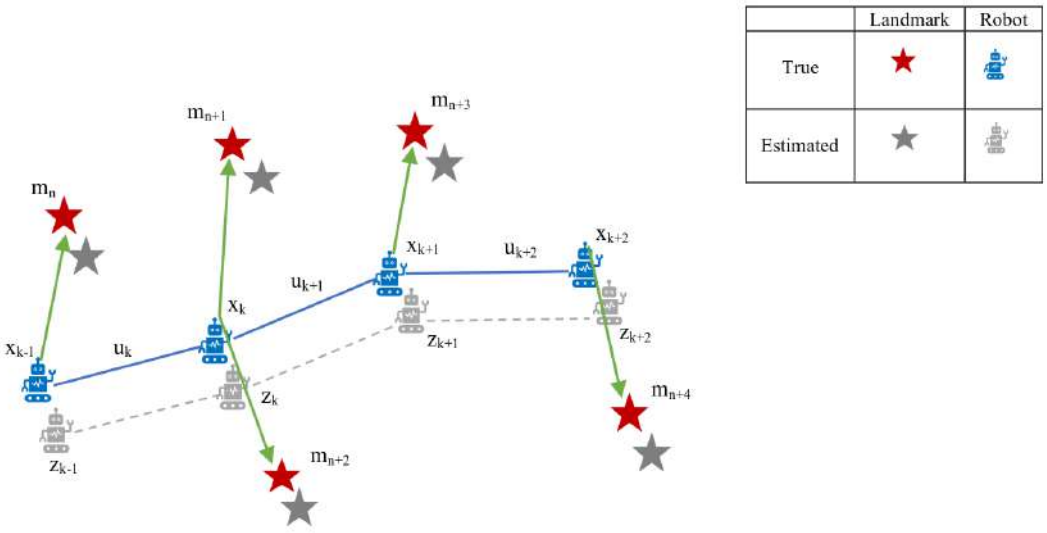

Figure 3.1: SLAM inputs and outputs definition. Adapted from Kazerouni et al. (2022).

Figure 3.1 illustrates the SLAM problem formula. Here, x_k represents the robot's state, including its orientation and position, at the time k. The control input u_k is what drives the robot from its previous state x_{k-1} to x_k. The sensors measurements are denoted by z_k, and m_n is the landmark observed

from the respective robot state. SLAM methods are used to find the robot's route x and landmark map m from controls inputs u and sensor data z.

3.1.1 Visual SLAM Systems

Visual SLAM, also known as V-SLAM, denotes SLAM systems that primarily utilize camera sensors to receive visual information of objects in an unknown environment (KAZEROUNI et al., 2022). Figure 3.2 details the scheme of a V-SLAM system.

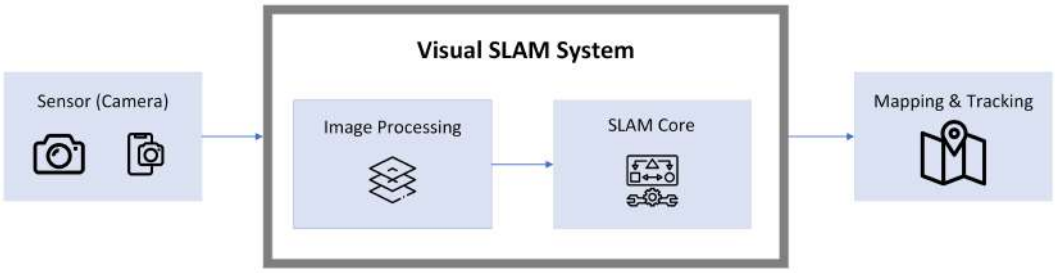

Figure 3.2: Diagram of a Visual SLAM System.

Among the different types of sensors, cameras have a lower operating cost while providing a rich amount of environmental information, enabling more precise recognition and mapping (DONG et al., 2022).

Monocular cameras are widely used, affordable, and typically come as standard equipment on many robots. There is a variety of algorithms, source code, literature, and scholarly articles available for tasks such as image processing, computer vision, and deep learning, all of which leverage the RGB images produced by these cameras. All of this could be used for SLAM techniques (KAZEROUNI et al., 2022).

3.2 Monocular Depth Estimation

Depth estimation involves the task of predicting a dense depth map from the input image(s) that corresponds to it. This process aims to capture the varying distances of objects within the scene, providing valuable spatial information for tasks such as ego-motion estimation, obstacle avoidance and scene understanding (DONG et al., 2022).

In the context of computer vision, monocular depth estimation refers to estimating the depth or distance of objects from a single 2D image. Unlike binocular vision, which relies on two eyes to perceive depth, monocular vision operates with only one input image. This gives it an advantage, as it uses

fewer resources and less data compared to other methods, such as those used in cameras that capture stereo images (BHOI, 2019).

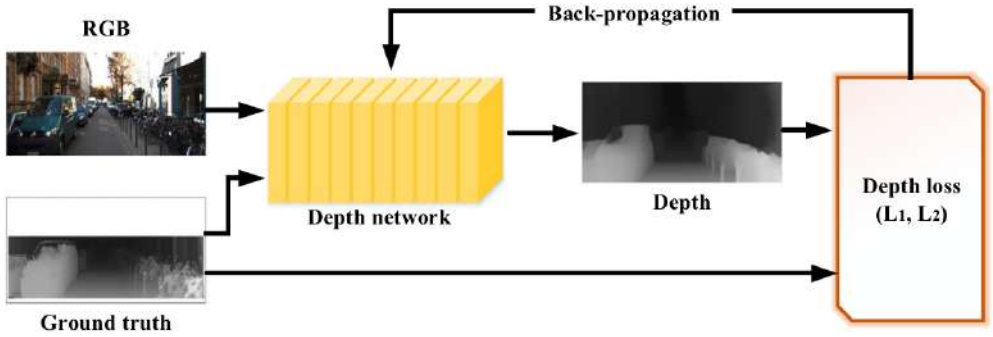

Figure 3.3: General model of supervised learning for monocular depth estimation. Adapted from Ming et al. (2021)

Mathematically speaking, the monocular depth estimation problem can be defined as follows: given a pair of an image I and its corresponding depth field D, the objective is to learn a non-linear mapping $\Phi : I \rightarrow D$ from a training set $T = \{(I_i, D_i)\}_{i=1}^{M}$. This formula may experience changes if applied to unsupervised algorithms, especially in cases where pixel-level ground truth is not widely available (BHOI, 2019). Figure 3.3 shows a general example of a monocular depth prediction model.

3.3 Visual Foundation Models

Visual Foundation Models (VFM) are capable of generating visual features designed to function seamlessly across various tasks, whether at the image level (such as image classification) or at the pixel level (such as segmentation). By utilizing these features "as they are", without any fine-tuning, one can achieve significantly improved performance on downstream tasks compared to what task-specific models can deliver. These models have the potential to significantly simplify image usage in any system by generating general purpose visual features (OQUAB et al., 2023).

3.3.1 DinoV2

For this purpose, DinoV2 (a VFM by Meta AI[1]) was trained using the self-supervised learning approach on a selected dataset containing 142 million images (LVD-142M). In self-supervised learning, visual features are learned directly from the images, without relying on labels. This allows for the learning

[1]https://dinov2.metademolab.com/

of complex information at both the image and pixel levels, which could be suppressed in a supervised approach.

To construct the image dataset used for pretraining DinoV2, the authors collected data from publicly available repositories and developed a pipeline to filter and curate those images. To ensure the quality and variety of the dataset when using unlabeled data, the authors applied a clustering method to retrieve images similar to those on annotated datasets. This allowed for the identification/selection of image classes without requiring manual annotations, resulting in a diverse corpus of 142 million images.

With a discriminative self-supervised method that combines iBOT and DINO losses, DinoV2 learns its features (CARON et al., 2020). To learn image-level features, DinoV2 applies the cross-entropy loss between the features extracted from a student and a teacher network. Regarding patch-level features, the model randomly masks some patches given to the student, but not to the teacher.

To evaluate the features extracted by DinoV2, the authors applied them on several downstream tasks from semantic segmentation to video understanding. When applied to the monocular depth estimation task, DinoV2's features produced images with fewer artifacts in comparison to those generated by OpenCLIP. That is why DinoV2 was chosen for the monocular depth estimation part of the methodology proposed on this work.

3.4 Autoencoders

Autoencoders can be defined as a neural network that is trained to reconstruct its input. The issue, as explicitly delineated in Baldi (2012), involves learning the functions $A : \mathbb{R}^n \rightarrow \mathbb{R}^p$ (encoder) and $B : \mathbb{R}^p \rightarrow \mathbb{R}^n$ (decoder) that fulfills the specified conditions for the following equation:

$$\arg\min_{A,B} \mathbb{E}[\Delta(x, B \circ A(x))] \tag{3-1}$$

Where $\mathbb{E}$ represents the expectation taken over the distribution of x, and Δ denotes the reconstruction loss function that quantifies the discrepancy between the decoder's output and the input. Typically, this loss function is defined using the ℓ_2-norm. Figure 3.4 shows an illustration of the encoder model.

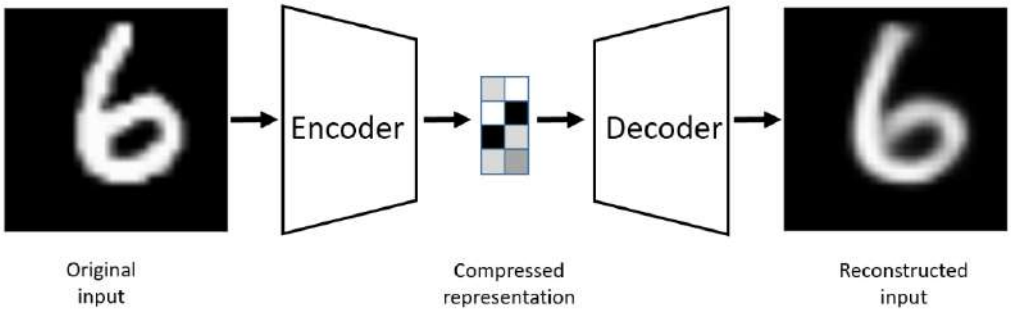

Figure 3.4: Example of a general structure of an Autoencoder. Adapted from Bank, Koenigstein e Giryes (2023).

3.5 Yolo

YOLO, an acronym for *You Only Look Once*, is a real-time object detection algorithm based on convolutional neural networks developed by (REDMON et al., 2016). Using a single-stage architecture, YOLO defines object detection as a regression problem, predicting the region an object is in along with its class probabilities all at once. This makes the network extremely fast, which contributed to its popularity and widely usage. Several subsequent versions of YOLO have been released in recent years, with the most recent one being YOLOv8.

3.5.1 YoloV8

For the removal of points of interest belonging to dynamic objects, such as a person walking through the environment, the methodology proposed on this dissertation used YOLOv8[2], the most recent version available at the time of writing this work.

YOLOv8 introduces new features and performance improvements, offering higher accuracy and speed in detection (WANG et al., 2023). To accelerate model convergence and enhance accuracy, the detection process is divided into two parts: object classification using cross-entropy as loss and regression to estimate the bounding box of the predicted region using distribution focal loss and CIoU (ZHENG et al., 2020). In this version, anchors are no longer used in detection. To detect objects of various sizes, input features are sub-sampled 5 times, accommodating objects of different scales. The choice of this model is due to its high performance in object detection and segmentation tasks while remaining feasible for real-time applications (REDMON et al., 2016).

[2]https://github.com/ultralytics/ultralytics

An overview of the general structure of YOLOv8 is depicted in Figure 3.5 and thoroughly explained by (WANG et al., 2023).

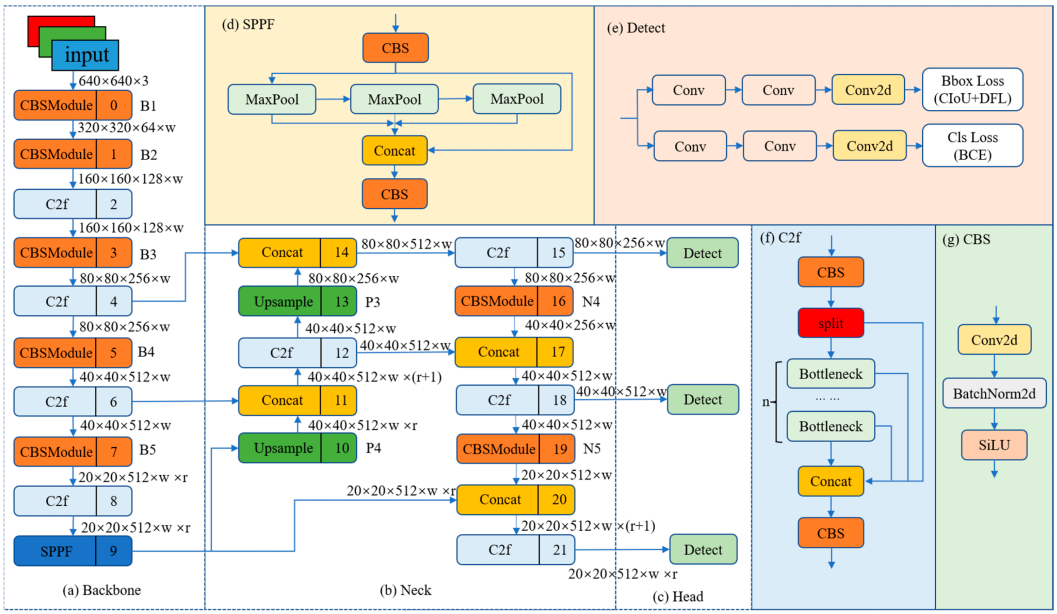

Figure 3.5: YOLOv8 architecture as presented by Wang et al. (2023).

4
Methodology

The methodology proposed in this work aims to investigate the advantages of using computer vision techniques for enhancements in visual SLAM systems. Initially, a method was developed based on the use of neural networks for object detection, with the purpose of identifying and filtering any objects of a dynamic nature that might interfere with mapping and tracking. Furthermore, the lack of information from depth sensors that are not present in a system that solely relies on a camera was addressed by exploring whether the use of VFMs to synthetically generate depth information from image frames contributes to superior performance compared to a purely monocular approach.

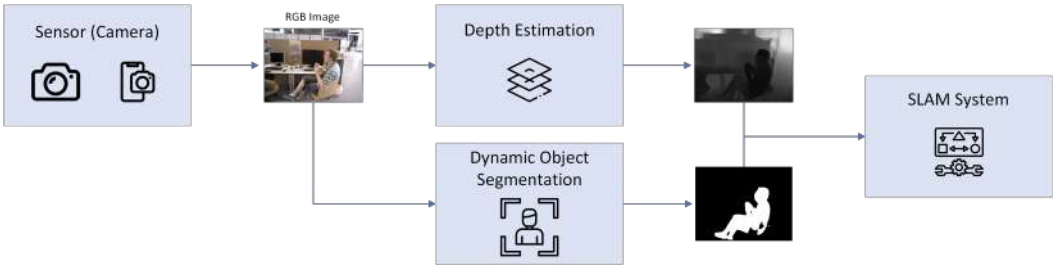

Figure 4.1: Proposed architecture.

Figure 4.1 provides an overview of the methodology. First, images are obtained through the main sensor (camera). After capture, a pair of images is generated from each frame: the first being a segmentation mask for dynamic objects present in the scene, and the second one, a depth image created using features provided by a VFM. These image pairs are fed into the SLAM system, where the device positions are localized and mapped.

This methodology is organized as following: first, we report the data acquisition process, then we discuss the detection and segmentation of dynamic objects in the environment; subsequently, we address the process of estimating monocular depth images; and finally, we discuss how the all those stages are integrated and utilized in the proposed SLAM system.

4.1
Data Acquisition

Datasets are essential for developing and refining algorithms, especially in areas that involve machine learning of some kind. The data used in this research consists of two well-established benchmarks for the area of SLAM in the literature: the TUM and KITTI datasets (STURM et al., 2012) (GEIGER; LENZ; URTASUN, 2012).

4.1.1 TUM Dataset

Provided by The Entrepreneurial University (TUM), the database consists of RGB and depth images captured by a Microsoft Kinect sensor. Along with the images, sensor trajectory information is provided, which serves as ground-truth for testing.

Figure 4.2: Example of sequence frames and their respective depth maps on TUM dataset.

The dataset is divided into several image sequences that were recorded in an office environment and an industrial hall, covering a large variety of scenes and camera motions. Each sequence was recorded at full sensor resolution of 640×480 at a video frame rate of 30hz (STURM et al., 2012). Figure 4.2 shows some examples of images and depth maps present in the dataset.

Sequence	**Duration**	**Length**
fr2/desk_with_person	142.08s	17.044m
fr3/sitting_static	23.63s	0.259m
fr3/sitting_xyz	42.50s	5.496m
fr3/sitting_halfsphere	37.15s	6.503m
fr3/sitting_rpy	27.48s	1.110m
fr3/walking_static	24.83s	0.282m
fr3/walking_xyz	28.83s	5.791m
fr3/walking_halfsphere	35.81s	7.686m
fr3/walking_rpy	30.61s	2.698m

Table 4.1: Segments used in the evaluation.

The Table 4.1 describes the segments used in the evaluation of this work, which belong to the category named *Dynamic Objects*; these are particularly

challenging due to the presence of moving objects in the environment. We can divide the sequences into two subgroups:

- **Medium dynamism**: characterized by the presence of people making movements in the scene while seated in a fixed location. The following segments belong to this group: *fr2/desk_with_person*, *fr3/sitting_static*, *fr3/sitting_xyz*, *fr3/sitting_halfsphere* and *fr3/sitting_rpy*;
- **High dynamism**: characterized by the presence of people moving around the scene while standing, walking through the environment almost constantly. Belonging to this group the following segments: *fr3/walking_static*, *fr3/walking_xyz*, *fr3/walking_halfsphere* and *fr3/walking_rpy*.

The other segments are utilized for fine-tuning the depth estimation model, as they contain numerous image pairs along with their respective depth ground-truth.[1] The Section 8.1 in the appendix details all the sequences available.

4.1.2 KITTI Dataset

The KITTI dataset consists of a series of videos captured by three high-resolution cameras (two of them in color and one in grayscale), along with their respective ground-truth trajectories captured by a Velodyne laser sensor and GPS (GEIGER; LENZ; URTASUN, 2012). As shown in Figure 4.3, the sensors are attached to a vehicle, which travels through various rural and urban parts of the city of Karlsruhe, Germany.

The dataset consists of 22 stereo sequences, with a total length of 39.2 km. The camera recorded the frames at 1392×512 resolution and 10hz. Figure 4.4 shows some examples of sequence images and their respective depth maps.

Due to the high presence of dynamic objects (approximately 15 cars and 30 pedestrians per image) (GEIGER; LENZ; URTASUN, 2012), the dataset is a suitable candidate for evaluating the proposed method. The segments used for evaluation are the ones belonging to the *Odometry* category.[2] Figure 4.4 shows some examples of sequence images and their respective depth maps.[3]

[1] Except sequences belonging to the categories *Validation Files* and *Calibration Files*
[2] Source: https://www.cvlibs.net/datasets/kitti/eval_odometry.php
[3] Sequence KITTI_03 overlaps with the public test set and hence was removed.

Figure 4.3: Sensors mounted on a vehicle. Adapted from Geiger, Lenz e Urtasun (2012).

Figure 4.4: Example of sequence frames and their respective depth maps on KITTI dataset.

4.2 Dynamic Object Segmentation

At this stage, the images captured by the sensor are subject to processing with the purpose of generating a segmentation mask for only the dynamic objects present in the environment. The YoloV8 model was selected for this task due to its superior capabilities in object detection and segmentation, maintaining suitability for real-time applications (REDMON et al., 2016).

In Figures 4.5 and 4.6, we can observe the resulting masks. It is important to note that only the classes considered dynamic (i.e., capable of moving within the target environment) are retained in the segmentation mask. The chosen dynamic classes are: *person*, *bicycle*, *car*, *motorcycle*, *bus*, *truck*, *bird*, *cat*, and *dog*. These classes are based on the YoloV8 model trained on the Microsoft Common Objects in Context (COCO) dataset (LIN et al., 2015).

Once generated, these masks are then sent to the SLAM system for the

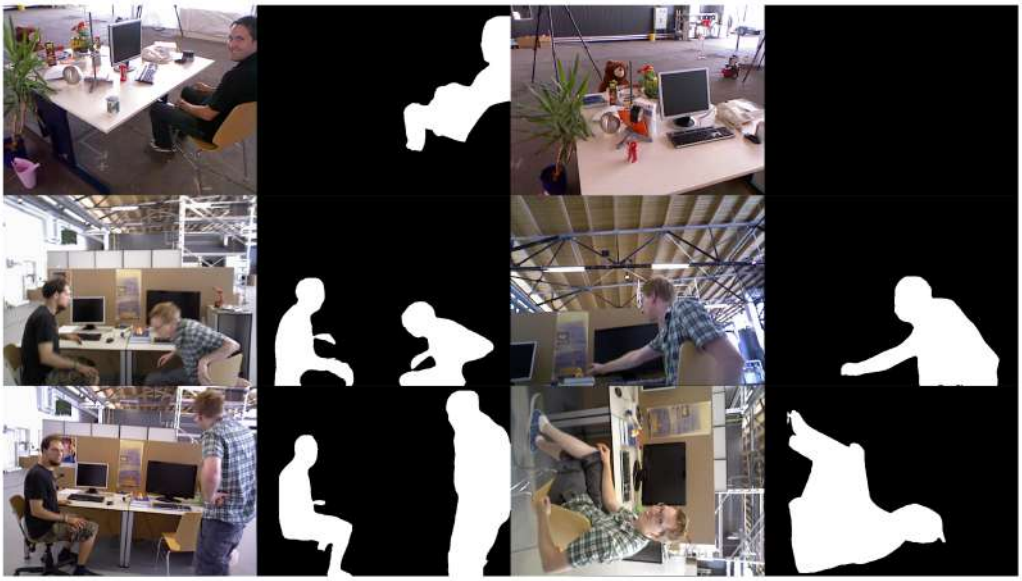

Figure 4.5: Example of sequence frames and their respective masks generated on TUM dataset.

Figure 4.6: Example of sequence frames and their respective masks generated on KITTI dataset.

management and proper filtering of these objects within the environment.

4.3 Monocular Depth Estimation

Alongside the object segmentation stage, depth maps are generated using the VFM DinoV2 (OQUAB et al., 2023). This model is capable of producing general-purpose visual representations (features) that can be applied to various types of images and tasks without requiring fine-tuning. Among these tasks, monocular depth estimation is included.

The Figures 4.7 and 4.8 present examples of depth maps created by utilizing features extracted from the model in conjunction with a pre-trained

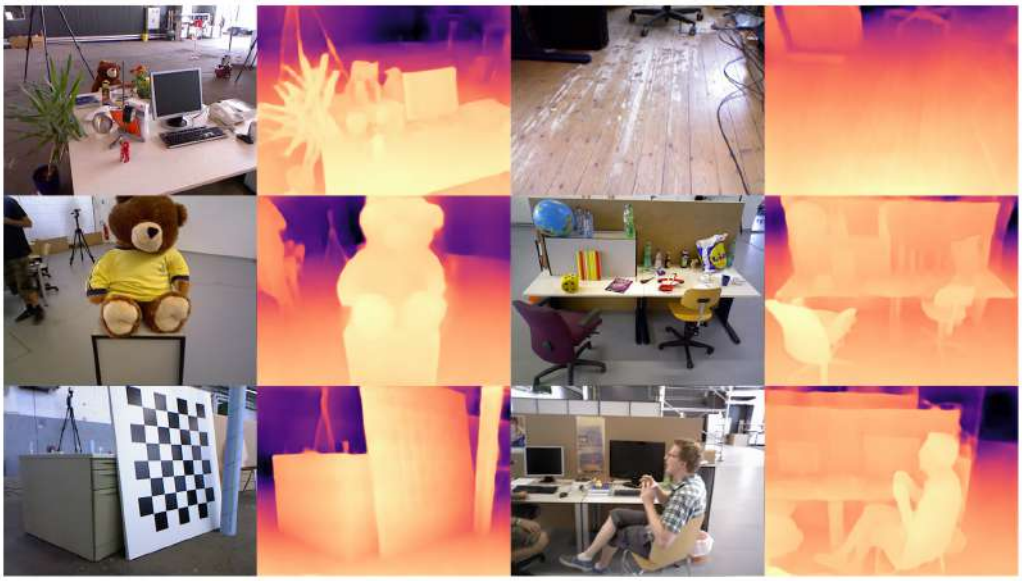

Figure 4.7: Example of sequence frames and their respective depth maps generated on TUM dataset.

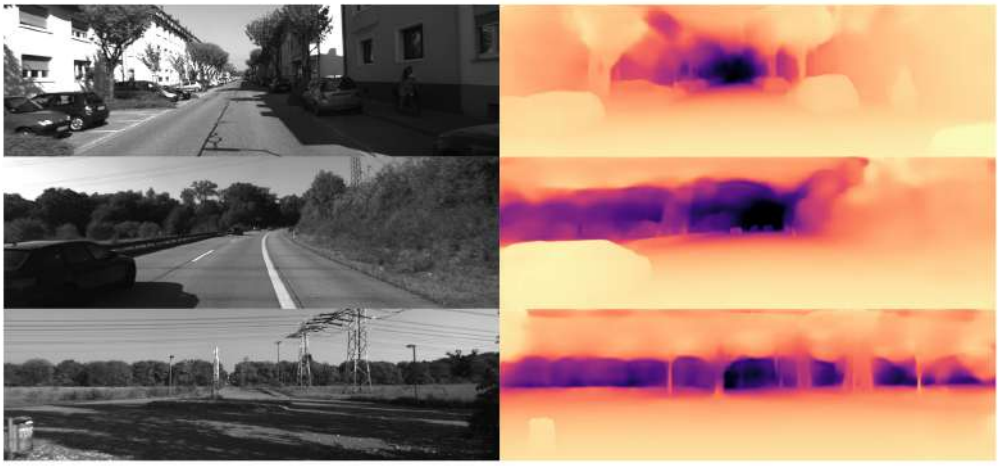

Figure 4.8: Example of sequence frames and their respective depth maps generated on KITTI dataset.

Dense Prediction Transformer(RANFTL; BOCHKOVSKIY; KOLTUN, 2021) head available in the DinoV2 repository.[4] For improved performance, only the smaller *ViT-S/14 distilled* backbone was utilized for the VFM in the SLAM system.

Despite Dinov2's ability to generate depth maps from images, these do not contain real-world depth information (in meters or centimeters), but only on a relative scale (RANFTL et al., 2020). To address this, it is necessary to perform fine-tuning on the datasets used.

[4]https://github.com/facebookresearch/dinov2

4.3.1 Metric Fine-tuning

For metric depth information, fine-tuning was conducted on the TUM dataset using a Variational Autoencoder model. Fine-tuning was not required on the KITTI dataset, as DinoV2 already provides a head trained on this dataset (OQUAB et al., 2023).

For training on the TUM dataset, all sequences that do not belong to the *Dynamic Objects* category were used as sources for model fitting. The images were divided into patches of 160×160. It is important to note that there is missing data in the ground-truth depth maps; therefore, only patches with fully populated depth maps were retained. Table 4.2 provides detailed information on the total number of patches used in training and validation. It is important to note that, to prevent data leakage, it was ensured that the same sequence only had patches in either the training or validation set, but never in both.

	Nº Patches	Patch Size
Training	19340	160x160
Validation	1250	160x160
Total	20590	

Table 4.2: Total number of patches used in training and validation on TUM dataset.

Figure 4.9: Original frames, Depth Ground-Truth and Estimated Depth Map on KITTI dataset.

Figures 4.9 and 4.10 show the results after metric depth is recovered from the estimated images, alongside the ground truth and the corresponding frame. As we can see, the estimated images do not exhibit the presence of 'missing data' that occurs in depth maps obtained by sensors.

Figure 4.10: Original frames, Depth Ground-Truth and Estimated Depth Map on TUM dataset.

4.4 SLAM

Figure 4.11: On the left is the frame with the points of interest generated by ORB-SLAM, on the right, after the removal of points of interest belonging to dynamic objects.

With the segmentation mask of dynamic objects, it is possible to identify the points of interest belonging to dynamic objects and remove them from the map to avoid accumulation of error in trajectory calculation. Figure 4.11 shows the result of this process. After the removal of points of interest, the

segmentation mask is discarded, and the execution of ORB-SLAM continues normally.

The implementation of ORB-SLAM used is already capable of processing depth images, so there is no need to adapt it to receive the images generated in Section 4.3. After receiving all the necessary data, the system will continuously update the map of the environment and localize itself.

5
Results

In this chapter, we present the results obtained by the proposed methodology for handling dynamic objects and depth map estimation using machine learning. Initially, the contributions were evaluated separately, with each result presented in its own section. Subsequently, the results obtained through the use of the complete methodology are discussed, along with some specific observations.

5.1
Metrics

To measure the accuracy of the proposed SLAM system, we use three widely popular metrics for this type of problem (PROKHOROV et al., 2019). The Absolute Trajectory Error (ATE_{rmse}) is the average deviation of the estimated trajectory from the ground truth per frame, which is mathematically defined as:

$$ATE_{rmse} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} || \; trans(E_i) \; ||^2} \tag{5-1}$$

where E_i is the matrix of absolute trajectory error at instant i.

The Relative Pose Error Translational (RPE_{trans}) corresponds to the drift of the robot's trajectory at the translation component. Given a relative pose error E at the time i, obtained from a sequence of m relative pose error matrices generated from n camera poses, RPE_{trans} can be calculated as:

$$RPE_{trans} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} || \; trans(F_i) \; ||^2} \tag{5-2}$$

where F_i is the matrix of the relative pose error at instant i.

Finally, the Relative Pose Error Rotational (RPE_{rot}) corresponds to the robot's rotation trajectory drift, defined by the following equation:

$$RPE_{rot} = \frac{1}{m}\sum_{i=1}^{m} \angle(rot(F_i^{\Delta})) \tag{5-3}$$

5.2
Experiments with Depth Generated Images

First, it was conducted experiments using our generated depth images in isolation to investigate any potential improvements in tracking. To define this experiment, we compared the use of our depth images with a monocular

approach and a second one that utilized the original depth maps (which were captured using a sensor specifically designed for this purpose).

5.2.1 TUM Dataset

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
freiburg2_desk_with_person	1.1518	0.0687	**0.0159**
freiburg3_sitting_halfsphere	0.2123	0.0854	**0.0204**
freiburg3_sitting_rpy	0.0589	**0.0269**	0.0278
freiburg3_sitting_static	0.0274	0.0115	**0.0079**
freiburg3_sitting_xyz	0.2931	0.0447	**0.0094**
freiburg3_walking_halfsphere	0.3545	0.2404	**0.1340**
freiburg3_walking_rpy	**0.1344**	0.5046	0.6129
freiburg3_walking_static	0.0215	0.0161	**0.0105**
freiburg3_walking_xyz	0.2995	**0.0849**	0.0956

Table 5.1: $ATE_{rmse}(m)$ of the segments from the TUM database with monocular approach, our generated depth maps and true depth maps.

Table 5.1 presents the results obtained from the TUM dataset. As observed, in comparison with the monocular approach, the gains were evident, indicating that the use of depth images generated by a Foundation Model contributes to an overall better performance of the system. When compared to the approach that uses real depth maps, it is apparent that our method yields results closer to this approach than the monocular one, and in some cases, it even achieves better results, as shown in the sequences *freiburg3_sitting_rpy* and *freiburg3_walking_xyz*. We can argue that due to the nature of these segments, one where there is a higher occurrence of rotational movements and almost no dynamic object movement, and another where there are quickly moving dynamic objects in large parts of the visible scene, the sensor might have been disadvantaged when capturing depth images in this setup when compared to the generative method.

Figure 5.1 displays the trajectories in the segment that experienced the greatest reduction in tracking error. As can be seen, the leap in performance between the monocular approach and ours highlights the importance of having some level of depth information present in a SLAM system to achieve better results.

Despite clear improvements in most sequences, there was one case where the monocular approach outperformed the other two, including ours. Observing Figure 5.2, which shows the trajectories in the segment *freiburg3_walking_rpy*, we can see that the behavior shifts; that is, depth information may not be as helpful in ORB-SLAM3 in a segment characterized by high camera rotation combined with dynamic objects moving simultaneously.

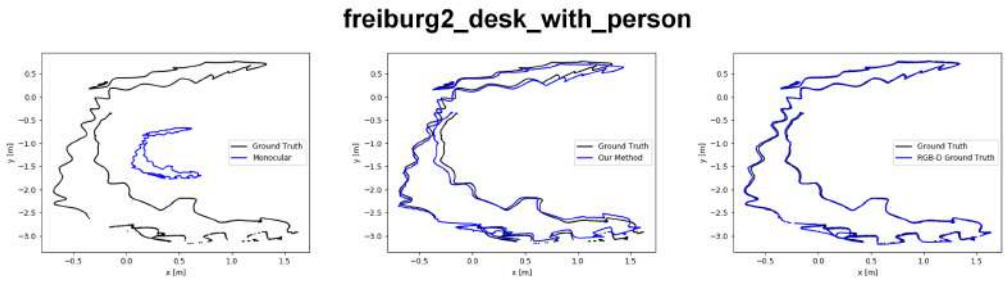

Figure 5.1: Plotted trajectory of sequence *freiburg2_desk_with_person* comparison between monocular, our method and RGB-D ground truth approach. In each plot, the black line represents the ground truth trajectory, and the blue line represents the estimated trajectory corresponding to each approach (monocular, our method and RGB-D).

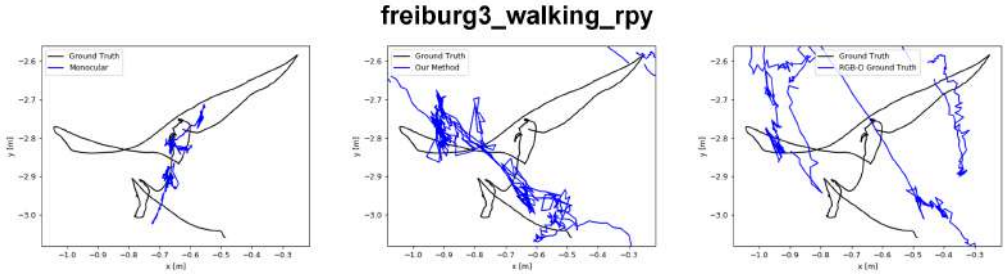

Figure 5.2: Plotted trajectory of sequence *freiburg3_walking_rpy* comparison between monocular, our method and RGB-D ground truth approach.

The metrics RPE_{trans} and RPE_{rot} for this experiment are detailed in the Appendix Section 8.3.1.

5.2.2 KITTI Dataset

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
KITTI_00	11.0357	3.8782	**1.1452**
KITTI_01	536.0013	665.6547	**165.5755**
KITTI_02	17.1554	7.3851	**4.5026**
KITTI_04	**0.8706**	2.4069	1.1440
KITTI_05	6.8373	4.7494	**0.8052**
KITTI_06	16.2867	3.9378	**1.8729**
KITTI_07	2.5578	1.0560	**0.4240**
KITTI_08	56.7354	8.1729	**4.8468**
KITTI_09	40.3568	1.8020	**1.3979**
KITTI_10	7.4552	2.6286	**1.8421**

Table 5.2: $ATE_{rmse}(m)$ of the segments from the KITTI database with monocular approach, our generated depth maps and true depth maps.

Observing Table 5.2, it is apparent that the use of generated depth maps

also results in performance gains in the scenario of outdoors environments, surpassing the monocular approach in most sequences of the dataset. Figures 5.3 and 5.4 display the cases with the most significant error reduction compared to the monocular approach. Here, it can be noted that our depth maps approximate the performance of RGB-D more closely than they do vs. purely visual approach.

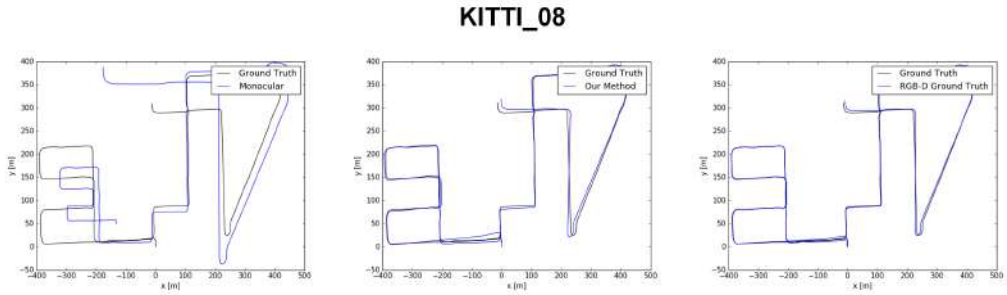

Figure 5.3: Plotted trajectory of sequence *KITTI_08* comparison between monocular, our method and RGB-D ground truth approach.

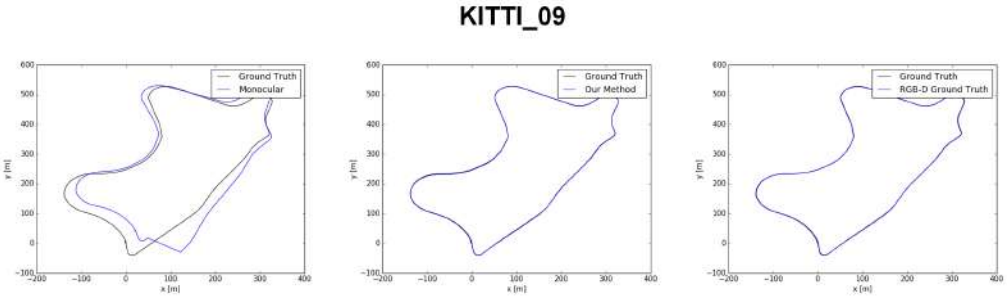

Figure 5.4: Plotted trajectory of sequence *KITTI_09* comparison between monocular, our method and RGB-D ground truth approach.

In the sequence *KITTI_01*, it is observed that for this specific case the use of our depth maps did not contribute to a positive performance, achieving a result inferior to the monocular approach. Looking at Figure 5.5, one could argue that the quality of the depth images generated for this segment is not satisfactory, as it is evident that accurate depth information does indeed contribute to improved performance, as seen in the RGB-D Ground Truth approach.

The metrics RPE_{trans} and RPE_{rot} for this experiment are detailed in the Appendix Section 8.3.2.

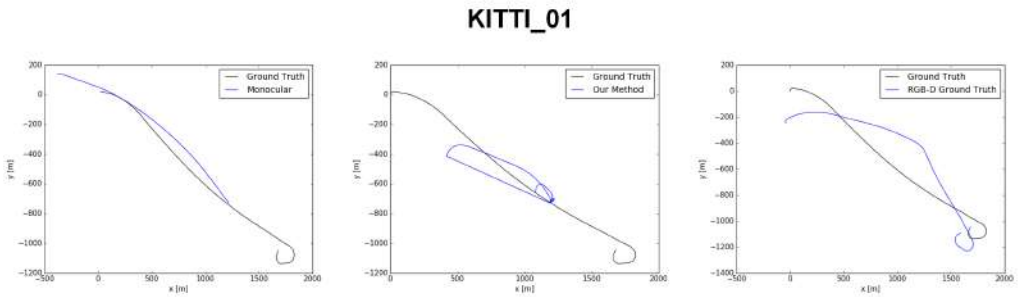

Figure 5.5: Plotted trajectory of sequence *KITTI_01* comparison between monocular, our method and RGB-D ground truth approach.

5.3 Experiments with Dynamic Object Masks

To assess the effectiveness of filtering dynamic objects in ORB-SLAM3, experiments were conducted in two ways: initially, we tested the use of a mask in a monocular scenario, followed by an evaluation of its effectiveness in a scenario that also utilizes depth information. The design of this experiment aims to determine whether segmenting and isolating certain objects could be advantageous in both scenarios if the environment is dynamic.

5.3.1 TUM Dataset

Sequence	Our Method ↓	Monocular ↓
freiburg2_desk_with_person	1.2781	**1.1518**
freiburg3_sitting_halfsphere	0.2991	**0.2123**
freiburg3_sitting_rpy	0.0604	**0.0589**
freiburg3_sitting_static	0.0345	**0.0274**
freiburg3_sitting_xyz	0.2998	**0.2931**
freiburg3_walking_halfsphere	0.4404	**0.3545**
freiburg3_walking_rpy	**0.1148**	0.1344
freiburg3_walking_static	**0.0196**	0.0215
freiburg3_walking_xyz	**0.2652**	0.2995

Table 5.3: $ATE_{rmse}(m)$ of the segments from the TUM dataset, comparing the use of segmentation masks against not using them, on monocular approach.

Tables 5.3 and 5.4 display the results for the monocular and RGB-D experiments, respectively. The most notable aspect of these results is that the performance gains are most apparent in the *walking* sequences. These segments share the common characteristic of more intense movement by the

Sequence	Our Method ↓	RGB-D Ground Truth ↓
freiburg2_desk_with_person	0.0169	**0.0159**
freiburg3_sitting_halfsphere	0.0251	**0.0204**
freiburg3_sitting_rpy	**0.0186**	0.0278
freiburg3_sitting_static	0.0088	**0.0079**
freiburg3_sitting_xyz	0.0126	**0.0094**
freiburg3_walking_halfsphere	**0.0256**	0.1340
freiburg3_walking_rpy	**0.0431**	0.6129
freiburg3_walking_static	0.0169	**0.0105**
freiburg3_walking_xyz	**0.0171**	0.0956

Table 5.4: $ATE_{rmse}(m)$ of the segments from the TUM dataset, comparing the use of segmentation masks against not using them, on RGB-D approach.

individuals captured by the camera, compared to other sequences where the subjects remain seated throughout their entire duration.

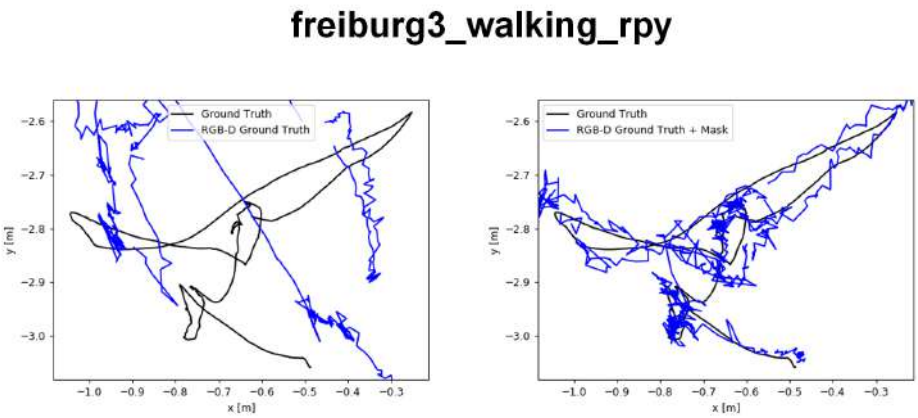

Figure 5.6: Plotted trajectory of sequence *freiburg3_walking_rpy* comparison between no-mask vs. mask, on RGB-D.

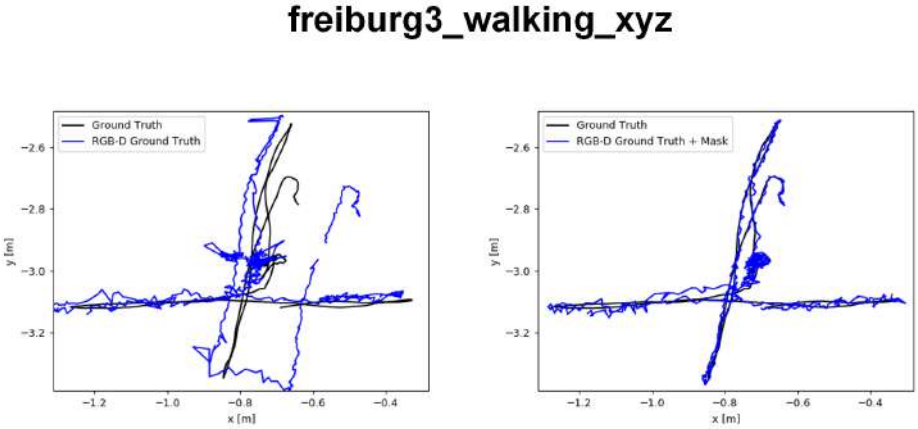

Figure 5.7: Plotted trajectory of sequence *freiburg3_walking_xyz* comparison between no-mask vs. mask, on RGB-D.

Figures 5.6 and 5.7 display the trajectories of two segments that performed better with the use of masks. It is noted that considering the behavior of these dynamic agents in mapping and tracking contributes to a reduction in precision error.

The metrics RPE_{trans} and RPE_{rot} for this experiment are detailed in the Appendix Section 8.4.1.

5.3.2 KITTI Dataset

Sequence	Our Method ↓	Monocular ↓
KITTI_00	**8.9666**	11.0357
KITTI_01	**419.7179**	536.0013
KITTI_02	27.4895	**17.1554**
KITTI_04	**0.6706**	0.8706
KITTI_05	**5.2133**	6.8373
KITTI_06	**16.3143**	16.2867
KITTI_07	**2.2088**	2.5578
KITTI_08	**7.6336**	56.7354
KITTI_09	**6.8851**	40.3568
KITTI_10	**7.3670**	7.4552

Table 5.5: $ATE_{rmse}(m)$ of the segments from the KITTI dataset, comparing the use of segmentation masks against not using them, on monocular approach.

Sequence	Our Method ↓	RGB-D Ground Truth ↓
KITTI_00	1.7325	**1.1452**
KITTI_01	719.3919	**165.5755**
KITTI_02	**4.0447**	4.5026
KITTI_04	1.4587	**1.1440**
KITTI_05	**0.5257**	0.8052
KITTI_06	**1.8153**	1.8729
KITTI_07	0.4530	**0.4240**
KITTI_08	5.0059	**4.8468**
KITTI_09	**1.2817**	1.3979
KITTI_10	**1.7325**	1.8421

Table 5.6: $ATE_{rmse}(m)$ of the segments from the KITTI dataset, comparing the use of segmentation masks against not using them, on RGB-D approach.

The results for the monocular and RGB-D experiments are presented in Tables 5.5 and 5.6. It can be noted that the use of segmentation masks was effective in the KITTI dataset, which could be argued to be due to the higher level of dynamism compared to the TUM, notably because it is in an

outdoor environment. The trajectory with the greatest error reduction in the monocular approach is shown in Figure 5.8.

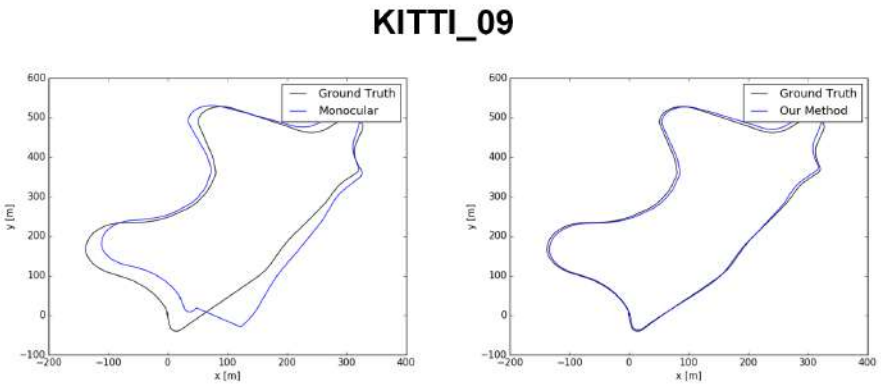

Figure 5.8: Plotted trajectory of sequence *KITTI_09* comparison between monocular and our method.

When compared using the RGB-D approach, our method maintained a lower error in half of the segments, reinforcing the effectiveness of these masks for this type of environment. But, it is observed that an unusual performance is re-noted for the segment *KITTI_01*, where in this case the mask contributed to a greater accumulation of error in the trajectory. This may allow us to begin categorizing this sequence as having behavior similar to an outlier.

The metrics RPE_{trans} and RPE_{rot} for this experiment are detailed in the Appendix Section 8.4.2.

5.4 Experiments with Depth Generated Images and Dynamic Object Masks

Finally, we conducted experiments by combining our two contributions, namely, dynamic segmentation masks and jointly generated depth maps. To assess the results, we compared them directly with the monocular and RGB-D approaches (noting that this approach uses the depth maps provided by the datasets themselves).

5.4.1 TUM Dataset

As shown in Table 5.7, the results exhibit a behavior similar to that discussed in Section 5.2. This time, our method appeared entirely superior to the monocular approach. The main difference is observed in the *freiburg3_walking_rpy* segment, where the monocular approach previously performed better, our method now demonstrates superior performance as presented in Figure 5.9.

Sequence	Monocular ↓	Our Method ↓	RGB-D ↓
freiburg2_desk_with_person	1.1518	0.0700	**0.0159**
freiburg3_sitting_halfsphere	0.2123	0.0754	**0.0204**
freiburg3_sitting_rpy	0.0589	0.0320	**0.0278**
freiburg3_sitting_static	0.0274	0.0086	**0.0079**
freiburg3_sitting_xyz	0.2931	0.0368	**0.0094**
freiburg3_walking_halfsphere	0.3545	**0.0431**	0.1340
freiburg3_walking_rpy	0.1344	**0.0542**	0.6129
freiburg3_walking_static	0.0215	**0.0070**	0.0105
freiburg3_walking_xyz	0.2995	**0.0213**	0.0956

Table 5.7: Comparing the performance of the entire method with the monocular approach and RGB-D Ground Truth on TUM dataset.

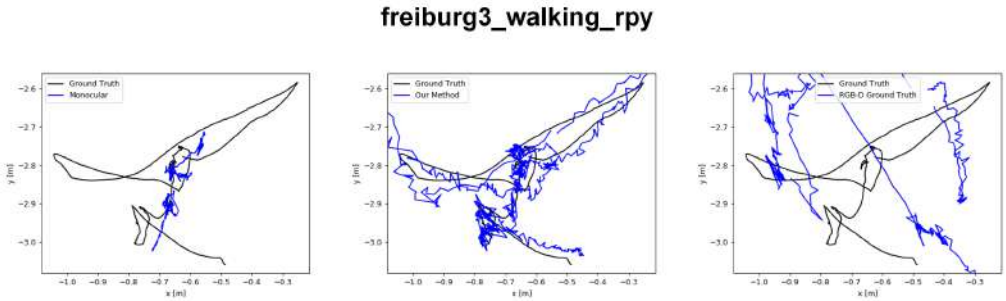

Figure 5.9: Plotted trajectory of sequence *freiburg3_walking_rpy* comparison between monocular, our method and RGB-D ground truth approach.

Another important point to note is that, compared to the RGB-D approach, our method performs better in scenarios with greater environmental dynamism, specifically in the segments from the group *walking*. The direct comparison of the trajectories in these segments can be seen in Figure 5.10.

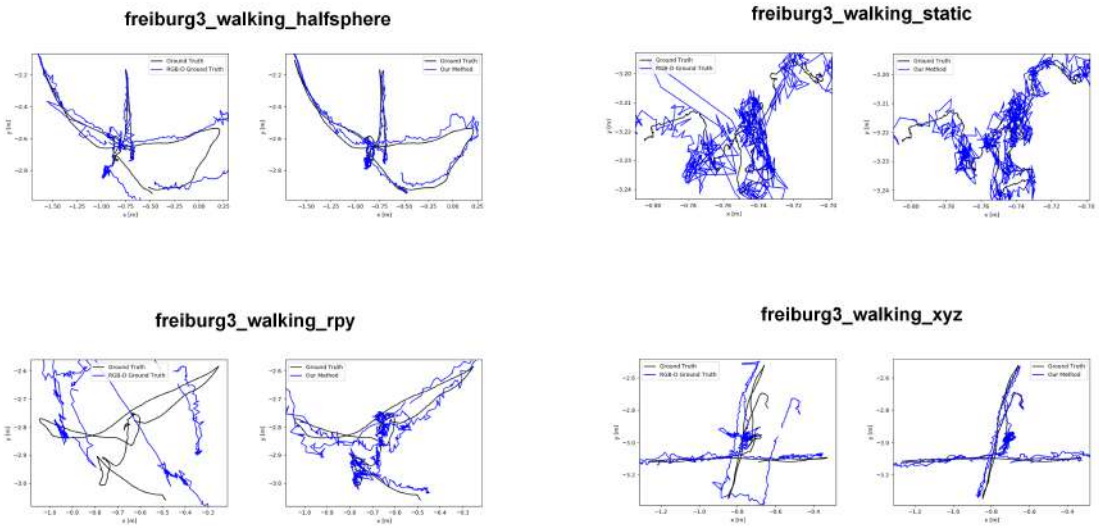

Figure 5.10: Plotted trajectory all *walking* sequences, on the left normal RGB-D, on the right our method.

The metrics RPE_{trans} and RPE_{rot} for this experiment are detailed in the Appendix Section 8.5.1.

5.4.2 KITTI Dataset

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
KITTI_00	11.0357	4.0760	**1.1452**
KITTI_01	536.0013	659.7643	**165.5755**
KITTI_02	17.1554	7.1659	**4.5026**
KITTI_04	0.8706	1.8693	**1.1440**
KITTI_05	6.8373	4.5593	**0.8052**
KITTI_06	16.2867	3.5056	**1.8729**
KITTI_07	2.5578	1.2163	**0.4240**
KITTI_08	56.7354	8.4895	**4.8468**
KITTI_09	40.3568	1.9775	**1.3979**
KITTI_10	7.4552	2.6563	**1.8421**

Table 5.8: Comparing the performance of the entire method with the monocular approach and RGB-D Ground Truth on KITTI dataset

The superior performance compared to the monocular approach is repeated in experiments with the KITTI dataset, as seen in Table 5.8. This demonstrates that using monocular estimation of depth maps is a viable alternative for purely visual systems, as there can be a significant performance gain without the need for a physical sensor such as LiDAR.

It is worth noting that the experiments were conducted on datasets with both indoor and outdoor characteristics. This demonstrates that the use of the techniques presented has a high diversity in its utility, especially if the environment is dynamic.

The metrics RPE_{trans} and RPE_{rot} for this experiment are detailed in the Appendix Section 8.5.2.

6
Conclusion

SLAM systems are widely used across several fields like robotics and autonomous vehicles, aiming to map environments accurately while tracking a camera's movement. The versatility of SLAM extends to drones, and virtual reality devices, attracting considerable interest from both academia and industry.

This dissertation presented the use of Visual Foundation Models to generate depth images, thereby eliminating the need for extra sensors in the SLAM system for localization and mapping. Additionally, it emphasized the importance of identifying and filtering dynamic objects within the environment using the latest version of YOLO.

Using YoloV8 for dynamic object detection and localization within an ORB-SLAM system has shown promising results, addressing one of its weaknesses: lower performance in its tracking tasks for highly dynamic environments. This adaptation has the potential to become a viable alternative for feature-based SLAM systems required in such environments.

By analyzing the provided results, it is possible to notice that the use of Visual Foundation Models (such as DinoV2), to generate depth information about the environment proves to be a viable alternative to both monocular approaches and physical depth sensors. This demonstrates the potential utility of these types of models in the field of SLAM.

For future works, we would like to investigate other applications of Visual Foundation Models in the domain of SLAM, such as using them for object detection, semantic mapping or even tracking tasks. Additionally, there is the possibility of conducting experiments on other benchmarks, such as datasets containing outdoor sequences and multiple classes of different dynamic objects. And finally, there is a need to test the feasibility of these systems using Visual Foundation Models in real-time settings, due to their computational demands.

7
Bibliography

BALDI, P. Autoencoders, unsupervised learning, and deep architectures. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. **Proceedings of ICML workshop on unsupervised and transfer learning**. [S.l.], 2012. p. 37–49.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. **Machine learning for data science handbook: data mining and knowledge discovery handbook**, Springer, p. 353–374, 2023.

BARROS, A. M. et al. A comprehensive survey of visual slam algorithms. **Robotics**, MDPI, v. 11, n. 1, p. 24, 2022.

BHOI, A. Monocular depth estimation: A survey. **arXiv preprint arXiv:1901.09402**, 2019.

CAMPOS, C. et al. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. **IEEE Transactions on Robotics**, IEEE, v. 37, n. 6, p. 1874–1890, 2021.

CARON, M. et al. Unsupervised learning of visual features by contrasting cluster assignments. **Advances in neural information processing systems**, v. 33, p. 9912–9924, 2020.

DONG, X. et al. Towards real-time monocular depth estimation for robotics: A survey. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 23, n. 10, p. 16940–16961, 2022.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: **Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2012.

KAZEROUNI, I. A. et al. A survey of state-of-the-art on visual slam. **Expert Systems with Applications**, Elsevier, v. 205, p. 117734, 2022.

LIN, T.-Y. et al. **Microsoft COCO: Common Objects in Context**. 2015.

MING, Y. et al. Deep learning for monocular depth estimation: A review. **Neurocomputing**, Elsevier, v. 438, p. 14–33, 2021.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. **IEEE transactions on robotics**, IEEE, v. 31, n. 5, p. 1147–1163, 2015.

MUR-ARTAL, R.; TARDÓS, J. D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. **IEEE transactions on robotics**, IEEE, v. 33, n. 5, p. 1255–1262, 2017.

OQUAB, M. et al. Dinov2: Learning robust visual features without supervision. **arXiv preprint arXiv:2304.07193**, 2023.

PROKHOROV, D. et al. Measuring robustness of visual slam. In: IEEE. **2019 16th International Conference on Machine Vision Applications (MVA)**. [S.l.], 2019. p. 1–6.

RANFTL, R.; BOCHKOVSKIY, A.; KOLTUN, V. Vision transformers for dense prediction. **ArXiv preprint**, 2021.

RANFTL, R. et al. **Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer**. 2020.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.

RUI, C. et al. A multi-sensory blind guidance system based on yolo and orb-slam. In: IEEE. **2021 IEEE International Conference on Progress in Informatics and Computing (PIC)**. [S.l.], 2021. p. 409–414.

STURM, J. et al. A benchmark for the evaluation of rgb-d slam systems. In: **Proc. of the International Conference on Intelligent Robot Systems (IROS)**. [S.l.: s.n.], 2012.

TOURANI, A. et al. Visual slam: What are the current trends and what to expect? **Sensors**, MDPI, v. 22, n. 23, p. 9297, 2022.

WANG, G. et al. Uav-yolov8: a small-object-detection model based on improved yolov8 for uav aerial photography scenarios. **Sensors**, MDPI, v. 23, n. 16, p. 7190, 2023.

WEBB, A. M.; BROWN, G.; LUJÁN, M. Orb-slam-cnn: lessons in adding semantic map construction to feature-based slam. In: SPRINGER. **Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019, London, UK, July 3–5, 2019, Proceedings, Part I 20**. [S.l.], 2019. p. 221–235.

WHYTE, H. D. Simultaneous localisation and mapping (slam): Part i the essential algorithms. **Robotics and Automation Magazine**, 2006.

WU, W. et al. Yolo-slam: A semantic slam system towards dynamic environment with geometric constraint. **Neural Computing and Applications**, Springer, p. 1–16, 2022.

ZHENG, Z. et al. Distance-iou loss: Faster and better learning for bounding box regression. In: **Proceedings of the AAAI conference on artificial intelligence**. [S.l.: s.n.], 2020. v. 34, n. 07, p. 12993–13000.

8
Appendix

8.1
TUM Dataset Information

In this section of the appendix, all segments of the TUM dataset used in this study are detailed.

Sequence Name	Duration	Length
fr1/xyz	30.09s	7.112m
fr1/rpy	27.67s	1.664m
fr2/xyz	122.74s	7.029m
fr2/rpy	109.97s	1.506m

Table 8.1: Sequences of **Testing and Debugging** category.

Table 8.1 details the sequences that belong to the *Testing and Debugging* category. This category comprises sequences that are well-suited for debugging purposes, as they are very straightforward.

Sequence Name	Duration	Length
fr1/360	28.69s	5.818m
fr1/floor	49.87s	12.569m
fr1/desk	23.40s	9.263m
fr1/desk2	24.86s	10.161m
fr1/room	48.90s	15.989m
fr2/360_hemisphere	91.48s	14.773m
fr2/360_kidnap	48.04s	14.286m
fr2/desk	99.36s	18.880m
fr2/large_no_loop	112.37s	26.086m
fr2/large_with_loop	173.19s	39.111m
fr3/long_office_household	87.09s	21.455m

Table 8.2: Sequences of **Handheld SLAM** category.

Table 8.2 details the sequences that belong to the *Handheld SLAM* category. This collection shows sequences simulating a handheld camera moving through larger scenes.

Sequence Name	**Duration**	**Length**
fr2/pioneer_360	72.75s	16.118m
fr2/pioneer_slam	155.72s	40.380m
fr2/pioneer_slam2	115.63s	21.735m
fr2/pioneer_slam3	111.91s	18.135m

Table 8.3: Sequences of **Robot SLAM** category.

Table 8.3 details the sequences that belong to the *Robot SLAM* category. The sequences of this group were captured from a robot-mounted camera navigating through an environment.

Sequence Name	**Duration**	**Length**
fr3/nostructure_notexture_far	15.79s	2.897m
fr3/nostructure_notexture_near_withloop	37.74s	11.739m
fr3/nostructure_texture_far	15.53s	4.343m
fr3/nostructure_texture_near_withloop	56.48s	13.456m
fr3/structure_notexture_far	27.28s	4.353m
fr3/structure_notexture_near	36.44s	3.872m
fr3/structure_texture_far	31.55s	5.884m
fr3/structure_texture_near	36.91s	5.050m

Table 8.4: Sequences of **Structure vs. Texture** category.

Table 8.3 details the sequences that belong to the *Structure vs. Texture* category. It is composed of sequences designed to test SLAM performance in environments with varying amounts of structure and texture.

Sequence Name	**Duration**	**Length**
fr2/desk_with_person	142.08s	17.044m
fr3/sitting_static	23.63s	0.259m
fr3/sitting_xyz	42.50s	5.496m
fr3/sitting_halfsphere	37.15s	6.503m
fr3/sitting_rpy	27.48s	1.110m
fr3/walking_static	24.83s	0.282m
fr3/walking_xyz	28.83s	5.791m
fr3/walking_halfsphere	35.81s	7.686m
fr3/walking_rpy	30.61s	2.698m

Table 8.5: Sequences of **Dynamic Objects** category.

Table 8.5 details the sequences that belong to the *Dynamic Objects* category. It has scenes with moving objects and people, challenging the robustness of SLAM algorithms.

Sequence Name	**Duration**	**Length**
fr1/plant	41.53s	14.795m
fr1/teddy	50.82s	15.709m
fr2/coke	84.55s	11.681m
fr2/dishes	100.55s	15.009m
fr2/flowerbouquet	99.40s	10.758m
fr2/flowerbouquet_brownbackground	76.89s	11.924m
fr2/metallic_sphere	75.60s	11.040m
fr2/metallic_sphere2	62.33s	11.813m
fr3/cabinet	38.58s	8.111m
fr3/large_cabinet	33.98s	11.954m
fr3/teddy	80.79s	19.807m

Table 8.6: Sequences of **3D Object Reconstruction** category.

Table 8.6 details the sequences that belong to the *3D Object Reconstruction* category. It is sequences focused on reconstructing the geometry of specific objects.

8.2 Trajectories Results

This portion details the trajectories obtained from the experiments conducted in Section 5.2.

8.2.1 TUM Dataset

Here, all the trajectories generated on TUM in the experiments presented in Section 5.2 are shown in Figures 8.1 to 8.9.

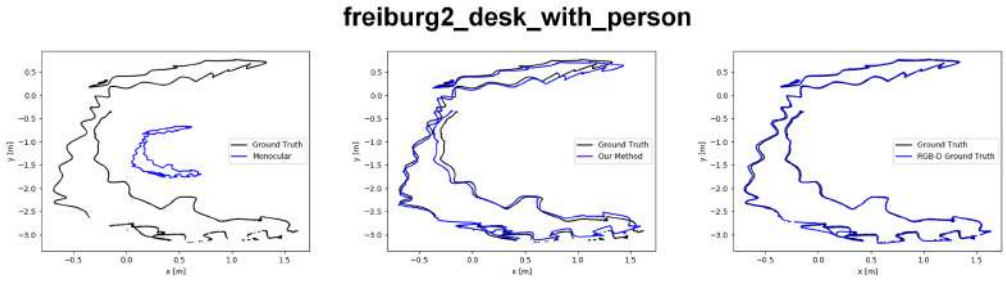

Figure 8.1: Plotted trajectory of sequence *freiburg2_desk_with_person*; comparison between monocular, our method and RGB-D ground truth approach.

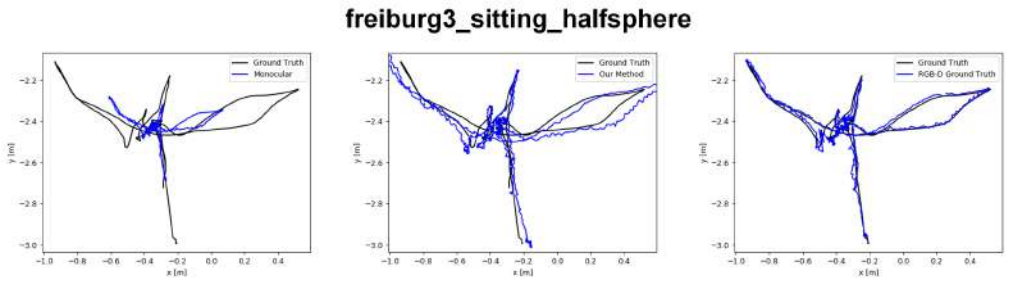

Figure 8.2: Plotted trajectory of sequence *freiburg3_sitting_halfsphere*; comparison between monocular, our method and RGB-D ground truth approach.

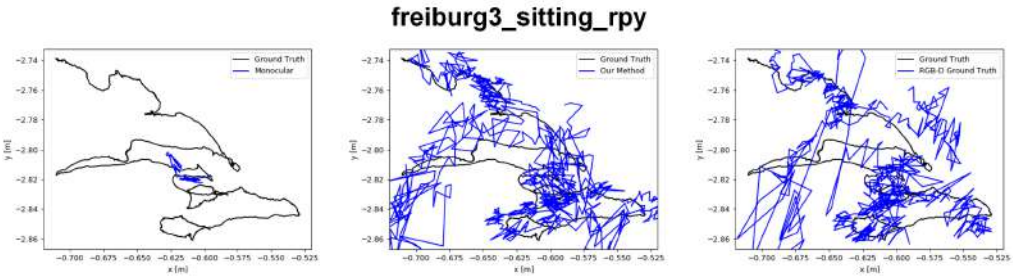

Figure 8.3: Plotted trajectory of sequence *freiburg3_sitting_rpy*; comparison between monocular, our method and RGB-D ground truth approach.

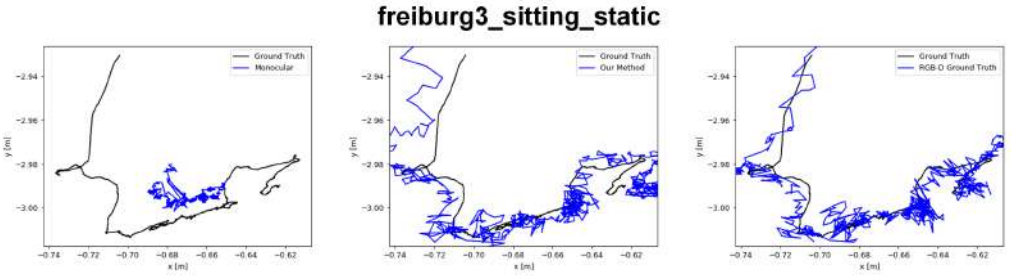

Figure 8.4: Plotted trajectory of sequence *freiburg3_sitting_static*; comparison between monocular, our method and RGB-D ground truth approach.

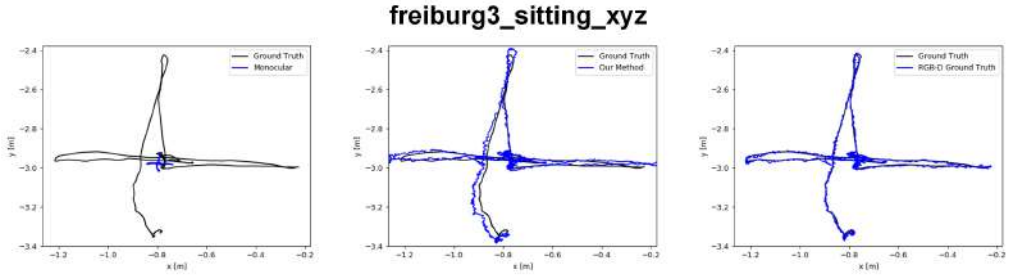

Figure 8.5: Plotted trajectory of sequence *freiburg3_sitting_xyz*; comparison between monocular, our method, and RGB-D ground truth approach.

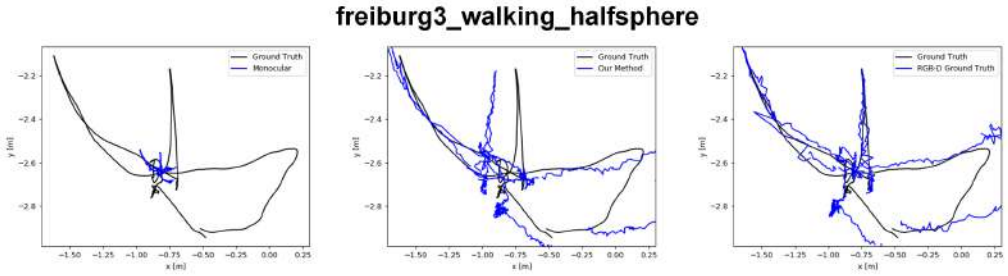

Figure 8.6: Plotted trajectory of sequence *freiburg3_walking_halfsphere*; comparison between monocular, our method and RGB-D ground truth approach.

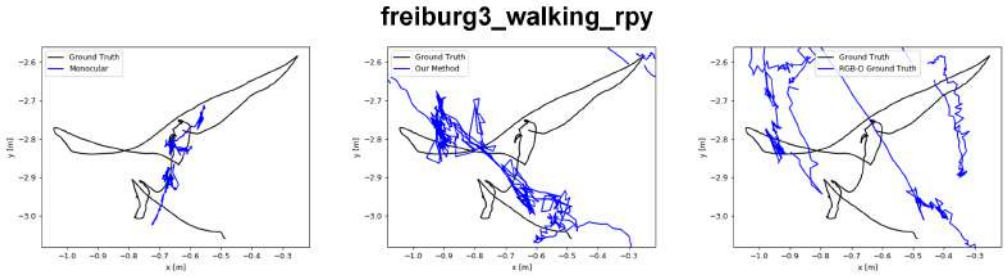

Figure 8.7: Plotted trajectory of sequence *freiburg3_walking_rpy*; comparison between monocular, our method and RGB-D ground truth approach.

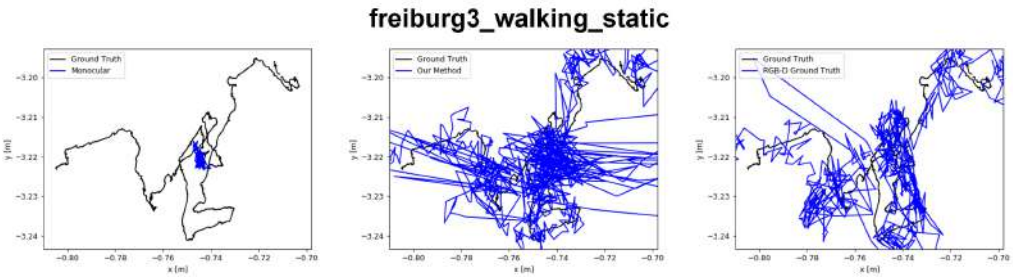

Figure 8.8: Plotted trajectory of sequence *freiburg3_walking_static*; comparison between monocular, our method and RGB-D ground truth approach.

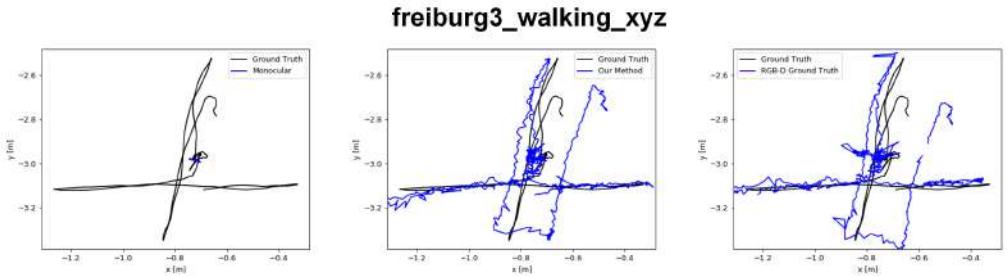

Figure 8.9: Plotted trajectory of sequence *freiburg3_walking_xyz*; comparison between monocular, our method and RGB-D ground truth approach.

8.2.2 KITTI Dataset

Here, all the trajectories generated on KITTI in the experiments presented in Section 5.2 are shown in Figures 8.10 to 8.19

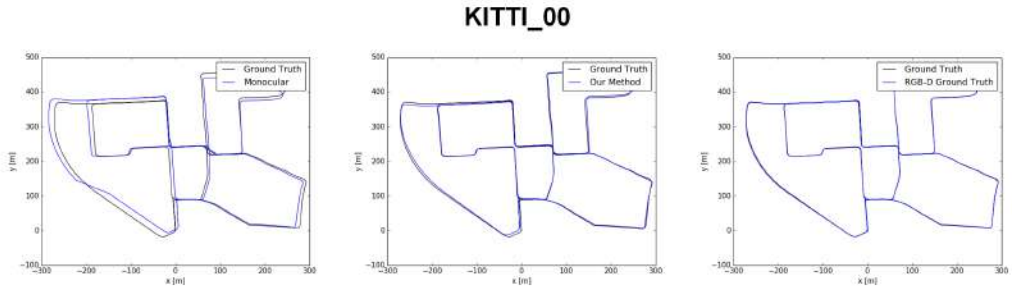

Figure 8.10: Plotted trajectory of sequence *00*; comparison between monocular, our method and RGB-D ground truth approach.

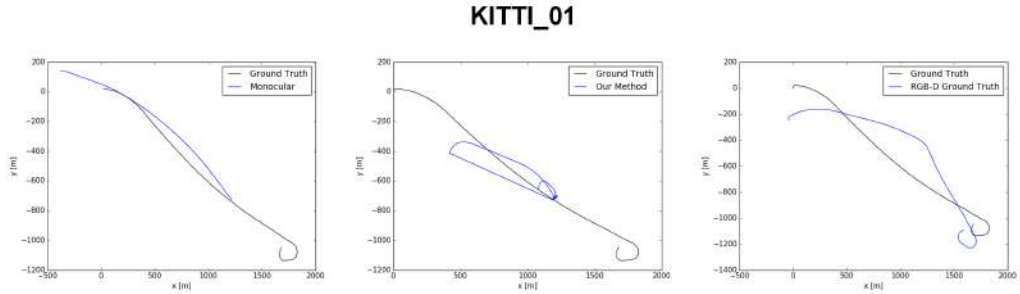

Figure 8.11: Plotted trajectory of sequence *01*; comparison between monocular, our method and RGB-D ground truth approach.

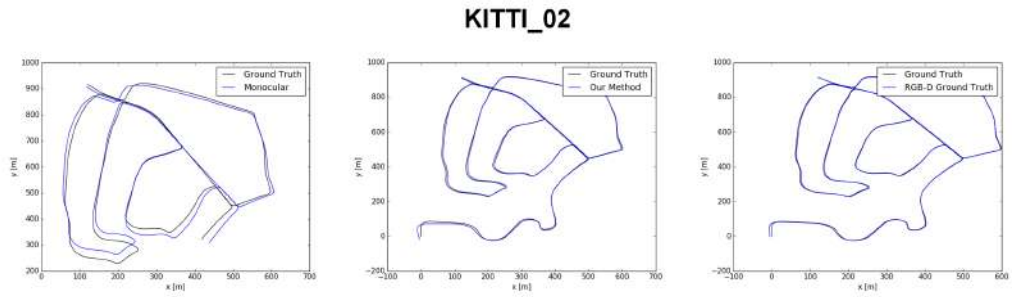

Figure 8.12: Plotted trajectory of sequence *02*; comparison between monocular, our method and RGB-D ground truth approach.

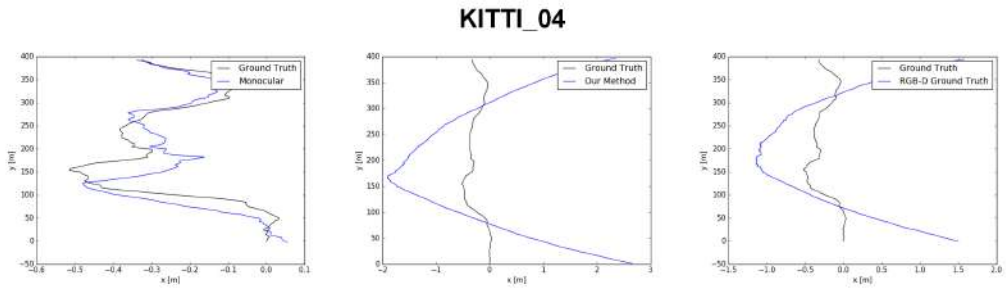

Figure 8.13: Plotted trajectory of sequence *04*; comparison between monocular, our method and RGB-D ground truth approach.

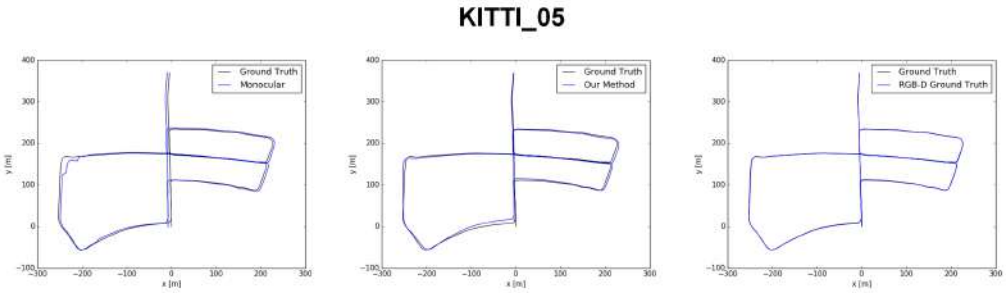

Figure 8.14: Plotted trajectory of sequence *05*; comparison between monocular, our method and RGB-D ground truth approach.

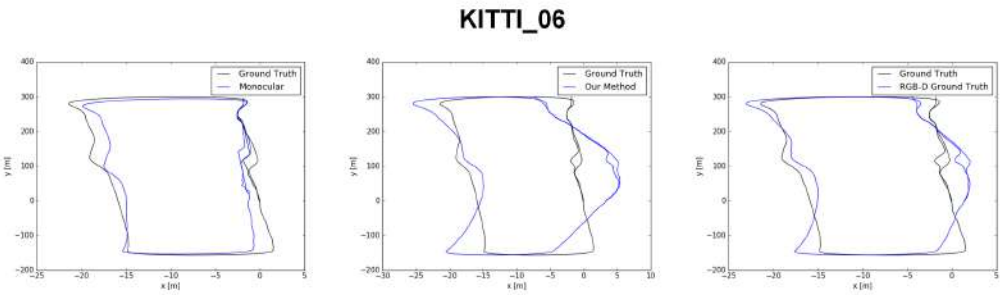

Figure 8.15: Plotted trajectory of sequence *06*; comparison between monocular, our method and RGB-D ground truth approach.

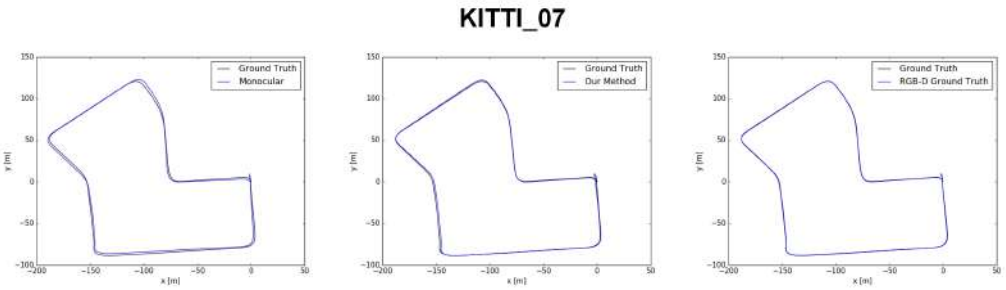

Figure 8.16: Plotted trajectory of sequence *07*; comparison between monocular, our method and RGB-D ground truth approach.

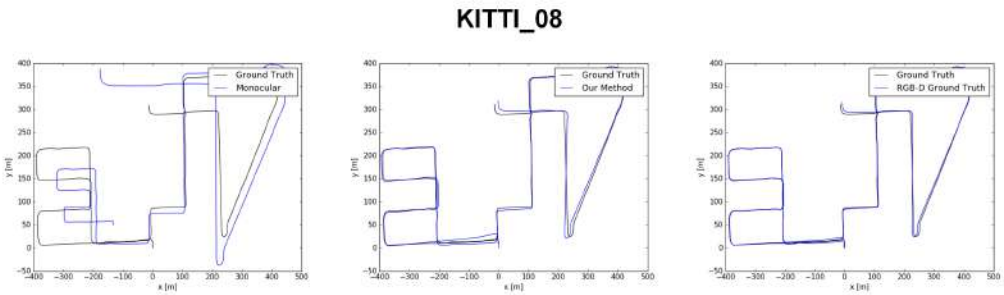

Figure 8.17: Plotted trajectory of sequence *08*; comparison between monocular, our method and RGB-D ground truth approach.

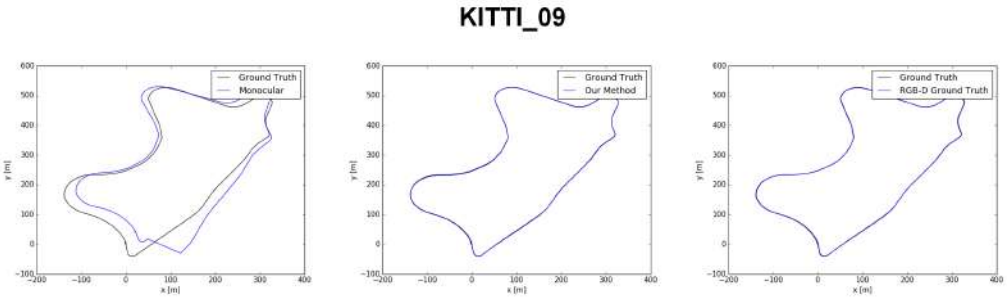

Figure 8.18: Plotted trajectory of sequence *09*; comparison between monocular, our method and RGB-D ground truth approach.

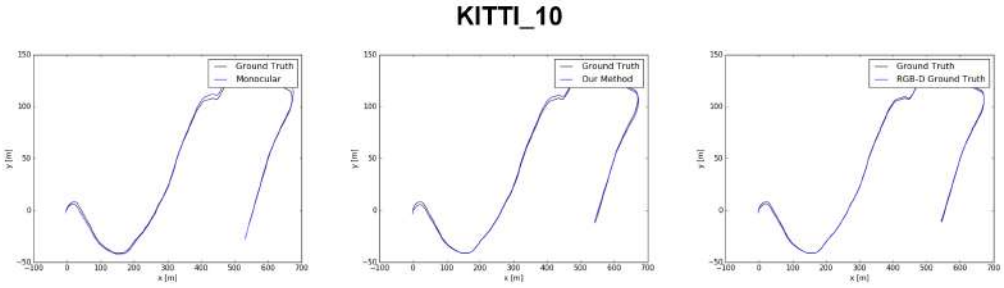

Figure 8.19: Plotted trajectory of sequence *10*; comparison between monocular, our method and RGB-D ground truth approach.

8.3 Experiments with Depth Generated Images

This section details the metrics for relative pose error in translation and rotation for the experiments conducted with the generated depth images in isolation.

8.3.1 TUM Dataset

The RPE results for the TUM Dataset are detailed in Tables 8.7 and 8.8.

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
freiburg2_desk_with_person	1.6073	0.1341	**0.0415**
freiburg3_sitting_halfsphere	0.2990	0.1226	**0.0295**
freiburg3_sitting_rpy	0.0858	**0.0393**	0.0403
freiburg3_sitting_static	0.0388	0.0164	**0.0131**
freiburg3_sitting_xyz	0.4131	0.0642	**0.0136**
freiburg3_walking_halfsphere	0.5020	0.3569	**0.1946**
freiburg3_walking_rpy	**0.1973**	0.7741	0.9157
freiburg3_walking_static	0.0306	0.0229	**0.0159**
freiburg3_walking_xyz	0.4303	**0.1206**	0.1354

Table 8.7: $RPE_{trans}(m)$ of the segments from the TUM database with monocular approach, our generated depth maps and true depth maps.

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
freiburg2_desk_with_person	**0.8164**	4.0892	1.3038
freiburg3_sitting_halfsphere	**0.7474**	2.4833	0.7819
freiburg3_sitting_rpy	2.5067	0.9782	**0.8227**
freiburg3_sitting_static	0.4329	0.4134	**0.3313**
freiburg3_sitting_xyz	0.5993	1.2178	**0.5777**
freiburg3_walking_halfsphere	**0.9060**	8.9389	2.1593
freiburg3_walking_rpy	14.1973	**13.9454**	17.9783
freiburg3_walking_static	0.4552	0.4563	**0.3400**
freiburg3_walking_xyz	3.3221	**1.5151**	1.5550

Table 8.8: $RPE_{rot}(deg)$ of the segments from the TUM database with monocular approach, our generated depth maps and true depth maps.

8.3.2 KITTI Dataset

Tables 8.9 and 8.10 present the RPE metrics obtained for the experiment with the depth-generated images in the KITTI dataset.

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
KITTI_00	2.2676	0.1695	**0.1288**
KITTI_01	2.1912	2.4028	**1.1074**
KITTI_02	1.7406	0.2666	**0.1217**
KITTI_04	0.3448	0.1009	**0.0744**
KITTI_05	3.4079	0.1348	**0.0818**
KITTI_06	0.7802	0.1294	**0.0706**
KITTI_07	1.1799	0.0695	**0.0666**
KITTI_08	1.0995	0.2643	**0.1986**
KITTI_09	1.9979	**0.1622**	0.1754
KITTI_10	0.8197	0.0873	**0.0680**

Table 8.9: $RPE_{trans}(m)$ of the segments from the KITTI database with monocular approach, our generated depth maps and true depth maps.

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
KITTI_00	**0.2510**	0.2946	0.2683
KITTI_01	0.3340	**0.0915**	0.2070
KITTI_02	0.1457	0.1362	**0.1204**
KITTI_04	**0.0493**	0.0971	0.0773
KITTI_05	0.1039	0.1257	**0.1023**
KITTI_06	0.0787	0.1045	**0.0697**
KITTI_07	**0.0568**	0.0755	0.0668
KITTI_08	**0.0640**	0.0889	0.0732
KITTI_09	0.0881	0.0737	**0.0648**
KITTI_10	**0.0726**	0.0927	0.0813

Table 8.10: $RPE_{rot}(deg)$ of the segments from the KITTI database with monocular approach, our generated depth maps and true depth maps.

8.4 Experiments with Dynamic Object Masks

This section details the metrics for relative pose error in translation and rotation for the experiments conducted using the dynamic object masks in isolation.

8.4.1 TUM Dataset

The RPE results for the TUM Dataset on a monocular approach are detailed in Tables 8.11 and 8.12.

Sequence	**Our Method ↓**	**Monocular ↓**
freiburg2_desk_with_person	1.7789	**1.6073**
freiburg3_sitting_halfsphere	0.4212	**0.2990**
freiburg3_sitting_rpy	0.0864	**0.0858**
freiburg3_sitting_static	0.0524	**0.0388**
freiburg3_sitting_xyz	0.4225	**0.4131**
freiburg3_walking_halfsphere	0.6185	**0.5020**
freiburg3_walking_rpy	**0.1813**	0.1973
freiburg3_walking_static	**0.0313**	0.0306
freiburg3_walking_xyz	**0.3754**	0.4303

Table 8.11: $RPE_{trans}(m)$ of the segments from the TUM dataset, comparing the use of segmentation masks against not using them, on monocular approach.

Sequence	**Our Method ↓**	**Monocular ↓**
freiburg2_desk_with_person	1.7429	**0.8164**
freiburg3_sitting_halfsphere	**0.4212**	0.7474
freiburg3_sitting_rpy	**0.0864**	2.5067
freiburg3_sitting_static	0.8468	**0.4329**
freiburg3_sitting_xyz	**0.5982**	0.5993
freiburg3_walking_halfsphere	**0.9001**	0.9060
freiburg3_walking_rpy	**4.4241**	14.1973
freiburg3_walking_static	3.6753	**0.4552**
freiburg3_walking_xyz	**0.5871**	3.3221

Table 8.12: $RPE_{rot}(deg)$ of the segments from the TUM dataset, comparing the use of segmentation masks against not using them, on monocular approach.

On Tables 8.13 and 8.14 the RPE metrics are detailed for the RGB-D approach.

Sequence	Our Method ↓	RGB-D Ground Truth ↓
freiburg2_desk_with_person	**0.0400**	0.0415
freiburg3_sitting_halfsphere	0.0356	**0.0295**
freiburg3_sitting_rpy	**0.0273**	0.0403
freiburg3_sitting_static	**0.0126**	0.0131
freiburg3_sitting_xyz	0.0179	**0.0136**
freiburg3_walking_halfsphere	**0.0367**	0.1946
freiburg3_walking_rpy	**0.0620**	0.9157
freiburg3_walking_static	0.0308	**0.0159**
freiburg3_walking_xyz	**0.0245**	0.1354

Table 8.13: $RPE_{trans}(m)$ of the segments from the TUM dataset, comparing the use of segmentation masks against not using them, on RGB-D approach.

Sequence	Our Method ↓	RGB-D Ground Truth ↓
freiburg2_desk_with_person	**1.2469**	1.3038
freiburg3_sitting_halfsphere	0.9496	**0.7819**
freiburg3_sitting_rpy	**0.7874**	0.8227
freiburg3_sitting_static	0.3703	**0.3313**
freiburg3_sitting_xyz	0.6009	**0.5777**
freiburg3_walking_halfsphere	**0.9327**	2.1593
freiburg3_walking_rpy	**1.3372**	17.9783
freiburg3_walking_static	0.5839	**0.3400**
freiburg3_walking_xyz	**0.6523**	1.5550

Table 8.14: $RPE_{rot}(deg)$ of the segments from the TUM dataset, comparing the use of segmentation masks against not using them, on RGB-D approach.

8.4.2 KITTI Dataset

The RPE results for KITTI on a monocular approach are detailed in Tables 8.15 and 8.16.

Sequence	**Our Method ↓**	**Monocular ↓**
KITTI_00	2.4100	**2.2676**
KITTI_01	**1.7626**	2.1912
KITTI_02	1.9218	**1.7406**
KITTI_04	**0.3220**	0.3448
KITTI_05	**3.2173**	3.4079
KITTI_06	**0.6918**	0.7802
KITTI_07	**1.1568**	1.1799
KITTI_08	1.3951	**1.0995**
KITTI_09	**1.3546**	1.9979
KITTI_10	**0.7843**	0.8197

Table 8.15: $RPE_{trans}(m)$ of the segments from the KITTI dataset, comparing the use of segmentation masks against not using them, on monocular approach.

Sequence	**Our Method ↓**	**Monocular ↓**
KITTI_00	0.2548	**0.2510**
KITTI_01	**0.2123**	0.3340
KITTI_02	**0.1178**	0.1457
KITTI_04	**0.0449**	0.0493
KITTI_05	**0.0698**	0.1039
KITTI_06	**0.0616**	0.0787
KITTI_07	**0.0558**	0.0568
KITTI_08	0.0804	**0.0640**
KITTI_09	**0.0620**	0.0881
KITTI_10	**0.0680**	0.0726

Table 8.16: $RPE_{rot}(deg)$ of the segments from the KITTI dataset, comparing the use of segmentation masks against not using them, on monocular approach.

Tables 8.17 and 8.18 detail RPE metrics for the RGB-D approach.

Sequence	Our Method ↓	RGB-D Ground Truth ↓
KITTI_00	0.1726	**0.1288**
KITTI_01	2.3001	**1.1074**
KITTI_02	**0.1086**	0.1217
KITTI_04	0.0906	**0.0744**
KITTI_05	**0.0771**	0.0818
KITTI_06	**0.0624**	0.0706
KITTI_07	**0.0638**	0.0666
KITTI_08	**0.1769**	0.1986
KITTI_09	**0.1730**	0.1754
KITTI_10	**0.0679**	0.0680

Table 8.17: $RPE_{trans}(m)$ of the segments from the KITTI dataset, comparing the use of segmentation masks against not using them, on RGB-D approach.

Sequence	Our Method ↓	RGB-D Ground Truth ↓
KITTI_00	**0.2502**	0.2683
KITTI_01	**0.0467**	0.2070
KITTI_02	0.1246	**0.1204**
KITTI_04	0.1039	**0.0773**
KITTI_05	**0.0913**	0.1023
KITTI_06	**0.0632**	0.0697
KITTI_07	**0.0624**	0.0668
KITTI_08	0.0736	**0.0732**
KITTI_09	0.0653	**0.0648**
KITTI_10	**0.0752**	0.0813

Table 8.18: $RPE_{rot}(deg)$ of the segments from the KITTI dataset, comparing the use of segmentation masks against not using them, on RGB-D approach.

8.5 Experiments with Depth Generated Images and Dynamic Object Masks

This section details the RPE metrics obtained using the complete methodology of this study. The complete methodology utilizes both depth-generated images and dynamic segmentation masks.

8.5.1 TUM Dataset

The RPE metrics obtained for the TUM dataset using the complete methodology are detailed in Tables 8.19 and 8.20.

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
reiburg2_desk_with_person	1.6073	0.1329	**0.0415**
freiburg3_sitting_halfsphere	0.2990	0.1083	**0.0295**
freiburg3_sitting_rpy	0.0858	0.0493	**0.0403**
freiburg3_sitting_static	0.0388	0.0126	**0.0131**
freiburg3_sitting_xyz	0.4131	0.0532	**0.0136**
freiburg3_walking_halfsphere	0.5020	**0.0772**	0.1946
freiburg3_walking_rpy	0.1973	**0.0789**	0.9157
freiburg3_walking_static	0.0306	**0.0103**	0.0159
freiburg3_walking_xyz	0.4303	**0.0535**	0.1354

Table 8.19: Comparing the performance of the entire method with the monocular approach and RGB-D Ground Truth on TUM dataset on $RPE_{trans}(m)$ metric.

Sequence	Monocular ↓	Our Method ↓	RGB-D Ground Truth ↓
freiburg2_desk_with_person	**0.8164**	3.9327	1.3038
freiburg3_sitting_halfsphere	**0.7474**	2.3668	0.7819
freiburg3_sitting_rpy	2.5067	1.0194	**0.8227**
freiburg3_sitting_static	0.4329	0.3712	**0.3313**
freiburg3_sitting_xyz	0.5993	1.1657	**0.5777**
freiburg3_walking_halfsphere	**0.9060**	2.0240	2.1593
freiburg3_walking_rpy	14.1973	**1.5396**	17.9783
freiburg3_walking_static	0.4552	**0.2730**	0.3400
freiburg3_walking_xyz	3.3221	**1.3405**	1.5550

Table 8.20: Comparing the performance of the entire method with the monocular approach and RGB-D Ground Truth on TUM dataset on $RPE_{rot}(deg)$ metric.

8.5.2 KITTI Dataset

The RPE metrics obtained on KITTI using the complete methodology are detailed in Tables 8.19 and 8.20.

Sequence	**Monocular ↓**	**Our Method ↓**	**RGB-D Ground Truth ↓**
KITTI_00	2.2676	0.1791	**0.1288**
KITTI_01	2.1912	1.5608	**1.1074**
KITTI_02	1.7406	0.2638	**0.1217**
KITTI_04	0.3448	**0.0703**	0.0744
KITTI_05	3.4079	0.1332	**0.0818**
KITTI_06	0.7802	0.1182	**0.0706**
KITTI_07	1.1799	0.0702	**0.0666**
KITTI_08	1.0995	0.2198	**0.1986**
KITTI_09	1.9979	**0.1564**	0.1754
KITTI_10	0.8197	0.0864	**0.0680**

Table 8.21: Comparing the performance of the entire method with the monocular approach and RGB-D Ground Truth on KITTI dataset on $RPE_{trans}(m)$ metric.

Sequence	**Monocular ↓**	**Our Method ↓**	**RGB-D Ground Truth ↓**
KITTI_00	**0.2510**	0.2620	0.2683
KITTI_01	0.3340	0.2570	**0.2070**
KITTI_02	0.1457	0.1295	**0.1204**
KITTI_04	**0.0493**	0.0802	0.0773
KITTI_05	0.1039	0.1145	**0.1023**
KITTI_06	0.0787	0.1468	**0.0697**
KITTI_07	**0.0568**	0.0746	0.0668
KITTI_08	**0.0640**	0.0879	0.0732
KITTI_09	0.0881	0.0745	**0.0648**
KITTI_10	**0.0726**	0.0897	0.0813

Table 8.22: Comparing the performance of the entire method with the monocular approach and RGB-D Ground Truth on KITTI dataset on $RPE_{rot}(deg)$ metric.