Signal Processing Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://sps.ewi.tudelft.nl/

SPS-2024-5144264

M.Sc. Thesis

Robust Monocular Depth Estimation For UAVs

Ibrahim Hassan B.Sc.

Abstract

This thesis presents an approach to monocular depth estimation for Unmanned Aerial Vehicles (UAVs). Monocular depth estimation is a critical perception task for UAVs, enabling them to infer depth information from visual data without relying on heavy or power-consuming sensors such as LiDAR or stereo cameras. Given the operational constraints of UAVs, such as limited payload and energy resources, robust and efficient depth estimation methods are required to facilitate safe navigation and environmental interaction.

The proposed methodology in this thesis integrates visual data from a monocular camera with inertial measurements from an Inertial Measurement Unit (IMU) sensor. This combination aims to address challenges such as scale ambiguity in the depth estimates and inaccuracies in dynamic environments that are common in aerial operations. The integration of IMU data with a differentiable camera-centric Extended Kalman Filter (EKF) allows for better ego-motion estimation, effectively calibrating the visual information with drone dynamics.

The method further incorporates depth map frame prediction, leveraging initial depth estimates along with temporal dynamics to predict future depth maps. This predictive capability improves efficiency by reducing the need for full depth estimation in every frame, allowing robotic agents to anticipate environmental changes. The evaluation on simulated and real-world datasets shows that while the algorithm performs well over short forecast horizons, accumulating errors from IMU data and the assumption of a static environment limit its long-term accuracy. The future depth map prediction algorithm reduced the need for DynaDepth from 10 runs per second to 2, and on the Mid-Air dataset, from 25 to 5. Additionally, this study provides a foundation for future work, including the integration of an object-oriented frame prediction algorithm.

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

Robust Monocular Depth Estimation For UAVs

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Ibrahim Hassan B.Sc.
born in Delft, The Netherlands

This work was performed in:

Signal Processing Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Robust Monocular Depth Estimation For UAVs"** by **Ibrahim Hassan B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: December 18, 2024

Chairman:	Dr. R.T. Rajan
Advisor:	Dr. R.T. Rajan
Committee Members:	Dr.ir. R.Sabzevari

Abstract

This thesis presents an approach to monocular depth estimation for Unmanned Aerial Vehicles (UAVs). Monocular depth estimation is a critical perception task for UAVs, enabling them to infer depth information from visual data without relying on heavy or power-consuming sensors such as LiDAR or stereo cameras. Given the operational constraints of UAVs, such as limited payload and energy resources, robust and efficient depth estimation methods are required to facilitate safe navigation and environmental interaction.

The proposed methodology in this thesis integrates visual data from a monocular camera with inertial measurements from an Inertial Measurement Unit (IMU) sensor. This combination aims to address challenges such as scale ambiguity in the depth estimates and inaccuracies in dynamic environments that are common in aerial operations. The integration of IMU data with a differentiable camera-centric Extended Kalman Filter (EKF) allows for better ego-motion estimation, effectively calibrating the visual information with drone dynamics.

The method further incorporates depth map frame prediction, leveraging initial depth estimates along with temporal dynamics to predict future depth maps. This predictive capability improves efficiency by reducing the need for full depth estimation in every frame, allowing robotic agents to anticipate environmental changes. The evaluation on simulated and real-world datasets shows that while the algorithm performs well over short forecast horizons, accumulating errors from IMU data and the assumption of a static environment limit its long-term accuracy. The future depth map prediction algorithm reduced the need for DynaDepth from 10 runs per second to 2, and on the Mid-Air dataset, from 25 to 5. Additionally, this study provides a foundation for future work, including the integration of an object-oriented frame prediction algorithm.

Acknowledgments

With this thesis, I conclude my academic journey, which began 20 years ago in primary school. Over the years, I have forged lifelong friendships, cherished unforgettable moments, and went through easy and challenging times that have profoundly shaped the person I am today. If I had to go back, I wouldn't change a single thing.

First and foremost, I would like to thank my parents, brothers, and sister for their unwavering support. To my friends—whom I consider my brothers—thank you for making our study sessions in the library, our time outside the university, and the trips we went on so memorable. I also want to express my gratitude to Raj for your support, advice, and guidance over the past nine months. It was a pleasure working with you. Lastly, I am deeply grateful to Professor Sabzevari for agreeing to serve on the thesis committee.

All good things must come to an end, and I am excited for the future!

الحمدُللّٰه.

Ibrahim Hassan B.Sc.
Delft, The Netherlands
December 18, 2024

Contents

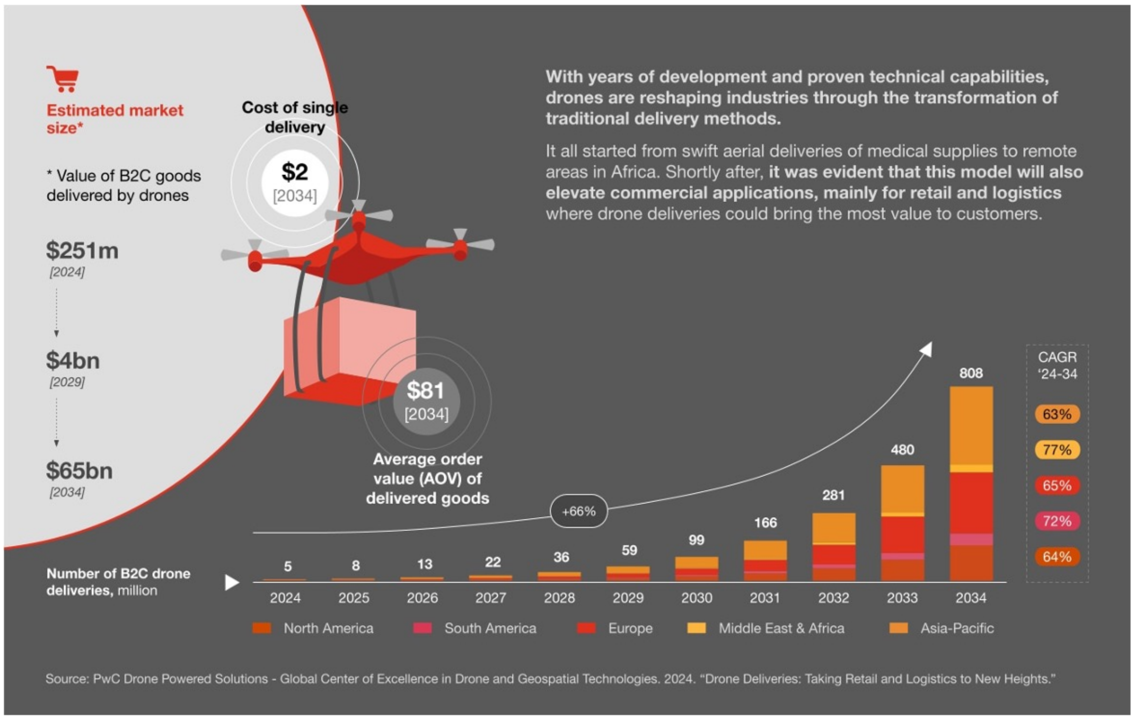
List of Figures

List of Tables

Introduction 1

1.1 Background: Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, are versatile remotely operated aircrafts that have gained widespread popularity and utility in various fields in recent years [8, 9, 10]. They range from small, consumer-grade quadcopters to large sized cargo drones. Drones are used in a variety of industries, including cinematography, agriculture, surveillance, search and rescue, and parcel delivery. Their adoption is increasing across different sectors. Notably, McKinsey has projected a record number of commercial drone delivery flights worldwide in 2023, following an 80 percent increase from 2021 to 2022 [11]. Figure 1.1 shows an outlook for the number of drone deliveries in the period 2024-2034, which will increase exponentially worldwide over the coming 10 years to a value of 65 billion dollars in delivered goods.



With years of development and proven technical capabilities, drones are reshaping industries through the transformation of traditional delivery methods.

It all started from swift aerial deliveries of medical supplies to remote areas in Africa. Shortly after, **it was evident that this model will also elevate commercial applications, mainly for retail and logistics** where drone deliveries could bring the most value to customers.

Source: PwC Drone Powered Solutions - Global Center of Excellence in Drone and Geospatial Technologies. 2024. "Drone Deliveries: Taking Retail and Logistics to New Heights."

Figure 1.1: Outlook drone delivery 2024-2034 [1].

Enhanced with Artificial Intelligence (AI), drones can perform complex tasks, such as autonomous navigation to predetermined destinations under adverse weather conditions with limited visibility and conducting agricultural tasks like ripeness assessments and crop spraying. Drones offer many advantages over human-centered alternative solutions. Their ability to reach remote, hazardous, or otherwise inaccessible areas quickly makes them invaluable in scenarios where time is of the essence. For instance, in search and rescue operations, drones can cover large areas rapidly, providing real-time data and potentially saving lives. Additionally, deploying drones can be more cost-effective and precise than using manned aircraft or ground personnel, especially for routine tasks such as agricultural monitoring, infrastructure inspection, and environmental surveying. Despite the availability of autonomous drones, significant technical challenges remain [12], such as in autonomous navigation, which is a central theme in the above-mentioned applications.

The typical pipeline for a sensor-based autonomous flying drone is depicted in Figure 1.2. These drones can be equipped with a range of sensors, including LIDAR, radar, cameras, Inertial Measurement Units (IMUs), and GPS, to collect data. To leverage the complimentary benefits of the different sensors, the data can be combined in a process known as sensor fusion. In the perception stage, the goal is to analyze the measurements to understand and interpret the environment. Using various algorithms, relevant information about the surroundings is extracted through typical perception tasks, such as object detection [13, 14, 15, 16, 17], depth estimation [6, 18, 19] and semantic segmentation [20]. With the extracted information, the localization stage aims to determine the drone's position on the map. Based on the perceived environment and the drone's location, algorithms generate a path or trajectory for the drone to follow [21]. The corresponding control signals are then transmitted to the control system to physically set the drone's course.

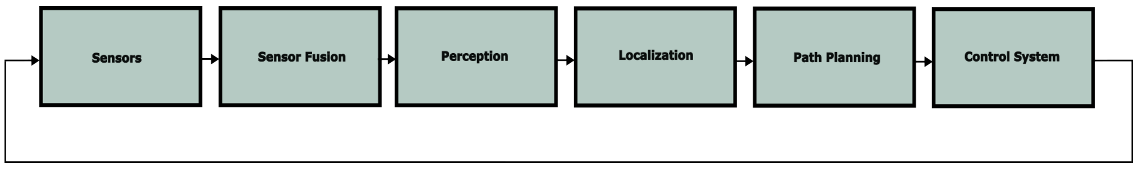


Figure 1.2: Pipeline autonomous drone operation.

1.2 Motivation

The focus of this thesis is depth estimation in the perception stage (see figure 1.2). Depth estimation is a particularly challenging perception task essential not only for drones but for all types of robotics. Many research papers on tasks following the perception module (e.g., path planning) often assume accurate depth estimates and build on that assumption. However, inaccurate depth estimates can significantly impact the performance of these algorithms, as errors propagate through the pipeline. Therefore, estimating accurate depth values and understanding how inaccurate depth values affect downstream task performance is paramount for the successful deployment of any robotic agent.

Depth estimation is the process of determining the distance of an object in a scene from a sensor (e.g., camera, LiDAR, radar). When a camera is used, a depth map is predicted based on the given image as shown in figure 1.3 where every pixel in the depth map stores the distance towards that point in the scene typically expressed in meters.

This depth data is valuable for extracting information about the three-dimensional surroundings of robots and is crucial for functions such collision avoidance and 3D-scene reconstruction [22]. It is often performed using a stereo camera system, which uses two cameras placed at a specific distance apart to capture images of the same scene from slightly different perspectives as illustrated in figure 1.4 . By comparing these images, the system can calculate the disparity between corresponding points in the images. This disparity is then used to determine the depth information for each point in the scene, based on the principles of triangulation.

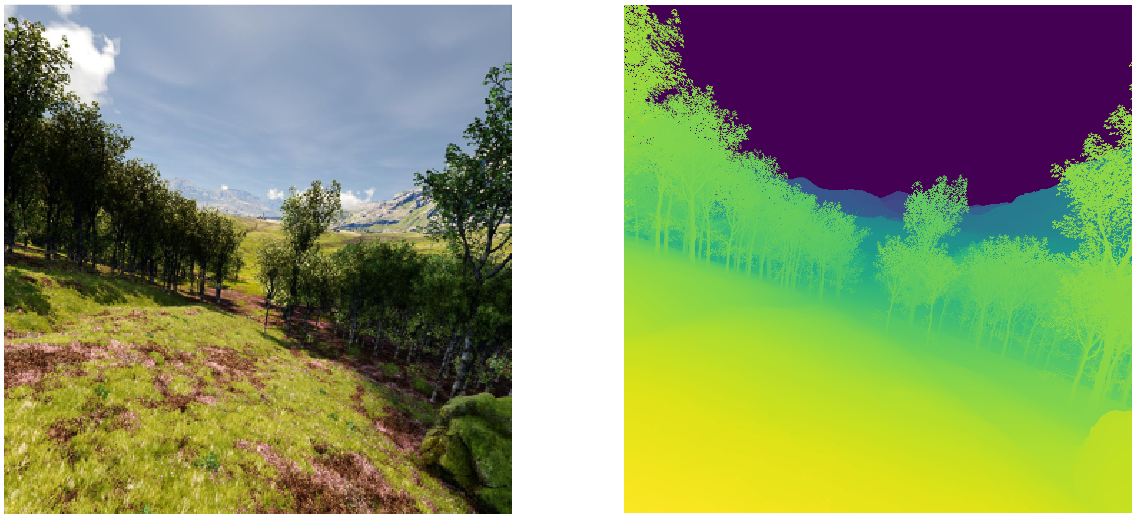


Figure 1.3: Depth map predicted from a given image, where the color of each pixel is proportional to the distance of a point in the scene from the camera, measured in meters [2].

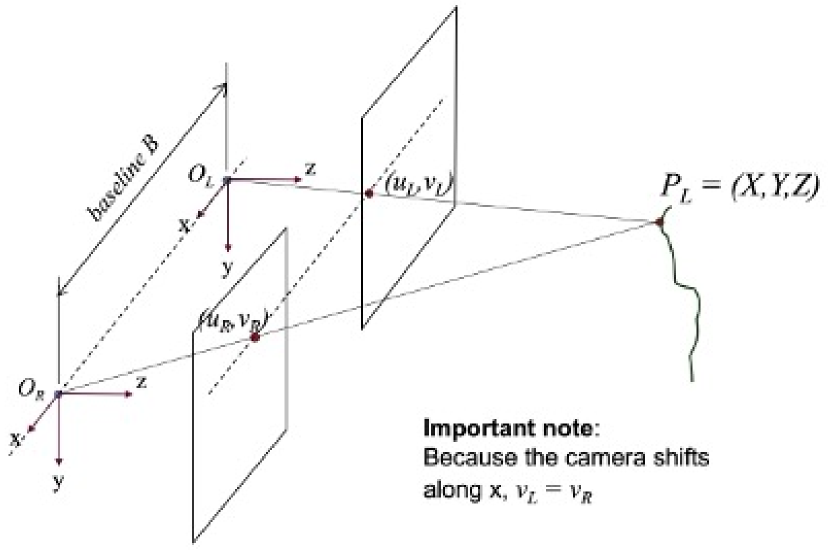


Figure 1.4: Stereo camera setup, capturing scene from two different perspectives with two cameras [3].

However, robotic agents do not have unlimited resources. Small robotic agents, such as Micro Air Vehicles (MAVs) [23], face energy and weight constraints. Therefore, ongoing efforts aim to make robotic agents lighter and more power-efficient. One way for satisfy those constraints is by reducing the number of sensors, thus performing depth estimation with just one camera. This method, known as monocular depth estimation, is not straightforward as it is an ambiguous task; many different 3D scenes can be projected into identical 2D coordinates [24].

Methods for monocular depth estimation often leverage machine learning and computer vision techniques to infer depth information from single or multiple images. These techniques include supervised learning approaches that use labeled datasets with depth information, as well as unsupervised methods that exploit geometric and photometric consistency across multiple frames or within a single frame. Recent advancements have demonstrated promising results, achieving performance levels comparable to stereo systems. This progress opens up new possibilities for lightweight and energy-efficient robotic applications [25, 26].

Monocular depth estimation is not only relevant for small, lightweight robotic agents but can also be applied to various other robotic applications as a backup system for stereo depth estimation, such as in moon rovers like Lunar Zebro [27]. Lunar Zebro, equipped with a stereo camera, is set to be deployed on the moon. Considering the harsh conditions on the moon and the financial and development time invested in the agent, it is crucial that the agent remains functional if any of its sensors fail. For Lunar Zebro, for instance, this would mean that the agent should still be capable of performing depth estimation even if one of the stereo cameras stops working.

1.3 Evaluation Datasets

The performance of the depth estimation models in this thesis are evaluated on two datasets: the KITTI dataset [7] and the Mid-Air dataset [28]. The KITTI dataset is a benchmark suite widely used in autonomous driving research. It provides real-world data collected by a moving vehicle equipped with high-resolution stereo cameras and laser scanners (see Figure 1.5) offering ground truth depth information for urban and semi-urban environments. This dataset is essential for assessing depth estimation models in automotive settings, where challenges include dynamic objects, varying lighting conditions, and complex scenes. Conversely, the Mid-Air dataset is a synthetic, photorealistic dataset specifically designed for aerial robotics applications. It simulates diverse flight scenarios with precise ground truth depth maps, enabling the evaluation of depth models in aerial contexts such as drone navigation.

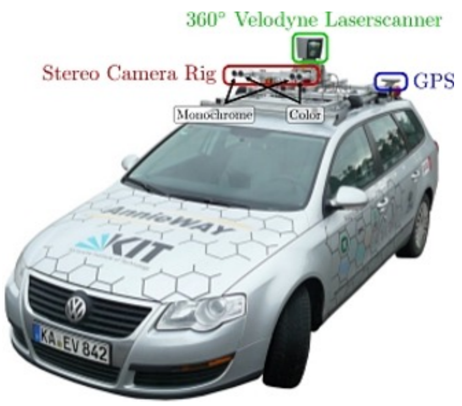


Figure 1.5: The vehicle used for capturing the KITTI dataset, equipped with high-resolution stereo cameras, a laser scanner, and precise GPS/IMU navigation systems for data collection in autonomous driving scenarios.

1.4 Research goal

In contrast to robotics operating on 2D-ground plane, drones operate in a three-dimensional space involving complex dynamics due to aerodynamic forces, gravity, wind disturbances, and the coupling between rotational and translational motions. Drone dynamics are captured by an

Inertial Measurement Unit (IMU) sensor, which measures both linear acceleration and angular velocity. The IMU sensor provides data about the drone's motion and orientation, enabling the drone to adjust its position and stabilize itself in response to environmental disturbances. This data reflects the drone's velocity, changes in acceleration, and rotational movements, all of which are vital for maintaining balance and achieving precise navigation.

Integrating dynamic data from the IMU with visual information from the onboard camera opens the possibility of improving depth estimation models. Traditional monocular depth estimation methods (e.g.,[29, 6, 18]) rely solely on visual data which lead to scale errors in depth estimates. Since these methods do not account for robot motion or dynamics, they struggle to determine the actual scale of objects in the environment accurately. This lack of absolute scale information often results in depth predictions that are inconsistent or misaligned with the true distances in the scene, which can be particularly problematic for UAVs navigating through complex three-dimensional spaces. Therefore, the goal of this thesis is to investigate how incorporating drone dynamics improves monocular depth estimation.

In addition, this thesis explores the prediction of future depth maps by leveraging initial depth estimates and temporal dynamics. The approach involves using an initial depth map generated by the depth estimation model together with IMU sensor data to predict how the depth structure of the environment will evolve over subsequent time frames. This predictive capability serves multiple crucial purposes. First, it enables robots to anticipate environmental changes before they occur, allowing for more proactive and smooth navigation decisions. Second, it addresses computational efficiency concerns by reducing the frequency of running the full depth estimation model on every frame, as interpolated predictions can fill the gaps between full model inferences.

1.5 Thesis outline

In summary, this thesis answers the following research questions:

1. How can incorporating drone dynamics improve monocular depth estimation?
2. How can future depth maps be predicted using an initial depth map estimate and temporal dynamics?

The structure of this thesis is as follows. Chapter 2 provides a review of the literature on depth estimation, discussing different types of sensors robots can be equipped with, the standard pinhole camera model, existing depth estimation methods ranging from traditional methods to deep learning-based methods and finally priors that can improve depth estimates. Chapter 3 discusses the proposed depth estimation model in detail. The model consists of several subnetworks including a depth network to regress depth and a camera-centric Extended Kalman Filter fusion block to mitigate scale ambiguity. Chapter 4 provides an overview of the datasets used to evaluate the depth estimation methods. Additionally, the chapter presents the conducted experiments and the evaluation of the results. The experiments are divided into two sections: the first section discusses the results of the proposed depth estimation model and the second part examines the proposed future depth prediction algorithm along with its evaluation on the datasets. Finally, Chapter 5 presents the thesis conclusion, summarizing the efforts and results achieved in relation to the proposed research objectives. It also explores possible future directions for the project, addressing the limitations encountered during the study and providing recommendations and suggestions for further research.

2 Literature review

This chapter reviews the key literature on monocular depth estimation, focusing on both theoretical foundations and practical methods. This literature review discusses sensor technologies, the traditional pinhole camera model, and various depth estimation approaches, including those enhanced with prior information.

2.1 Sensors

Robots can use various types of sensors, such as monocular cameras, stereo cameras, LiDAR, and radar, to improve their situational awareness and response capabilities. However, each sensor has limitations, and integrating them into robots imposes constraints related to their weight and size. LiDAR and radar sensors, for example, provide sparse measurements, limiting their effectiveness in achieving dense data recovery [22]. Additionally, LiDAR and radar sensors are energy-consuming and costly compared to cameras, and their weights make them less attractive for lightweight robotics like drones [24]. On the other hand, stereo cameras provide denser measurements but are limited by the baseline distance between the two cameras [22], resulting in less accurate measurements for objects farther from the cameras. In contrast, monocular RGB cameras are cheap, provide high-density measurements, are lighter than the stereo camera, and consume less energy compared to other sensors.

2.2 Camera Model

A commonly used theoretical framework for representing the camera model is the pinhole camera model. This model describes the relationship between a point in the real world and its projection onto an image plane. Figure 2.1 shows an overview of the pinhole camera model.

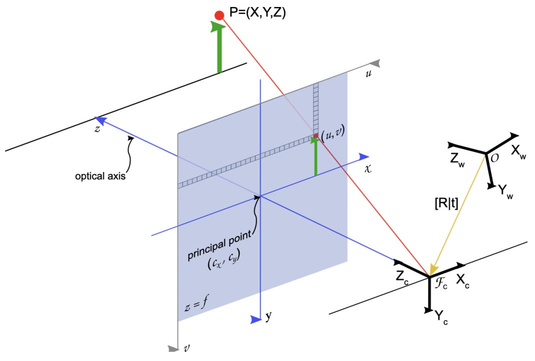


Figure 2.1: Pinhole camera model[4].

where the three-dimensional point $\boldsymbol{P} = [X, Y, Z]$ is mapped to an image with pixel coordinates (u, v). The principal point (c_x, c_y) is the point where the camera's optical axis - the line passing through the camera's optical center and perpendicular to the image plane - intersects the image plane. In an ideal pinhole camera model, the principal point would be at the center of the image plane. However, due to various imperfections in the lens or sensor alignment, the actual principal point may have an offset from the center. The focal length f is the distance from the optical center to the image plane.

The camera model can be represented by the intrinsic matrix $\boldsymbol{K}$ defined as:

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where f_x and f_y denote the horizontal and vertical focal lengths of the camera in pixel units, respectively, c_x and c_y represent the camera's principal point in pixels, with its origin projected onto the image plane, typically at the image center, and s denotes a scale factor [4]. Points in 3-dimensional space, represented as $\boldsymbol{P_w}$ in the world frame, are transformed into 3-dimensional points $\boldsymbol{P_c}$ in the camera frame using the following transformation:

$$\boldsymbol{P_c} = \boldsymbol{R}\boldsymbol{P_w} + \boldsymbol{t} \tag{2.2}$$

where $\boldsymbol{R}$ is a 3×3 rotation matrix and $\boldsymbol{t}$ the translation vector. This operation, known as the extrinsic transformation, can be efficiently represented using homogeneous coordinates. This is achieved by augmenting the 3D world coordinate point with 1: $\boldsymbol{P_w} = [X, Y, Z, 1]^T$, and forming the extrinsic translation:

$$\boldsymbol{P_c} = \boldsymbol{T}\boldsymbol{P_w} \tag{2.3}$$

where $\boldsymbol{T} = [\boldsymbol{R}|\boldsymbol{t}]$. Finally, transforming the points in the camera frame to image frame is performed by multiplying the camera coordinates $\boldsymbol{P_c}$ with intrinsic camera matrix $\boldsymbol{K}$:

$$s \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{1} \end{bmatrix} = \boldsymbol{K}\boldsymbol{P_c} \tag{2.4}$$

where s is a scaling factor, $\boldsymbol{u}$ and $\boldsymbol{v}$ are the coordinates on the image plane, expressed in pixels, and $\boldsymbol{1}$ is a vector consisting entirely of ones.

In a stereo camera setup, depth (distance in the z-direction) is derived from geometry principles. Figure 2.2 illustrates a stereo camera configuration with two cameras, L and R, spaced by baseline b. Point P in the scene is projected onto point x_L on the image plane of the left camera and onto x_R on the image plane of the right camera. The difference in the projection positions of point P between the left and right camera images is known as disparity.

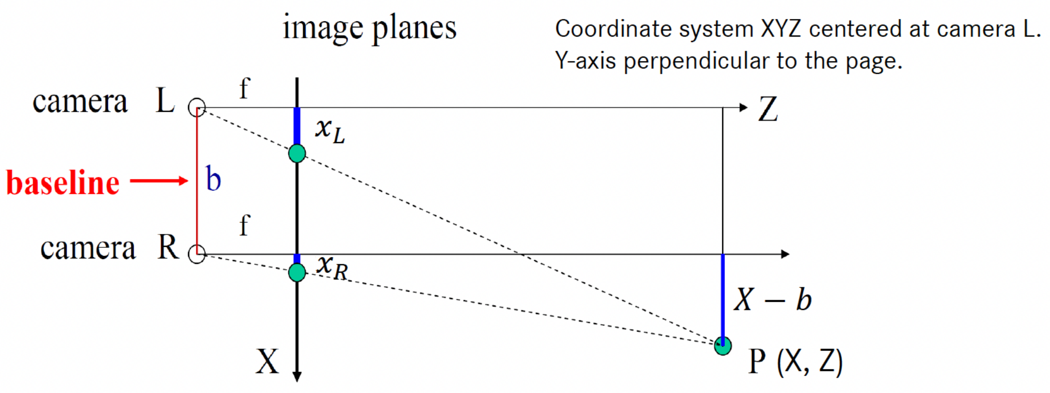


image planes
Coordinate system XYZ centered at camera L.
Y-axis perpendicular to the page.
camera L
f
x_L
Z
baseline
b
f
camera R
x_R
$X - b$
X
P (X, Z)

Figure 2.2: Depth coplanar stereo configuration [5]

.

Using geometry, the following relationships can be derived:

$$\frac{Z}{f} = \frac{X}{x_L} \qquad \frac{Z}{f} = \frac{X - b}{x_R} \tag{2.5}$$

Rewriting equations 2.5 in terms of x_L and x_R and substituting them into the disparity equation in 2.6, the following expression for depth is obtained:

$$d = x_L - x_R \tag{2.6}$$

$$Z = \frac{fb}{x_L - x_R} \tag{2.7}$$

where d represents the disparity, Z is the depth, f is the focal length, and b is the baseline distance between the two cameras.

2.3 Depth Estimation

Depth estimation using monocular cameras can be defined as a non-linear mapping Ψ: $I \rightarrow D$, where I refers to a single RGB image of size $w \times h$ and D is the resulting depth map of the same size [24]. Due to projection ambiguity, this problem is considered ill-posed. Many monocular depth estimation algorithms leverage image features such as texture, occlusion, lighting, and other visual cues to address this challenge. Monocular depth estimation methods often rely on deep learning models that learn depth cues from large datasets. Alternatively, depth estimation can also be performed using multi-view techniques like Structure from Motion (SfM), which reconstruct depth using multiple images from different viewpoints. These methods establish and utilize feature correspondences across images to compute depth, either through handcrafted feature-based techniques or deep learning-based models.

2.3.1 Structure from Motion

Structure from Motion (SfM) is based on finding correspondences between features detected in multiple images of a scene. These correspondences can be established through direct methods, which use pixel intensities, or indirect (feature-based) methods, which rely on identifiable features within the image. In a feature-based approach, the features to match can be either handcrafted, such as corners or edges, or learned using deep learning based models.

Once correspondences are identified, SfM triangulates the matched points across the images to reconstruct the three-dimensional structure of the scene. The depth values are derived from this triangulation, with the accuracy depending heavily on the precision of the correspondences. Environmental factors such as lighting, texture, and contrast variations can significantly influence the results. The computational cost of SfM increases with the number of images and features used.

2.3.2 Handcrafted feature-based methods

Handcrafted feature-based methods build on the indirect approach to finding correspondences. This method divides an image into smaller patches known as superpixels, which are groups of pixels that share similar attributes [22]. Features selected by human experts- such as color, location, texture, motion, and geometric context are then extracted from each superpixel to compute depth cues. A Markov Random Field (MRF) is used to integrate the depth estimates of each superpixel with information from its neighboring superpixels, resulting in the construction of a final depth map. The main drawback of this method is its reliance on handcrafted features, which may fail to capture certain scenarios if the features are not sufficiently relevant. Additionally, deploying this method in real-time on UAVs is challenging due to the necessity of pre- and post-processing, which increases computational resource requirements.

2.3.3 Deep learning-based methods

Deep learning-based methods have revolutionized depth estimation by learning to establish correspondences directly from data. These methods utilize neural networks to automatically extract and match features, bypassing the need for manual feature engineering. Depending on the availability of ground truth, they can be categorized into supervised, semi-supervised, and unsupervised approaches [22].

Supervised Learning

In supervised learning, the input image I and its corresponding dense depth map ground truth D^* are known to the network. The network's task is to learn the mapping $I \rightarrow D^*$ by minimizing a loss function $L(D^*, D)$. However, supervised learning methods require large amounts of ground-truth depth data for training. Acquiring such data is challenging because it often involves expensive LIDAR or RGB-D camera sensors, and calibration errors can result in incorrect depth estimates.

To address the shortage of labeled data, the MiDaS framework [30] provides robust Monocular Depth Estimation (MDE) by training models on a collection of mixed-label datasets. These datasets contain varying depth representations, introducing scale and shift ambiguities. To overcome these challenges, MiDaS applies various loss functions and mixing strategies to make the datasets compatible for training. Leveraging mixed training data mitigates dataset bias and enables zero-shot transfer capability, enhancing generalization without fine-tuning, addressing limitations of single-dataset approaches. Over the years, several versions of MiDaS have been released, featuring new models with more powerful backbones, such as MiDaS v3 [31].

MiDaS is designed for relative depth estimation and does not support metric depth estimation. Some approaches, such as ZeroDepth [32], Metric3D [33], and ZoeDepth [34], have attempted to estimate metric depths; however, they show poorer generalization ability compared to MiDaS v3, as noted in [35]. Nevertheless, ZoeDepth [34], which is also based on the MiDaS framework, has demonstrated that, with refinement using metric depth data, the model can effectively perform in broader metric depth estimation tasks.

Semi-supervised Learning

By leveraging semi-supervised learning methods, the reliance on a fully labeled dataset is reduced. The dataset does not need to be fully labeled, as semi-supervised learning algorithms can effectively use both labeled and unlabeled data to improve model performance. These methods begin by training the model on the labeled data to learn the mapping from input to output $I \rightarrow D^*$. The model then generates predictions for the unlabeled data, which are used as pseudo labels. The pseudo-labeled data, combined with the original labeled data, forms a new dataset, and the network is retrained in a supervised manner using this expanded dataset.

Unsupervised Learning

Although semi-supervised learning reduces the reliance on labeled data, some labeled data is still required. Unsupervised learning, on the other hand, completely eliminates the need for labeled data. For depth estimation, unsupervised methods process video sequences with small changes in camera positions between consecutive frames, allowing successive frames to be used for depth estimation [22].

Unsupervised learning methods treat depth estimation as an image reconstruction task, where depth maps are part of the image reconstruction process. Instead of directly reconstructing the target view from the input view, these methods are forced to learn intermediate predictions of geometry in the image, including depth estimation [6].

The depth estimation accuracy of unsupervised methods is generally lower compared to supervised methods. To address this gap, many different unsupervised approaches have been proposed over the years. One of the most prominent research topics in depth estimation within the domain of unsupervised learning is self-supervised learning. Monodepth, proposed by Godard et al. [29], was one of the first works to incorporate self-supervised learning into monocular depth estimation.

Building on Monodepth, Zhou et al. [6] proposed a method that removes the reliance on stereo camera pairs, instead utilizing monocular video sequences. Their approach infers depth from unlabeled video sequences captured by a moving monocular camera and simultaneously estimates the camera pose. The framework uses view synthesis as a supervisory signal: for a given input view of the scene, a new image is synthesized from a different camera pose [6].

The framework consists of two networks: a depth network and a pose network. The pose network takes three frames as input: a target frame I_t and source frames I_s - { I_{t-1}, I_{t+1} } and outputs the relative camera poses $\hat{T}_{t\rightarrow t-1}$ and $\hat{T}_{t\rightarrow t+1}$. The depth network processes the target frame I_t and generates a depth map $\hat{D}_t$. The synthesized view is created by warping a source view I_s to the target frame's coordinate system, based on the depth map and the estimated pose (i.e., reconstructing the target view). The model is then trained by minimizing the error between the target view and the synthesized view. Figure 2.3 shows an overview of the model pipeline.

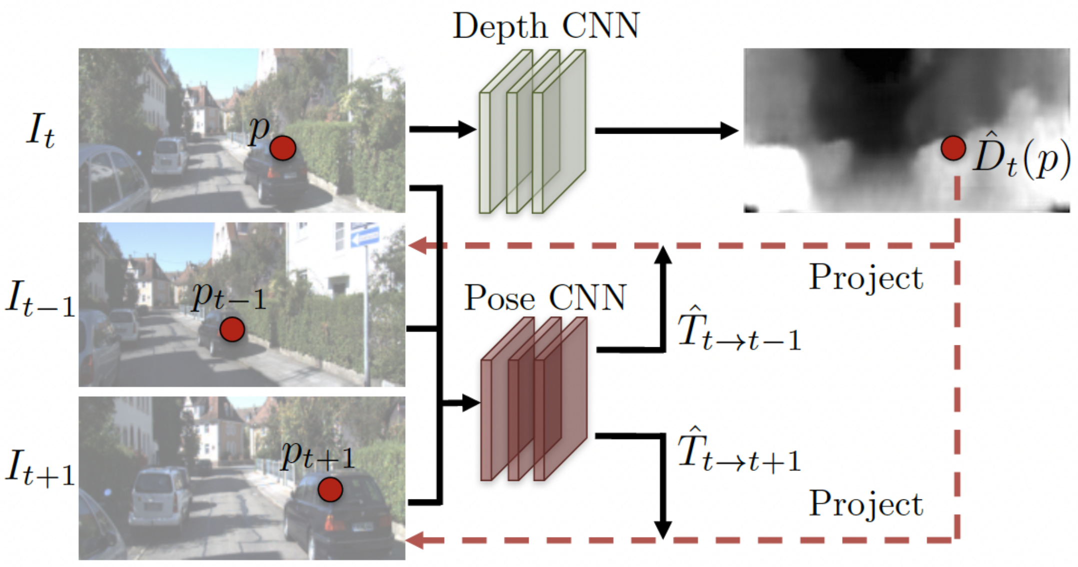


Figure 2.3: Zhou et al. Model pipeline [6].

Monodepth2 [18] builds upon the framework of Zhou et al. [6] by refining the loss functions and improving the handling of occlusions and static regions, resulting in more accurate and visually consistent depth estimates. Over the years, significant advancements in Monocular Depth Estimation (MDE) have been made by works such as MonoViT [36], PlaneDepth [37], and SQLdepth [38].

2.3.4 Physics priors in monocular depth estimation

Drones can operate in various settings, often dynamic environments that involve both their own motion (ego-motion) and the movement of other objects around them. In such conditions, relying on static assumptions—that the environment is stationary—can severely degrade performance. The accuracy of the depth estimates can be improved by injecting physics priors in the model such as temporal priors, geometric priors, optical flow, optics and illumination physics.

Temporal physics

In general, single-frame self-supervised depth estimation models use adjacent source image frames during training that are mapped onto the current image plane with depth map prediction [6]. However, during test time, only a single frame is used for inference, neglecting additional cues such as temporal features. Several methods leverage multi-frame data during both training and test time to improve depth estimates, such as SLAM [39, 40, 41]. Another approach involves test-time refinement, where the training method is replicated during test-time inference using multi-frame data, as seen in typical single-frame self-supervised methods [42, 43, 19]. However, these methods are more computationally expensive, making them less attractive for drones. Another method is incorporating recurrent layers in the monocular network, but this is also computationally expensive and unable to reason about geometry during inference [19, 44].

Watson et al. [19], inspired by multi-view stereo (MVS), proposes a framework that incorporates self-supervised monocular depth estimation with cost-volumes from MVS to handle multi-frames during inference. Cost volumes learn appearance features and measure the geometric correspondence of depth values of the pixels across the target frame and the source frames. This allows reasoning about geometry through temporal sequences of images. If the prediction is precise, the re-projected images should match the actual current frame image. However, the cost function of Watson et al. [19] assumes static environments, so this method does not work well with dynamic environments as the mapping to the target frame will fail.

Feng et al. [44] proposes a method that disentangles object motion from depth estimation by a Dynamic Object Motion Disentanglement (DOMD) module that uses depth priors to address discrepancies in object motion and mapping. DOMD leverages masks to identify and separate dynamic objects for the rest of the scene. Additionally, cost volume is redesigned to handle occlusion and an occlusion-aware re-projection loss is introduced to mitigate problems related to motion occlusion in training.

A drawback of cost volumes is that they are computationally and memory intensive. Works that leverage other methods (e.g. [45]) do not jointly infer depth for consecutive frames and cannot learn the underlying dynamic motions or trajectories of objects along with their spatial information. Yasarla et al. [46] introduce the FutureDepth model that leverages future prediction and adaptive masked reconstruction to improve depth estimation. FutureDepth employs a dual-network approach involving a Future Prediction Network (F-Net) and a Reconstruction Network (R-Net), which function collaboratively to optimize the use of multi-frame spatial and temporal features while maintaining computational efficiency. F-Net predicts future frame features to enhance the understanding of motion and scene dynamics iteratively. In parallel, the R-Net employs a masked auto-encoding technique on multi-frame features to refine the model's perception of structural elements within scenes over time.

Geometric physics

Recovering fine-grained details from depth estimation maps is a challenging problem. Many methods that estimate depth directly from immediate visual features fail to recover these fine-grained scene details. Wang et al. [38] propose a self-supervised depth estimation method for single-image inference that estimates depth from self-cost volumes, which store relative depth representations instead of inferring directly from immediate visual features. The philosophy behind this approach is that pixel depth is correlated with the depths of adjacent pixels and related objects within an image. The fine-grained scene details are captured by the self-cost volume in a so-called Self Query Layer (SQL).

MonoViT [36] is a self-supervised monocular depth estimation framework that integrates Vision Transformers (ViTs) with convolutional networks. It addresses the limitations of conventional convolutional neural networks (CNNs) by introducing global reasoning through ViTs. These Transformers enhance depth estimation by modeling long-range geometric relationships between objects in the scene, acting as a form of geometric prior. The global receptive field of ViTs allows MonoViT to capture interactions between objects that are far apart, improving the model's ability to distinguish foreground and background objects, especially in complex environments. This geometric reasoning enables MonoViT to recover more fine-grained depth details and better handle scenes with occlusion and varying spatial scales.

Furthermore, the combination of local and global feature extraction allows MonoViT to effectively reason about both local geometry (object boundaries and textures) and global scene structure (the relationships between objects at different depths). As a result, MonoViT achieves state-of-the-art depth estimation performance, particularly in scenes where depth relationships are difficult to infer from local features alone, thereby providing a more comprehensive understanding of scene geometry.

Most depth estimation methods are developed for general depth estimation tasks in applications such as Augmented Reality, virtual reality, or autonomous driving. However, only a few methods are specifically tailored for depth estimations from UAVs. Depth estimation for UAVs presents its own unique challenges, such as the influence of non-uniform depth distributions in low-altitude settings. In such settings, depth may be concentrated either on the background or foreground of a frame, such as roofs and walls [47]. Additionally, scale variations significantly affect the accuracy of depth estimates. Another challenge arises from the fact that photometric-consistency based depth estimation models are not well-suited for UAVs due to scale variations and occlusion of large areas.

To address these challenges, [47] proposes a self-supervised framework tailored for UAVs. This framework includes a global and local mixed multi-scale feature enhancement network capable of handling the dynamic environment of UAVs with large scale-variations. Furthermore, a Global Scene Attention module is designed to establish semantic connections across distant parts of the input feature map and integrate contextual details into the feature map's channel representation.

Pose network

Physics can be infused into the pose network by injecting IMU sensor data along with the image data to estimate ego-motion. The IMU sensor reads acceleration and rotational motion data from the accelerometer and gyroscope, respectively. Unlike a monocular camera, which struggles to extract quality data from low-texture images or due to motion blur at high speeds, the IMU is scene-independent and does not suffer from these issues. However, IMU data is not accurate in low-angular speed and low-acceleration scenarios. Moreover, the IMU suffers from sensor drift which results in accumulation of positioning errors [48].

Various approaches have been proposed for estimating ego-motion and depth maps from images and IMU data [49], with some employing optical flow ([50]. Aslan et al. [48] propose a method in which camera frames and IMU data are fused to estimate the pose. This is accomplished by processing the IMU data from consecutive frames through a Bidirectional Long Short Term Memory (BiLSTM) network layer, which estimates the pose. Simultaneously, the visual data is processed, also resulting in a pose estimation. These two estimates are then fused using a BiLSTM network, producing the final pose estimates. The downside of this method is that it uses a supervised learning scheme, which limits its applicability.

Liu et al. [51] propose an attention-guided deep framework for visual-inertial odometry (ATVIO), diverging from traditional recurrent neural network (RNN)-based methods for IMU data processing. Instead, [51] introduces an attention framework designed to effectively extract features from IMU data, addressing the inefficiencies often associated with RNN methods. To fuse the visual and inertial data, Liu et al. [51] develop an attention-guided visual-inertial fusion framework. This approach aims to establish a correlation between visual and inertial data, effectively bridging the gap in the distributions between the two, a gap not adequately explored in previous works. However, scale information from the IMU data is lost due to scale ambiguity in the framework [52].

Zhang et al. [53] proposes DynaDepth which integrates IMU sensor data with image data to estimate depth. It solves the scale ambiguity problem that occurs in many method as the warping process is equivalent up to an arbitrary scaling factor with respect to depth and translation. Obtaining median ratios between ground-truth depth and predictions, commonly used for method evaluation by rescaling each prediction map, presents practical challenges. Nonetheless, the absolute scale metric can be recovered through IMU motion dynamics. Similarly, Zhang et al. [52] introduce a self-supervised method that enhances scale recovery in VIO. This framework includes a self-attention-based IMU network (IMUSAtt) designed to denoise IMU data, thereby facilitating more accurate pose estimation from these inputs. Additionally, the Decoupled PoseNet

(D-PoseNet) is employed to process rotation and translation estimates separately, which significantly enhances the accuracy of each.

2.4 Conclusion

The literature review in this chapter provides an overview of existing research in the field of monocular depth estimation. The key insights drawn from this analysis are summarized below:

- Compared to the extensive work on monocular depth estimation in general image-based contexts or automotive applications, research specifically targeting UAV applications remains relatively limited.
- While some methods for automotive applications incorporate vehicle dynamics to improve monocular depth estimates, even fewer approaches integrate UAV-specific dynamics into their models. As a result, introducing scale information into depth estimates for drone-based applications remains largely underexplored.

Building on the objectives outlined in Section 1.5 of Chapter 1—investigating how incorporating drone dynamics can enhance monocular depth estimation and predicting future depth maps from initial estimates—this literature review further refines and confirms the relevance of the thesis goals. Specifically:

1. Quantify the impact of integrating unique drone dynamics to address scale ambiguity and the effects on monocular depth estimation.
2. Develop a methodology for leveraging initial depth estimates with temporal dynamics to predict future depth maps enhancing accuracy and operational efficiency in depth estimation tasks.

Methodology 3

This chapter details the research methodology used in developing a monocular depth estimation model for UAVs. The approach combines visual data from a monocular camera with inertial data from an IMU to address common challenges such as scale ambiguity. By integrating IMU data and leveraging an EKF framework, the model enhances the accuracy of depth estimation by introducing scale-awareness. This chapter covers the model architecture, IMU motion dynamics, and the loss functions employed to train the network.

3.1 Depth Estimation Model

The proposed depth estimation model, DynaDepth, is inspired by the work of Zhang et al. [53]. The architecture of DynaDepth is illustrated in Figure 3.1. During training, the model takes as input a snippet consisting of two source images, a target image, and IMU data between the consecutive frames, as shown in Figure 3.2. The two source images are taken at adjacent timesteps, with one frame captured one timestep before the target frame, and the other source frame one timestep after the target frame. The goal is to estimate the depth of the target image. By leveraging these source images, the model is trained in a self-supervised manner, predicting depth through reconstructing the target image from the perspective of the source images.

DynaDepth is composed of several subnetworks: a depth network $\mathcal{M}_d$, which is a fully convolutional U-Net used to estimate depth; a pose network $\mathcal{M}_p$, which estimates the ego-motion using an encoder-decoder structure with a ResNet backbone, outputting both rotation and translation; and a gravity and velocity networks, $\mathcal{M}_g$ and $\mathcal{M}_v$ respectively, that estimates the camera-centric gravity and velocity.

DynaDepth addresses the scale ambiguity problem that many previous vision-only models suffer from by incorporating IMU motion dynamics, which enable the recovery of the absolute scale. To fully leverage the complementary information provided by visual and IMU sensors, DynaDepth employs a differentiable, camera-centric extended Kalman filter (EKF) framework. DynaDepth recalibrates the preintegrated IMU terms when new ego-motion predictions are observed from visual data, taking into account the propagated IMU error states and the covariances of the visual predictions. In this way, the inherent noise in IMU data is corrected by more reliable visual data. Furthermore, the EKF-framwork provides an uncertainty measure for the predicted ego-motion. During inference, only the depth network $\mathcal{M}_d$ is used, while the other networks shown in Figure 3.1 are used during training to support $\mathcal{M}_d$ by refining depth estimates and learning the scale specific to the deployment environment.

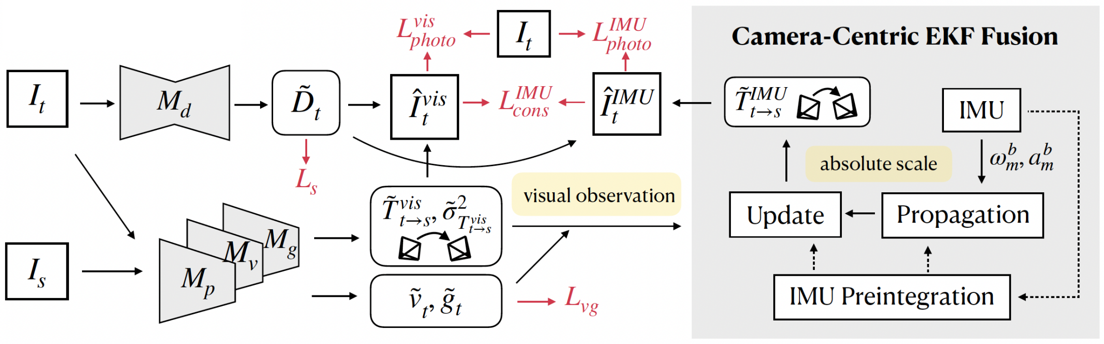
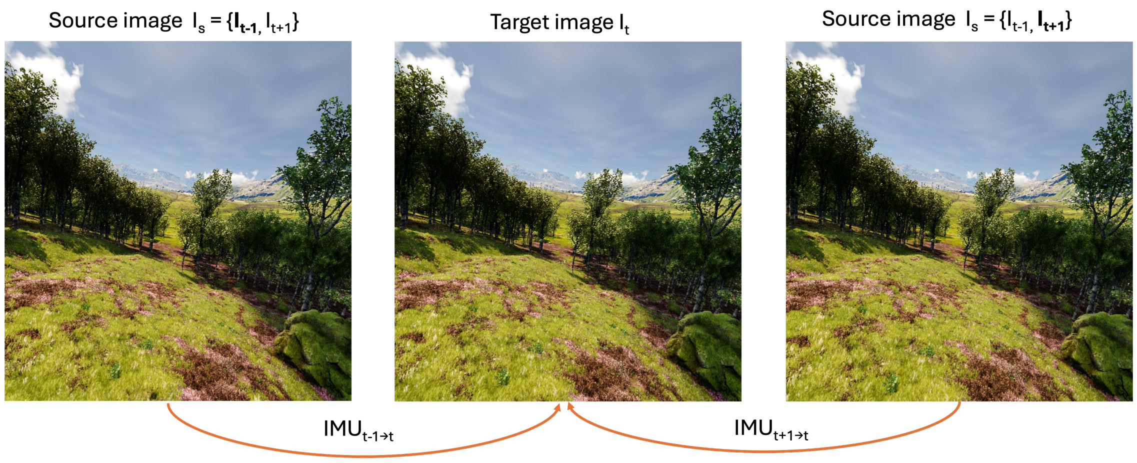


Figure 3.1: The DynaDepth framework includes four networks: a depth-network $\mathcal{M}_d$, pose-network $\mathcal{M}_p$, velocity-network $\mathcal{M}_v$, and gravity-network $\mathcal{M}_g$. The depth network $\mathcal{M}_d$ takes the target image I_t as input and predicts a depth map $\tilde{D}_t$, with smoothness loss L_s reducing the sharpness in the depth values. The pose, velocity, and gravity networks take source frames I_s and target frame I_t as input and provide ego-motion, ego-motion uncertainty, velocity, and gravity, optimized through loss L_{vg}. Using the estimated pose, source frames I_s are warped to the target frame using $\tilde{D}_t$, resulting in the reconstructed frame $\hat{I}_t^{vis}$, optimized with photometric loss L_{photo}^{vis}. Additionally, IMU measurements are also used for pose estimation via preintegration, with source frames warped to the target frame as $\hat{I}_t^{IMU}$ and optimized with L_{photo}^{IMU}. A consistency loss L_{cons}^{IMU} aligns the IMU-based and vision-based pose estimates.



Source image $I_s = \{I_{t-1}, I_{t+1}\}$
Target image I_t
Source image $I_s = \{I_{t-1}, I_{t+1}\}$
$IMU_{t-1\to t}$
$IMU_{t+1\to t}$

Figure 3.2: Input snippet for the model, consisting of two source images, a target image, and IMU data between consecutive frames.

3.2 IMU Motion Dynamics

An IMU sensor measures angular velocity $\boldsymbol{\omega}_m$ and linear acceleration $\boldsymbol{a}_m$ in the sensor/body frame b. The relationship between the true angular velocity and acceleration with their measured counterparts is expressed as follows:

$$\boldsymbol{\omega}_m^b = \boldsymbol{\omega}^b + \mathbf{b}^g + \mathbf{n}^g \tag{3.1}$$

$$\boldsymbol{a}_m^b = \mathbf{R}_{bw}(\boldsymbol{a}^w + \boldsymbol{g}^w) + \mathbf{b}^a + \mathbf{n}^a \tag{3.2}$$

where $\boldsymbol{R}_{bw}$ denotes the rotation matrix from the world frame to the body frame and $\mathbf{g}^w$ represents the gravity factor in the world frame. Furthermore, $\{\mathbf{b}^g, \mathbf{b}^a\}$ and $\{\mathbf{n}^g, \mathbf{n}^a\}$ refer to the Gaussian biases and random walks of the gyroscope and accelerometer, respectively.

Let $\{\boldsymbol{p}_{wb_t}, \boldsymbol{q}_{wb_t}\}$ represent the translation and rotation (denoted in quaternion form) of the body frame relative to the world frame at time t, respectively, and let $\boldsymbol{v}_t^w$ denote the velocity in the world frame. The IMU motion dynamics from timestep i to j are as followed:

$$\mathbf{p}_{wb_j} = \mathbf{p}_{wb_i} + \mathbf{v}_i^w \Delta\mathbf{t} + \iint_{t\in[i,j]} \left(\mathbf{R}_{wb_t}\mathbf{a}^{b_t} - \mathbf{g}^w\right) dt^2, \tag{3.3}$$

$$\mathbf{v}_j^w = \mathbf{v}_i^w + \int_{t\in[i,j]} \left(\mathbf{R}_{wb_t}\mathbf{a}^{b_t} - \mathbf{g}^w\right) dt, \tag{3.4}$$

$$\mathbf{q}_{wb_j} = \int_{t\in[i,j]} \mathbf{q}_{wb_t} \otimes \left[0, \frac{1}{2}\mathbf{w}^{b_t}\right]^T dt, \tag{3.5}$$

where $\Delta\mathbf{t}$ denotes the time difference between step i and j. The above equations use the fact that the first derivatives of $\{\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{q}\}$ are:

$$\dot{\mathbf{p}}_{wb_t} = \mathbf{v}_t^w \tag{3.6}$$

$$\dot{\mathbf{v}}_t^w = \mathbf{a}_t^w \tag{3.7}$$

$$\dot{\mathbf{q}}_{wb_t} = \mathbf{q}_{wb_t} \otimes \left[0, \frac{1}{2}\mathbf{w}^{b_t}\right] \tag{3.8}$$

3.3 IMU preintegration

The IMU measurements are recorded at a much higher sampling rate compared to the camera measurements. Using the raw IMU measurements directly and integrating the IMU data to extract position or orientation is computationally expensive. Every time the optimizer adjusts a state (like position or orientation), naively using the high frequency IMU measurements requires reintegration of all IMU measurements between keyframes. This process is computationally expensive because the integration must be performed repeatedly during each optimization iteration. For example, if there are hundreds of IMU measurements between keyframes, and the optimizer runs multiple iterations, the same integration calculations (involving rotation, velocity, and position updates) must be recomputed thousands of times, since the integration result depends on the changing state estimates.

IMU preintegration [54] solves this by performing the integration once, relative to a local frame, independent of the changing state estimates. The key insight is that IMU measurements can be combined into a single relative motion constraint between two keyframes, expressed in the local frame of the first keyframe. This preintegrated measurement becomes a fixed quantity that does not need to be recomputed during optimization. The IMU preintegration terms for velocity, position, and rotation are the defined as follows:

$$\boldsymbol{\beta}_{b_ib_j} = \int_{t\in[i,j]} \left(\mathbf{R}_{b_ib_t}\mathbf{a}^{b_t}\right) dt \tag{3.9}$$

$$\boldsymbol{\alpha}_{b_i b_j} = \int\int_{t\in[i,j]} \left(\mathbf{R}_{b_i b_t}\mathbf{a}^{b_t}\right) dt^2 \tag{3.10}$$

$$\boldsymbol{q}_{b_i,b_j} = \int_{t\in[i,j]} \boldsymbol{q}_{b_i,b_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}_{b_t} \end{bmatrix} dt \tag{3.11}$$

Equation 3.9 represents the preintegrated velocity change due to acceleration between keyframes i and j, where $\mathbf{R}_{b_i b_t}$ denotes the rotation matrix and $\mathbf{a}^{b_t}$ represents acceleration, both expressed in the IMU body frame. Equation 3.10 integrates the velocity preintegration term in equation 3.9 to estimate position change due to acceleration. Finally, the rotation preintegration term in Equation 3.11 describes the preintegrated rotation change due to IMU angular velocity measurements integrated from keyframes i to j. Here, $\boldsymbol{q}_{b_i,b_t}$ is a quaternion representing the rotation from $\boldsymbol{\omega}_{b_t}$ at time t to the reference frame of b_i. They are combined using quaternion multiplication $\otimes$ with the angular velocity $\boldsymbol{\omega}_{b_t}$, as measured by the IMU at time t.

When state estimates change, only a small correction term needs to be applied to account for changing gravity and bias estimates, rather than redoing the entire integration. This dramatically reduces computational cost since the expensive numerical integration is performed only once, not at every optimization iteration. The derivation of the preintegration terms is detailed in Appendix A. Using the IMU preintegration terms, the camera-centric IMU preintegration ego-motion is defined as:

$$\check{\boldsymbol{R}}_{c_k c_{k+1}} = \boldsymbol{R}_{cb}\mathcal{F}^{-1}(\boldsymbol{q}_{b_k b_{k+1}})\boldsymbol{R}_{bc} \tag{3.12}$$

$$\check{\boldsymbol{p}}_{c_k c_{k+1}} = \boldsymbol{R}_{cb}\boldsymbol{\alpha}_{b_k b_{k+1}} + \check{\boldsymbol{R}}_{c_k c_{k+1}}\boldsymbol{R}_{cb}\boldsymbol{p}_{bc} - \boldsymbol{R}_{cb}\boldsymbol{p}_{bc} + \tilde{\boldsymbol{v}}_{c_k}\Delta t_k - \frac{1}{2}\tilde{\boldsymbol{g}}_{ck}\Delta t_k^2 \tag{3.13}$$

where $\mathcal{F}$ is the transformation from rotation matrix to quaternion, $\{\boldsymbol{R}_{cb}, \boldsymbol{p}_{cb}\}$ and $\{\boldsymbol{R}_{bc}, \boldsymbol{p}_{bc}\}$ denote the extrinsics between the IMU sensor and the camera frames. Lastly, $\tilde{\boldsymbol{v}}_{c_k}$ and $\tilde{\boldsymbol{g}}_{c_k}$ represent the velocity and gravity components in the camera frame at time step k.

3.3.1 Camera-centric EKF framework

The EKF framework fuses measurements from the camera and IMU sensors to reduce noise in the IMU measurements. Zhang et al. [53] introduce the following state transition model:

$$\boldsymbol{x_t} = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{w}_t \tag{3.14}$$

where $\boldsymbol{u}_t$ is the IMU data at time t, $\boldsymbol{w}_t$ is the noise term and $\boldsymbol{x}_t$ represents the state defined as:

$$\boldsymbol{x_t} = \begin{bmatrix}\boldsymbol{\varphi}_{c_k b_t}^T & \boldsymbol{p}_{c_k b_t}^T & \boldsymbol{v}^{c_k T} & \boldsymbol{g}^{c_k T} & \boldsymbol{b}_w^{b_t T} & \boldsymbol{b}_a^{b_t T}\end{bmatrix}^T \tag{3.15}$$

All state variables, except for the IMU biases $\{\boldsymbol{b}_w, \boldsymbol{b}_a\}$, are expressed in the camera frame c_k at time t_k. The IMU biases are represented in the body frame, denoted by the superscript $\{b_t\}$ The term $\boldsymbol{\varphi}_{c_k b_t}$ in the state vector represents the $\mathfrak{so}(3)$ Lie algebra of the rotation matrix $\boldsymbol{R}_{c_k b_t}$, such that $\boldsymbol{R}_{c_k b_t} = \exp([\boldsymbol{\varphi}_{c_k b_t}]^\wedge)$, where $[\cdot]^\wedge$ denotes the mapping from a $\mathfrak{so}(3)$ vector to its corresponding skew-symmetric matrix. The final state variables are position $\boldsymbol{p}_{c_k b_t}$ mapped from body frame b_t at time t to camera frame at time t_k, velocity $\boldsymbol{v}^{c_k T}$ and gravity $\underline{\boldsymbol{g}}^{c_k T}$.

The state variables can be split into two terms: nominal states denoted by $\overline{(\cdot)}$ and error states $\delta\boldsymbol{x}_{b_t} = \left[\delta\boldsymbol{\phi}_{c_k b_t}^T, \delta\boldsymbol{p}_{c_k b_t}^T, \delta\boldsymbol{v}^{c_k T}, \delta\boldsymbol{g}^{c_k T}, \delta\boldsymbol{b}_w^{b_t T}, \delta\boldsymbol{b}_a^{b_t T}\right]^T$. The nominal states are computed through IMU preintegration, while the error states are estimated by the EKF. The use of an EKF is specifically required because the error states are nonlinear; it solves this by linearizing the state transition model at each time t using a first-order Taylor approximation.

Acceleration and rotation are continuous physical phenomena, and with the IMU's high sampling rate, the state is represented in continuous time. However, the data is processed in discrete time steps, and since the EKF operates on discrete data, a continuous-discrete extended Kalman filter (CD-EKF) is used to bridge the gap between continuous-time dynamics and discrete-time updates.

The continuous-time propagation model for the error states is defined as:

$$\delta \dot{\boldsymbol{x}}_{b_t} = \boldsymbol{F} \delta \boldsymbol{x}_{b_t} + \boldsymbol{G} \boldsymbol{n} \tag{3.16}$$

where $\boldsymbol{n}$ represents a noise vector, $\boldsymbol{F}$ is the system matrix and $\boldsymbol{G}$ the process noise matrix, which are defined as:

$$\boldsymbol{F} = \begin{bmatrix} -\left[\bar{\boldsymbol{w}}^{b_t}\right]^\wedge & 0 & 0 & 0 & -\boldsymbol{I}_3 & 0 \\ 0 & 0 & \boldsymbol{I}_3 & 0 & 0 & 0 \\ -\bar{\boldsymbol{R}}_{c_k b_t}\left[\bar{\boldsymbol{R}}_{c_k b_t}^T \bar{\boldsymbol{g}}^{c_k} + \bar{\boldsymbol{a}}^{b_t}\right]^\wedge & 0 & 0 & -\boldsymbol{I}_3 & 0 & -\bar{\boldsymbol{R}}_{c_k b_t} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.17}$$

$$\boldsymbol{G} = \begin{bmatrix} -\boldsymbol{I}_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\bar{\boldsymbol{R}}_{c_k b_t} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{I}_3 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{I}_3 \end{bmatrix} \tag{3.18}$$

where $\bar{\boldsymbol{w}}^{b_t} = \boldsymbol{w}_m^{b_t} - \bar{\boldsymbol{b}}_w^{b_t}$ and $\bar{\boldsymbol{a}}^{b_t} = \boldsymbol{a}_m^{b_t} - \bar{\boldsymbol{R}}_{c_k b_t}^T \bar{\boldsymbol{g}}_{c_k} - \bar{\boldsymbol{b}}_a^{b_t}$.

Given the continuous error propagation model with the initial condition $\boldsymbol{\Phi}(t_\tau, t_\tau) = \boldsymbol{I}_{18}$, the discrete state transition matrix $\boldsymbol{\Phi}(t_{\tau+1}, t_\tau)$ can be obtained by solving the equation $\dot{\boldsymbol{\Phi}}(t_{\tau+1}, t_\tau) = \boldsymbol{F}_{t_{\tau+1}} \boldsymbol{\Phi}(t_{\tau+1}, t_\tau)$:

$$\boldsymbol{\Phi}_{t_{\tau+1}, t_\tau} = \exp\left(\int_{t_\tau}^{t_{\tau+1}} \boldsymbol{F}(t) dt\right) \approx \boldsymbol{I}_{18} + \boldsymbol{F}\delta t + \frac{1}{2}\boldsymbol{F}^2 \delta t^2, \quad \delta t = t_{\tau+1} - t_\tau. \tag{3.19}$$

With the discrete state-transition matrix, the regular EKF equations can be defined. The prior covariance estimate is written as:

$$\check{\boldsymbol{P}}_{t_{\tau+1}} = \boldsymbol{\Phi}_{t_{\tau+1}, t_\tau} \hat{\boldsymbol{P}}_{t_\tau} \boldsymbol{\Phi}_{t_{\tau+1}, t_\tau}^T + \boldsymbol{Q}_{t_\tau}, \tag{3.20}$$

$$\boldsymbol{Q}_{t_\tau} = \int_{t_\tau}^{t_{\tau+1}} \boldsymbol{\Phi}_{t, t_\tau} \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T \boldsymbol{\Phi}_{t, t_\tau}^T dt \approx \boldsymbol{\Phi}_{t_{\tau+1}, t_\tau} \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T \boldsymbol{\Phi}_{t_{\tau+1}, t_\tau}^T \delta t, \tag{3.21}$$

where the process noise covariance $\boldsymbol{Q}$ is

$$\boldsymbol{Q} = \mathcal{D}([\sigma_w^2 \boldsymbol{I}_3, \sigma_{bw}^2 \boldsymbol{I}_3, \sigma_a^2 \boldsymbol{I}_3, \sigma_{ba}^2 \boldsymbol{I}_3]) \tag{3.22}$$

with $\mathcal{D}$ denoting the diagonalization function. The observation model is in general defined as $\boldsymbol{\xi}_{k+1} = h(\boldsymbol{x}_{k+1}) + \boldsymbol{n}_r$ with $\boldsymbol{n}_r \sim N(0, \boldsymbol{\Gamma}_{k+1})$ where $\boldsymbol{\xi}_{k+1}$ denotes the observation measurement from the camera and $\boldsymbol{\Gamma}_{k+1}$ the corresponding covariance.

The corresponding EKF update equations are as follows:

$$\boldsymbol{K}_{k+1} = \check{\boldsymbol{P}}_{k+1}\boldsymbol{H}_{k+1}^T \left(\boldsymbol{H}_{k+1}\check{\boldsymbol{P}}_{k+1}\boldsymbol{H}_{k+1}^T + \boldsymbol{\Gamma}_{k+1}\right)^{-1}, \tag{3.23}$$

$$\hat{\boldsymbol{P}}_{k+1} = \left(\boldsymbol{I}_{18} - \boldsymbol{K}_{k+1}\boldsymbol{H}_{k+1}\right)\check{\boldsymbol{P}}_{k+1}, \tag{3.24}$$

$$\delta\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{K}_{k+1}\left(\boldsymbol{\xi}_{k+1} - h(\check{\boldsymbol{x}}_{k+1})\right). \tag{3.25}$$

where $\boldsymbol{K}_{k+1}$ is the Kalman gain and $\hat{\boldsymbol{P}}_{k+1}$ denotes the posterior covariance. $\delta\hat{\boldsymbol{x}}_{k+1}$ is the estimated error states and $h(\check{\boldsymbol{x}}_{k+1})$ denotes the IMU states predicted by preintegration. $\boldsymbol{H}_{k+1}$ follows by taking the derivative of $h(\boldsymbol{x}_{k+1})$ with respect to the error states: $\boldsymbol{H}_{k+1} = \frac{\partial h(\boldsymbol{x}_{k+1})}{\partial \delta\boldsymbol{x}_{k+1}}$. They are defined as follows.

$$h(\check{\boldsymbol{x}}_{k+1}) = \begin{bmatrix} \bar{\boldsymbol{\phi}}_{c_k c_{k+1}} \\ \bar{\boldsymbol{R}}_{c_b b_{k+1}}\boldsymbol{p}_{bc} + \bar{\boldsymbol{p}}_{c_k b_{k+1}} \end{bmatrix}, \tag{3.26}$$

$$\boldsymbol{H}_{k+1} = \begin{bmatrix} J_l(-\bar{\boldsymbol{\phi}}_{c_k c_{k+1}})^{-1}\boldsymbol{R}_{cb} & 0 & 0 & 0 & 0 & 0 \\ -\bar{\boldsymbol{R}}_{c_b b_{k+1}}[\boldsymbol{p}_{bc}]^\wedge & \boldsymbol{I}_3 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.27}$$

The observation measurement is obtained through the prediction of the ego-motion by the pose network in the model as $\boldsymbol{\xi}_{k+1} = \left[\tilde{\boldsymbol{\phi}}_{c_k c_{k+1}}^T, \tilde{\boldsymbol{p}}_{c_k c_{k+1}}^T\right]^T$ and the corresponding covariance $\boldsymbol{\Gamma}_{k+1}$ is also derived from the pose network.

The corrected ego-motion estimates can be obtained by adding the error states to the nominal states as follows:

$$\boldsymbol{R}_{c_k b_t} = \bar{\boldsymbol{R}}_{c_k b_t}\exp\left(\left[\delta\boldsymbol{\phi}_{c_k b_t}\right]^\wedge\right), \quad \boldsymbol{p}_{c_k b_t} = \bar{\boldsymbol{p}}_{c_k b_t} + \delta\boldsymbol{p}_{c_k b_t}, \tag{3.28}$$

$$\boldsymbol{v}^{c_k} = \bar{\boldsymbol{v}}^{c_k} + \delta\boldsymbol{v}^{c_k}, \quad \boldsymbol{g}^{c_k} = \bar{\boldsymbol{g}}^{c_k} + \delta\boldsymbol{g}^{c_k}, \tag{3.29}$$

$$\boldsymbol{b}_w^{b_t} = \bar{\boldsymbol{b}}_w^{b_t} + \delta\boldsymbol{b}_w^{b_t}, \quad \boldsymbol{b}_a^{b_t} = \bar{\boldsymbol{b}}_a^{b_t} + \delta\boldsymbol{b}_a^{b_t}. \tag{3.30}$$

The working principle of the camera-centric EKF framework can be summarized in the following pseudocode:

Algorithm 1 Camera-Centric EKF Pseudocode

```
Initialization: begin
        $x_{\text{nominal}} \leftarrow [\phi_{c_k b_t}^\top, p_{c_k b_t}^\top, v^{c_k T}, g^{c_k T}, b_w^{b_t T}, b_a^{b_t T}]^\top$    // Nominal State Vector in Camera Frame
    1   $\delta x \leftarrow \mathbf{0}_{18\times 1}$    // Initial Error State
    2   $P \leftarrow I_{18}$    // Initial Covariance Matrix
    3   $Q \leftarrow \text{Diagonal}([\sigma_w^2 I_3, \sigma_{bw}^2 I_3, \sigma_a^2 I_3, \sigma_{ba}^2 I_3])$    // Process Noise Covariance
    4   $t_{\text{current}} \leftarrow t_0$    // Set Initial Time
end
Main Loop: while data is available do
    if IMU measurement received at time $t_{\tau+1}$ then
        EKF_Propagation($x_{\text{nominal}}$, $P$, $u_{\tau+1}$, $\delta t$)    // EKF Propagation Step
    end
    if Camera measurement $\xi$ received at time $t_{k+1}$ then
        EKF_Update($x_{\text{nominal}}$, $P$, $\xi_{k+1}$, $\Gamma_{k+1}$)    // EKF Update Step
    end
end
EKF_Propagation Function: Function EKF_Propagation($x_{nominal}$, $P$, $u$, $\delta t$):
        // Propagate Nominal State using IMU Data
    5   $F, G \leftarrow \text{Compute\_F\_G}(x_{\text{nominal}}, u_{\tau+1})$    // Compute F and G matrices
    6   $\Phi \leftarrow I_{18} + F\delta t + 0.5F^2\delta t^2$    // Discretize the State Transition Matrix
    7   $Q_{\text{discrete}} \leftarrow \Phi G Q G^\top \Phi^\top \delta t$    // Compute Process Noise Covariance for Discrete Time
    8   $P_{\text{prior}} \leftarrow \Phi P \Phi^\top + Q_{\text{discrete}}$    // Propagate Covariance
    9   $P \leftarrow P_{\text{prior}}$    // Update the global covariance
    10  return $x_{\text{nominal}}, P$
EKF_Update Function: Function EKF_Update($x_{nominal}$, $P_{prior}$, $\xi$, $\Gamma$):
        $h_x \leftarrow h(x_{\text{nominal}})$    // Compute Observation Model based on Nominal State
    11  $y \leftarrow \xi - h_x$    // Compute Observation Residual
    12  $H \leftarrow \text{Compute\_H}(x_{\text{nominal}})$    // Compute Observation Jacobian
    13  $S \leftarrow H P_{\text{prior}} H^\top + \Gamma$    // Compute Kalman Gain
    14  $K \leftarrow P_{\text{prior}} H^\top S^{-1}$
    15  $P_{\text{post}} \leftarrow (I_{18} - KH)P_{\text{prior}}$    // Update Covariance
    16  $\delta\hat{x} \leftarrow Ky$    // Update Error State Correction
    17  $x \leftarrow \text{Apply\_Correction}(x_{\text{nominal}}, \delta\hat{x})$    // Apply Error Correction to Nominal State
    18  $\delta x \leftarrow \mathbf{0}_{18\times 1}$    // Reset Error State
    19  $P \leftarrow P_{\text{post}}$    // Update the global covariance
    20  return $x, P$
```

3.4 Loss-functions

The final output of the depth estimation model is an estimated depth map $\boldsymbol{D}_t$ based on a target image $\boldsymbol{I}_t$, a pose estimate derived from only IMU data $\boldsymbol{T}_{t\to s}^{\text{imu}}$, a pose estimate $\boldsymbol{T}_{t\to s}^{\text{vis}}$, velocity $\boldsymbol{v}_t$, and gravity $\boldsymbol{g}_t$ based on visual data only .

The self-supervised model learns by view synthesis, where the target image $\boldsymbol{I}_t$ is reconstructed from the view of the source images $\boldsymbol{I}_s$, constrained by an intermediate depth variable $\boldsymbol{D}_t$ to learn depth. However, this problem is ill-posed [18] as many incorrect depth pixel values can still reconstruct a correct target view.

View synthesis begins by generating a depth map $\boldsymbol{D}_t$ from the target image. The depth map provides the distance of each pixel in the target frame from the camera. With the depth map $\boldsymbol{D}_t$ and the intrinsic camera parameters $\boldsymbol{K}$, the 2D pixel coordinates (u, v) of target frame $\boldsymbol{I}_t$ are projected into 3D space as:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \boldsymbol{D}_t(u, v)\boldsymbol{K}^{-1}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{3.31}$$

Relative pose $\boldsymbol{T}_{t\to s}$ represents the rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ between the source and the target image. Using this relative pose $\boldsymbol{T}_{t\to s}$, the 3D point clouds of the target image are projected into the coordinate frame of the source image $\boldsymbol{I}_s$ as:

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \boldsymbol{R} \left(\boldsymbol{D}_t(u,v)\boldsymbol{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \right) + \boldsymbol{t} \tag{3.32}$$

which is equivalent to

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \boldsymbol{T}_{t\to s}\boldsymbol{D}_t(u,v)\boldsymbol{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{3.33}$$

The 3D points clouds in the coordinate frame of source image $\boldsymbol{I}_s$ are subsequently reprojected onto a 2D image plane using camera intrinsic matrix $\boldsymbol{K}$ as:

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \boldsymbol{K}\boldsymbol{T}_{t\to s}\boldsymbol{D}_a(u,v)\boldsymbol{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{3.34}$$

which can also be written as:

$$p_s \sim \boldsymbol{K}\boldsymbol{T}_{t\to s}\boldsymbol{D}_t(p_t)\boldsymbol{K}^{-1}p_t \tag{3.35}$$

It is not guaranteed that the reprojected points in the 2D plane will fall exactly on the integer pixel grid of the source frame, so differentiable bilinear interpolation is used to address that. Using this relationship between the source and target image pixels (equation 3.35), the pixels of the source image are warped towards the pixels of the target frame $\boldsymbol{I}_t$. The resulting image is the synthesized target image, or the reconstructed target, frame from the perspective of the source image. The model is then trained to minimize the error between the reconstructed image frame and the actual target image by photometric reprojection loss similar to [6, 18]. The visual photometric reprojection loss is defined as

$$L_{photo}^{vis} = \frac{1}{N}\sum_{i=1}^{N} \min_{\delta\in\{-1,1\}} \mathcal{L}(\boldsymbol{I}(\boldsymbol{y}_i), \boldsymbol{I}_\delta(\psi(\boldsymbol{K}\tilde{\boldsymbol{R}}_\delta\boldsymbol{K}^{-1}\boldsymbol{y}_i + \frac{\boldsymbol{K}\tilde{\boldsymbol{p}}_\delta}{\tilde{z}_i}))) \tag{3.36}$$

where $\mathcal{L}(\boldsymbol{I}, \boldsymbol{I}_\delta)$ is denoted as

$$\mathcal{L}(\boldsymbol{I}_t, \boldsymbol{I}_\delta) = \frac{\alpha}{2}\left(1 - \mathrm{SSIM}(\boldsymbol{I}_t, \boldsymbol{I}_\delta)\right) + (1-\alpha)\left\|\boldsymbol{I}_t - \boldsymbol{I}_\delta\right\|_1 \tag{3.37}$$

where $\boldsymbol{I}_t$ denotes the target frame, $\boldsymbol{I}_\delta$ is the reconstructed source image, $\boldsymbol{K}$ refers to the camera intrinsics, N denotes the number of pixels used - due to masking - $\boldsymbol{y}_i$ and $\tilde{z}_i$ are the pixel coordinates in target frame $\boldsymbol{I}_t$ and the corresponding depth map. $\boldsymbol{I}(\boldsymbol{y}_i)$ is the pixel intensity at coordinate $\boldsymbol{y}_i$ and ψ is the depth normalization function. Finally, $\{\tilde{\boldsymbol{R}}_\delta, \tilde{\boldsymbol{p}_\delta}\}$ is the ego-motion predicted by the visual pose network.

SSIM in equation 3.37 refers to the Structural Similarity Index (SSIM) [55], which considers changes in structural information, luminance, and contrast to measure the similarity between two images. Luminance is defined as the mean intensity value μ. The luminance function, which measures the similarity in brightness between $\mathbf{x}$ and $\mathbf{y}$, is defined as:

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{3.38}$$

where μ_x and μ_y are the means of $\mathbf{x}$ and $\mathbf{y}$, which represent two nonnegative image spatial patches from both images that are spatially aligned. C_1 denotes a constant that ensures robustness in case the denominator becomes small and is defined as:

$$C_1 = (K_1 L)^2 \tag{3.39}$$

where L is the dynamic range of the pixel values, typically 255 for 8-bit grayscale images, and $K_1 \ll 1$. The contrast function measures the similarity of contrast between the compared images, focusing on the differences in pixel intensity values. The contrast function is defined as

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{3.40}$$

where σ_x and σ_y are the standard deviations of pixel intensities in images $\mathbf{x}$ and $\mathbf{y}$, respectively. C_2 is a stabilization factor, similarly defined as in equation 3.41, and is given as:

$$C_2 = (K_2 L)^2 \tag{3.41}$$

Finally, the structure comparison function assesses the correlation of structural information between two images. Structural information is defined by correcting an image by the luminance, i.e., $\mathbf{x} - \mu_x$, and normalized by the standard deviation σ_x.

$$s(\mathbf{x}) = \frac{\mathbf{x} - \mu_x}{\sigma_x} \tag{3.42}$$

The structural similarity between images $\mathbf{x}$ and $\mathbf{y}$ is defined by the inner product of the structural information of each image.

$$s(\mathbf{x}, \mathbf{y}) = \left\langle \frac{\mathbf{x} - \mu_x}{\sigma_x}, \frac{\mathbf{y} - \mu_y}{\sigma_y} \right\rangle \tag{3.43}$$

which is similar to

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} \tag{3.44}$$

where C_3 is again a stabilization term and σ_{xy} is denoted as

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \tag{3.45}$$

Combining the luminance, contrast and structural simililarity terms, the SSIM index can be defined as:

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^{\alpha} [c(\mathbf{x}, \mathbf{y})]^{\beta} [s(\mathbf{x}, \mathbf{y})]^{\gamma} \tag{3.46}$$

where α, β, and γ are weights. Setting these values to 1 and $C_3 = C_2/2$ leads to the final form:

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x \mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.47}$$

The final term in the photometric reprojection loss in equation 3.37, α, adjusts the weight between the two losses and is set to 0.85.

Similarly, an IMU photometric reprojection loss can be defined as:

$$L_{photo}^{IMU} = \frac{1}{N}\sum_{i=1}^{N}\min_{\delta\in\{-1,1\}}\mathcal{L}(\boldsymbol{I}(\boldsymbol{y}_i), \boldsymbol{I}_\delta(\psi(\boldsymbol{K}\hat{\boldsymbol{R}}_\delta\boldsymbol{K}^{-1}\boldsymbol{y}_i + \frac{\boldsymbol{K}\hat{\boldsymbol{p}}_\delta}{\tilde{z}_i}))) \tag{3.48}$$

where $\{\hat{\boldsymbol{R}}_\delta, \hat{\boldsymbol{p}}_\delta\}$ is the IMU-camera sensor fusion based ego-motion estimates.

In addition to the IMU and visual photometric losses, Zhang et al. [53] introduce a cross-sensor photometric consistency loss, aimed at synchronizing the ego-motion from IMU preintegration and the vision-based pose network, which is defined as:

$$L_{\text{photo}}^{\text{cons}} = \frac{1}{N}\sum_{i=1}^{N}\min_{\delta\in\{-1,1\}}\mathcal{L}(\boldsymbol{I}_\delta(\psi(\mathbf{K}\tilde{\boldsymbol{R}}_\delta\mathbf{K}^{-1}\boldsymbol{y}_i + \frac{\mathbf{K}\tilde{\boldsymbol{p}}_\delta}{\tilde{z}_i})), \boldsymbol{I}_\delta(\psi(\mathbf{K}\hat{\boldsymbol{R}}_\delta\mathbf{K}^{-1}\boldsymbol{y}_i + \frac{\mathbf{K}\hat{\boldsymbol{p}}_\delta}{\tilde{z}_i})))) \tag{3.49}$$

Additionally, a smooth loss is defined in equation 3.50 to reduce the sharpness of depth values.

$$L_s = |\partial_x \boldsymbol{D}_t^*|\, e^{-|\partial_x \boldsymbol{I}_t|} + |\partial_y \boldsymbol{D}_t^*|\, e^{-|\partial_y \boldsymbol{I}_t|} \tag{3.50}$$

The gravity network is trained using the gravity loss function, denoted by equation 3.51, to estimate $\boldsymbol{g}^W$, as the gravity vector varies for each snippet. This is because gravity estimates from previous frames are unknown to the unsupervised network. In the gravity loss equation, $\boldsymbol{g}^{enu}$ represents the known gravity vector in the East-North-Up (ENU) frame.

$$L_g = \left\| \|\mathbf{g}^W\| - \|\mathbf{g}^{\text{enu}}\| \right\|_1 \tag{3.51}$$

Finally, the velocity loss is defined as

$$L_v = \left\| \boldsymbol{v}^W - \boldsymbol{v}^{imu} \right\|_1 \tag{3.52}$$

where $\boldsymbol{v}^W$ is the estimated camera-centric velocity from the velocity network and $\boldsymbol{v}^{imu}$ denotes the velocity measured by the IMU sensor.

Combining the photometric reprojection, smoothness, gravity, and velocity losses leads to the final loss equation, denoted as:

$$L_{total} = L_{photo}^{vis} + \lambda_1 L_s + \lambda_2 L_{photo}^{IMU} + \lambda_3 L_{photo}^{cons} + \lambda_4 L_{vg} \tag{3.53}$$

where the λ terms are used to adjust the weights of the losses and are determined empirically.

3.5 Implementation Details

As previously mentioned, the DynaDepth model consists of a fully convolutional U-Net depth network, a pose network and velocity and gravity networks. Following [18], the minimum depth is set to 0.001 meters and the maximum depth is set to 80 meters. The depth network, $\mathcal{M}_d$, is the same network as in MonoDepth2 [18] with a ResNet-50 backbone configuration pretrained on ImageNet [56]. The visual pose estimation model $\mathcal{M}_p$ is adopted from [18], but the output dimension is expanded from 6 to 12 to accommodate uncertainty predictions. The gravity and velocity networks, $\mathcal{M}_g$ and $\mathcal{M}_v$, share the same architecture as $\mathcal{M}_p$, though their output dimensions are configured to 3.

The model was trained under three distinct settings: (1) depth estimation using only a monocular camera, (2) depth estimation using both a monocular camera and IMU measurements, and (3) depth estimation using a monocular camera combined with IMU measurements filtered through an EKF. Minimal preprocessing was applied to the datasets, consisting mainly of adding noise and distortions to simulate real-world conditions. The images were normalized, and augmentations like random brightness, contrast adjustments, and flips were applied to improve generalization.

The weighting factors in the total loss function, equation 3.53, are set to $\lambda_1 = 0.001$, $\lambda_2 = 0.5$, $\lambda_3 = 0.01$, $\lambda_4 = 0.001$. The model was trained for 30 epochs with a learning rate of 1e-4, which was reduced to 1e-5 after completing the first 15 epochs using the Adam optimizer [57]. The entire model is trained end-to-end with the same learning rate for all components to ensure consistency.

The DynaDepth network is implemented using PyTorch [58]. The three model configurations trained on different datasets are conducted on NVIDIA Tesla V100 and NVIDIA Geforce RTX 2080 GPUs. Training time took up till 36 hours for 30 epochs. All code and configuration files for training and evaluation can be found in this repository.

3.6 Conclusion

In this chapter, the DynaDepth model is introduced as a monocular depth estimation method for UAVs, addressing the key challenge of scale ambiguity that many monocular depth estimation methods suffer from. The following summarizes the main points discussed.

- The DynaDepth model was developed to perform monocular depth estimation and addresses the limitations of traditional visual data-based methods, particularly scale ambiguity and robustness, by incorporating IMU data alongside visual data to achieve improved depth estimation accuracy and introduce scale-awareness, crucial for UAV applications.

- The Extended Kalman Filter (EKF) framework is used to fuse IMU and visual data, enhancing depth estimation reliability, further refining scale estimation. In chapter 4, DynaDepth is extended with a future depth prediction algorithm.

- The implementation details of the DynaDepth model and the loss functions used for training which include photometric reprojection loss, smoothness loss, gravity loss, and velocity loss.

Results 4

This chapter provides an overview of the dataset used for training, testing, and validating the depth estimation model, along with a discussion of the model's performance on these datasets. Additionally, a future depth prediction algorithm is introduced and evaluated. The section is organized into three parts: the first part describes the datasets, the second part discusses the model's performance, and the third part examines the future depth map prediction algorithm and presents the evaluation results.

4.1 Datasets

Two datasets are used to evaluate the depth estimates generated by the depth estimation model: the KITTI [7] and Mid-Air [59] datasets. KITTI is a widely used real-world driving dataset that serves as a benchmark for many robotic perception algorithms. It is well known for its extensive collection of sensor data from real-world urban and highway driving scenarios, making it the standard tool for testing perception and localization algorithms. The dataset was collected using camera, LiDAR scanners, GPS and IMU sensors. The LiDAR measurements provide ground truth for depth estimation tasks using other sensors. Figure 4.1 shows an overview of the various scene scenarios included in the KITTI dataset.

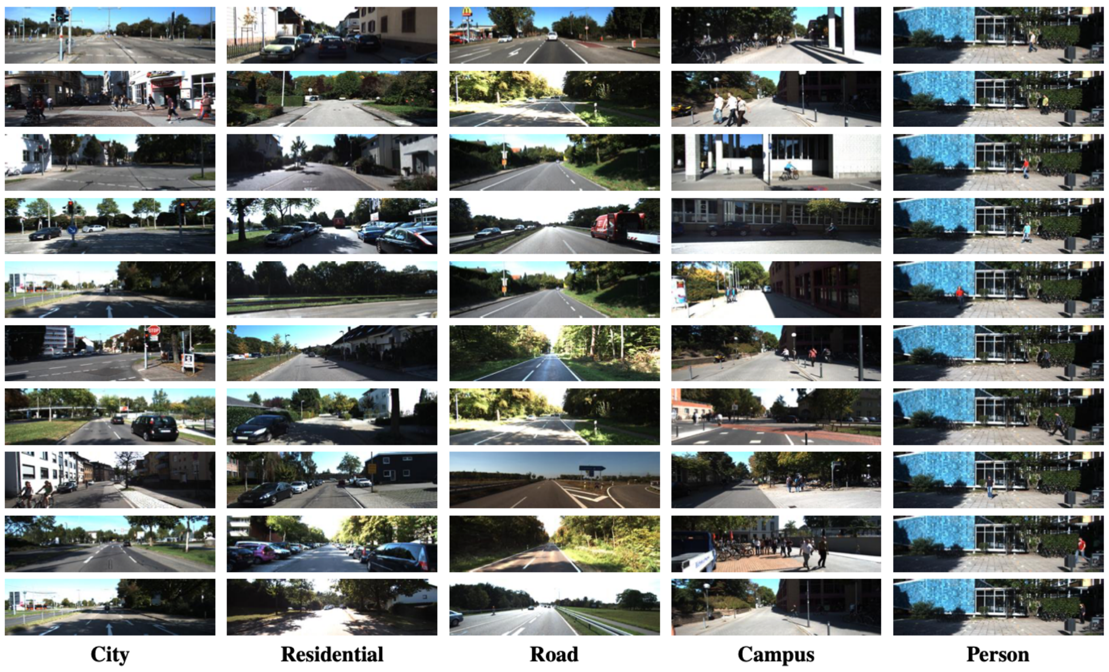


Figure 4.1: Examples from the KITTI dataset captured by the left color camera image [7].

The Mid-Air dataset is a simulated dataset designed for training and evaluating models in drone-related applications, particularly for depth estimation and visual perception tasks in outdoor environments. It simulates low-altitude drone flights in unstructured, real-world settings, addressing gaps left by traditional datasets, which are predominantly focused on autonomous vehicle scenarios. The Mid-Air dataset provides synchronized data from multiple sensors, capturing diverse environmental conditions across a range of flight trajectories.

The dataset simulates the perspective of a quadcopter drone equipped with various sensors, including front-facing cameras, a downward-facing camera, an IMU, and a GPS receiver. The cameras capture a range of outputs such as RGB images, depth maps, normal maps, stereo disparity maps, occlusion masks, and semantic segmentation maps. All images are rendered using Unreal Engine [60]. For this research, only data from one front-facing camera and IMU measurements are utilized. Additionally, the Mid-Air dataset includes sensor data recorded under four different weather conditions: sunny, cloudy, foggy, and sunset. The sunny and sunset conditions are particularly challenging due to the complex lighting and shadows.

4.2 Evaluation Metrics

Depth estimation models are typically evaluated using a set of quantitative metrics that measure the discrepancy between the predicted depth $\hat{\mathbf{d}}$, and the ground-truth depth $\mathbf{d}$. These metrics include the absolute relative error (Abs$_{rel}$), squared relative error (Sq$_{rel}$), regular and logarithmic root mean squared errors (RMSE and RMSE$_{log}$, respectively), and three threshold metrics. This section elaborates on these evaluation metrics and how they relate to the performance of the depth estimation models.

4.2.1 Relative Error Metrics

The absolute relative error (Abs$_{rel}$) computes the average absolute difference between the predicted and ground-truth depths, normalized by the ground truth values as shown in equation 4.1.

$$Abs_{rel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{d}_i - \hat{\mathbf{d}}_i|}{\mathbf{d}_i} \tag{4.1}$$

Abs$_{rel}$ provides a relative measure of the error, indicating the significance of the prediction error in proportion to the actual depth. The metric is scale-invariant due to the normalization, which gives equal importance to errors in both near- and far-depth.

The squared relative error is given by:

$$Sq_{rel} = \frac{1}{N}\sum_{i=1}^{N}\frac{(\mathbf{d}_i - \hat{\mathbf{d}}_i)^2}{\mathbf{d}_i} \tag{4.2}$$

Sq$_{rel}$ calculates the squared difference between the predicted and ground-truth depths, normalized by the ground truth. This metric is particularly sensitive to prediction errors for nearby objects due to both the squaring of the difference and normalization by the true depth value. In depth estimation tasks, this means that regions where the model's predictions are significantly off for close-range measurements - such as nearby obstacles, pedestrians, or surfaces with abrupt depth changes - will contribute more heavily to the overall Sq$_{rel}$ metric. This sensitivity to near-field errors aligns well with real-world applications where accurate depth estimation of nearby objects is crucial for safety and navigation.

4.2.2 Absolute Error Metrics

The root mean squared error (RMSE) measures the absolute deviation of predicted depths from ground truth as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{d}_i - \hat{\mathbf{d}}_i)^2} \tag{4.3}$$

RMSE provides an overall measure of the model's predictive performance across all depth values. A lower RMSE indicates that the predicted depth maps are generally close to the ground truth, which is crucial for tasks like 3D reconstruction and obstacle avoidance, where absolute depth accuracy is important.

The logarithmic root mean squared error (RMSE$_{log}$) computes the RMSE of the logarithm of the depth values and is defined as follows:

$$RMSE_{log} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\Big(\log(\mathbf{d}_i) - \log(\hat{\mathbf{d}}_i)\Big)^2} \tag{4.4}$$

This metric emphasizes the relative error between the predicted and ground-truth depths, which is important when the depth values span several orders of magnitude. In depth estimation, RMSE$_{log}$ is particularly useful for evaluating the model's performance across varying depth ranges, ensuring that both near and far objects are accurately estimated.

4.2.3 Threshold Accuracy Metrics (σ_1, σ_2, σ_3)

The threshold metrics evaluate the proportion of predicted depths that fall within a specified factor of the ground truth, defined as:

$$\sigma_n = \mathbb{E}\left(\mathcal{T}(\mathbf{d}, \hat{\mathbf{d}}) < \tau^n\right), \quad n = 1, 2, 3 \tag{4.5}$$

where τ is 1.25 and threshold $\mathcal{T}(\mathbf{d}, \hat{\mathbf{d}})$ is defined as

$$\mathcal{T}(\mathbf{d}, \hat{\mathbf{d}}) = \max\left(\frac{\mathbf{d}}{\hat{\mathbf{d}}}, \frac{\hat{\mathbf{d}}}{\mathbf{d}}\right) \tag{4.6}$$

These metrics quantify the accuracy of depth predictions by measuring the percentage of pixels where the predicted depth is within a specified threshold of the ground truth. For instance, σ_1 indicates the proportion of predictions within a factor of 1.25 of the actual depth. Higher values for δ metrics suggest that the model reliably predicts depths within acceptable error margins, which is essential for applications like navigation and collision avoidance, where safety relies on accurate depth perception.

4.3 Monocular Depth Estimation KITTI Dataset

In this section, the performance of the DynaDepth framework on the KITTI dataset is evaluated using the Eigen split [61]. The KITTI dataset includes a variety of driving scenarios, providing complex urban and highway environments with dynamic objects like cars, pedestrians, and cyclists. Accurate depth estimation in such settings is crucial, especially for vehicles, as it directly impacts tasks such as collision avoidance, path planning, and autonomous navigation.

Table 4.1 presents the quantitative results of the DynaDepth framework evaluation on the KITTI data set. Static frames are removed from the dataset and frames without corresponding IMU measurements are also disregarded. The scale is calculated by taking the ratio between the median depth from LiDAR-based ground truth and the predicted depth.

Models	Methods			Scale	Error↓				Accuracy↑		
	Visual	IMU	EKF		Abs$_{rel}$	Sq$_{rel}$	RMSE	RMSE$_{log}$	$\sigma_1 < 1.25$	$\sigma_2 < 1.25^2$	$\sigma_3 < 1.25^3$
DynaDepth	✓			30.675±0.81	0.114	0.844	4.725	0.191	0.878	**0.961**	0.981
DynaDepth	✓	✓		**5.515**±0.076	0.113	0.860	4.814	0.193	0.876	0.959	0.981
DynaDepth (full)	✓	✓	✓	5.538±**0.073**	**0.110**	**0.800**	**4.714**	**0.190**	**0.879**	0.960	**0.982**

Table 4.1: Comparison of DynaDepth configurations and their performance metrics on KITTI dataset.

Table 4.1 shows that incorporating IMU measurements significantly improves scale accuracy, reducing it from approximately 30.6 in the monocular camera-only setup to 5.5 when combining the monocular camera with IMU sensors. The inclusion of the EKF further refines scale estimation, as evidenced by the lowest standard deviation in scale, indicating more consistent depth predictions, although the scale is slightly larger compared to the camera-IMU-only setup.

Analyzing the error metrics, the model configuration with visual input, IMU, and EKF achieves the lowest absolute relative error and squared relative error. The lower absolute relative error indicates that the predicted depths are closer to the ground-truth in proportion to the actual depth values, which is critical for accurately estimating the depth of cars at varying distances. The relative squared error which is sensitive to errors in nearby objects, indicates improved performance in estimating depths of close-range vehicles—an essential factor for collision avoidance systems. Examining the RMSE and RMSE$_{log}$ metrics demonstrates improved model performance with the Visual+IMU+EKF configuration. A lower RMSE reflects the overall predictive accuracy in meters, while a lower RMSE$_{log}$ indicates better performance when considering the logarithmic scale of depth values.

The threshold accuracy metrics (σ_1, σ_2, σ_3) assess the proportion of depth predictions within specific error margins relative to the ground-truth. The Visual+IMU+EKF configuration shows the best performance for σ_1 and σ_3, indicating a higher percentage of predictions within acceptable error thresholds. Specifically, a σ_1 of 0.879 means that approximately 87.9% of the predicted depths are within a factor of 1.25 of the ground truth. However, for σ_2, the visual only configuration marginally outperforms the other setups.

Figure 4.3 shows a depth map estimated by DynaDepth on the target frame in figure 4.2. The depth map represents the distances to various objects, including multiple cars on the street. The figure shows that the model can accurately differentiate between the cars at different distances, which is essential for tasks like obstacle detection and maintaining safe following distances.

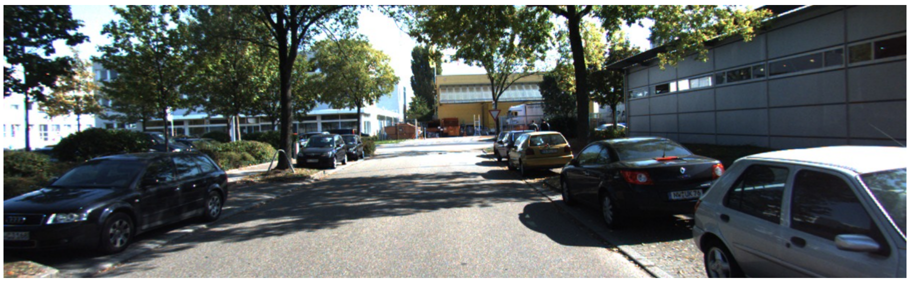


Figure 4.2: Target frame of street with multiple objects in KITTI.

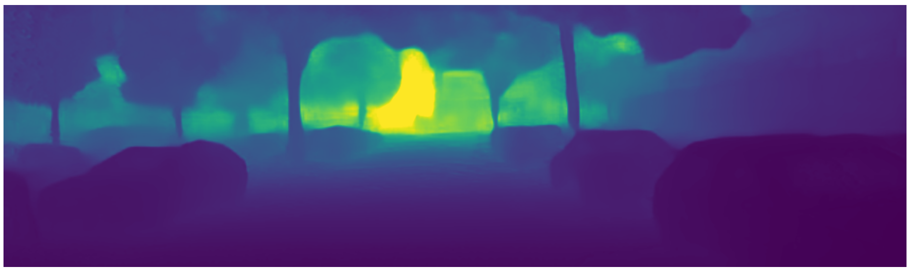


Figure 4.3: Depth map for target frame in KITTI.

4.4 Impact of KITTI Environmental Context on DynaDepth

This section examines how different environmental settings in the KITTI dataset influence the performance of the DynaDepth model. Specifically, it examines how varying environments—City, Residential, and Road as shown in Figure 4.1—affect the algorithm's ability to accurately estimate depth. Each environment presents unique challenges due to differences in object density, dynamic elements, and scene complexity, which can influence the effectiveness of monocular depth estimation. The three environments were selected by categorizing the Eigen split [61] into the aforementioned categories, resulting in 13 trajectories for the City environment, 9 trajectories for the Residential environment, and 6 trajectories for the Road environment.

The City environment is characterized by densely populated urban areas with numerous dynamic objects such as vehicles, pedestrians, and complex infrastructures. These conditions present challenges for monocular depth estimation due to the high density of objects in close proximity to the car and rapid depth changes. Evaluation results in Table 4.2 indicate that the Visual+IMU+EKF configuration achieves the best performance in this setting. Similarly, the Residential environment, which typically features narrower roads, fewer dynamic objects, and closely spaced static obstacles like parked cars, trees, and buildings, poses challenges due to object proximity and potential occlusions. As shown in Table 4.3, the Visual+IMU+EKF setup continues to outperform other configurations in this environment. In contrast, the Road environment generally consists of open highways or rural roads with minimal surrounding objects and fewer dynamic elements, where the primary challenge is the lack of nearby visual cues necessary for determining scale and distance. Nonetheless, as illustrated in Table 4.4, the Visual+IMU+EKF configuration again provides the most accurate and consistent depth estimations. Overall, across all three environments—City, Residential, and Road—the Visual+IMU+EKF setup consistently delivers superior performance, demonstrating its robustness and adaptability in varying conditions.

By comparing the tables for the three different environments, it can be concluded that DynaDepth performs best in crowded settings where many objects are static and closely spaced around the vehicle, as evidenced by the superior overall metrics and better scale estimations. In scenarios where the environment is more densely populated with various dynamic objects, the model's performance declines, as shown in Table 4.2. The performance further deteriorates on open highways and rural roads with minimal surrounding objects and fewer dynamic elements, as illustrated in Table 4.4.

Models	Methods			Scale	Error↓				Accuracy↑		
	Visual	IMU	EKF		Abs$_{rel}$	Sq$_{rel}$	RMSE	RMSE$_{log}$	$\sigma_1 < 1.25$	$\sigma_2 < 1.25^2$	$\sigma_3 < 1.25^3$
DynaDepth	✓			30.638$\pm$**0.068**	0.120	1.001	5.050	0.205	0.873	**0.954**	**0.977**
DynaDepth	✓	✓		**5.502**$\pm$0.070	0.119	0.979	5.140	0.206	0.871	0.952	**0.977**
DynaDepth (full)	✓	✓	✓	5.523$\pm$0.070	**0.116**	**0.933**	**5.036**	**0.204**	**0.874**	**0.954**	**0.977**

Table 4.2: Comparison of DynaDepth configurations and their performance metrics on City scenario.

Models	Methods			Scale	Error↓				Accuracy↑		
	Visual	IMU	EKF		Abs$_{rel}$	Sq$_{rel}$	RMSE	RMSE$_{log}$	$\sigma_1 < 1.25$	$\sigma_2 < 1.25^2$	$\sigma_3 < 1.25^3$
DynaDepth	✓			30.241$\pm$**0.078**	0.119	0.698	4.109	0.191	0.867	0.958	**0.982**
DynaDepth	✓	✓		5.507$\pm$0.083	0.117	0.671	4.072	0.191	0.867	**0.959**	**0.982**
DynaDepth (full)	✓	✓	✓	**5.488**$\pm$0.083	**0.114**	**0.650**	**4.052**	**0.190**	**0.869**	0.957	**0.982**

Table 4.3: Comparison of DynaDepth configurations and their performance metrics on Residential scenario.

Models	Methods			Scale	Error↓				Accuracy↑		
	Visual	IMU	EKF		Abs$_{rel}$	Sq$_{rel}$	RMSE	RMSE$_{log}$	$\sigma_1 < 1.25$	$\sigma_2 < 1.25^2$	$\sigma_3 < 1.25^3$
DynaDepth	✓			32.618$\pm$0.088	0.136	**1.242**	5.708	0.217	0.843	**0.955**	**0.979**
DynaDepth	✓	✓		5.732$\pm$0.077	0.138	1.404	5.910	0.219	0.841	0.953	0.978
DynaDepth (full)	✓	✓	✓	**5.703$\pm$0.073**	**0.134**	1.282	**5.706**	**0.214**	**0.844**	0.953	**0.979**

Table 4.4: Comparison of DynaDepth configurations and their performance metrics on Road scenario.

4.5 Monocular Depth Estimation Mid-Air Dataset

Table 4.5 presents the results of DynaDepth evaluated on the Mid-Air dataset. The results demonstrate that incorporating IMU data significantly improves scale estimation. However, this improvement comes at the cost of increased error metrics (e.g., RMSE increaes to 14.181 and Abs$_{rel}$ rises to 0.275) and decreased accuracy (σ_1 drops to 69.9%), suggesting that the raw fusion of visual data and IMU data may introduce inconsistencies. The full DynaDepth configuration, which integrates visual data, IMU, and EKF, achieves the best performance across all metrics. It further refines the scale to 4.964$\pm$0.400 and significantly improves error metrics (e.g., Abs$_{rel}$ reduces to 0.149, RMSE from 14.181 to 8.531 m) and accuracy (σ_1 increases to 81.2%). This indicates that the EKF effectively integrates the visual and IMU inputs by accurately estimating error states and reducing the inherent noise in IMU measurements, which is more prominent in drone dynamics due to their higher degrees of freedom compared to car dynamics in the KITTI dataset. Figure 4.4 shows a target frame with its corresponding ground-truth depth map and the prediction made by DynaDepth.

Models	Methods			Scale	Error↓				Accuracy↑		
	Visual	IMU	EKF		Abs$_{rel}$	Sq$_{rel}$	RMSE	RMSE$_{log}$	$\sigma_1 < 1.25$	$\sigma_2 < 1.25^2$	$\sigma_3 < 1.25^3$
DynaDepth	✓			25.918$\pm$0.700	0.243	5.896	11.391	0.341	0.784	0.869	0.902
DynaDepth	✓	✓		5.176$\pm$0.684	0.275	6.425	14.181	0.372	0.699	0.812	0.876
DynaDepth (full)	✓	✓	✓	**4.964$\pm$0.400**	**0.149**	**2.252**	**8.531**	**0.233**	**0.812**	**0.920**	**0.960**

Table 4.5: Comparison of DynaDepth configurations and their performance metrics on Mid-Air dataset.

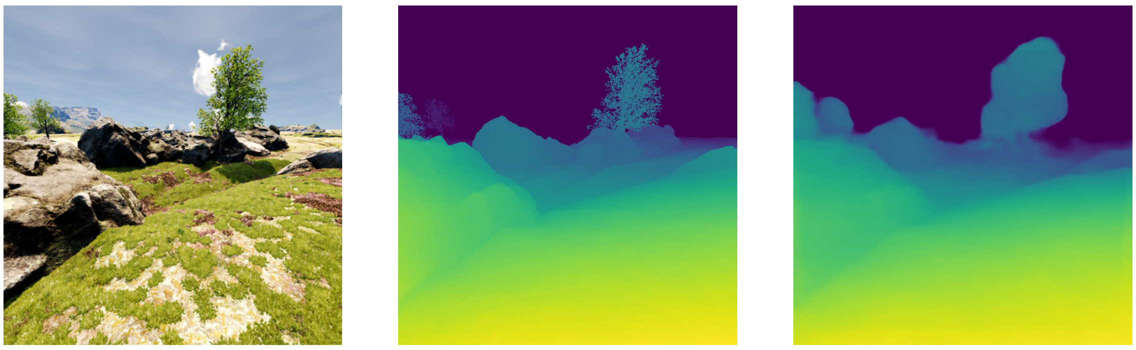


Figure 4.4: Comparison of the target frame, ground-truth depth map, and estimated depth map using DynaDepth: (a) Target frame, (b) Ground-truth depth map of the target frame, (c) Estimated depth map produced by DynaDepth.

4.6 IMU Depth Map Generation

The camera frequencies for the KITTI and Mid-Air datasets are 10 Hz and 25 Hz, respectively. This means that KITTI captures 10 frames per second, while the Mid-Air dataset captures 25 frames per second. Since the depth estimation model generates a depth map for each image, it produces 10 depth maps per second for KITTI and 25 per second for Mid-Air. For energy-constrained applications, however, more energy-efficient solutions are necessary. One approach is to predict future depth maps based on IMU movement and an initial depth map. By integrating IMU measurements as described in section 3.2, a new pose can be estimated. Using an initial depth map generated by the model and the estimated pose, future depth maps can then be predicted.

Assuming a trajectory with constant velocity, the future depth map generation process consists of 4 main steps as illustrated in figure 4.5: First, the depth map is transformed into 3D depth point clouds. Second, the inverse pose is applied to these 3D depth point clouds to extract new depth values of the environment relative to the car from the point clouds. Third, as the robot moves in the z-direction, new pixel coordinates are computed by transforming the point clouds with the pose. Finally, the new depth map is warped with the new depth values to the new pixel grid.

The future depth prediction algorithm is evaluated on the KITTI dataset using ten trajectories, with the car driving at a constant velocity with only static objects present in the scene. Figure 4.6 presents the evaluation results, expressed as the RMSE relative to the forecast horizon in frames. The trajectories were recorded at an average speed between 8 and 11 meters per second (m/s). The results show that as the forecast horizon increases, the corresponding RMSE generally increases linearly. The initial error at frame 0 is derived from the evaluated depth map generated directly by the DynaDepth model. In most cases, the error increases marginally for the first five frames before increasing with a higher rate. Considering that the car's speed ranges from 8 to 11 m/s, in a span of five frames (or half a second), it travels approximately 4 to 6 meters. This distance is significant, allowing for objects or elements in the scene that were not visible in the initial frame to become apparent. Figure 4.7 illustrates the depth predictions over a 20-frame prediction horizon. The plot shows the depth maps at 5-frame intervals along with their corresponding target frames. It can be observed that the DynaDepth model accurately predicts the depth map initially, with clearly distinguishable objects in the frames. By prediction frame 5, the objects remain distinguishable; however, as the prediction horizon extends further, the frame prediction algorithm struggles to

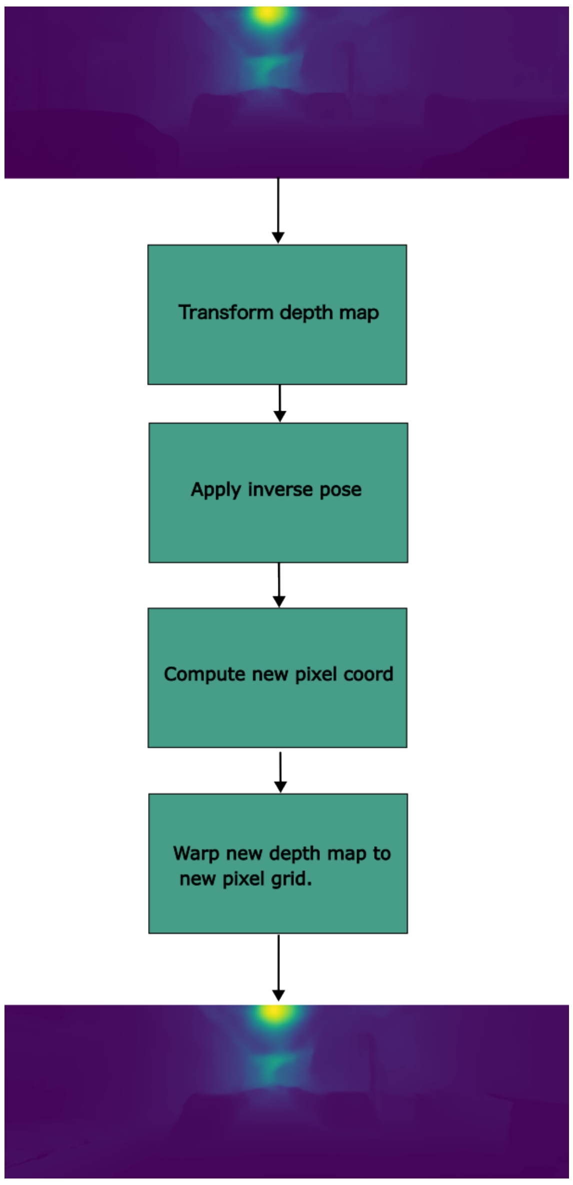


Figure 4.5: Future depth map prediction algorithm.

preserve the shapes of the objects.

Figure 4.8 shows the results of the future depth prediction algorithm on the Mid-Air dataset for a 2 second trajectory (camera-frequency: 25 frames per second). A total of 22 trajectories are analyzed after removing outliers using the depth map prediction algorithm. Unlike the evaluation on KITTI, no specific trajectories with constraints were selected other than the agent is not allowed to rotate completely around the yaw axis for the Mid-Air dataset. This is because the Mid-Air dataset represents an open environment with minimal objects other than stones and trees and does not face the same challenges as KITTI, such as limited visibility of distant surroundings (e.g., at 20 meters, a house lawn, playground, or corner may appear) and occasional interactions

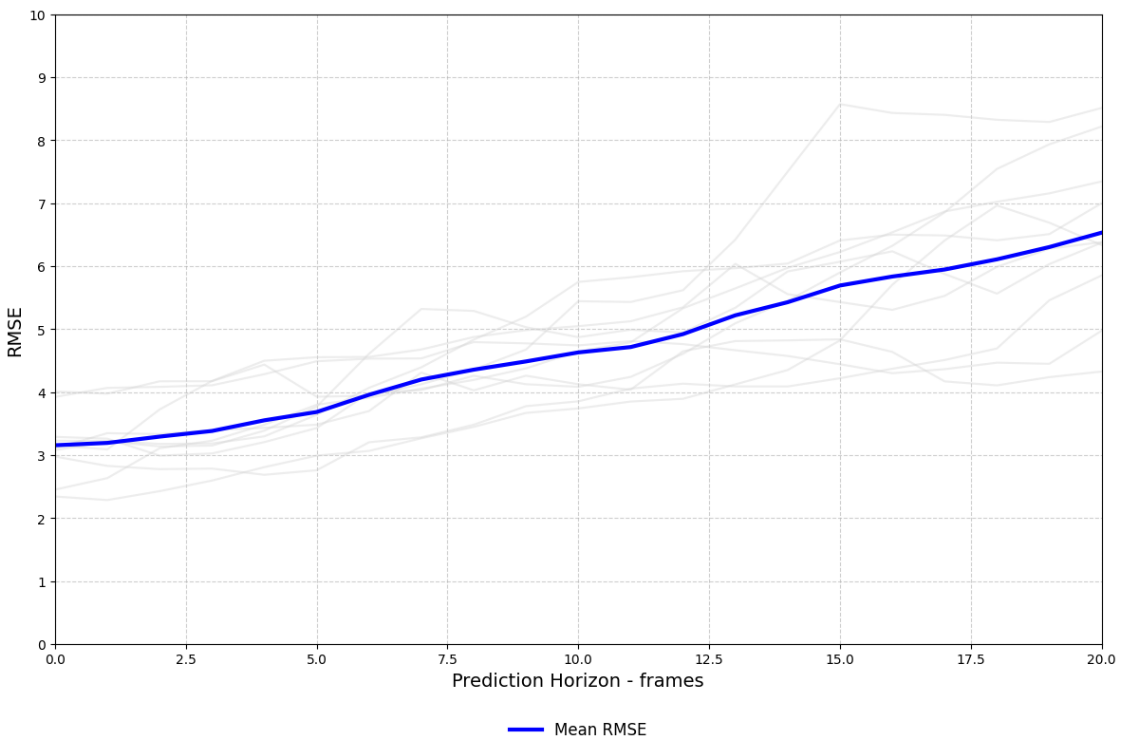


Figure 4.6: Forecast Horizon IMU Depth Map Generation on KITTI.

with other road users like cyclists and pedestrians. The initial depth map is predicted with the Visual+IMU+EKF configuration of DynaDepth. The dynamics of the drone (e.g., yaw, pitch, and roll) make these trajectories challenging to predict. The blue line in Figure 4.8 indicates the average RMSE computed over the prediction horizon that increases at a linear rate. In other words, as the prediction horizon increases, the error tends to increase proportionally. Given that the Mid-Air dataset has a camera frequency of 25 frames per second, predicting the depth for a prediction horizon of 5 frames—thereby would reduce the usage of the model from 25 times per second to 5 times per second. The IMU deviates from the true trajectory as the prediction horizon increases. Figure 4.9 shows a sequence along with a predicted depth map in the second row for a prediction horizon of 20 frames. In this setting, the drone is relatively stable and moves in the positive z-direction. As the prediction horizon increases the estimated depth gets more distored, this is shown in figure 4.10, which demonstrates that the IMU-based predictions accumulate errors over time, leading to significant deviations in depth estimation.

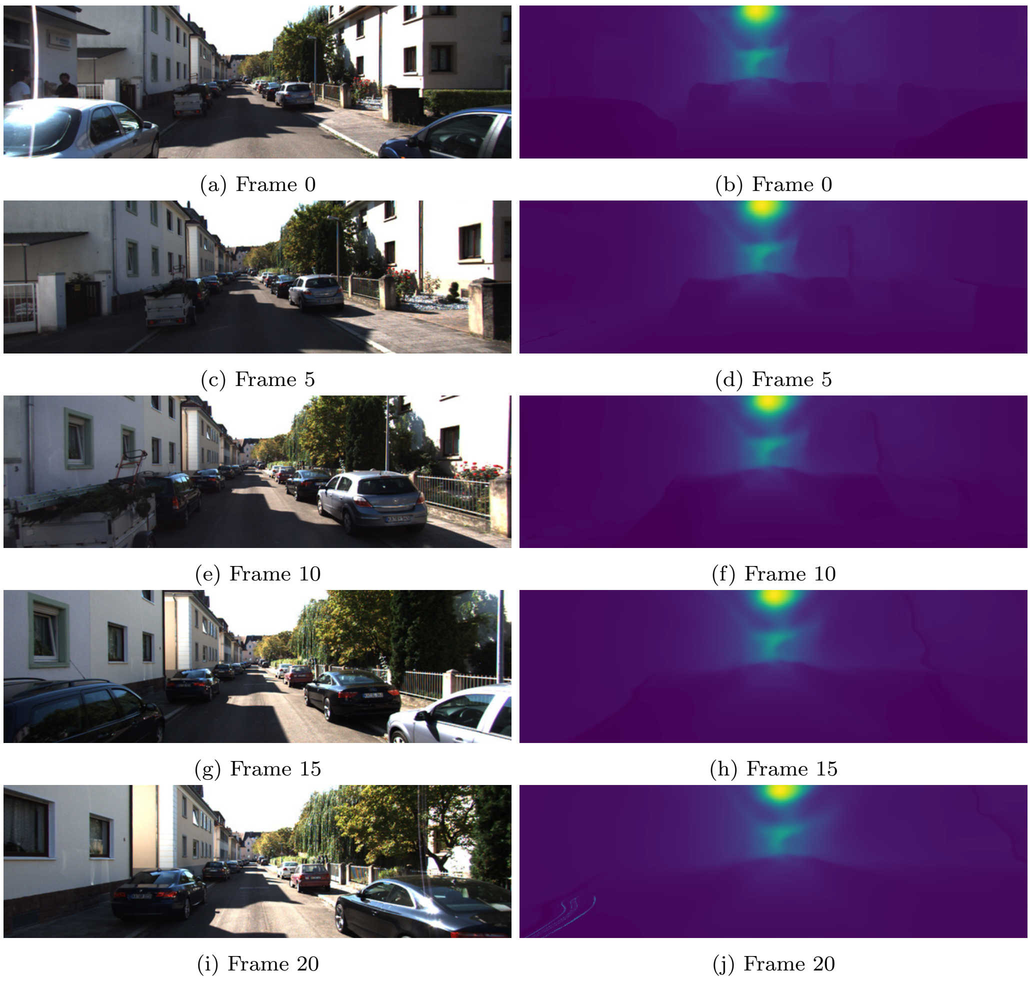

(a) Frame 0 (b) Frame 0
(c) Frame 5 (d) Frame 5
(e) Frame 10 (f) Frame 10
(g) Frame 15 (h) Frame 15
(i) Frame 20 (j) Frame 20

Figure 4.7: Depth map prediction with prediction horizon of 20 frames.

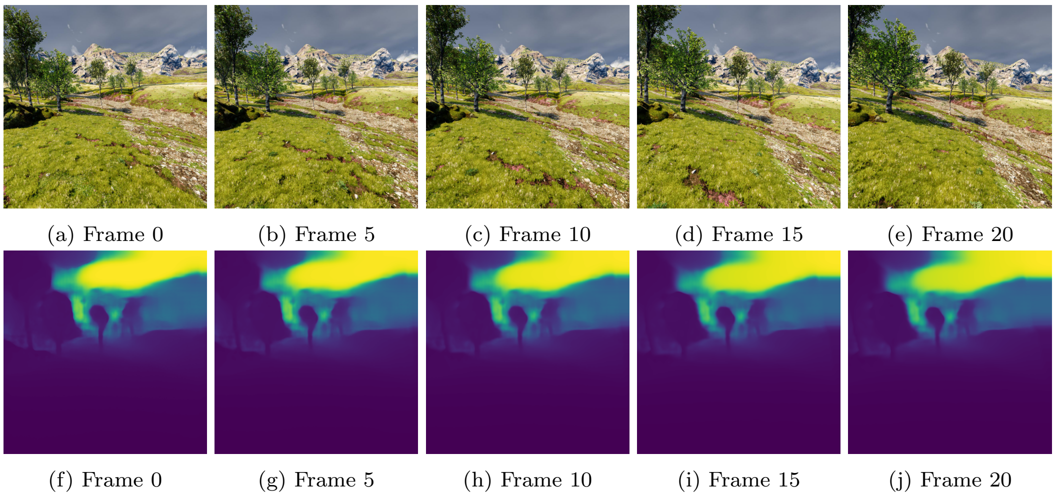

(a) Frame 0 (b) Frame 5 (c) Frame 10 (d) Frame 15 (e) Frame 20
(f) Frame 0 (g) Frame 5 (h) Frame 10 (i) Frame 15 (j) Frame 20

Figure 4.9: Depth map prediction frames 0-20.

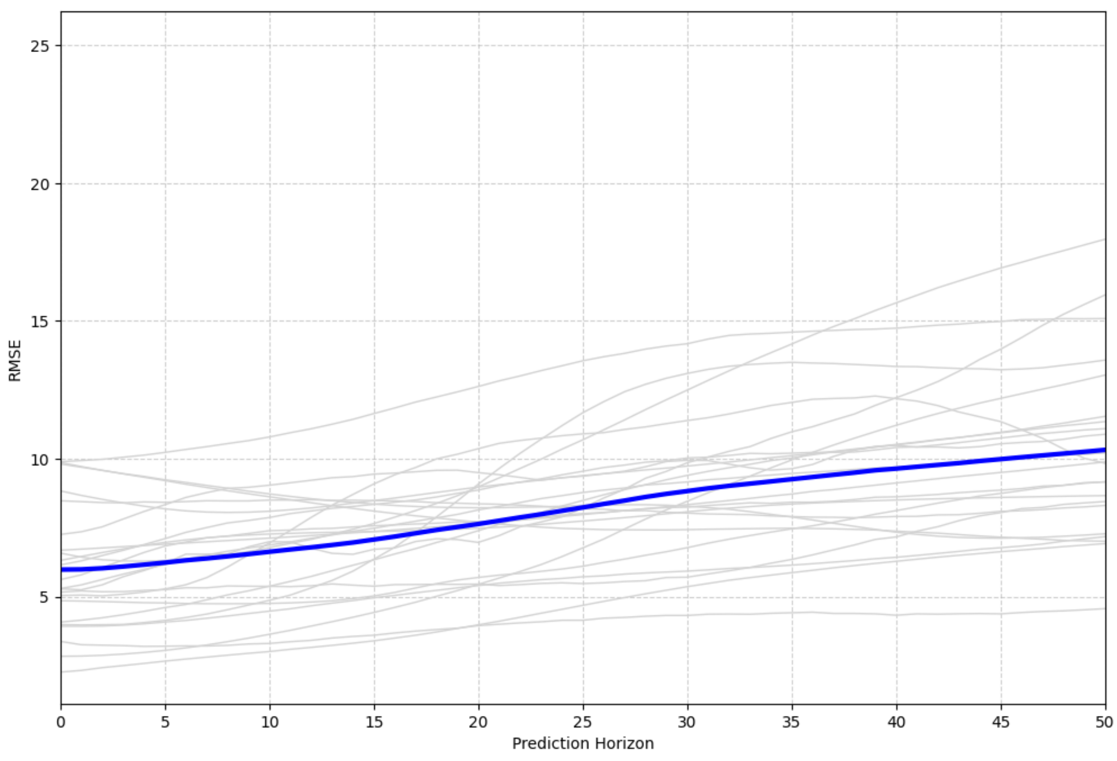
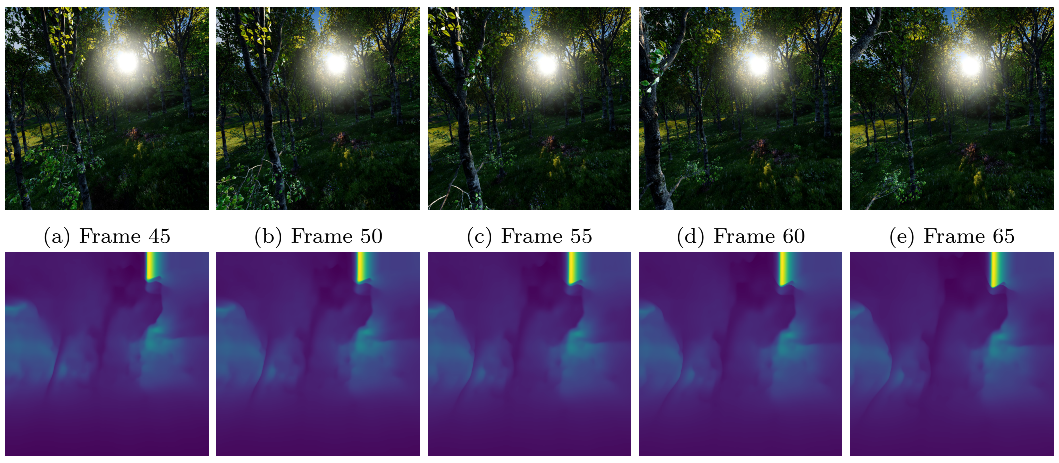


Figure 4.8: Forecast Horizon IMU Depth Map Generation on Mid-Air.



(a) Frame 45 (b) Frame 50 (c) Frame 55 (d) Frame 60 (e) Frame 65

(f) Frame 45 (g) Frame 50 (h) Frame 55 (i) Frame 60 (j) Frame 65

Figure 4.10: Depth map prediction frames 45-65.

The future frame prediction algorithm has a few limitations. Firstly, it assumes a static environment, and tests show that unexpected behavior occurs when a moving object is present in the frame. Secondly, as the agent moves, new areas may come into view while others become occluded, and this method cannot incorporate new information not present in the original depth map. Finally, IMU data is susceptible to noise and drift over time, which can lead to inaccuracies in pose estimation and, consequently, in the depth map transformation.

4.7 Conclusion

This chapter presented an evaluation of the DynaDepth framework for monocular depth estimation using the KITTI and Mid-Air datasets. The evaluation focused on the effectiveness of integrating visual data with inertial measurements and an EKF to improve depth estimation accuracy and scale consistency. The following summarizes the key points discussed.

- On the KITTI dataset, which features complex urban driving scenarios with dynamic elements, the full DynaDepth setup (using visual data, IMU, and EKF) outperformed other configurations. It achieved lower absolute and squared relative errors, enhancing both near and far-depth estimations. The scale error was notably reduced from about 30.6 in the visual-only model to around 5.5 with the full configuration. Threshold accuracy metrics indicated that the model with the Visual+IMU+EKF configuration was more reliable than other configurations in predicting depths within acceptable error margins. This suggests that the combined use of visual data, inertial measurements, and the EKF enhances the robustness of depth estimation, reducing errors more effectively compared to alternative setups.

- The impact of different environmental contexts within the KITTI dataset—City, Residential, and Road—was also analyzed. The DynaDepth framework performed best in environments with many static objects, such as residential areas, where accurate depth estimation is critical due to the proximity of obstacles. In open road scenarios with fewer visual cues, performance slightly declined, suggesting areas for potential improvement in less structured settings.

- For the Mid-Air dataset, which simulates drone flights in outdoor environments, integrating visual and inertial data with EKF once again led to the best results. The full configuration reduced error rates and improved accuracy across all metrics. The EKF effectively mitigated the noise inherent in IMU measurements, which is more present in drone dynamics due to higher degrees of freedom compared to ground vehicles. These results confirm the observed performance decrease in the KITTI dataset when operating in less structured environments, as the Mid-Air dataset represents an unstructured setting. This emphasizes that unstructured environments pose greater challenges for depth estimation models.

- Although integrating IMU measurements substantially reduced the scale error—by a factor of approximately 5.5 on the KITTI dataset and about 5.2 on the Mid-Air dataset—the final scale error remained in the range of 5 to 5.5. Moreover, improvements in other error metrics were relatively minor. Therefore, these enhancements can be regarded as an incremental improvement.

- The chapter also introduced a future depth map prediction algorithm that leverages IMU data and an initial depth map to predict depth maps without relying on continuous visual input. This approach is especially beneficial for energy-constrained applications where reducing the frequency of computationally intensive depth estimations is desirable. The evaluation showed that while the algorithm performs well over short forecast horizons, accumulating errors from IMU data and the assumption of a static environment limit its long-term accuracy. On the KITTI dataset, with a camera frequency of 10 frames per second, the future depth map prediction algorithm can predict up to 5 frames ahead, effectively reducing the need for DynaDepth from 10 runs per second to 2. Similarly, for the Mid-Air dataset, this reduces the usage of DynaDepth from 25 times per second to just 5.

5 Conclusion & Future work

This chapter presents the final reflections on the outcomes of this thesis and outlines potential avenues for further research. The first section provides a summary of the key findings and contributions, examining the research questions and the conclusions drawn from them. The second section explores future research directions, suggesting opportunities to expand on the present work, and improve methodologies.

5.1 Conclusion

This thesis proposes an extension of DynaDepth through a future frame prediction algorithm to enhance monocular depth estimation for UAVs. By integrating visual data with drone dynamics, this thesis addresses limitations of traditional depth estimation methods. Unlike existing approaches that rely solely on visual inputs and struggle with scale inconsistencies, this study leverages IMU data to generate more precise depth estimates. This approach is critical for navigating UAVs through complex 3D environments. Additionally, this thesis explores predicting future depth maps using initial estimates and temporal dynamic data, which enables proactive navigation and improved computational efficiency by reducing the frequency of depth estimation in real-time applications

The monocular depth estimation model, DynaDepth, proposed by [53] is introduced in Chapter 3 to perform depth estimation while addressing the scale limitations faced by state-of-the-art methods. This is achieved by incorporating an IMU sensor alongside the camera, providing scale awareness. The scale is further refined using an EKF to correct the inherent noise in the IMU measurements.

DynaDepth is evaluated on the automotive driving dataset KITTI [7] and on the simulated drone flight Mid-Air dataset [59]. The DynaDepth setup, which combines visual data, IMU, and EKF, outperformed other configurations on the KITTI dataset by reducing depth estimation errors and improving threshold accuracy metrics. It performed particularly well in residential areas with more static objects, but its performance declined slightly in less structured open road scenarios. On the Mid-Air dataset, integrating visual, inertial data, and EKF also yielded the best results, with EKF mitigating IMU noise in drone dynamics. However, the challenges of unstructured environments for depth estimation were noted.

The integration of IMU measurements significantly reduced scale error by approximately 5.5 on the KITTI dataset and 5.2 on the Mid-Air dataset, though the final scale error still remained in the 5 to 5.5 range. Despite the reduction in scale error, other performance metrics showed only marginal improvements, indicating that in this setup it is an incremental improvement.

The depth map prediction algorithm introduced in Chapter 4 aims to predict future depth maps using IMU data and an initial depth map, rather than relying on continuous visual input. The evaluation demonstrated that the algorithm performs well in short forecast horizons, which makes it effective in energy-constrained scenarios. However, the future frame prediction algorithm is limited by its assumption of a static environment, leading to inaccuracies when dynamic elements are present or when new areas come into view. Additionally, IMU data is prone to noise and drift, causing errors in pose estimation and depth map transformation over time. Despite these limitations, the algorithm's capability to maintain reasonably accurate depth estimations in the short term demonstrates its potential to complement existing depth estimation techniques. On the KITTI dataset, with a camera frequency of 10 frames per second, the future depth map prediction algorithm can predict up to 5 frames ahead, effectively reducing the need for DynaDepth from 10 runs per second to 2. Similarly, for the Mid-Air dataset, this reduces the usage of DynaDepth from 25 times per second to 5.

5.2 Future work

The results of this thesis contribute to monocular depth estimation for UAVs, yet there are several opportunities for improving and expanding upon the current work. First, to conclusively validate the benefits of the IMU-sensor integration with a camera, more rigorous evaluations under the same conditions should be conducted against other visual-based monocular depth estimation models on the Mid-Air dataset. However, only a few models have been publicly evaluated on the Mid-Air dataset, as seen on resources like "Papers with Code" [62]. Some effort may be required to reproduce monocular depth estimation models and evaluate them on the Mid-Air dataset.

Second, since the performance of the visual-IMU-based model is evaluated on simulated low-altitude UAV trajectories, an interesting research question would be how well the current setup translates to real-world operating drones and whether the same performance on the metrics can be achieved in those conditions.

Third, due to the current configuration of the DynaDepth model, a depth network $\mathcal{M}_d$ trained on dataset X cannot be directly applied to another dataset for zero-shot applications. This limitation arises because the IMU sensor provides the model with scale-awareness, which is inherently tied to the scale of the training dataset. As a result, when the model is applied to a different dataset, it lacks the necessary scale information and, under the current setup, cannot adapt to new scales in real-time. This presents an interesting research opportunity to develop methods that enable the model to learn scale online, allowing it to generalize effectively across diverse datasets.

Fourth, the current depth map generation algorithm cannot preserve the shape of objects in the scene. An interesting research question, therefore, would be how to predict future depth maps based on IMU measurements while maintaining object structure within the scene. One possible approach could involve adapting the method by [63], which focuses on object-oriented frame prediction. This paper employs a transformer model to learn consecutive perspective and affine transformations based on previous frames. A significant part of the paper discusses building the dataset for training the transformer network, and this code can also be found in the repository.

To apply this approach for future depth map prediction, the instance segmentation model would need to be fine-tuned on depth map data. However, to generate a complete future depth map, an additional model should be used to predict the remainder of the frame. For this purpose, the future depth map prediction algorithm proposed in this thesis could be integrated.

Bibliography

[1] PricewaterhouseCoopers, "Drone Deliveries: Taking Retail and Logistics to New Heights." Available: https://www.pwc.com/c1/en/drone-powered-solutions/drone-deliveries-taking-retail-and-logistics-to-new-heights.html. Accessed: Nov. 28, 2024.

[2] M. Fonder and M. Van Droogenbroeck, "Mid-air: A multi-modal dataset for extremely low altitude drone flights," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 553–562, 2019.

[3] J. Lambert, "Stereo and Disparity." Available: https://johnwlambert.github.io/stereo/, 2018. Accessed: Nov. 8, 2024. Figure titled "Two cameras with optical centers O_L and O_R are separated by a baseline B.".

[4] NVIDIA, "VPI - Vision Programming Interface: Pinhole Camera Model." Available: https://docs.nvidia.com/vpi/appendix_pinhole_camera.html. Accessed: Nov. 28, 2024.

[5] Y. Zheng, P. Liu, L. Qian, S. Qin, X. Liu, Y. Ma, and G. Cheng, "Recognition and depth estimation of ships based on binocular stereo vision," *Journal of Marine Science and Engineering*, vol. 10, p. 1153, 08 2022.

[6] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6612–6619, 2017.

[7] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, Aug. 2013.

[8] S. M. R. Islam, "Drones on the Rise: Exploring the Current and Future Potential of UAVs," *arXiv preprint arXiv:2304.13702*, 2023.

[9] N. Elmeseiry, N. Alshaer, and T. Ismail, "A Detailed Survey and Future Directions of Unmanned Aerial Vehicles (UAVs) with Potential Applications," *Aerospace*, vol. 8, no. 12, 2021.

[10] K. Telli, O. Kraa, Y. Himeur, A. Ouamane, M. Boumehraz, S. Atalla, and W. Mansoor, "A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs)," *Systems*, vol. 11, no. 8, 2023.

[11] McKinsey Company, "Clouds or Clear Skies? Prospects for Future Air Mobility." Online. Available: https://www.mckinsey.com/industries/aerospace-and-defense/our-insights/future-air-mobility-blog/clouds-or-clear-skies-prospects-for-future-air-mobility, 2024. Created: Jan. 23, 2024.

[12] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, pp. 982–987, 8 2023.

[13] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *arXiv preprint arXiv:1311.2524*, vol. abs/1311.2524, 2013.

[14] R. Girshick, “Fast r-cnn,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[15] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *arXiv preprint arXiv:1506.01497*, vol. abs/1506.01497, 2015.

[16] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *arXiv preprint arXiv:1506.02640*, vol. abs/1506.02640, 2015.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” *arXiv preprint arXiv:1512.02325*, vol. abs/1512.02325, 2015.

[18] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, “Digging into self-supervised monocular depth estimation,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3827–3837, 2019.

[19] J. Watson, O. M. Aodha, V. Prisacariu, G. Brostow, and M. Firman, “The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth,” *arXiv preprint arXiv:2104.14540*, 2021.

[20] Y. Liao, S. Kang, J. Li, Y. Liu, Y. Liu, Z. Dong, B. Yang, and X. Chen, “Mobile-seed: Joint semantic segmentation and boundary detection for mobile robots,” *IEEE Robotics and Automation Letters*, vol. 9, p. 3902–3909, Apr. 2024.

[21] Y. Yang, X. Xiong, and Y. Yan, “UAV Formation Trajectory Planning Algorithms: A Review,” *Drones*, vol. 7, no. 1, p. 62, 2023.

[22] X. Dong, M. A. Garratt, S. G. Anavatti, and H. A. Abbass, “Towards real-time monocular depth estimation for robotics: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 16940–16961, 2022.

[23] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. Reddi, “Mavbench: Micro aerial vehicle benchmarking,” in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 894–907, 2018.

[24] Y. Zhang, Q. Yu, K. H. Low, and C. Lv, “A Self-Supervised Monocular Depth Estimation Approach Based on UAV Aerial Images,” in *Proceedings of the 2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, pp. 1–8, 2022.

[25] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, “Fastdepth: Fast monocular depth estimation on embedded systems,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6101–6108, 2019.

[26] L. Papa, P. Russo, and I. Amerini, “Meter: A mobile vision transformer architecture for monocular depth estimation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 10, pp. 5882–5893, 2023.

[27] “Lunar Zebro — Nano Rover TU Delft.” Available: https://zebro.space/. Accessed: Nov. 28, 2024.

[28] M. Fonder and M. V. Droogenbroeck, “Mid-air: A multi-modal dataset for extremely low altitude drone flights,” in *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, June 2019.

[29] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, 2022.

[31] R. Birkl, D. Wofk, and M. Müller, "MiDaS v3.1 – A Model Zoo for Robust Monocular Relative Depth Estimation," *arXiv preprint arXiv:2307.14460*, 2023.

[32] V. Guizilini, I. Vasiljevic, D. Chen, R. Ambrus, and A. Gaidon, "Towards Zero-Shot Scale-Aware Monocular Depth Estimation," *arXiv preprint arXiv:2306.17253*, 2023.

[33] W. Yin, C. Zhang, H. Chen, Z. Cai, G. Yu, K. Wang, X. Chen, and C. Shen, "Metric3D: Towards Zero-shot Metric 3D Prediction from A Single Image," *arXiv preprint arXiv:2307.10984*, 2023.

[34] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth," *arXiv preprint arXiv:2302.12288*, 2023.

[35] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[36] C. Zhao, Y. Zhang, M. Poggi, F. Tosi, X. Guo, Z. Zhu, G. Huang, Y. Tang, and S. Mattoccia, "Monovit: Self-supervised monocular depth estimation with a vision transformer," in *2022 International Conference on 3D Vision (3DV)*, IEEE, Sept. 2022.

[37] R. Wang, Z. Yu, and S. Gao, "PlaneDepth: Self-supervised Depth Estimation via Orthogonal Planes," *arXiv preprint arXiv:2210.01612*, 2023.

[38] Y. Wang, Y. Liang, H. Xu, S. Jiao, and H. Yu, "SQLdepth: Generalizable Self-Supervised Fine-Structured Monocular Depth Estimation," *arXiv preprint arXiv:2309.00526*, 2023.

[39] T. Laidlow, J. Czarnowski, and S. Leutenegger, "Deepfusion: Real-time dense 3d reconstruction for monocular slam using single-view depth and gradient predictions," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4068–4074, 2019.

[40] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[41] R. Li, S. Wang, and D. Gu, "Deepslam: A robust monocular slam system with unsupervised deep learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2021.

[42] Y. Chen, C. Schmid, and C. Sminchisescu, "Self-supervised Learning with Geometric Constraints in Monocular Video: Connecting Flow, Depth, and Camera," *arXiv preprint arXiv:1907.05820*, 2019.

[43] X. Luo, J. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent Video Depth Estimation," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, vol. 39, no. 4, 2020.

[44] Z. Feng, L. Yang, L. Jing, H. Wang, Y. Tian, and B. Li, "Disentangling Object Motion and Occlusion for Unsupervised Multi-frame Monocular Depth," *arXiv preprint arXiv:2203.15174*, 2022.

[45] R. Li, D. Gong, W. Yin, H. Chen, Y. Zhu, K. Wang, X. Chen, J. Sun, and Y. Zhang, "Learning to Fuse Monocular and Multi-view Cues for Multi-frame Depth Estimation in Dynamic Scenes," *arXiv preprint arXiv:2304.08993*, 2023.

[46] R. Yasarla, M. K. Singh, H. Cai, Y. Shi, J. Jeong, Y. Zhu, S. Han, R. Garrepalli, and F. Porikli, "FutureDepth: Learning to Predict the Future Improves Video Depth Estimation," *arXiv preprint arXiv:2403.12953*, 2024.

[47] R. Chang, K. Yu, and Y. Yang, "Self-Supervised Monocular Depth Estimation Using Global and Local Mixed Multi-Scale Feature Enhancement Network for Low-Altitude UAV Remote Sensing," *Remote Sensing*, vol. 15, no. 13, 2023.

[48] M. F. Aslan, A. Durdu, A. Yusefi, and A. Yilmaz, "Hvionet: A deep learning based hybrid visual–inertial odometry approach for unmanned aerial system position estimation," *Neural Networks*, vol. 155, pp. 461–474, 2022.

[49] Y. Almalioglu, M. Turan, A. E. Sari, M. R. U. Saputra, P. P. B. de Gusmão, A. Markham, and N. Trigoni, "SelfVIO: Self-Supervised Deep Monocular Visual-Inertial Odometry and Depth Estimation," *arXiv preprint arXiv:1911.09968*, 2020.

[50] L. Han, Y. Lin, G. Du, and S. Lian, "DeepVIO: Self-supervised Deep Learning of Monocular Visual Inertial Odometry using 3D Geometric Constraints," *arXiv preprint arXiv:1906.11435*, 2019.

[51] L. Liu, G. Li, and T. H. Li, "Atvio: Attention guided visual-inertial odometry," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4125–4129, 2021.

[52] T. Zhang, D. Zhu, W. Shi, Y. Liu, G. Zhang, X. Zhang, and J. Li, "Self-supervised scale recovery for decoupled visual-inertial odometry," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1612–1619, 2024.

[53] S. Zhang, J. Zhang, and D. Tao, "Towards Scale-Aware, Robust, and Generalizable Unsupervised Monocular Depth Estimation by Integrating IMU Motion Dynamics," *arXiv preprint arXiv:2207.04680*, 2022.

[54] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Robotics: Science and Systems XI*, 2015.

[55] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[57] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2017.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv preprint arXiv:1912.01703*, 2019.

[59] M. Fonder and M. Van Droogenbroeck, "Mid-air: A multi-modal dataset for extremely low altitude drone flights," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 553–562, 2019.

[60] E. Games, "Unreal Engine web site." Available: https://www.unrealengine.com. Accessed: Oct. 30, 2024.

[61] D. Eigen and R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture," *arXiv preprint arXiv:1411.4734*, 2015.

[62] Papers with Code, "The latest in machine learning papers with code." https://paperswithcode.com. Accessed: Dec. 10, 2024.

[63] S. Mokssit, D. B. Licea, B. Guermah, and M. Ghogho, "An object-oriented deep learning method for video frame prediction," in *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pp. 398–402, 2024.

Deriviation preintegration terms

A

Prerequisites

Forster et al. [54] define preintegration terms for IMU measurements to combine multiple IMU readings into a single, more manageable representation for visual-inertial odometry (VIO). The following equations show IMU measurements incorporating the effect of gravity, noise and biases.

$$\begin{aligned} {}^{b}\tilde{\boldsymbol{\omega}}_{wb}(t) &= {}^{b}\boldsymbol{\omega}_{wb}(t) + \mathbf{b}^{g}(t) + \boldsymbol{\eta}^{g}(t) \\ {}^{b}\tilde{\mathbf{a}}(t) &= \mathbf{R}_{wb}^{\top}(t)({}^{w}\mathbf{a}(t) - {}^{w}\mathbf{g}) + \mathbf{b}^{a}(t) + \boldsymbol{\eta}^{a}(t) \end{aligned} \tag{A.1}$$

where superscripts {b,w} refer to the body and world frames, $\boldsymbol{R}_{wb}^{T}$ denotes the rotation matrix from the world frame to the body frame and ${}^{W}\mathbf{g}$ represents the gravity factor in the world frame. Furthermore, ${}^{b}\boldsymbol{\omega}_{\mathrm{wb}}$ refers to angular velocity of the body frame relative to world frame, ${}^{w}\boldsymbol{a}$ is acceleration expressed in the world frame and $\{\mathbf{b}^{g}, \mathbf{b}^{a}\}$ and $\{\mathbf{n}^{g}, \mathbf{n}^{a}\}$ refer to the Gaussian biases and random walks of the gyroscope and accelerometer, respectively.

By transforming the IMU measurements into changes in rotation $\boldsymbol{R}$, velocity $\boldsymbol{v}$, and position $\boldsymbol{p}$ between frames i and j, the following preintegration terms are derived:

$$\boldsymbol{R}_{ij} = \boldsymbol{R}_{i}^{T}\boldsymbol{R}_{j} = \prod_{k=i}^{j-1} \exp\left(\tilde{\boldsymbol{\omega}}_{k} - \boldsymbol{b}_{g_k} - \boldsymbol{\eta}_{k}^{g}\right)\Delta t \tag{A.2}$$

$$\boldsymbol{v}_{ij} = \boldsymbol{R}_{i}^{T}(\boldsymbol{v}_{j} - \boldsymbol{v}_{i} - \boldsymbol{g}\Delta t_{ij}) = \sum_{k=i}^{j-1} \boldsymbol{R}_{ik}\left(\tilde{\boldsymbol{a}}_{k} - \boldsymbol{b}_{a_k} - \boldsymbol{\eta}_{k}^{a}\right)\Delta t \tag{A.3}$$

$$\begin{aligned} \boldsymbol{p}_{ij} &= \boldsymbol{R}_{i}^{T}\left(\boldsymbol{p}_{j} - \boldsymbol{p}_{i} - \boldsymbol{v}_{i}\Delta t_{ij} - \frac{1}{2}\boldsymbol{g}\Delta t_{ij}^{2}\right) \\ &= \sum_{k=i}^{j-1}\left(\boldsymbol{v}_{ik}\Delta t + \frac{1}{2}\boldsymbol{R}_{ik}\left(\tilde{\boldsymbol{a}}_{k} - \boldsymbol{b}_{a_k} - \boldsymbol{\eta}_{k}^{a}\right)\Delta t^{2}\right) \\ &= \sum_{k=i}^{j-1} \frac{3}{2}\boldsymbol{R}_{ik}\left(\tilde{\boldsymbol{a}}_{k} - \boldsymbol{b}_{a_k} - \boldsymbol{\eta}_{k}^{a}\right)\Delta t^{2} \end{aligned} \tag{A.4}$$

where $\boldsymbol{R}_{ik} = \boldsymbol{R}_{i}^{T}\boldsymbol{R}_{k}$ and $\boldsymbol{v}_{ik} = \boldsymbol{v}_{k} - \boldsymbol{v}_{i}$.

Additionally, the time derivatives of rotation, velocity, and position are defined as:

$$\dot{\mathbf{R}}_{wb} = \mathbf{R}_{wb}\,{}^{\mathrm{b}}\boldsymbol{\omega}_{wb}^{\wedge}, {}^{\mathrm{w}}\dot{\mathbf{v}} = {}^{\mathrm{w}}\mathbf{a} = \mathbf{R}_{wb}\,{}^{\mathrm{b}}\tilde{\mathbf{a}} + {}^{\mathrm{w}}\mathbf{g}, \quad {}^{\mathrm{w}}\dot{\mathbf{p}} = {}^{\mathrm{w}}\mathbf{v}, \tag{A.5}$$

The skew-symmetric angular velocity matrix, $\boldsymbol{\omega}$, is represented as a vector using the hat operator:

$$\boldsymbol{\omega}^{\wedge} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathfrak{so}(3). \tag{A.6}$$

Velocity preintegration term

Differentiating the velocity preintegration term in equation A.3 with respect to time results into:

$$\boldsymbol{v}_{ij} = \boldsymbol{R}_{wb_i}^T(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{ij}) \tag{A.7}$$

$$\dot{\boldsymbol{v}}_{ij} = \frac{d}{dt}\left[\boldsymbol{R}_{wb_i}^T(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{ij})\right] = \boldsymbol{R}_{wb_i}^T\left(\frac{d}{dt}[\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{ij}]\right) \tag{A.8}$$

where the rotation term $\boldsymbol{R}_{wb_i}^T$ and $\boldsymbol{v}_i$ are constants as they are with respect to the initial frame and do not change over time. Substituting the velocity derivative from A.5 for v_j into A.8, removing the subscripts for readability, leads to

$$\dot{\boldsymbol{v}}_{ij} = \boldsymbol{R}_{wb_i}^T(\boldsymbol{R}_{wb_j}(t)\boldsymbol{a}(t) + \boldsymbol{g} - \boldsymbol{g}) = \boldsymbol{R}_{wb_i}^T\boldsymbol{R}_{wb_j}(t)\boldsymbol{a}^b(t) \tag{A.9}$$

using the rotation property:

$$\boldsymbol{R}_{wb_i}^T\boldsymbol{R}_{wb_j} = \boldsymbol{R}_{b_iw}\boldsymbol{R}_{wb_j} = \boldsymbol{R}_{b_ib_j} \tag{A.10}$$

gives the final velocity preintegration terms as denoted in equation A.11.

$$\boldsymbol{\beta}_{b_ib_j} = \boldsymbol{v}_{ij} = \int_{t\in[i,j]} \left(\mathbf{R}_{b_ib_t}\mathbf{a}^{b_t}\right) dt \tag{A.11}$$

Position preintegration term

The position preintegration term is derived by taking the integral of the velocity preintegration expression as:

$$\boldsymbol{\alpha}_{b_ib_j} = \boldsymbol{p}_{ij} = \int\int_{t\in[i,j]} \left(\mathbf{R}_{b_ib_t}\mathbf{a}^{b_t}\right) dt^2 \tag{A.12}$$

Rotation preintegration term

Differentiating the rotation term in equation A.2 is :

$$\begin{aligned}\frac{d}{dt}(\boldsymbol{R}_{ij}) &= \frac{d}{dt}\left(\boldsymbol{R}_{wb_i}^T\boldsymbol{R}_{wb_j}\right)\\ &= \boldsymbol{R}_{wb_i}^T\frac{d}{dt}(\boldsymbol{R}_{wb_j})\end{aligned} \tag{A.13}$$

Substituting the rotation derivative defined in equation A.5 leads to

$$\frac{d}{dt}(\boldsymbol{R}_{ij}) = \boldsymbol{R}_{wb_i}^T\boldsymbol{R}_{wb_j}(t)^{\mathrm{b}}\boldsymbol{\omega}^\wedge(t) \tag{A.14}$$

Using the rotation property in equation A.10 simplifies equation A.14 to

$$\frac{d}{dt}(\boldsymbol{R}_{ij}) = \boldsymbol{R}_{b_ib_j}{}^{\mathrm{b}}\boldsymbol{\omega}^\wedge(t) \tag{A.15}$$

The derivative of quaternion is given as

$$\frac{d}{dt}(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{q}\otimes\begin{bmatrix}0\\ {}^{\mathrm{w}}\boldsymbol{\omega}\end{bmatrix} \tag{A.16}$$

where $\boldsymbol{q}$ is the rotation denoted in quaternion form and $\otimes$ refers to the quaternion multiplication operator.

Using equation A.16 and the tranformation of angular velocity to the body frame in equation A.17, equation A.18 is derived.

$$\boldsymbol{\omega}_{b_t} = \boldsymbol{R}_{wb_i}^{T}\,{}^{\mathrm{w}}\boldsymbol{\omega}(t) \tag{A.17}$$

$$\frac{d}{dt}(\boldsymbol{q}_{b_i,b_j}) = \boldsymbol{q}_{b_i,b_j} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}_{b_t} \end{bmatrix} \tag{A.18}$$

Integrating over interval $[i, j]$ gives the final expression for the rotation preintegration term as

$$\boldsymbol{q}_{b_i,b_j} = \int_{t\in[i,j]} \boldsymbol{q}_{b_i,b_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}_{b_t} \end{bmatrix} dt \tag{A.19}$$