Lightweight Mapping of Unstructured Environments

A Method of Reconstruction of Agricultural Regions for Mobile Robotics

Author

Michael Emmanuel Katsoulis

University of Cape Town
Department of Electrical Engineering

Rondebosch, Cape Town
South Africa

MSc.(Eng.) thesis submitted in partial fulfillment of the requirements for the degree of MSc. in Mechatronic Engineering in the Department of Electrical Engineering at the University of Cape Town

---

Keywords: 3D Reconstruction; Unstructured Environments; Agricultural Robotics

Declaration

I, Michael Emmanuel Katsoulis, hereby:

1. grant the University of Cape Town free licence to reproduce the above thesis in whole or in part, for the purpose of research;
2. declare that:
   (a) this thesis is my own unaided work, both in concept and execution, and apart from the normal guidance from my supervisors, I have received no assistance except as stated in my bibliography and acknowledgments.
   (b) neither the substance nor any part of the above thesis has been submitted in the past, or is being, or is to be submitted for a degree at this University or at any other university.

Signed by candidate

Michael Emmanuel Katsoulis

Department of Electrical Engineering
University of Cape Town

Monday 27th November, 2023

Contents

Abstract

Lightweight Mapping of Unstructured Environments

Michael Emmanuel Katsoulis

Monday 27th November, 2023

We present a computationally, size and cost-lightweight monocular reconstruction pipeline that produces high-quality reconstructions in an unstructured agricultural environment consisting of orchards and vineyards.
The pipeline has to be deployed and tested on ground vehicles equipped only with a single monocular camera sensor. Running on a CPU only, while achieving a sufficient resolution to identify individual plants without limitations on maximum path length.

We show that a simple visual odometry system is capable of providing performance that is more accurate than GPS, with a relative transform error of 0.19m, without the need for techniques such as bundle adjustment or loop closure. Towards this contribution, we evaluate the impact of the choice of image feature on the accuracy of the visual odometry as well as the impact of the choice of disparity estimation method on the accuracy.

Additionally, we show that state-of-the-art unsupervised monocular depth networks can outperform stereo techniques in terms of accuracy achieved when estimating the depth in an agricultural setting. We also show that lightweight pyramid-based methods are able to match the performance of deep monocular depth networks at the task of disparity estimation.

The pipeline presented is optimised for application in agricultural environments and is lightweight in terms of size, weight, power and computational requirements. The pipeline functions using only a single camera and without any other sensors or sources of additional information making

Acknowledgements

As with any endeavor of significant effort and length, this work would not have been possible without the help of many people. While I cannot name everyone who has helped me along the way, there are a few people who deserve special mention.

A special thanks needs to be given to the following people whom I would not have been able to complete this work without.

To my family—my parents, my brother, and my sister—your unwavering love and support have been my bedrock. Like patient gardeners, you provided me with the grounding for my growth and the nurturing care that allowed me to flourish.
I owe much of who I am and where I stand today to each of you.

To my supervisor Dr Paul Amayo for all the guidance, patience and inspiration as well as for being a ceaseless source of help. You were always a message or a call away and have been a beacon along the way. I am profoundly thankful for the time you dedicated to my progress.

To David and Matt, my steadfast colleagues, your friendship, camaraderie, and infectious laughter have transformed this academic pursuit into a joyous venture. I am truly blessed to have shared this journey with you.

A heartfelt thanks to the friends and colleagues in the lab whose encouragement and shared experiences added color to our academic paths. And to the broader community around me, I express my gratitude to all who helped me along the way.

And lastly, to my mysterious examiners, your commitment to evaluating this work is sincerely appreciated. Your constructive feedback has been instrumental in refining my work, and I am grateful for the time and consideration you devoted to its assessment.

List of Figures

List of Tables

Introduction

1.1 Motivation for Research

The problem

Modern advancements in technology have improved the field of robotics and machine learning such that robots are increasingly being used in our everyday environment. Autonomous robots are now used for everything from cleaning household floors to running the complex internal logistics of warehouses and factories. The rapid adoption of robotic systems underscores the pressing need for enhanced autonomous navigation capabilities. [8, 9, 10]

Being able to effectively navigate an environment may seem intuitive to us, but giving a machine this ability is a non-trivial task. Robotic navigation, at its core, revolves around three fundamental aspects: spatial awareness, motion tracking, and path planning. Therefore, to successfully navigate as a robot, the task is to know where you are, where you have been, and where you are going.

In order to understand where you are, you need to understand both what is in your surroundings as well as where you are relative to these objects in 3D space.
Knowing where you have been involves understanding your motion relative to your surroundings. The complexity of this task increases when your surroundings are similar, change significantly as you move, or if the objects around you are moving as well.
Knowing where you are going requires knowledge of the road ahead of you and what is in your environment between you and your target location such that you can travel the best path around any obstacles and reach your destination.

Challenges in the real world

Traditionally, the use of robots has been limited to structured environments such as the aforementioned urban regions, factories, and warehouses. These environments typically consist of flat, planar surfaces and are often well-lit. In these environments, navigation is simpler as maps can be pre-built and assumptions can be made about the environment. In some circumstances, such as in warehouses, the environment can even be controlled and modified to suit the needs of the robot.

Unstructured environments challenge assumptions made in structured environments. Objects and surfaces are uncontrolled and organic, resulting in them being irregular and non-planar.

Lighting conditions are also uncontrolled, with shadows and reflections often present. Structural features such as walls and floors are also often absent, making it difficult to build maps and plan paths. Additionally, many mainstay sensors such as GPS are hampered when working in agricultural settings where trees and plants often reach overhead, occluding signal and increasing GPS error. The vehicles used are also further away from infrastructure and so have to be more independent.

The need to reconstruct the surroundings and paths travelled is, therefore, a direct result of the inherent complexity of unstructured environments. Unlike structured settings, unstructured environments pose unique challenges, with irregularity and organic shapes and layouts that defy conventional assumptions pivotal to structured navigation.

In such scenarios, the ability to reconstruct the paths travelled becomes paramount. This reconstruction thus aids in overcoming environmental challenges, giving a virtual world that can be referenced, contributing to the robot's spatial awareness and motion tracking.

Figure 1.1: A photo of a robot in an agricultural setting showing the unstructured, rugged nature of the environment. [1]

Physical and hardware demands in unstructured environments

Physical and hardware constraints are more severe when a robot is operating in an unstructured environment. By nature of the environment, robots need to be more mobile and agile, requiring them to be smaller and lighter. Furthermore, robots need to be able to operate for longer periods without human intervention, requiring them to be more power efficient. These constraints are further exacerbated by the need for robots to be more independent, requiring them to be more robust.

These constraints are especially important in the agricultural sector where robots are used to automate tasks such as crop monitoring, harvesting, and spraying. In these applications, robots need to be able to operate for long periods in a variety of conditions, including rain, mud, and

dust (see fig. 1.1 for a visual demonstration). These conditions are often detrimental to the operation of sensors and electronics, requiring them to be more robust and sealed. These robots also need to be affordable to be viable for use by small-scale farmers who often need to operate on thin margins.

Impact of a lighter, more efficient pipeline

The study and the proposed end-to-end monocular reconstruction pipeline tackle these challenges through the application of computer vision techniques, reconstructing the environment using a single camera. Unlike the prohibitively expensive Lidar sensors, cameras offer a cost-effective alternative, thereby increasing the accessibility of robotic technologies. Monocular cameras typically come at a cost of $100, in contrast to the starting price of $4500 for typical Lidar sensors [11, 12]. This significant cost difference underscores the transformative potential of vision-only approaches in democratizing the field of robotics.

The research in end-to-end monocular reconstruction not only confronts the limitations of current multi-sensor navigation methods but also paves the way for more economical and efficient robotic navigation. By empowering robots to reconstruct their environment and paths, this methodology aligns with the broader objective of developing adaptable and versatile robotic systems.

In summary, the integration of a vision-based reconstruction pipeline marks a pivotal advancement in enhancing robotic navigation capabilities. Its impact transcends the confines of structured environments, making it a crucial contribution to the domains of robotics and mechatronic engineering. As the pursuit continues to equip robots with greater autonomy and proficiency in unstructured environments, the significance of this research becomes increasingly apparent, shaping the trajectory of autonomous robotics in the future.

Figure 1.2: An example of a stereo image pair from the dataset collected at the Lindenhoff farm. See section 11.2 for more information.

Contributions of this research

Classical reconstruction pipelines use stereo cameras for depth prediction and VO. This allows for dense and fast depth estimation from relatively cheap sensors. While being computationally lightweight and mature in implementation, the accuracy of far-field estimations is dependent on a large baseline which, in many applications, is impractical for deployment on some platforms. Furthermore, the increased mass and power requirements of an additional camera present a limitation.

Alternative pipelines utilise lidar sensors for high-accuracy depth sensing. With modern advancements, lidar sensors have become increasingly higher resolution while also becoming more affordable. Implementations such as [13] have been shown to produce highly accurate reconstructions with the sparse points obtained from lidar sensors. The price of these sensors is however still unaffordable for most field applications.

In the field of agricultural robotics, dense reconstructions are needed to capture not only the geometry of the environment but also to provide valuable information about the condition of crops and produce while being useful for localisation, path planning, and other applications. To this effect, we propose a pipeline that utilises a lightweight monocular depth prediction network on a single camera. Our pipeline is lightweight in size, weight, power, and computational requirements - running on CPU only - while producing dense reconstructions in near real-time.

In this work, we investigate the methods and decisions behind constructing a size, weight, and power lightweight pipeline for reconstruction in agricultural regions. We investigate appropriate structures for this pipeline. Additionally, we explore methods of visual odometry for tracking the position of a platform as well as the methods of disparity estimation for obtaining geometrically accurate information about the 3D environment which can be fused to create a reconstruction that faithfully represents the path travelled by a robot.

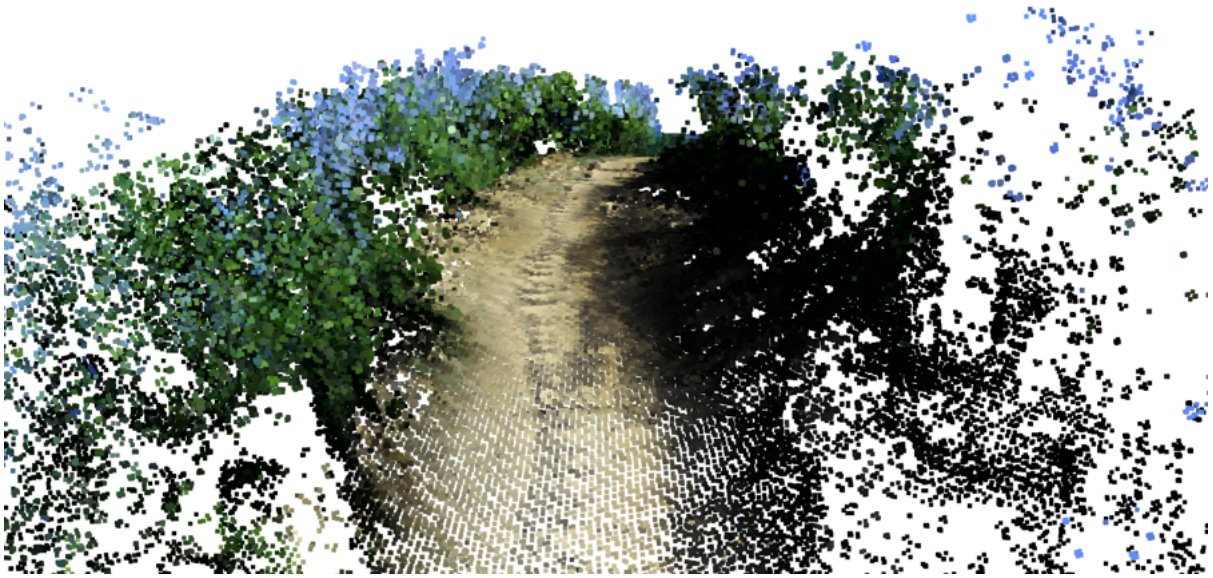


Figure 1.3: A screenshot of a reconstruction of a farm road with difficult shadows and occlusions using the proposed monocular pipeline.

Preliminaries

This work presents a pipeline for the reconstruction of a 3D environment in a manner that is lightweight in terms of computational, size, weight and power requirements (SWaP constrained). While this pipeline can be used agnostic to a particular application, the focus area of this research is the application in mobile robotics. Namely, we focus the implementation on ground-based robotic rovers as the platform of choice. Therefore, this chapter introduces the sensors as well as some of the preliminary mathematical tools that are utilised in mobile robotic platforms. We begin by presenting the models of the sensors used before explaining the coordinate reference frames as well as the transforms between them.

2.1 Sensors

2.1.1 Pinhole Camera Model

For the cameras used in this work, we utilise the pinhole camera model that is used in computer vision to mathematically describe the manner in which three-dimensional points/objects are projected onto the camera sensor. In this model, the camera center is located behind the image plane. When projecting a point onto the image plane, the location of the point is determined by the intersection of the image plane with the ray from the camera center to the point. The distance of the image plane from the camera center is determined by the geometry of the lens and the camera. This distance is referred to as the focal length and is measured between the camera center and the center of the image plane.

In this model, the camera intrinsic matrix describes the parameters that determine the location of the plane relative to the camera center as well as the optical center of the image plane.
The Camera Intrinsic Matrix is given by the following matrix:

$$K = \begin{pmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

Where f_u and f_v describe the focal distance in the u and v directions. Similarly c_u and c_v describe the location of the optical center in the image plane when measured from the top left corner. The parameter s is normally approximately zero and describes the 'skew' warping of the image.

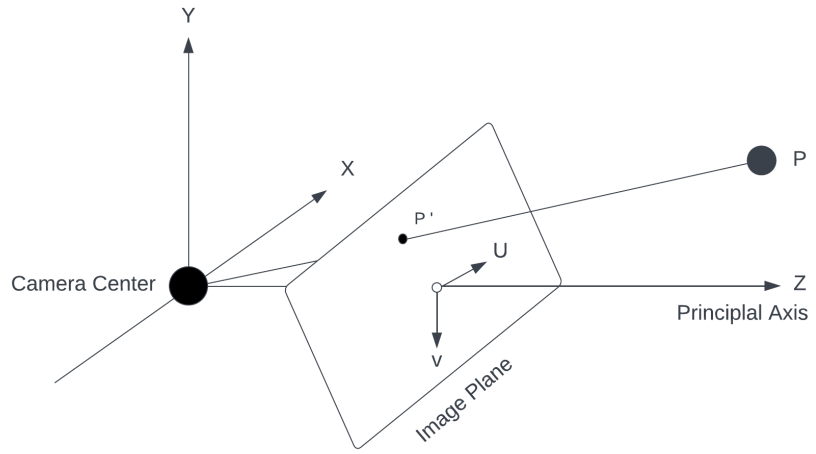


Figure 2.1: Diagram of the pinhole camera model

2.1.2 Stereoscopic Camera Model

Stereoscopic cameras capture images while also giving a perception of depth.
Stereoscopy, derived from the greek words *στερεός* (stereos) 'firm, solid', and *σκεπέω* (skopeo) 'to look, to see', uses two simultaneous views from different locations to give an impression of 3D structure or depth.

The invention of the stereo camera and stereo images is attributed to the English physicist Sir Charles Wheatstone (1832) who and later improved by Sir David Bewster (1849) [14]. Their stereo images involved using two cameras placed approximately as wide apart as the human eyes. The developed positives of the images were then placed in an apparatus representing a pair of eyeglasses. When looking at the images through the eye holes in the apparatus, the image would give the impression of depth when interpreted by the human brain. As camera imaging technology advanced along with computer processing, the role of interpreting depth could be performed by computers in the stead of a brain.

Stereo vision has evolved as the biologically dominant method of depth perception in the animal kingdom. With different roles in the food chain, the placement of an animal's eyes is directly linked to its requirements of depth perception. Animals that require good depth perception have eyes that have significantly overlapping fields of view whereas the converse is true for species that are generally prey as their emphasis is on the widest field of view. [15] The same concepts apply when designing stereo camera sensors. The image in Figure 2.2 shows how a point in space is projected onto the image planes of the two sensors in a stereo camera, similar to the eyes of an animal.

By aligning two pinhole cameras a known distance apart with their optical axis parallel, a stereo

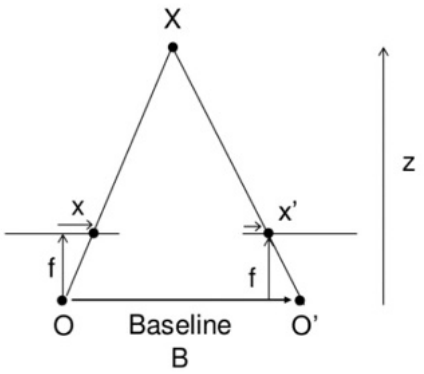


Figure 2.2: Diagram of a stereo camera [2]

camera is formed from two monocular cameras. Given that we know the distance between the two optical centers, referred to as the baseline, the depth of a point can be calculated using the shift in its location in the two images. The overlapping region between the cameras fields of view is where depth can be estimated since the point needs to appear in both images.

Calculating the depth of a point is a matter of solving similar triangles.
This leads to the depth being given by:

$$z = f\frac{b}{x_l - x_r} \tag{2.2}$$

Where x_l and x_r are the x positions of the point in the rectified image plane and f is the focal length of the camera.

2.1.3 Lidar Sensors

Light Detection and Ranging, or LiDAR, sensors are time-of-flight sensors that measure the distance to an object using the time it takes for the reflected light to return to the receiver.

Radar had historically been an effective way of ranging targets using time-of-flight methods. The first systems that used light to range an object were developed in the late 1930s where light pulses from spotlights were used to measure the height of clouds in the atmosphere. In 1960, the invention of the laser by Theodore Maiman [16] provided a more instantaneous and focused light source that increased the accuracy in ranging objects.

In 1961, the Hughes Aircraft Company introduced a system to range and track satellites by combining an imaging system with a powerful laser. This laser-based range-finding method was given the name Colidar, meaning "Coherent light detecting and ranging". Although the methods for ranging proved rather accurate, lidar's main application at the time was still meteorological measurements of clouds, pollution and other atmospheric properties. In the late 1960s, NASA expanded the use of lidar to topographical mapping from aircraft and in 1971, a lidar system was deployed on the Apollo 15 mission and used to map the surface of the moon [17, 18].

As technology has evolved, lidar systems have grown increasingly sophisticated and accurate. This has enabled a multitude of different applications. The highly accurate depth measurements provided by lidar systems have made it a pivotal sensor in geometric mapping and reconstruction in uses ranging from discovering ancient ruins in the Amazon rainforest to 3D scanners used in engineering modeling and design.

Most modern Lidars use low-powered lasers that have several beams of light usually from the non-visible spectrum. By measuring the reflection time for each of the beams, the depth to points along each of the beam rays is determined. The set of beams is then rapidly swept across the environment to sample the depth across the scene. The result is a cloud of points around the sensor that gives a sparse representation of the 3D space.

The lidar used in this work has 32 beams and rotates at 10Hz to give a dense pointcloud however, there are other versions that do not rotate or only have a single beam. Precise timing is needed to measure each beam while very high-speed computational hardware is required to solve the distance of each of the points measured and assemble them into a pointcloud. These two demanding requirements mean that modern state-of-the-art lidar sensors are orders of magnitude more expensive than a stereo camera. While the calculation for the depth of a beam is relatively simple;

$$d = \frac{c \cdot t_{reflection}}{2} \tag{2.3}$$

the speed of light (c) is a fixed parameter which is the defining agent in the difficulty of manufacturing increasingly accurate, affordable and point-dense lidar sensors.

2.2 Coordinate Reference Frames

Reference frames in robotics, as well as our 3D reconstruction application, are used to describe the position of a sensor or robot at any point in time. In 3D, they allow us to efficiently describe the position in terms of translation in x, y and z as well as the rotation (*roll*, *pitch* and *yaw*) for the 6-DOF coordinates of the platform or sensor. In this section, we introduce the special Euclidean Group which mathematically describes the reference frames in robotics.

2.2.1 The Special Euclidean Group and Rigid Body Motion

In Mathematics, an Euclidean group is a set of possible rotations and translations that preserve the Euclidean distance between points. The special Euclidean Group $\mathbb{SE}(n)$ represents a set of all possible rigid-body transformations, in terms of translations and rotations, that can occur in n-dimensional space, typically 2D or 3D space. In this work, the $\mathbb{SE}(3)$ transformations are used to represent the transforms between different poses of the robot in the environment.

The $\mathbb{SE}(n)$ transformation consists of an $n \times n$ rotational matrix as well as a translational vector of dimension n. Therefore, given an $\boldsymbol{n}$ dimensional positional vector $\boldsymbol{p}$, it can be transformed to a new positional vector $\hat{\boldsymbol{p}}$ via the equation

$$\hat{\boldsymbol{p}} = \boldsymbol{R}\boldsymbol{p} + \boldsymbol{t} \tag{2.4}$$

This transformation can be combined into a single matrix for simplicity. Thus, the transformation from the source position B to the destination position A can be described by T_B^A as defined by

$$T_A^B = \begin{bmatrix} \boldsymbol{R}_A^B & \boldsymbol{t}_A^B \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix} \tag{2.5}$$

Applying the transformation T to the n-dimensional homogenous vector p is done as follows

$$\hat{p} = T \begin{bmatrix} p \\ 1 \end{bmatrix} \tag{2.6}$$

Since the rotational matrix $\boldsymbol{R}$ is orthogonal, the inverse of the transform can efficiently be computed As

$$(T_A^B)^{-1} = T_B^A = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix} \tag{2.7}$$

2.2.2 Convention

In order to describe the position of the platform or sensor at one point in time relative to another, the $\mathbb{SE}(3)$ transformations described above are used. In this subsection, the conventions and notations for using these transforms are described.

The position of any point of space is defined relative to an origin and a reference frame. In this work, two reference frames are used, namely the robot and camera reference frames. The robot reference frame describes the position of the vehicle in world coordinates relative to the position at the start of the platform's travels. The camera frame describes a point in terms of its observed location relative to the camera itself.

To illustrate the use of the reference frames, let us look at the example where a camera observes a point $\boldsymbol{p}_{camera} = (x_c, y_c, z_c, 1)^T$. The location of this point in the robot frame is determined using the $\mathbb{SE}(3)$ transform between the reference frames via,

$$\boldsymbol{p}_{robot} = \boldsymbol{T}_{camera}^{robot} \boldsymbol{p}_{camera} \tag{2.8}$$

and vice versa for determining the position of a point in the camera frame from its position in the robot frame

$$\boldsymbol{p}_{camera} = (\boldsymbol{T}_{camera}^{robot})^{-1} \boldsymbol{p}_{robot} = \boldsymbol{T}_{robot}^{camera} \boldsymbol{p}_{robot} \tag{2.9}$$

When connecting transforms between positions along a path, it quickly becomes cumbersome to represent one transform after another, defining a unique reference frame for each sensor in each position. To mitigate this problem, we introduce the transform composition operator $\oplus$ which allows transforms to be combined as follows,

$$T_a^c = T_a^b \oplus T_b^c \tag{2.10}$$

Therefore a long series of transforms can be represented by its combined version as follows,

$$T_a^e = T_a^b \oplus T_b^c \oplus T_c^d \oplus T_d^e = (T_e^a)^{-1} \tag{2.11}$$

where the inverse of the composed transform is defined by,

$$T_e^a = (T_a^b \oplus T_b^c \oplus T_c^d \oplus T_d^e)^{-1} = (T_a^e)^{-1} \tag{2.12}$$

Part I

State of the Art

Introduction

The focus of this work is on the computational efficiency and geometric accuracy of the reconstruction pipeline. Since we are not trying to redefine the methods of visual odometry or depth estimation, this literature review will focus largely on these methods. Later, we will present our contributions to the field through our experimentation evaluating the impact of depth estimation methods and feature selection on the accuracy and computational efficiency of the pipeline.

In section 4.1, we present an overview of 3D reconstruction approaches, focusing on high-level perspectives from a sensor standpoint and lightly exploring their implementations. Thereafter we present our proposed topography for the reconstruction pipeline in chapter 8. In this presentation of the state of the art, emphasis is placed on the ramifications of system-level choices and their influence on the overall pipeline. To avoid redundancy and keep this work succinct, more detailed discussions on the implementation specifics - such as depth estimation methods, identification of key points, and feature tracking - are reserved for subsequent chapters.

We provide a literature review of the field of Visual Odometry in Chapter 5 where we discuss the different methods of estimating the motion of a platform using only a video stream. We delve into the importance of the selection of features and the methods of tracking them. Additionally, we examine odometry calculation methods employing alternative sensors like depth sensors and lidar sensors, contrasting them with the prevalent use of GPS sensors which are the de facto choice for platform odometry in agricultural machinery.

For the disparity estimation methods, we categorise the literature according to the underlying principles. We explore the literature on disparity estimation using lightweight, geometric methods in Section 6.1 before delving into the realm of learned methods for dense depth estimation in Section 6.2. The key distinction lies in the computational requirements. Geometric methods are computationally lightweight but less accurate, while learned methods are more accurate yet computationally intensive. Both fields have witnessed extensive research and development and as such, we present a literature review of both domains.

Reconstruction

4.1 Approaches towards 3D Reconstruction

3D reconstruction is a powerful tool that allows us to capture digital models of the world around us. In the field of robotics, 3D reconstructions allow robots the ability to act more autonomously and better interact with their environments.

Therefore, the field of 3D mapping and reconstruction has seen significant research and contributions [19, 20, 21, 22, 23]. Many different approaches have been taken towards the mapping problem from methods that utilise lidar and depth sensors [13, 24, 25], to those that use stereo cameras [21, 19, 26] and those that only utilise monocular cameras [27, 28, 29, 30, 31].

These methods presented in [13, 24, 25] focus on the use of lidar and depth sensors for 3D reconstruction. And, while these methods can produce extremely accurate reconstructions, the financial and weight requirements of these sensors make have prevented these methods from being viable options for lightweight platforms. Lighter, cheaper, and low-power alternatives can be used at the cost of depth density.

Methods such as [32, 24, 33, 34] add a monocular camera to a few-beam (usually 4-beam) lidar sensor to increase the density of the resulting reconstruction through interpolation. They show that this approach yields results that are comparable to those of significantly more expensive lidar sensors. All at a fraction of the cost and power requirements of a full 64-beam lidar sensor. This is performed using the monocular camera to guide the shape of objects which are then ranged using the lidar sensor. These shapes are then interpolated either by simply assigning the lidar's depth measurement to the segments of a segmented image or by using a more complex interpolation that utilises image gradients in conjunction with depth gradients to determine the depth of each pixel in the image. We find that these methods are extremely effective at increasing the density of the resulting reconstruction. However, the addition of a camera to the sensor array increases the cost and power requirements of the system which does not solve the problem for lightweight platforms.

However, much of the inspiration for this work comes from the methods used by [26, 28, 30] and as such, we will focus on the approaches of these works for this literature review. All three works feature a similar general pipeline which works in three steps. The first is to obtain the depth estimates of trackable image features then secondly, using those depths and the motion of features an estimate for the ego-motion of the platform is obtained. By integrating this estimate of motion, the current position of the tracked points relative to the initial starting point can be calculated. Then thirdly, the 3D location of the tracked points is combined into a digital reconstruction.

Greene et al. [26] propose a method for computationally lightweight reconstruction of an environment. Their method utilises stereo imagery to estimate the depths of points which are tessellated into a mesh. By tracking the relative motion of these points, they can add new points to the mesh when they are encountered by the moving vehicle at the next frame. This approach results in a constantly growing mesh that represents the surface of the environment and thus is an extremely lightweight method of reconstruction. The benefit of this approach is the high framerate of their pipeline which can be implemented on high-speed vehicles however this is at a cost to reconstruction density. As such, this approach represents the most lightweight of the reviewed methods of reconstruction in terms of computation requirements to expand their mesh surface and therefore the reconstruction. The disadvantage of this method is that the resulting mesh surface is extremely sparse which, in an outdoor setting, would result in a reconstruction where individual trees would not be differentiable from one another. For this reason, a denser method of reconstruction is required for our purposes.

In [28] proposed by Rosinol et al. use a sparse set of features to create a mesh reconstruction of surfaces in the environment. To this end, they develop a Visual Inertial Odometry based pipeline which utilises variational smoothing of the depth of the tracked points using a mesh. The depth of these points is obtained using inverse depth estimation from the different poses obtained from the VIO. The results of their method is an accurate mesh representation of the environment, using a sparse set of features results in a good reconstruction in indoor environments with planar surfaces which are uniform due to the regularisation. While the sparseness of their resulting reconstruction is appropriate for a structured environment, the complex shapes of an unstructured, outdoor environment would not be adequately represented. This method does however highlight the need for regularisation of depth estimates and surfaces that will be integrated into a 2D reconstruction.

In contrast to the previous two methods, the work presented by Yokozuka et al [30] provides a method for extremely dense reconstruction using many tracked features. Their proposed method uses VIO and feature detection and tracking based on the local maxima of an image gradient function. The result is a massive number of points that provide a detailed reconstruction of the environment. Their method required the engineering of a new descriptor-less as the matching of conventional descriptors for the tracking and depth estimation of their dense set of points would have been too computationally expensive. Once their method has determined the depth of the dense set of points as well as the position of the camera, they utilise a truncated signed distance field to represent the location of reconstructed surfaces in 3D space. This approach allows them to quickly integrate new surfaces into a reconstruction while being computationally lightweight and robust to outliers. The resulting reconstructions from their method are sufficiently dense for identifying small features while their pipeline can run in near real-time. However, their method of estimating the depth of each feature could potentially be improved upon by using a dense depth estimate from either a stereo camera or a monocular network considering the speeds at which modern methods can estimate depth for both of these potential sources.

From the review of the literature, an efficient and lightweight pipeline for reconstruction has been determined. before this pipeline can be implemented in an agricultural setting, an investigation is needed to determine the best choices for depth estimation as well as methods of determining the current location of the platform, all without using multiple sensors. This choice of using one sensor aims to reduce the power and space requirements of the reconstruction pipeline and makes it better suited for size, weight, and power-constrained platforms in robotics.

Visual Odometry

5.1 Introduction to Visual Odometry

Humans and animals have the ability to infer their own ego-motion as they move through an environment, being able to estimate their path travelled even when their environment is unfamiliar. They can do this by using their vision to track the movement of objects in their environment relative to themselves. This ability extends to estimating the path of a camera when watching videos.
An extreme example that is a testament to this ability is that of human drone pilots. Pilots of drones and other remote vehicles must know exactly where their vehicle is and where it has travelled, often with the only information being a video stream [35]. Incredibly, these pilots can estimate the path of their vehicle with high accuracy and precision, even when the vehicle is moving at high speeds and the environment is unfamiliar. Furthermore, their performance does not degrade significantly even when the authors of [35] introduce missing frames into the video stream. Giving a machine the ability to estimate its path using only a video stream would allow it to act more autonomously and would be a valuable tool for autonomous vehicles and robots. As such, it is a field that has seen significant research and development. This ability to estimate the path of a vehicle using only a video stream is known as Visual Odometry (VO) and in this chapter, we provide an introduction to the field.

Robots and autonomous vehicles would benefit from being able to know where they have travelled. Whether the robot is exploring a new environment or navigating through a familiar one, accurate knowledge of its position allows a vehicle to act more autonomously [36, 37]. This opens up possibilities such as navigating environments that are unsafe for humans, automating the paths of vehicles in farms and factories, and allowing robots to be integrated into the world without placing them, people or their environment at risk of harm.

Visual Odometry is defined as the process of estimating the motion of a robot, in terms of rotation and translation, using a sequence of images as the input. While various sensors and methods enable direct position measurement, visual odometry only requires cameras to determine ego-motion. For robots and vehicles that are already equipped with cameras, this implies that VO can be implemented without any additional sensors which would add the cost of expense, added weight and power consumption.

Modern GPS sensors can give positions with accuracies of 1-10m and are useful for determining the location of a vehicle over large distances where most other sensors would be affected by drift. The downside of purely GPS-based systems is the noise that affects measurements, especially when surrounded by objects such as trees and buildings that all can cause backscattering. To quantify the impact this has in agricultural regions, [38] shows that the accuracy of GPS

measurements is affected by the number of trees in the vicinity of the receiver. They show that in static measurements, which are the best-case scenario for GPS, the impact of trees on the accuracy of GPS measurements is in the order of 1-2m. This is a significant error when compared to the accuracy of GPS measurements which were shown to be 2-5m in the same study. This evidence supports the claim that GPS is unsuitable for high-fidelity tracking. There are, however, techniques for filtering noise, adding additional sensors such as inertial units or expanding systems with methods such as Real Time Kinematic (RTK) compensation which increase the performance of GPS. But these solutions all add cost and complexity to the problem while still being dependent on a good GPS signal with low dilation of precision.

Other methods such as triangulating the location of a robot relative to beacons at known locations are reliable and proven methods. However, it is dependent on beacons at a known location which adds setup and cost. Vehicles cannot be used in locations without the appropriate facilities.

Onboard high-fidelity depth sensors for accurate ego-motion estimation. Low-drift and real-time lidar odometry and mapping (LOAM) presented in [13] is one such example of an algorithm that uses the highly accurate and dense depth measurements of a lidar pointcloud to estimate the ego-motion of a robot. the results obtained from LOAM are significantly more accurate than that of a GPS for small movements. Despite being real-time, their method achieves a relative transform accuracy of 2.8% in an outdoor environment and 1.1% in an indoor environment. This accuracy is high enough that it significantly outperforms GPS even when drift is taken into account, even with paths with lengths of >300m. The problem with utilising a method such as LOAM is the prohibitive expense for a lidar sensor (upwards of $4 600 [12]) that can provide accurate, dense enough measurements required for this performance. In this work, we are fortunate enough to have a capable lidar sensor and as such, LOAM is utilised as a ground truth benchmark against which we can compare our visual odometry methods as well as the onboard GPS sensor.

5.2 Methods and Development of Visual Odometry

One of the first methods that used visual features to estimate the ego-motion of a robot was implemented in the 1980s by Moravec [39] at Stanford University. Their method was implemented on a mobile robot that navigated through a cluttered indoor environment, calculating a potential path while estimating its own motion through the environment. Using a single camera that could move on a rail as a form of a stereo camera, features were extracted from a set of stereo images and correlated to reconstruct the 3D structure of the environment. The motion of the camera was then estimated by aligning the 3D reconstructed points with the locations they are observed in at another pose. While their method was slow at the time, it showed that ego-motion estimation was possible and could be used reliably when navigating a scene. Many modern visual odometry methods still use similar methods of motion estimation that are routed in the method outlined here.

The term "Visual Odometry" was first introduced in the literature by [40] who noted the similarities between the concepts of visual odometry and wheel odometry. While they proposed methods for obtaining ego-motion for both stereo and monocular cameras, one of their major contributions was the implementation of RANSAC [41] for outlier rejection. This contribution made their method significantly more robust than those which it built upon [42, 43].

One of the most famous uses of VO in recent times has been its use on the Mars rover missions of Spirit and Opportunity [44] where the two rovers used VO to map their trajectories over the

surface of the dusty planet. While this is an extreme example of deploying VO for autonomous vehicles that are unable to use GPS, it demonstrates the capabilities of a good VO system. By using a stereo camera with a significant baseline, the rovers can obtain good depth estimates which, along with other methods, are used to obtain trajectory estimates over paths that are hundreds of meters long. This is a good example of the potential for VO in an unstructured environment and parallels can be drawn to justify its deployment in agriculture which, being more structured, should yield comparable results if not better.

A common trend among all the aforementioned methods of VO is that they rely on matching features between images in order to estimate their movement as well as 3D position. While feature matching is a mature field that has been researched extensively, types of feature extractors and feature descriptors are areas that are still undergoing much change. Features such as Harris Corners [45], SIFT [6], SURF [46] and ORB [47] are all dependent on descriptors which may not match an object if the camera's relative position changes.

The authors of [30] present a method for monocular SLAM that does not use image descriptors to solve the problem of descriptor drift and mismatching due to the change in the appearance of an object as a camera moves around it. Their proposed method instead utilises an enormous quantity of points that are obtained from the local maxima of an image curvature filter. This alternative approach with such a massive quantity of tracked points would not be possible at the real-time frame rates they achieve if descriptor matching of features was used. Instead, they match their local maxima of curvature of one frame to the next after predicting the movement using dominant flow estimation. While their method is presented as a SLAM and 3D reconstruction method, the authors claim that their motion estimation and trajectory estimation results outperform state-of-the-art SLAM methods, such as DSO [48], ORB-SLAM[19, 20], and LSD-SLAM [23]. Their method provides a valuable method of feature tracking without the use of descriptors and as such, it is explored in this work.

Modern open-source libraries such as ORBSLAM3 [20], LSD-SLAM [23], Mono-SLAM [49] and Kimera [29] provide the ability to perform visual odometry with a wide range of camera sensors ranging from simple monocular and stereo cameras to structured light and RGB-D cameras. These libraries are easy to implement and use when compared to what researchers such as [39] used while being able to run at near real-time frame rates. These libraries are significantly large and computationally intensive however, they are still able to provide high frame rates. They are also able to perform bundle adjustment, loop closure, and map merging (sometimes on-the-fly) which significantly boosts their accuracy.

Depth Estimation

6.1 Sparse Methods

6.1.1 Stereo Depth Estimation

Stereo cameras have been around since the 1830s [14] and the mathematics of estimating depth from them has remained the same ever since (see Section 2.1.2 in the Preliminaries for an explanation of the sensors). Research in modern times has however been concerned with bettering the methods of matching features and parts of images to determine the disparity.

These algorithms have progressed from the simple corner detection and matching of Harris [45], to increasingly complex feature extractors, descriptors and feature-matchers. However, fundamentally the principle has remained the same. For a review of historic stereo matching algorithms and their progressive development, the reader is referred to the work by Lazaros et al. in [50] as well as that of Bleyer [51].

The method proposed by [52] uses image segmentation to take advantage of the way that depth discontinuities and image segmentation boundaries both are biased to occur at the edges of objects. This approach allows them to accurately estimate the depth of regions that have little to no image texture by assuming it is represented by a planar surface of the same object. While this method is valuable in structured, planar environments such as cities, it does not hold well in unstructured outdoor environments such as rural and agricultural regions.

Semi-Global Block Matching (SGBM) was proposed by [53] and has become the baseline against which all other stereo disparity estimations are measured. The method works by initially calculating a piecewise matching cost along the epipolar lines of the rectified stereo images. After identifying the points with a low matching cost, they calculate the directional cost along several paths radiating outwards in different directions from the point of interest before penalising any jump discontinuities. For example, with four paths the algorithm would calculate directional costs by travelling up, down, left and right from the point. By aggregating cost along these lines as well, the algorithm prevents any mismatches as a result of noise in the first matching cost step. The resulting best-matched disparities for each pixel are chosen by selecting the index of the lowest cost from the aggregated directional costs. The final step is to post-process the disparity with a uniqueness function and disparity validation step. This results in an algorithm that was state of the art at the time of release and is widely used in stereo disparity calculations (such as those implemented by OpenCV [2]).

Other approaches such as that of [54] use a convolutional neural network to compute the matching cost at each position for all the disparities that can be considered. This matching cost is

used along with several post-processing steps such as cross-based cost aggregation, left-right consistency checks and smoothness constraints using semi-global matching. While their post-processing steps are not novel, they demonstrate how combining these with a learned method can produce good estimates of disparity that achieved state-of-the-art performance at the time as well as having almost half the error of a census-matched stereo algorithm. This emphasises the importance of post-processing for stereo-depth algorithms for error rejection, smoothing and regularisation.

6.1.2 Lightweight Regularisation of Sparse Depth Estimates

While various method of post-processing of stereo disparity has been shown to significantly improve the performance, adding steps to any process will increase the time it takes to run. In order to keep the framerate the same, either the algorithm as a whole needs to be simplified or the hardware capabilities need to be improved. Since post-processing steps such as uniqueness functions, left-right consistency and photometric checks are all algorithms that are performed on a near-every-pixel basis, they require many calculations for large images which, in turn, requires time to compute. If the number of iterations could be reduced, it would reduce the additional time required by regularising steps. In this subsection, alternative methods of regularising sparse disparity estimations is explored in the literature.

Work such as that of [55, 56] have shown there is benefit in representing sparse disparity estimations as meshes. Since meshes allow for even and smooth interpolation, the result is sparse disparity estimates can be efficiently interpolated to produce a dense disparity estimate. While this achieves a somewhat equivalent output to dense methods such as SGBM and census matching, this approach is significantly more lightweight to compute. The approach of obtaining disparity from matched features is, however, prone to errors due to mismatched descriptors. As such, outliers and noise need to be regularised.

In their method of FLaME [55], the authors argue that every-pixel disparity prediction and dense reconstruction algorithms greatly oversample their environment since many points are redundant and never used. As an alternative, they propose a method to produce a lightweight monocular disparity estimation and 3D mesh reconstruction that works well on resource-constrained platforms. The proposed method results in sparse, keyframe-based disparity estimation that is refined as the monocular camera obtains alternate views of the environment. For their sparse disparity, they use a mesh constructed from matched features. This mesh is then regularised using Non-Local Second Order Variational Cost (NLTGV$^2 - L_1$) to reject outliers and smooth the result before joining it to the global mesh map of their environment. Their method keeps this optimised 3D surface available at the current pose making this research particularly valuable for mobile robotics. Their reconstruction algorithm can run at an extremely high framerate (230Hz) on CPU only due to the way that they pose the reconstruction problem as a non-local variational optimization over a time-varying Delaunay graph of the scene geometry.

Their work, What's best for my mesh [56], builds on the regularisation work presented in [55] and performs a survey of different methods of regularisation. They compare the results of TV, TGV, LogTV and LogTGV on a stepped, piecewise planar surface as well as angled planes to determine which method is most faithful after regularising a noisy mesh. This work outlines the significance of the choice of regulariser as well as demonstrates the performance of popular convex and non-convex regularisers on a Delaunay triangulation.
Both of these works show promising results and suggest that a mesh regularised disparity estimation may be able to perform equivalently to an SGBM disparity estimation while potentially being significantly faster (especially in the case of FLaME's method)

6.2 Learned Depth Estimation Methods

6.2.1 Introduction and Advantages of Learned Methods of Depth Estimation

Stereo depth estimation is a mature field that has been shown to produce accurate results[57]. The pitfall of stereo methods is that they require a rigid baseline between the two cameras and, especially for far-field depth estimation, the size of the stereo sensor becomes significantly large due to the increased baseline. When considering that the use-case of this research is size weight and power (SWaP) constrained vehicles, the size and weight of sensors are rigid constraints on the system that need to be minimised where possible. Additionally, stereo cameras that are not placed in rigid, monolithic housing are at risk of losing their calibrated orientation due to impacts when working in real-world environments.

With the advent of convolutional neural networks (CNNs) combined with significant improvements in computational capabilities, it became possible to estimate the depth from images without using geometric techniques [58, 3, 59, 60, 4]. Initially, CNNs were applied to computer vision problems such as reading handwriting, segmentation and object detection. CNNs performed well at these tasks as they were able to learn the set of convolutional filters that extracted the relevant information from the input images. Given the shortcomings of traditional, geometric depth estimation in stereo cameras, it was a natural step to try to use data-driven learned methods to predict the stereo disparity and subsequently the depth.

One of the first networks to show that depth predictions from data-driven methods could rival that of traditional stereo was by Eigen et al.[3] showed that at inference time, the network could provide accurate depth predictions from a single monocular image. Subsequent contributions to state-of-the-art learned methods by [59, 60, 61, 4, 62] resulted in methods that are equivalent to or, in some cases, outperformed the results achieved by traditional stereo techniques.
These depth prediction networks have the advantage that they are independent of a baseline or additional sensors, only being dependent on the quality of the input image and the quantity of training data. Therefore, the physical size and weight of the sensor equipment required for obtaining the 3D layout of the surrounding environment only needs to be that of the monocular camera used to obtain an image. As a result of using these networks, size-constrained vehicles can achieve depth prediction accuracy rivalling that of unconstrained vehicles that can facilitate large stereo camera setups.

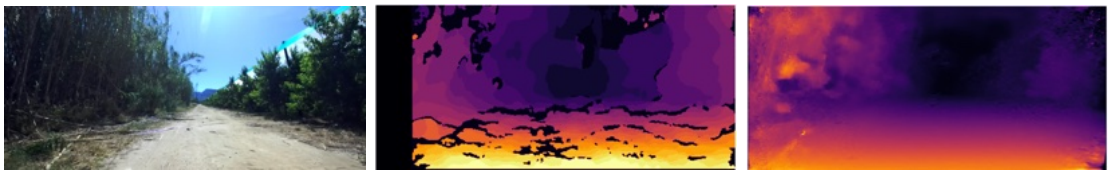
An additional benefit of learned methods is that different monocular cameras do not require individual spatial calibration, unlike stereo cameras. Provided the input images are rectified, the same depth prediction network can be used on different cameras (with the same resolution and focal length).

6.2.2 Supervised Depth Networks

As with all learning, human, animal or machine, feedback is required to understand performance and therefore improve. The supervision signal provides the correct outcome that should have been predicted for a given input. By knowing what the correct answer is, the error between the prediction and the correct outcome can be quantified. Once the error is understood, changes can be made to the method of prediction to improve the accuracy in future.

In the realm of machine learning, supervised depth prediction networks rely on a labelled depth map as the ground truth which is used to quantify the accuracy of estimations before backprop-

agating the corrections required to improve the result. Sources for these supervision signals for depth estimation include direct depth measurement sensors such as lidar, simulated environments captured by simulated cameras, or by using depth estimation methods such as stereo SGBM disparity.



(a) Input image

(b) Supervision depth signal from stereo camera

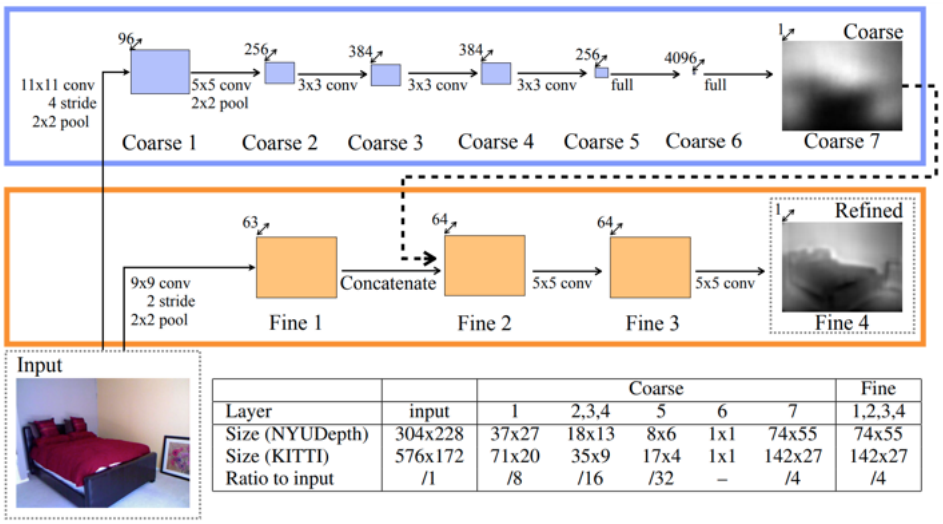
(c) Output depth prediction (after training)

Figure 6.1: Example of the supervision pipeline for a depth prediction network

One of the first methods that demonstrated that depth can be accurately estimated from a single image using learned methods is the work of Saxena et al. [58]. The authors proposed a network to predict the 3D structure of an image by using a Markov Random Field (MRF) to infer the 3D location and orientation of small homogenous patches of an image. These patches function as superpixels which are used pieced together as small planes to reconstruct a 3D environment from the image. The parameters of each of these planes are predicted by a linear regression model that is trained offline on a set of laser scans of the environments. While their results are not comparable to modern state-of-the-art methods, they showed that learned methods can provide promising results at the time of publication. Their method assumes that the environment is made of planar structures and relies on the ground plane in the images to be horizontally aligned. As a result, the disadvantage of this method is that it fails to generalise to less controlled settings and predictions do not have the global context to produce accurate outputs as predictions are made in a local context.

In Eigen et al. [3] the authors present a new method that shows it is possible to predict depth using only a single RGB camera image as input by using two deep CNN network stacks. This image does not need to be orientated correctly as was the case with [58] and therefore can be used in less controlled applications. Their approach is to have two separate tasks performed by each of the networks. The first task is a coarse, global depth prediction while the second task is to locally refine the global prediction to resolve the depth of finer details. To train the combined network, the authors utilise ground truth labels supplied by the KITTI and NYU datasets, achieving an RMSE of 0.41m for the NYUDepth dataset and thereby definitively demonstrating the capabilities of CNNs for depth estimation. The key contribution, other than demonstrating performance rivaling stereo methods, is their model learns to predict depth directly from the raw pixel values and does not rely on hand-crafted features or initial over-segmentation of the image.

Liu et al. [63] present a deep convolutional neural field model for monocular depth prediction. They improve on the work of [3] by leveraging the strengths of conditional random fields (CRF) to augment the performance of a convolutional network. Their method is supervised by ground-truth depth maps supplied by datasets such as KITTI, NYU v2 and Make3D. These ground truth depth labels are obtained from lidar or using structured light sensors such as the Kinect camera. The presented method for their dense convolutional neural field network achieves better performance than [3] on the NYU v2 and Make3D datasets while results are comparable on the KITTI dataset. While their work provides a good alternative to the structure and method of [58], one weakness of their method is that the process of using superpixels to over-segment the image results in artifacts in their predicted depth maps.



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Figure 6.2: Architecture proposed by Eigen et al which showed the feasibility of CNNs for monocular depth estimates. [3]

DispNet, proposed by Mayer et al. [64] builds on the architecture set out in FlowNet [65] which features an encoder-decoder architecture but with long-range links between layers (also referred to as skip-connections). By using skip-layer connections the architecture prevents the typical over-compression and loss of detail that occurs due to the data bottleneck in standard encoder-decoder networks. The authors of DispNet provide two significant contributions. The first is that they show how synthetic datasets can be used to train depth prediction networks for real-world deployment. The second is their refinement and expansion of the FlowNet architecture to produce smoother disparity maps by adding convolutions to the decoding/expanding part of the architecture. Their improved architecture is named DispNet and provides the foundation for current state-of-the-art high-performance encoder-decoder networks (See Monodepth [4])

6.2.3 Un-supervised Depth Networks

The training of the monocular depth estimation networks described in Section 6.2.2 requires a substantially large amount of data, all with accurate depth labels. Obtaining these labels presents its own set of challenges. Firstly, sufficiently dense and accurate time-of-flight sensors such as state-of-the-art lidar sensors are prohibatively expensive. After the significant capital outlay to purchase a lidar, the pointclouds returned are often too sparse for use as a supervision signal without some form of densification algorithm being applied [33, 66]. In turn, these densification algorithms introduce their own set of errors to the supervision signal.

Secondly, if a stereo camera is used in place of a dense lidar, the baseline of the stereo camera used to obtain a disparity signal then becomes a significant limitation for the performance of the network. Without additional metrics, the performance of a network is limited by its supervision signal. If the supervision signal is inaccurate or limited by nature then one cannot expect the network to perform better. Stereo matching algorithms have improved thanks to techniques such as photometric loss verification, however, their performance is still unable to rival the accuracy of lidar sensors. By using stereo disparity as a supervision signal, the best performance possible from our network will still imitate the shortfalls of using descriptors or block matching to obtain disparity.

Given the difficulties with obtaining a dense, accurate and faithful ground truth depth signal,

Godard et al. [4] showed that a depth prediction network could be trained unsupervised with only stereo images as input. This has a marked advantage over supervised methods as stereo imagery is easy and cheap to collect in large quantities. Therefore, a larger training dataset can be leveraged to train a more robust and potentially more accurate network.

The architecture of MonoDepth [4] is loosely inspired by the skip-connections encoder-decoder network of DispNet [64]. Along with some modifications to the convolutional layers, the main contribution of MonoDepth is the introduction of an image similarity-based loss for unsupervised supervision. As a result, they are the first to show that an entirely unsupervised network can achieve state-of-the-art results after training on stereo images.
To calculate the loss during training, the predicted disparity ($\tilde{d}$) is used to reconstruct the left image from the right image by shifting the pixels of the right image by their predicted disparity. By calculating the similarity between the stereo left image and the reconstructed left image, an error for the predicted disparity can be calculated. The authors strengthen their loss calculation by also adding other costs for disparity smoothness and left-right disparity consistency. The resulting performance of MonoDepth was impressive enough to spur other researchers towards unsupervised depth prediction networks considering the benefits in performance as well as the ease of obtaining training data using stereo imaging.

A relatively different approach to the unsupervised depth prediction problem was presented in Unsupervised Learning of Depth and Ego-Motion from Video by Zhou et al. [60]. Their method utilises monocular video streams as the input and uses view synthesis as a supervision signal to train two networks simultaneously. Given a stream of monocular camera images, the network aims to synthesise a view based on the rest of the video stream. This is done using an estimated depth of an image as well as the estimated pose to warp nearby views to the target image. Since the two predictions from the networks are tightly coupled to the view synthesis error, the networks responsible for each respective prediction can both be trained simultaneously. However, at test time, they function as two independent networks and can be applied independently. By only requiring a single-camera video stream as the input to the network, this method has a substantial advantage in the ease of obtaining training datasets. Therefore, this network would be a good choice for applications where test data cannot be gathered using a stereo camera setup. The results for depth prediction were comparable to that of MonoDepth [4] which is a significant achievement however, for this reason, it was not explored in this work.

Pyramidal Networks

While the aforementioned networks constitute many of the state-of-the-art methods at the time of writing, the greatest disadvantage to deep-depth prediction networks is the computation demands required to run them. Networks such as [4, 60, 64, 3] and many others all require a GPU to run at a framerate that is appropriate for real-time depth prediction (>5Hz). Additionally, since they are such massive networks they require large amounts of RAM to store the network at run time which, on constrained platforms, leaves fewer resources for other processes.

Recent advancements towards real-time monocular depth prediction such as Pydnet [67] and PSMNet [68] have produced comparable results to massively-deep state-of-the-art methods as detailed in [69, 60, 4]

In their 2007 paper, A pyramidal neural network for visual pattern recognition, Phung et al. [70] proposed an architecture for classifying objects and patterns in images. They build on the concepts of local receptive fields as well as image pyramids to create a multilayered architecture

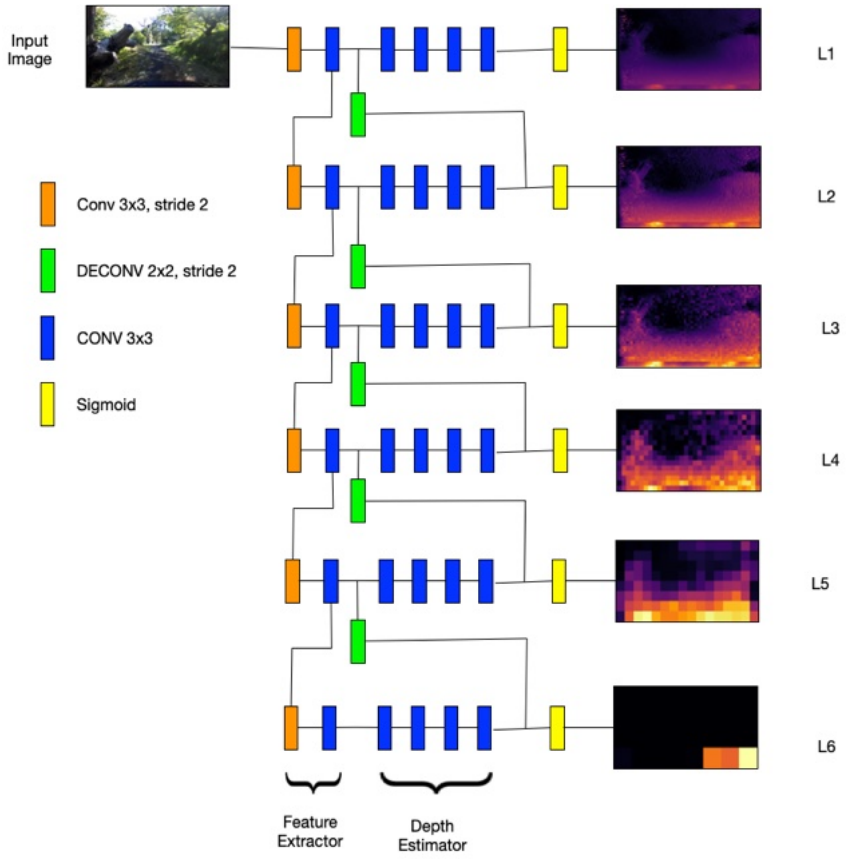


Figure 6.3: Pyramidal feature layout for Pydnet

with what they propose as 2D pyramidal layers. These layers can extract features and encode image data in a many-to-few manner, reducing the size of the image for later layers. By cascading these pyramids, they can compress the input image into something that can be fed into a 1D classification layer which their method uses to classify an output. Their results showed that the features and encoding provided by a pyramidal structure could provide sufficient information to accurately classify images such as the gender of faces which is their chosen test dataset.

While traditional convolutional networks gained much traction in the years following [70], they relied on being massively deep in order to have good performance. The work by Han et al. [71] on deep pyramidal residual networks showed that a pyramidal network could outperform the state-of-the-art CNN methods on the CIFAR-10 and CIFAR-100 image classification datasets. This performance was achieved while being significantly less deep (nearly 10x shallower than ResNet) which argues that the use of a pyramidal architecture provides a more efficient architecture.

Pyramidal Depth Prediction Networks

Pyramidal depth prediction networks build on the successes of pyramidal feature extractors in other fields where they have successfully been used for object visual recognition as well as optical flow estimation. The first majorly successful adaption to the task of depth prediction by Pydnet [67] which built on the unsupervised learning methods of MonoDepth [4] to learn depth from stereo imagery. Their method utilised much of the same training methods as well as very similar loss functions however, their network was completely different. Loosely inspired by the encoder-decoder architectures of the PWC-Net [72] and U-net [73], their method utilises pyramidal feature extractors in an encoder-decoder network with skip-connections. As a result of this change in network architecture, they achieve comparable results to Monodepth while

using only 6% of the total number of parameters. As such, their network runs at approximately 8Hz on a standard PC CPU.

The work of [22] sought to further speed up the training of Pydnet by further removing layers from the encoder-decoder architecture. They showed that the low resolution at the top layers of the pyramid were causing the depth predictions to lose fine detail. By removing two layers, they increased the accuracy of fine details while also significantly improving inference speed. A downside of this approach is that training the network became more challenging. Their solution was to include a proxy supervision signal in the training in the form of SGBM stereo disparity. The result is a network that outperformed the original Pydnet in both accuracy and speed.

The original authors of Pydnet later released Pydnet2 [5] which implemented the improvements of [22]. In addition, they proposed a random-crop training method instead of rescaling the images. This was done to prevent the loss of small details during the downscaling process as well as to improve learning on a global scale. Nevertheless, they claim a reduction in memory use of >2x as well as a near 100% increase in framerate over the original Pydnet.

Part II

Method

Overview of Methodology

In this chapter, we present the methods implemented and evaluated in the process of building our reconstruction pipeline. Building a pipeline that is lightweight in terms of computational, power and size requirements requires several different tasks to be optimised towards that aim. Our method was to split the task of reconstruction down into three separate tasks. Namely depth estimation of points for spatial awareness of the environment, estimation of the robot's ego-motion using visual odometry and finally placement of points into a digital 3D object. In this way, each task can be explored and evaluated to establish the approach that is most suited and will give the best appropriate trade-off between performance and accuracy.

We draw inspiration for the topography of the pipeline in this work from the works of [74, 55, 30, 31]. We attempt to implement a simpler version of a reconstruction pipeline compared to these works for two main reasons. The first is that a simpler pipeline is more computationally lightweight and as such, helps us achieve higher frame-rates on a standard CPU such that we achieve near real-time performance. The second is that a simpler pipeline lets us better evaluate the impact made by changes to each of the component tasks and therefore simplifying the task of choosing the most appropriate methods for reconstruction. This pipeline and its topography are introduced in Chapter 8.

Our research into various methods of depth estimation is presented next. For the purpose of this work, the task of depth estimation has been broken into two different categories based on the approach. The geometrically driven approaches using stereo depth cameras are presented in Chapter 9 while the data-driven, learned monocular methods are presented in Chapter 10. Thereafter, we present our research into the impact of feature selection as well as the impact of different depth estimation techniques on the accuracy of VO. This work on visual odometry is presented in Chapter 13.

After presenting each of the methods explored for each of the reconstruction tasks, an evaluation of their task-relative performance is presented at the end of each relevant section. This evaluation provides the results needed to make an informed selection of the methods to be used by the final reconstruction pipeline for the culminating evaluation of the reconstruction performance. The final reconstruction pipelines are evaluated in comparison to a ground truth reconstruction created using lidar scans and their performance is presented in Chapter 14.

Reconstruction Pipeline Topography

The pipeline we present in this work is minimal and lightweight with the intention to achieve the best possible performance at a high framerate running purely on a CPU. This minimal approach also ensures that the quality of the output reconstruction is directly influenced by the methods of interest and minimises the impact of variation in other processes in the pipeline. Methods to improve on the uncertainty of the visual odometry, such as loop closures and bundle adjustment, are trivial to include in the pipeline in future.

This pipeline topography is a simplified version of the pipelines used in state-of-the-art works such as [74, 55, 30, 31]. and its effectiveness is supported by those works. A high-level overview of the pipeline at a task level is shown in Figure 8.1.

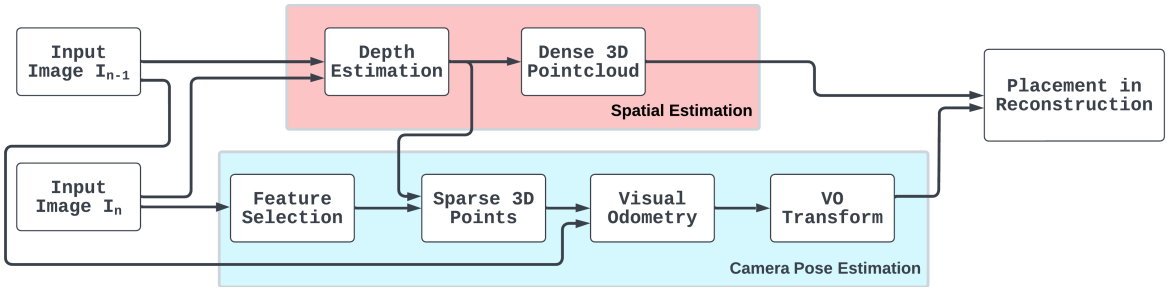


Figure 8.1: High-level overview of the reconstruction pipeline

The fundamental tasks of the pipeline can be broken into three categories. Spatial 3D estimation, pose estimation and reconstruction.

The first task can be more completely defined as the spatial estimation of the position of points in the surrounding environment relative to the vehicle at the current instance in time. Since the use of direct 3D position measurement with time-of-flight depth measurement sensors is prohibitively expensive, it is not feasible to deploy them in a low-cost setting. And since a method of reconstruction needs to be affordable for it to be adopted, costs need to be minimised where possible. The solution to reducing the cost of the system is to utilise sensors that are cheaper and can be used for other tasks as well. This leaves the option of cameras as they are relatively cheap (∼$100 [11]), versatile sensors that are often already equipped onto vehicles such as drones, ground-based rovers, and legged robots.

Cameras are the best choice of multifunctional sensors in this case as they can be used for a wide variety of tasks. The problem when determining the 3D position of points in the environment from a camera the following. When images are captured, a point anywhere along the same ray

through the aperture (in a pinhole model) will appear at the same location in an image. This means the 3D position of any point in an image can lay anywhere along that same ray. However, if we can determine the depth from the camera to the point in an image, we can calculate where along the ray the points lay and therefore determine the 3D location from a 2D image.
This, therefore, redefines the first task from one of estimating the 3D location of features in the surrounding environment to the task of depth estimation of pixels in an image.

Stereo cameras are a mature technology that can be used to efficiently estimate depth given a sufficiently wide baseline. As shown in the literature review section, recent advances in monocular depth estimation networks have also significantly reduced the computational requirements for depth estimation. Their accuracy has in some circumstances [3, 63, 4, 67, 5] even surpassed that of stereo algorithms. These two methods of determining the depth of a point in an image utilise two different approaches, namely geometric-driven stereo verses data-driven monocular networks. With these two approaches identified, the task at hand is determining which of them is most suited to a lightweight reconstruction pipeline for use in agricultural environments.

With the understanding of the position of points in the camera's coordinate frame, the second task is to estimate the camera's current motion and location relative to the global coordinate frame of the environment. Once the position of the vehicle is known in the global frame, the position of the points from the surrounding environment can be determined as well. This is needed so that we know where to place the 3D points obtained in the first task in the reconstruction.
Considering the choice to only use a camera on the end reconstruction pipeline, the motion estimation needs to run entirely from the images from the camera and without the use of any other sensors. If the camera provides images at regular intervals, the motion of the vehicle through the environment manifests as changes in the images since the view of the camera changes. Therefore, if we can accurately estimate the odometry from sequential images, we can integrate along the chain of transforms to determine the current position of the vehicle relative to the starting point. In practice, there is uncertainty in the odometry estimation, and this uncertainty accumulates over time. This means that the longer the vehicle travels, the more uncertain the position estimate becomes. This accumulated error is known as drift.

Once the positions of features are known in the global coordinate frame, the third task for the pipeline is placing them in the digital reconstruction. To place points in the 3D reconstruction, we utilise an off-the-shelf software package called Voxblox [75] which utilises a Truncated Signed Distance Field (TSDF) to incrementally integrates new information from each new frame into a larger volumetric map of the environment.

A TSDF is a way of discretising a volume into voxels where the value of each voxel is labeled by a distance to the nearest surface of an object. To add information to the TSDF, we first extract a coloured pointcloud from the current image and dense depth estimate. This pointcloud is then used to create a mesh which is regularised using one of the regularisers presented in chapter 9 to remove outliers. Then, using the current coordinates of the camera obtained from the visual odometry, this mesh can be placed in the correct location in the TSDF. To merge the mesh with existing surface data in the TSDF, we average the values for the voxel distance to the nearest mesh surface. For storage and later viewing the resultant mesh reconstruction is stored using the Polygon File Format or ".ply" so that it can be easily viewed using many mainstream 3D viewing software packages. The whole process of creating the reconstruction is computationally lightweight enough to be run in real-time onboard an embedded CPU.

Sparse Depth Estimation Methods

The work presented in this chapter evaluates two methods of stereo disparity. The first is the method of stereo SGBM. SGBM acts as a baseline of comparison for all depth estimates (for depth accuracy as well as visual odometry) presented in this work. This choice of baseline of comparison is because SGBM represents the standard of obtaining depth from stereo cameras in the field and as such, all methods presented need to outperform SGBM in order to contribute to the field of reconstruction.
The second method presented is our proposed mesh regularised method of obtaining dense depth estimates from sparse feature matches. We present our methods for mesh construction, interpolation as well as different variational approaches to regularisation which results in a smooth, dense and computationally light disparity method.

Both of these methods are used directly as depth estimates for the task of understanding the spatial position of points in the environment. This depth is what is used to determine the 3D position of points in an image before they can be placed in the digital reconstruction. Additionally, these methods provide the depth that is used in the process of visual odometry in order to determine where the robot is at a given moment in time.

9.1 Introduction to Stereo Depth Estimation

Stereo Depth Estimation is the technique of estimating the depth of a point from the difference in its position in the field of view of two cameras.
Generally, the two cameras are mounted in a manner such that their optical axes are collinear. With this setup, the disparity in the position of a point in the two images will be purely horizontal.

Stereo sensors are favourable over time-of-flight sensors such as lidar due to the cost of the latter. In recent years, the price of lidar sensors has fallen sharply and there have been significant advancements in the technology which has increased the density of lidar pointclouds substantially. However, the price of these sensors is still inaccessible for the majority of applications, especially when compared to stereo camera sensors. For comparison, a mid-range Velodyne Lidar such as the 16-beam Velodyne Puck costs around $4 600 at the time of writing [12] while a stereo camera such as the ZED 2 costs $449 [76]. As a result, stereo cameras are still a popular choice for depth estimation in many applications.

To calculate the depth of a point in a stereo image, the disparity of the point must be known. The disparity of a point is the difference in the horizontal position of the point in the two images. However, the two images need to be in the same focal plane to compare the horizontal

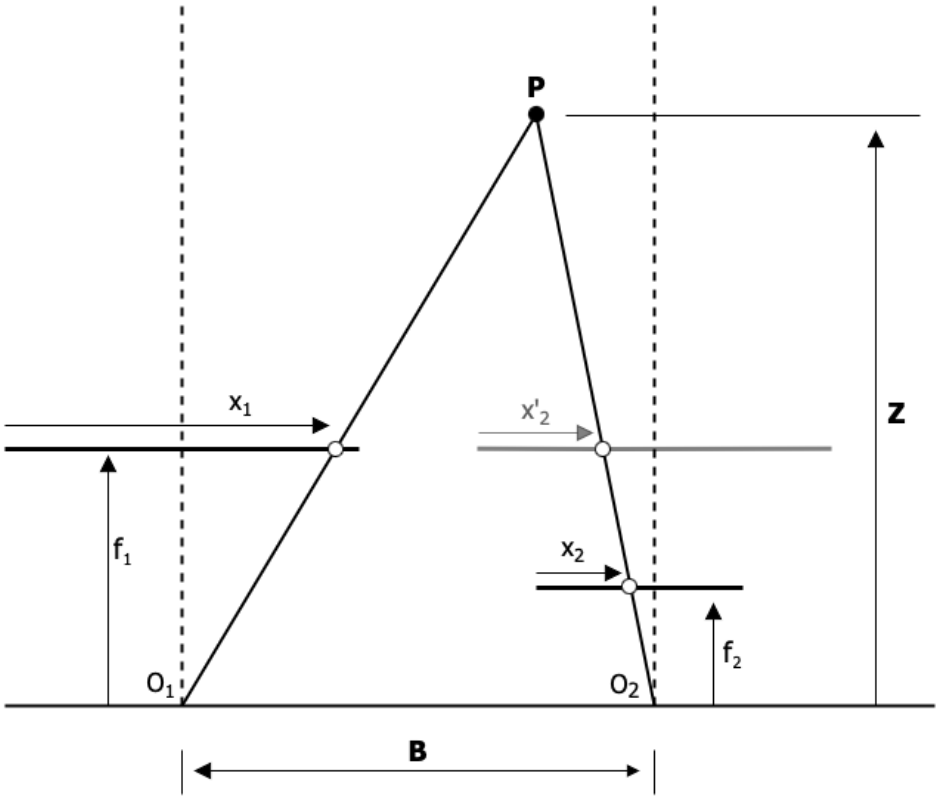


Figure 9.1: Diagrammatic view of a stereo camera featuring two different focal lengths.

position of the point.

In this simplified diagram, we need to project the position of the point on one image onto the same plane as the other image using the focal lengths. Convention dictates that we use our left camera as the reference frame and so we will project the point x_2 from the right image onto the same plane as the left image, now x_2'.

Using similar triangles,

$$\frac{x_2}{f_2} = \frac{x_2'}{f_1}$$

$$\therefore\ x_2' = \frac{f_1}{f_2} x_2 \tag{9.1}$$

Now, the disparity d of the point can be calculated as the difference in the horizontal position of the point in the two images. This is simply the difference between the position of the point in the left image, x_1 and the position of the point in the right image, x_2'.

Once the disparity of a point is known, similar triangles can be used to obtain the estimate of the depth since disparity is inversely proportional to the depth of a point. This assumes that the axis of the two cameras are parallel and that the baseline between the cameras is known. If the axis of the cameras are not parallel, the points on image planes will have to be projected onto a common plane. This process is referred to as rectification and is a common pre-processing step for stereo images.

In the diagram above, the two similar triangles of interest are given by

$$\triangle x_1 x' 2P \sim \triangle O_1 O_2 P \tag{9.2}$$

Since these triangles are similar, we can use the ratio of the sides to obtain the following relationship between the disparity and the depth of a point.

$$\frac{B}{Z} = \frac{B - (x_1 - x_2')}{Z - f_1} \tag{9.3}$$

If we simplify this equation by substituting $d = x_1 - x_2'$ and, solving for Z, we obtain the following equation for the depth of a point in a stereo image given its disparity.

$$Z = \frac{B f_1}{d} \tag{9.4}$$

Where B is the baseline between the two cameras and Z is the depth of the point.

Now that we have established the process to obtain the depth of a point in a stereo image given its disparity, there is one remaining question. How do we identify and match the pixels representing the same object in each image of the stereo image such that a dense disparity estimation can be produced? This is the task of stereo matching and is the focus of the rest of this chapter.

9.2 Semi Global Block Matching

Semi-global block matching is a well-established method of obtaining stereo disparity as set out in the foundational paper [53]. The implementation used in this work is from OpenCV [2].

In the implementation of this method, the parameters presented in Table 9.1 were hand-tuned to provide the best results on both datasets and their chosen values are as presented in the table.

Table 9.1: Parameters used for SGBM after optimising

Parameter	Weighting
Number of disparities on 720x1280px Image	128
Number of disparities 256x512px Image	52
blockSize	23
preFilterCap	2
uniquenessRatio	12
speckleRange	7
speckleWindowSize	10
disp12MaxDiff	6
minDisparity	1

9.3 Mesh Regularised Disparity

While many depth estimation techniques produce reliable results, there are times when estimations are inaccurate. For example, features of an image such as repeating objects and patterns can cause stereo-image features to be incorrectly matched, resulting in incorrect depth estimates. For our method to be reliable and valuable in real-world applications, it needs to be robust to outliers and errors in depth estimations.

Inspired by the findings and results of [56, 55], we propose a mesh regularised disparity that interpolates a sparse set of disparities to produce a dense disparity estimation. The choice of features used was $BRISK$ due to the speed at which features could be extracted and matched. The locations and disparities obtained from these feature matches resulted in an extremely sparse set of points, approximately 2 500 points as opposed to the 13 1072 pixels in 256x512px depth prediction image returned from a network.

The sparse set of depth estimates from the matched features was used as the supports for the tessellation of a Delaunay triangulation.

This mesh could then be regularised using total variational techniques such as those used in both [56] and [55]. The methods and definitions of these variational techniques are explained in the next section, Section 9.3.1

9.3.1 Regularising Methods for Mesh Regularised Disparity

Errors from depth estimation can take the form of small amplitude noise as well as large outliers. Both contribute unwanted artefacts, making estimations inaccurate and therefore less useful. The goal of a regularising function is to smooth noise without losing faithful depth estimates, thus yielding an accurate representation of the true surface. While simple in principle, this task is challenging due to the complexities of real-world environments. Small, thin objects, textures and curves all introduce difficulties that prevent simple solutions from providing satisfactory performances as the details for these environmental features would be lost.

Regularisation of images for noise reduction is an area of research that has been a subject of significant effort. Methods such as TV TGV NLTGV, NLTGV^2 etc. have been developed for this task and have shown good results when used to regularise a mesh [55, 56]. As such, we explore the uses of these for regularising our stereo mesh disparity.

Total Variation

The Total Variation norm is a useful metric for calculating the amount of variational energy in a signal. Intuitively, it can be explained as an L_1 norm of the derivatives of a signal [77]. When using variational methods of regularising, the aim is to minimise an objective energy functional ($E(f)$) which constitutes the following form:

$$\min_{\Omega} E(f) = \min_{\Omega} E_{smoothing}(f) + \lambda E_{data}(f) \tag{9.5}$$

Where f is the scalar functional $f : \Omega \to \mathbb{R}$ and the balance of influence between the data-fidelity term (E_{data}) and the regularising term ($E_{smoothing}$) being determined by the scalar

$\lambda > 0$ This formulation allows the regularising action to provide satisfactory smoothing while still preserving as many of the high-frequency components of the underlying desired signal as possible. This lets us remove noise while keeping fine texture details and thin/small objects.

A common choice for the data-fidelity term is the L_1 norm. This provides an outlier-robust measure where $z : \Omega \to \mathbb{R}$ is a noisy signal.
Therefore, we define E_{data} as

$$E_{data}(f) = \int_{\Omega} |f(\omega) - z(\omega)| d\omega \tag{9.6}$$

When defining the method of Total Variation for noise removal, [77] proposed the regularising term be defined according to

$$E_{smoothing}(f) = TV(f) = \int_{\Omega} |\nabla f| \tag{9.7}$$

assuming that the original data f_0 has been modified by some random noise n and A, a linear operator (such as blurring) such that $f = Au_0 + n$. This is defined under the assumption that the noise has zero mean.

Minimisation of the energy functional in the case of a mesh is different from the methods that would be used in an image. In our case, the edges formed between the vertices can be performed according to the primal-dual optimisation as proposed by Chambolle et al. [78]. A thorough and good explanation of the method of implementation is laid out in Pilbrough et al. [56] and we refer the reader to this work for the implementation details.

Total Generalised Variation (TGV)

Second-order Total Generalised Variation (TGV^2) has been shown to be a powerful regulariser with good trade-offs between computational complexity and accuracy [79], For TGV^2, an auxiliary function is introduced $\boldsymbol{\psi} \to \mathbb{R}^2$.
Thus, TGV^2 is defined as,

$$E_{smoothing}(f) = TGV^2(f) = \alpha_1 \int_{\Omega} |\nabla f(\boldsymbol{\omega}) - \psi(\boldsymbol{x})| + \alpha_0 \int_{\Omega} |\nabla \psi(\boldsymbol{\omega})| \tag{9.8}$$

Where $\alpha_0, \alpha_1 \in \mathbb{R}^+$ are scalar weighting parameters penalising jumps and second-order discontinuities respectively.
An important note is that $TGV^2(f) = 0$ if f is an affine polynomial (polynomial of order < 2). As a result, using a TGV^2 regularised tends to produce piecewise affine results.

Non-Local Total Generalised Variation (NLTGV)

Non-local total generalised variation is another powerful first-order variational method that incorporates global information through a regularisation term to improve upon TGV. NLTGV is defined as follows,

$$E_{smoothing}(f) = NLTV(f) = \int_{\Omega}\int_{\Omega} \alpha(x,y)|f(x) - f(y)|dydx \tag{9.9}$$

Where the function $\alpha(x, y)$ represents a weighting function that is used to strengthen the regularisation in large areas or those with ambiguous data terms.

Non-Local Second-order Total Generalized Variation (NLTGV^2)

NLTGV^2 introduces another auxiliary regularisation term to further improve upon the first-order NLTGV which leads to smooth piecewise affine results. The NLTGV^2 variational energy is defined by,

$$\begin{aligned} E_{smoothing}(f) = NLTGV^2(f) = \min_{\omega(x)\in\mathbb{R}^2} \int_{\Omega}\int_{\Omega} \alpha_0(\boldsymbol{x},\boldsymbol{y})|f(\boldsymbol{x}) - f(\boldsymbol{y}) - \langle \boldsymbol{\gamma}(\boldsymbol{x}), \boldsymbol{x}-\boldsymbol{y}\rangle d\boldsymbol{y}d\boldsymbol{x} \\ + \sum_{i=1}^{2} \int_{\Omega}\int_{\Omega} \alpha_1(\boldsymbol{x},\boldsymbol{y})|\,\gamma^i(\boldsymbol{x}) - \omega^i(\boldsymbol{y})|d\boldsymbol{y}d\boldsymbol{x} \end{aligned} \tag{9.10}$$

where super-scripts denote the vector components. i.e. $\boldsymbol{\gamma}(x) = (\gamma^1(x), \gamma^2(x))^T$ and the weighting functions $\alpha_0(\boldsymbol{x},\boldsymbol{y})$ and $\alpha_1(\boldsymbol{x},\boldsymbol{y})$ both ≥ 0

9.3.2 Construction of meshes

In order to create a mesh regularised disparity, motivated by the works of [55, 56], a mesh is needed. In this subsection, we describe the methods of creating a mesh which can be used to regularise the sparse disparity estimates from stereo matches and thus create a smooth stereo disparity estimate.

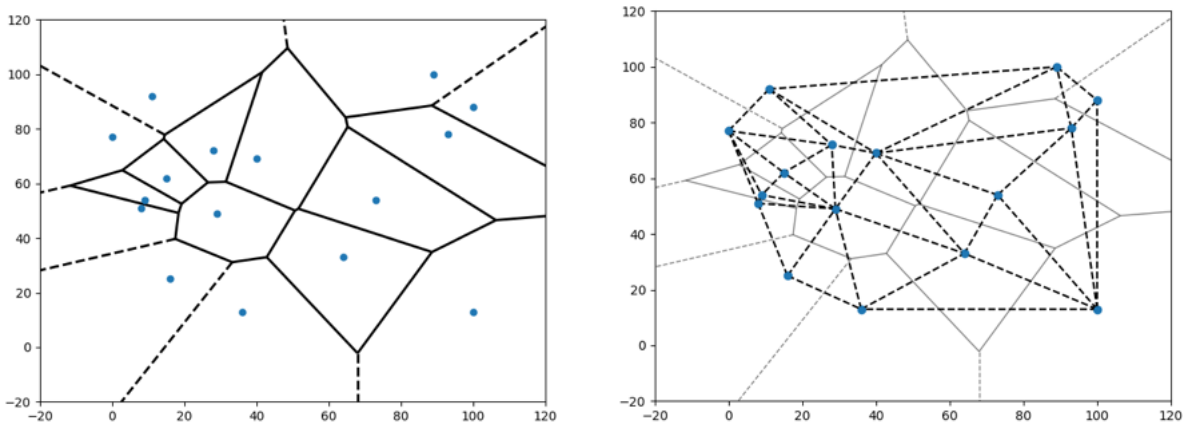
Definition of a Delaunay triangulation

We start with a set of points that will be used for vertices. Let this set of points be $\mathcal{V} = \{v_1, \ldots, v_n\}, N \geq 3\}$ where $v_1 = (x_1, y_1)$ and we assume that the points are not all collinear.

Now, if we set each of the points in $\mathcal{V}$ as the centres of a set of cells that all grow outwardly at the same rate until they touch the border of another cell, we will have a set of polygons where only the cells on the convex hull of $\mathcal{V}$ are still expanding. This set of cells defines the Voronoi Diagram.

The Delaunay Triangulation of a set of points corresponds to the dual graph of the Voronoi Diagram of the same set since the circumcenters of the triangles of the Delaunay Triangulation correspond to the vertices of the Voronoi Diagram. Therefore, the Delaunay triangulation can be calculated trivially from the Voronoi Tessellation.

For further information, [80] provides an in-depth discussion on two different algorithms for calculating the Delaunay triangulation of a set of discrete points.



(a) Veronoi graph using 16 points

(b) The Delaunay Triangulation constructed from the Voronoi Diagram in Figure 9.2a

Figure 9.2: Example of a Delaunay Triangulation

Extending meshes from the camera frame to the 3D robot frame

A mesh constructed using the u and v coordinates of image features can be extended to the estimated 3D location of the features in the world frame provided an estimate of the depth of the feature to the camera is known. Using the estimated depth, the camera-frame 3D coordinates $f_{cam} = (x_{cam}, y_{cam}, z_{cam})$ can be calculated from the image feature location $f_{img} = (u, v, d)$ as follows:

Given the camera intrinsic matrix K Where f_u and f_v are the focal lengths in the respective directions while $c_u$andc_v are the locations of the image centre and for our experiments axis skew $s \approx 0$,

$$K = \begin{pmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{pmatrix} \tag{9.11}$$

The coordinates of a point in the camera frame is given by,

$$x_{cam} = \frac{(u - c_u) \times d}{f_u}, \quad y_{cam} = \frac{(v - c_v) \times d}{f_v}, \quad z_{cam} = d \tag{9.12}$$

Transforming camera frame coordinates to the robot frame coordinates is then calculated with,

$$f_{robot}(x, y, z) = R_{cam}^{robot} \, f_{cam} \tag{9.13}$$

Dense Depth Methods

10.1 Unsupervised Deep Depth Networks

10.1.1 Monodepth

Monodepth network was proposed in [4] where the authors pose depth estimation as a method for re-projection to reconstruct one image from another given pair of stereo images during training. At training time, their method predicts a disparity d^r which would reconstruct the right image I^r of the stereo pair when given the input, left image I^l from the stereo pair. By training on rectified stereo images, their network produces a disparity estimation that can be used to recover the predicted depth using the typical depth formula for a rectified stereo image. Namely,

$$depth = \frac{bf}{d}$$

Given that the baseline for the training images b and the focal length of the cameras f is known.

Monodepth achieved state-of-the-art results on the Kitti dataset when it was released, outperforming all depth prediction methods at the time of release. Additionally, although the network is massively deep, it executes quickly enough to be used for CPU-only predictions (at the cost of high memory requirements due to a large number of parameters). For these reasons, Monodepth is used in this work as a benchmark standard for the performance of a dense learned method. This gives a good point of comparison for lighter networks while also evaluating the feasibility of using the Monodepth network in a reconstruction pipeline as presented in this work.

Structure of the network

Monodepth draws some inspiration for their architecture from Disp-Net [64] but expands on the convolutional encoder-decoder structure, deepening the network significantly. Monodepth's encoder-decoder architecture also utilises skip connections (see [81]) from their encoder's activation blocks to enable higher-resolution details to be resolved.
See Figure 10.1 for a graphical representation of the architecture of Monodepth. In addition to the encoder-decoder network, the authors design the network to produce two disparity estimations simultaneously from the same left input image during training, see Figure 10.2. This double-view disparity is used to enforce the left-right mutual consistency of the output to improve the quality of the output predictions.

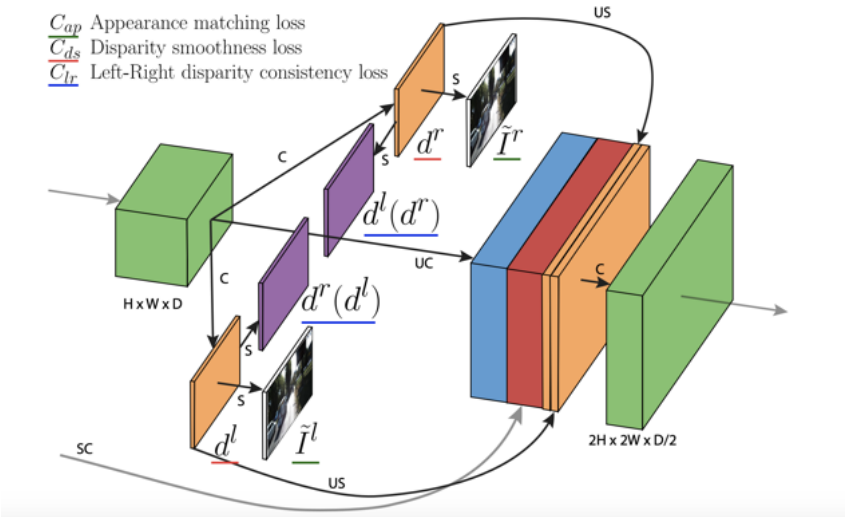


Figure 10.1: Graphic showing the flow of data through the architecture of the Monodepth network. [4]

Training Loss

The training loss refers to the metric used to determine the performance of the network with respect to the target. In the case of Monodepth, the training loss is determined by the following equation

$$C = \sum_{s=1}^{4} \alpha_{ap}(C_{ap}^l + C_{ap}^r) + \alpha_{ds}(C_{ds}^l + C_{ds}^r) + \alpha_{lr}(C_{lr}^l + C_{lr}^r) \tag{10.1}$$

Where l_ap is the appearance matching loss, l_ds is the disparity smoothness loss an l_{lr} is the left-right consistency loss and α defines a scalar weighting for each particular loss.

These losses are defined as follows:

Appearance Matching Loss
The appearance matching cost or photometric image reconstruction loss C_{ap}^l is described by the equation:

$$C_{ap}^l = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM(I_{ij}^l, \tilde{I}_{i,j}^l)}{2} + (1-\alpha)||I_{ij}^l \tilde{I}_{ij||}^l \tag{10.2}$$

Where $\alpha = 0.85$ and SSIM is simplified with a 3×3 block filter. I_{ij}^l represents the input image while the image reconstructed from the predicted disparity and the right image is given by $\tilde{I}_{ij}^l$. The number of pixels is given by N.

Disparity Smoothness Loss
Encourage smooth disparities by penalising prediction gradients that do not correspond to image gradients ∂I.

$$C_{ds}^l = \frac{1}{N}\sum_{i,j} \partial_{i,j}^l |\, e^{||\partial_x I_{ij}^l||} \; + \; \partial_y d_{i,j}^l |\, e^{-||\partial I_{ij}^l||} \tag{10.3}$$

Where ∂_x and ∂_y are the partial derivatives in the image/prediction x and y axis respectively.

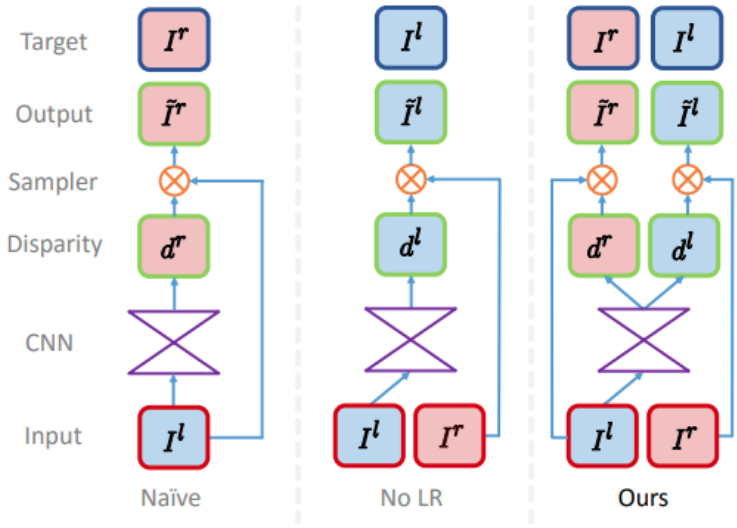


Figure 10.2: Comparison of different view enforcement strategies for input images when training. From left to right - Naive (only left image), left-right without enforcement, Monodepth Training Left-Right mutual view enforcement. [4]

Left-Right Disparity Consistency Loss

A left-right disparity consistency loss term is added to the cost function to improve prediction accuracy. By penalising differences between the predicted left image disparity and the right image disparity map (projected onto the left view using the predicted left disparity), the authors enforce congruency between two predictions that should represent the geometric scene. An L_1 distance measurement is used for the difference.

$$C_{lr}^{l} = \frac{1}{N} \sum_{ij} |d_{ij}^{l} - d_{ij+d_{ij}^{l}}^{r}| \tag{10.4}$$

Cost for Right Image Disparity Map

The loss calculation is mirrored and repeated for all of the output scales for the right disparity map.

10.2 Pyramidal Networks

10.2.1 Pydnet

Structure of the Pydnet network

Inspired by the encoder-decoder networks of [82, 3, 83, 63] , the authors of Pydnet [67] presented a network that used pyramidal feature extractors that were previously used in object detection to predict disparity between two stereo images. Their use of pyramidal features allows for faster execution times with lower computational requirements. An additional benefit of their proposed architecture is that size of the training images does not have to be the same size as the images used at inference time.

They utilise the same methods presented in [4] to formulate depth as an image reconstruction problem.

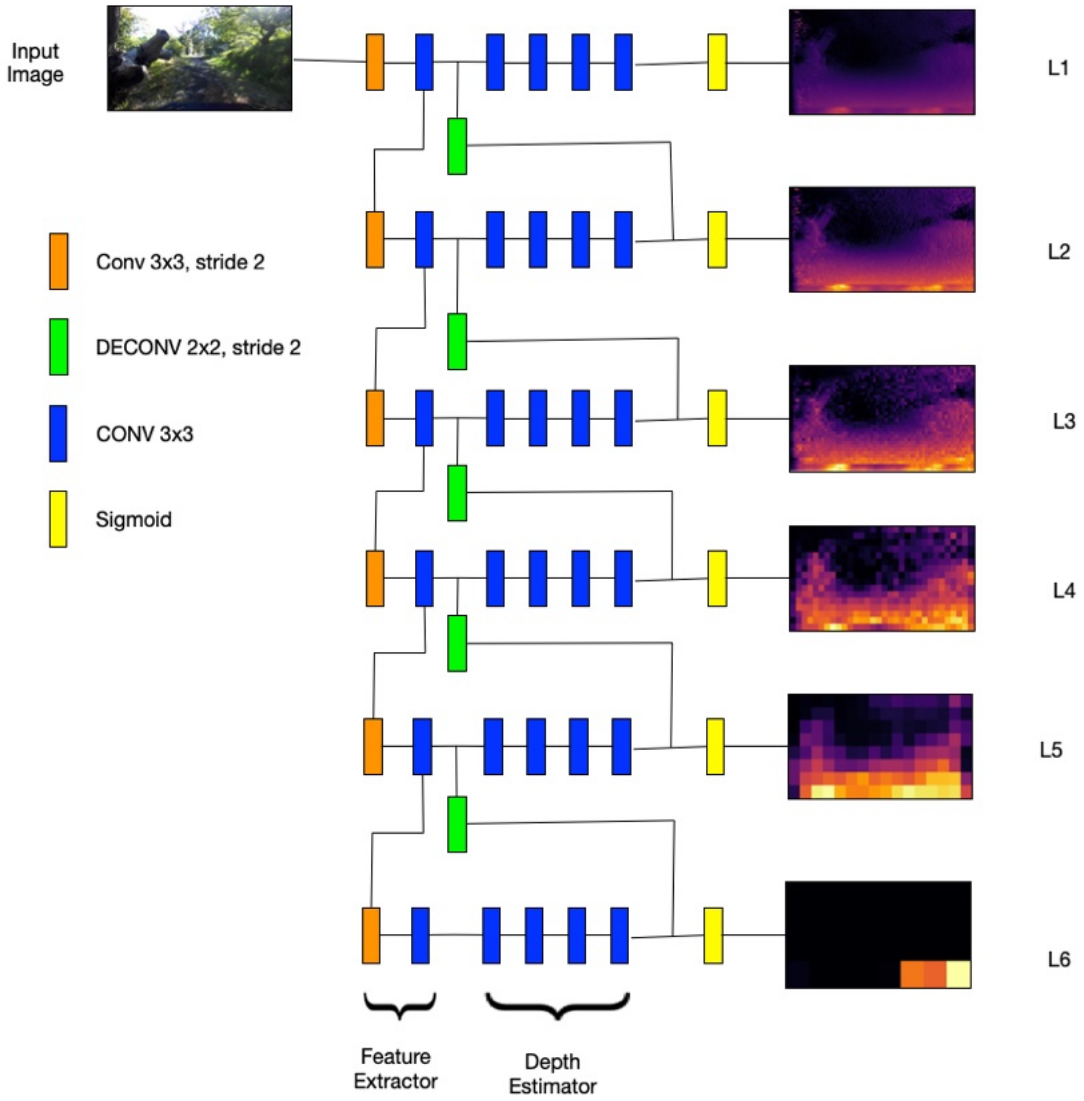


Figure 10.3: Figure showing the architecture of the Pydnet network

Pyramidal feature extractor

At the first level of the pyramid, two consecutive convolutional layers act on the input image with a stride of 2 and 1 respectively. Each of the convolutional layers deploys 3×3 kernels and is followed by a leaky ReLU with $\alpha = 0.2$. This results in an output half the size of the input image which is then fed into the second layer of the pyramid which contains two of the same convolutional layers as before. This is repeated up until the 6th, and highest, level of the pyramidal feature extractor which has an output image size of $\frac{1}{64}$ of the original input image size. By having a small resolution at the highest level of the pyramid, the authors ensure that the extracted features include global context in their predictions. Higher resolution outputs at lower resolutions of the pyramid allow for the refinement of details while the *coarse-to-fine* strategy employed by linking the layers reduces the number of parameters significantly and consequently reduces the memory required by the network.

Upsampling and Depth Decoding

Extracted features from each level of the Pydnet pyramid are decoded by a 4 layer convolutional depth decoder. At the highest level (and lowest resolution) of the pyramid, the output of the

depth decoder has two functions.
The first is producing a disparity map prediction by passing through a sigmoid operator.
The second use of the decoded features is concatenation with the features extracted by the next lower level of the pyramid. The decoded features are first processed by a deconvolution layer (2×2 and stride of 2) which doubles the spatial resolution of the decoded features which is the same resolution as the lower layer. The concatenated features are then processed by the new depth decoder of the lower layer. This output is used to predict the depth of the current layer and is also upsampled and passed to the next layer as the process is repeated until the lowest layer (and therefore the highest resolution).

For more details about the architecture of Pydnet, we refer the reader to their contribution [67].

Training cost

To train Pydnet, we minimise a loss function composed of three terms:

- Photo-consistency loss, $l_p(x)$
- Left-right consistency loss, $l_{lr}(x)$
- Disparity regularization term, $l_r(x)$

Each of these loss terms is defined at each image scale $s \in [1, \cdots, 6]$

Therefore, the loss function can be written as:

$$\begin{aligned} L_s = &\sum_{i=0}^{6} \alpha_p(l_p(I_l^i, \hat{I}_l^i) + l_p(I_r^i, \hat{I}_r^i)) \\ &+ \alpha_{lr}(l_{lr}(D_l^i, D_r^i) \\ &+ \alpha_r(l_r(D_l^i) + l_r(D_r^i)) \end{aligned} \tag{10.5}$$

Here $D_l : \Omega \to \mathbb{R}$ represents the disparity map that warps the right image I_r to the left image I_l. Similarly, we let D_r represent the disparity map that warps the left image to the right image.

The first term representing the image reconstruction error or photometric error l_p, measures the difference between the input left image I_l and the image warped using the estimated disparity $\hat{I}_l$. Using the approach presented in [4] we define l_p as a combination of structural similarity(SSIM) and a simple L_1 pixel error:

$$\begin{aligned} l_p(I_l, \hat{I}_l) = &\frac{1}{N} \sum_{u \in \Omega} \alpha \frac{1 - SSIM(I_l(u), \hat{I}_l(u))}{2} \\ &+ (1 - \alpha)||(I_l(u), \hat{I}_l(u)|| \end{aligned} \tag{10.6}$$

Here N is the number of pixels and $\alpha > 0$ controls the weighting between the SSIM and L_1 terms.

A pure photometric error results in predicted disparities that produce very good image similarity however that does not necessarily produce an accurate disparity. The left-right consistency check

l_{lr} enforces coherence between predicted left D_l and right D_r disparity maps. The addition of this term results in more realistic disparity predictions.

$$l_{lr} = \frac{1}{N} \sum_{u \in \Omega} |D_l(u) - D_r(u + D_l(u)| \tag{10.7}$$

The disparity smoothness term, l_{ds}, discourages disparity discontinuities according to an L_1 penalty unless a gradient ∇_I occurs on the image.

$$\begin{aligned} l_{ds} = \frac{1}{N} \sum_{u \in \Omega} & |\nabla_x D(u)| e^{-||\nabla_x I(u)||} \\ & + |\nabla_y D(u)| e^{-||\nabla_y I(u)||} \end{aligned} \tag{10.8}$$

10.2.2 Proxy Supervision of Pydnet

Proxy supervision aims to influence the training of the network using a known accurate/ground-truth signal, hastening training and impressing structure from the proxy supervision signal into the prediction, while still allowing the unsupervised loss metrics of the network to guide the training [22, 5]. The goal of this method of training is that the unsupervised metrics will allow the resulting network to outperform the proxy supervision signal while the proxy signal will speed up training and allow the user to promote desirable behaviors by careful selection of the signal.

In this work, two proxy supervision signals were investigated. Both are based on stereo depth estimations. The first is the traditional Semi-Global Block Matching algorithm which has been proven to provide reliable predictions. The second is a Mesh-Regularised Disparity which resulted from our research in the use of barycentric coordinates to regularise meshes. The implementation of this signal is as presented in Section 9.3.

Modifications to Pydnet Network

Training Loss

To train PydnetProxy, we minimise a cost function that features the original three terms but includes a proxy supervision term $l_{px}(x)$.

Therefore, the loss function L_s can be written as:

$$\begin{aligned} L_s = & \sum_{i=0}^{6} \alpha_p (l_p(I_l^i, \hat{I}_l^i) + l_p(I_r^i, \hat{I}_r^i)) \\ & + \alpha_{lr}(l_{lr}(D_l^i, D_r^i) \\ & + \alpha_r(l_r(D_l^i) + l_r(D_r^i)) \\ & + \alpha_{px}(l_{px}(D_l^i, \check{D}_l^i)) \end{aligned} \tag{10.9}$$

Where $D_l : \Omega \to \mathbb{R}$ still represents the disparity map that warps the right image I_r to the left image I_l. Similarly,D_r represents the disparity map that warps the left image to the right image. We introduce $\check{D}_l$ to denote the proxy disparity signal that is used as a supervisory term.

The proxy supervision loss l_{px} is calculated using a backwards Huber loss [84] as shown below in 10.10:

$$l_{px}(D_l^i, \check{D}_l^i) = \frac{1}{N} \sum_{i,j} Huber(D_{i,j}^l, \check{D}_{i,j}^l, c) \tag{10.10}$$

Where:

$$Huber(D_{i,j}^l, \check{D}_{ij}^l, c) = \begin{cases} |D_{i,j}^l - \check{D}_{i,j}^l| & \text{if } |D_{i,j}^l - \check{D}_{i,j}^l| \leq c \\ \frac{\check{D}_{i,j}^l - \check{D}_{i,j}^l|^2 - c^2}{2c} & \text{otherwise} \end{cases} \tag{10.11}$$

To determine the most effective proxy weight (α_{px}), an ablation study is performed on the zoo dataset to establish which weight provides the most accurate depth estimates. The results are shown in Table 10.1.

Table 10.1: Results of Ablation study of proxy disparity weighting after evaluating on the old zoo dataset

Method	Metric	Pydnet Disparity Weighting				
		0.1	0.3	0.6	0.8	1.0
Mesh Pydnet Disparity	$t_{rel}[\%]$	**38.76**	38.86	40.13	39.14	41.34
	$r_{rel}[^o/50m]$	83.86	**82.40**	84.50	82.79	83.37
	ATE $[m]$	26.65	13.72	21.67	**12.82**	17.37
	RPE $[m]$	**0.3834**	0.3962	0.3986	0.4123	0.4127
	RPE $[deg]$	3.085	3.079	3.079	**3.079**	3.079
SGBM Pydnet Disparity	$t_{rel}[\%]$	42.21	**42.16**	44.05	45.75	44.78
	$r_{rel}[^o/50m]$	**78.37**	79.69	80.07	79.87	78.94
	ATE $[m]$	16.58	**14.49**	17.45	23.43	19.47
	RPE $[m]$	0.6956	**0.6912**	0.705	0.7229	0.7333
	RPE $[deg]$	3.244	**3.241**	3.245	3.248	3.245

From the results from the proxy weight ablation study, the best weighting for SGBM supervision was selected as $\alpha_{px} = 0.3$ while a weighting of $\alpha_{px} = 0.8$ was chosen for the mesh regularised disparity. An interesting trend to note is how most of the errors progressively increase with increased SGBM proxy weighting after a weighting of 0.3 implying that the appearance-driven metrics used by Pydnet are more accurate than the SGBM signal. This hypothesis is tested in the comparison of the depth estimation accuracy for all the methods which is presented in Chapter 12.

10.2.3 Pydnet 2

After the release of Pydnet in 2018, in 2020 Greene et al. [22] showed that the use of a proxy supervision signal improved the performance of the Pydnet network which was the inspiration for our work in the previous section towards investigating the effects of performance in an unstructured agricultural environment. In 2022, Poggi et al. [5] confirmed the results of Greene et al. while also proposing their new, improved version of Pydnet2. The method of implementation of this network is discussed in this section.

Changes in Architecture

In their paper proposing Pydnet2, [5], the authors showed that a 4-level version of Pydnet ran 1.2 times faster than the original Pydnet and 13 times faster than Monodepth. They argue that the low-resolution feature maps produced at the high levels of the Pydnet pyramid produce a prediction that is too coarse. This coarseness subsequently propagates through to lower levels of the pyramid. As a result, the fine detail predictions are blurred by the coarse features which reduce the overall accuracy of the network. By removing these pyramid levels, they remove 13 of 41 convolutional layers from the network. This greatly reduces the number of parameters to 700k which is 36% of Pydnet[67] and 2% of Monodepth[4]. By corollary, the memory requirements of the network as well as the inference time are greatly reduced. The authors found their implementation was 40% quicker than the original Pydnet.

This results in an architecture as shown in Figure 10.4. Note the changes between Pydnet2 and the original Pydnet.

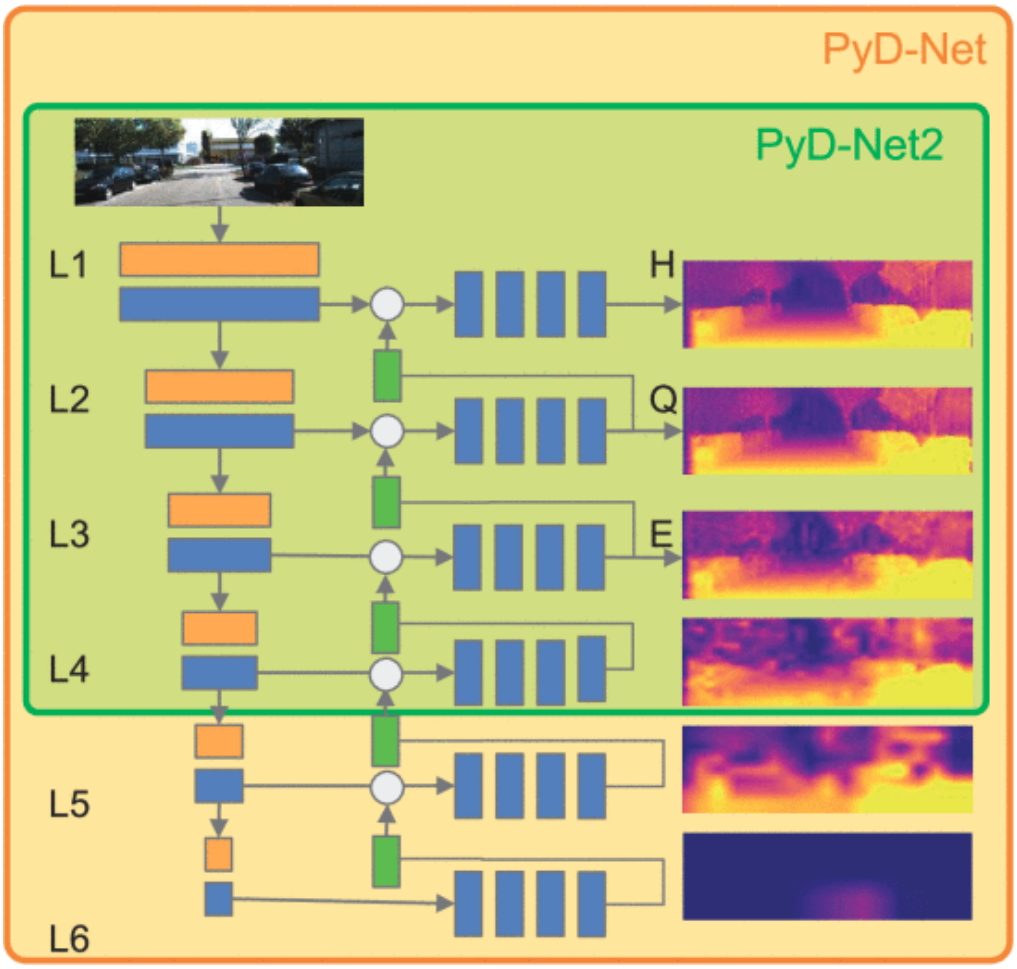


Figure 10.4: Figure from Pydnet2 [5] showing the difference in the architecture of the Pydnet2 network versus Pydnet1

Modifications to The Training of Pydnet2

In [67], during the training of the Pydnet network, training images are scaled down to the network input size. In the full images, a region of pixels on the left of the input image will have

no overlap with the right image. Therefore, there's a region of no ground truth disparity in the network during training. Methods such as flipping the pair of training images horizontally aid in reducing the impact of this region by switching the area of no overlap to the right of the image but these methods do not solve the underlying issue. Additionally, when using scaled images, training occurs on a fraction of the potential input image resolution. Therefore, a large portion of the details of the input image are lost during downsampling leading to lower-quality input. Simply put, garbage in, garbage out.
Instead, by randomly cropping the stereo images input to the network during training, the network is trained on higher levels of detail. This means that, at inference time, depth predictions should capture smaller features and objects in the image with significantly greater accuracy while still preserving the global scale and accuracy.

Training Loss

For the training of Pydnet2, the new network inherits the image similarity error metric approach that was presented in the original Pydnet. The loss function is a combination of image re-projection (Equation (10.6)), disparity smoothness (Equation (10.8)), left-right consistency (Equation (10.7)) and a proxy supervision term (Equation (10.10)).
The result is a training loss function that is the same as described in the proxy-supervised Pydnet presented in Section 10.2.2. The difference is that the proxy supervision weight $\alpha_{px} = 1$.

After accounting for the reduced number of layers as well as the change to the proxy weight, the loss for Pydnet2 is described by

$$\begin{aligned} L_s = &\sum_{i=0}^{4} \alpha_p (l_p(I_l^i, \hat{I}_l^i) + l_p(I_r^i, \hat{I}_r^i)) \\ &+ \alpha_{lr}(l_{lr}(D_l^i, D_r^i) \\ &+ \alpha_r(l_r(D_l^i) + l_r(D_r^i)) \\ &+ l_{px}(D_l^i, \check{D}_l^i) \end{aligned} \tag{10.12}$$

Where the proxy supervision signal ($\check{D}_l$) is the SGBM disparity.

10.3 Implementation Details

For the implementation and training of the different learned methods, all the networks had the following implementation details in common All networks were implemented using Tensorflow APIs [85]. They were trained on each of the two datasets (see Section 11.2) for evaluation in the unstructured Rhodes Zoo environment as well as an Agricultural environment.

When training, images were resized to 256x512px using a bilinear downsampling This downsampling from the original 720x1280px images was used to improve the execution speed. Higher input resolutions will provide more detailed depth predictions however the marginal improvement does not justify the significant requirements of memory and computation that are presented by larger inputs. Similarly, smaller images rapidly become too coarse for depth prediction of small objects. For all training, data augmentation was performed by utilising random horizontal flips as well as shifts in colour, gamma, and brightness. Each augmentation occurs with a

50% chance and the values for shifts in contrast, brightness, and gamma are randomly assigned from a uniform distribution. The values for the colour shift are in the range [0.8, 1.2] for each respective channel while gamma is in the range [0.8, 1.2] and brightness is [0.5, 2.0].

All networks were trained using two NVIDIA 2080 RTX GPUs for a duration of 50 epochs in batches of 4 using Adam [86] optimisation. The parameters used for Adam are $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$

10.3.1 Monodepth

Monodepth is a significantly deep network to train with a total of 31M parameters and therefore takes approximately 36hrs to train on the Lindenhoff Farm dataset consisting of 75k images. The initial learning rate for Monodepth is set to 1×10^{-4} halving after 30 epochs and then again after 40 epochs.

Table 10.2: Weights used for training Monodepth for 50 epochs

Parameter	Weighting
Photo-consistency α_{ap}	0.85
Left-right α_{lr}	1.0
Disparity smoothness α_{ds}	0.1
Batch Size	8
Number of Epochs	50
Initial Learning Rate	1e-4
Number of Parameters Trained	31.6M

10.3.2 Pydnet

Six pyramid levels were used in the version of Pydnet for training and testing. This was done in keeping with the experimental practices of the authors [67]. This strategy provides fast predictions while giving good depth prediction performance. Increasing the number of layers does not improve depth prediction accuracy sufficiently to offset the cost in computation speed or memory as it rapidly becomes a diminishing return. Further increasing the size of the input images and therefore the resolution of the predictions was shown in [67] to not improve depth prediction performance significantly as it only increases the small details of the depth prediction. Depth prediction performance is largely impacted by the accuracy on an object level (trees in an image) and the depth of small features (such as the leaves on the trees) does not affect the depth map accuracy significantly.

For training Pydnet, the parameters for the loss function in Equation (10.5) were set as follows:

10.3.3 Pydnet Proxy supervised

Modifications to the Pydnet network were performed as described in the method description. An ablation study was performed to determine the best weighting for each of the SGBM and mesh disparity proxy supervision signals with $\alpha_{px} \in 0.1, 0.3, 0.6, 0.8, 1$

Table 10.3: Weights used for training Pydnet for 50 epochs

Parameter	Value
Photo-consistency α_{ap}	0.85
Left-right α_{lr}	1.0
Disparity smoothness α_{ds}	0.1
Batch Size	8
Number of Epochs	50
Initial Learning Rate	1e-4
Number of Parameters Trained	1.9M

The results of the ablation study are presented in Table 10.1. The best proxy weighting was used to evaluate the performance of the networks.

The training protocol was modified to include proxy supervision images during training. Since no modifications were made to the layers of Pydnet, the network was used in the same manner as Pydnet at test time.

10.3.4 Pydnet2

Table 10.4: Weights used for training Pydnet2

Parameter	Value
Photo-consistency α_{ap}	0.85
Left-right α_{lr}	1.0
Disparity smoothness α_{ds}	0.1
Proxy supervision weight α_{px}	$\{0.5, 0.8, 1\}$
Batch Size	8
Number of Epochs	50
Initial Learning Rate	1e-4
Number of Parameters Trained	700K

In implementing Pydnet2, we used the same training protocol as Pydnet. Our implementation followed the methodology outlined in [5], removing the top two levels of the pyramid and connecting the output of the 4th level of the coder to the decoder. This is as shown in fig. 10.4.

Additionally, trained the network first using random crops, following the methods proposed in [5, 22]. When this failed to produce usable results (training was divergent), we also attempted to train the new network using the same methods as Pydnet, using scaled images as opposed to random crops.

When performing the random crops for training, we used input images at their full resolution of 720x1280 px which were then cropped to the same input size of 256x512px with the cropped pixel coordinates matching in the left and right images. This was done to ensure that the left and right images were still aligned after cropping.

When the results from the random crops were not usable, we attempted to train the network using scaled images as was done in the original Pydnet. This was done by first scaling the images to 256x512px in the data preparation step, as was done for the aforementioned networks. While this did not produce usable results, the results were not as divergent as the random crops.

Instead, the results mainly featured noisy and scattered disparity predictions. In addition, the network predicted that the majority of the pixels in the image had a disparity of less than 5 which, when evaluated, results in depths that are too large to be feasible.

Proxy supervision was then attempted using SGBM disparity in the same manner as Pydnet Proxy supervised. The addition of a proxy supervision signal caused the original Pydnet to converge more quickly, however, the results from the pydnet2 network were irregular and mainly represented the outlines of shapes in the image when evaluated. Therefore, this also failed to produce usable results.

In future endeavours to implement the Pydnet2 network, we would recommend a meticulous focus on the way that the network is shortened. While the authors of Pydnet2 [5] state that their architecture simply removes the top two levels of the pyramid, we believe that there is either an error in our implementation, or that there are additional changes that were obvious to the authors but not to us.

This is unfortunate as the results presented in [5] show that the Pydnet2 network is significantly faster than the original Pydnet while also producing superior results. This would have been a significant advantage in the unstructured agricultural environment where the speed and size of the network are significant factors in the usability of the network.

Data Collection

Before we can evaluate the depth estimation methods presented above, we need to collect data that can be used for training and testing. This chapter presents the datasets used for this work as well as the platform used for data collection.
We present two separate datasets to ensure that the data is representative of real-world environments where the methods developed in this work will be used. The first dataset is collected in a semi-structured environment where the methods can be developed and tested. The second dataset is collected in an unstructured agricultural environment where the methods can be tested in a more realistic setting.

11.1 Platform for data collection

In order to gather data and test the methods presented in this work, a ground-based vehicle is needed to travel the environment and host sensors. This vehicle needed to be small enough to fit between rows of crops while being rugged and robust enough to handle the offroad driving conditions of the floor between crops. For these reasons, a Clearpath Husky A200 Unmanned Ground Vehicle was used as the platform. A photo of the vehicle at the UCT Zoo is shown in Figure 11.1

Figure 11.1: Clearpath Husky A200 fitted with sensors used for data collection and testing

The vehicle was equipped with a powerful Mini-ITX computer which has performance that is equivalent to a mid to high-tier laptop.

Type	Mini-ITX Single board computer
Processor	Intel i5-520M 2.4GHz
RAM	8GB DDR3
GPU	None
Storage	1TB SSD

Table 11.1: Computer Specifications onboard the Husky A200 used for data collection

The platform is equipped with a dense 32-beam, 360°Velodyne Lidar, a 720x1280px ZED stereo camera and Garmin GPS, all of which enabled the collection of data for this work.

Data was recorded at 10Hz for the Lidar and 15Hz for the stereo cameras and stored using ROS bag files for processing later. This choice of data storage ensured that all data was timestamped and could be time synchronised appropriately to ensure data fidelity. The images recorded were stereo images which could be used for stereo algorithms as well as for the training of the monocular networks. During the testing of monocular methods, the left image was used and the right image was discarded.

11.2 Datasets

11.2.1 Semi-structured Outdoor Environment

Rhodes Zoo - University of Cape Town

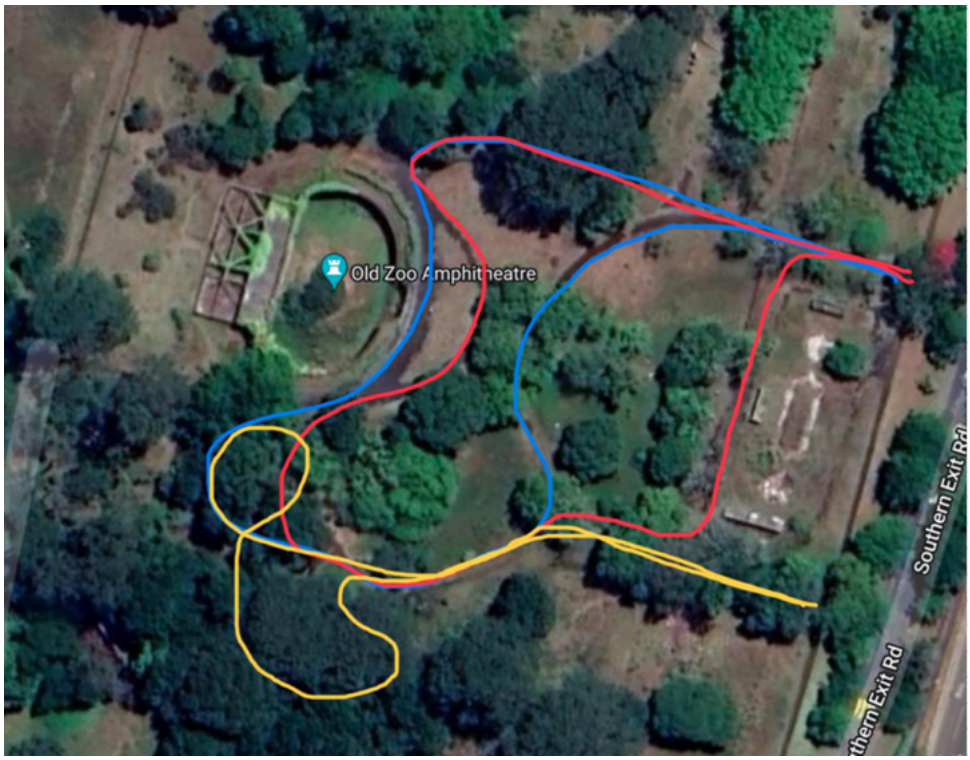


Figure 11.2: Outline of routes at the Rhodes Old Zoo superimposed on a satellite image. Blue is Route-A, Red is Route-B and Route-C is yellow.

The unstructured environment of the old Rhodes Zoo at UCT was used as the test and proving ground for the development of the algorithms presented. It consists of a semi-urban wooded area with asphalt paths for the robot to travel along. A few small buildings are present, distributed amongst the trees, shrubs, and grassy regions. The environment is 100m North to South by 300m East to West.

The UCT old zoo was identified as the best location for testing within close proximity to the UCT Mechatronics Laboratory. Other environments around the university campus were evaluated for suitability when testing in an outdoor environment but this was the most appropriate. The mixture of unstructured, natural vegetation with semi-urban features such as paved footpaths and buildings provided a dataset that was useful for identifying the strengths and weaknesses of algorithms as they were developed.

(a) Image showing a narrow section of the path with nearfield objects.

(b) Image showing an open area of the environment.

(c) Another example of the unstructured part of the environment.

(d) Example of the buildings obscured by trees and foliage.

Figure 11.3: Images as examples of the environment at the Rhodes Old Zoo where data was collected.

11.2.2 Unstructured Agricultural Environment

Lindenhoff Farm

While the UCT old zoo was useful for developing and testing algorithms, the goal of this work is to produce a pipeline that is useful for agricultural robots. Therefore, real-world testing on agricultural regions needed to be performed. Two farms were visited for testing, both being completely different in terms of the environments that they present. Data from these farms was used as the primary test case to determine the performance of the systems.

Lindenhoff Farm is situated in Paarl, Western Cape, South Africa and as a result, represents a typical stone-fruit farm that can be found in the Cape region. The methods of cultivating fruits are based on trellised orchards where trees are actively trained and managed. The majority

Figure 11.4: Annotated Google Maps satellite image of the Lindenhoff Farm. The different crops are noted as follows: orange and blue represent nectarines under shade. The magenta shows a vineyard of Chennin Blanc grapes.

of their stone fruit (nectarines and plums) are grown under shade cloth to protect the trees and fruit from hail and high winds, giving a higher quality yield for the export market. This is representative of most high-end farms in Southern Africa where exported fruit is more lucrative.

The data collected at Lindenhoff was split into two parts. One for training and validation, which occurred in the nectarine orchard annotated by the orange overlay in Figure 11.4. The test data was obtained in another nectarine orchard (blue overlay). The magenta overlay represents the vineyard that can be used to test how the methods used generalise to an unseen crop. The training portion of the dataset consisted of 75k stereo image pairs collected in the nectarine and plum training orchards as well as on the roads between the two. The test dataset consisted of 5k stereo images taken on the route through the test nectarine orchard. Examples of images from Lindenhoff Farm dataset can be found in Figure 11.5.

(a) Nectarine orchard used for training

(b) Farm infrustructure

(c) Nectarine orchard used for testing

(d) Dirtroad data past different vegitation showing shadow occlusions

Figure 11.5: Sample images showing the environment at the Lindenhoff Farm.

Evaluation of Depth Estimation

To evaluate the error of the disparity estimations, the laser scans from the Velodyne lidar on the husky robot were used as a ground truth. The problem is that the disparity estimates and the lidar points are both in different coordinate frames. To correct for this, the set of lidar points p_l^t at time t were transformed to the image frame using the transformation matrix T_{lidar}^{cam}. This transformation was obtained during lidar-camera calibration of the system. The (u, v) pixel coordinates of the projected points in the image frame were then used as locations to sample the dense disparity predictions. After converting the ground truth depth into a disparity measurement, the pairs of points could be compared and the error metrics for the method of disparity estimation could be calculated.

When comparing the errors in the estimates, it is tempting to compare the two in terms of depth. However, this is misleading as the real output of a stereo camera is in terms of disparity between the left and right images. Therefore, to accurately compare the performance, the depth of the ground truth needs to be converted into disparity such that it represents what a 'perfect stereo camera' would measure. Only then can we benchmark our camera based methods. Using a depth measurement for error introduces bias to the error since at far distances, a small change in disparity results in a large change in depth. This relationship between depth and disparity is shown in Figure 12.1 better illustrate this effect, take the example where an error of 2px of disparity occurs on our system where images are 1280px wide.The error in depth between disparities of 2 and 4 is 15.8m whereas the difference between disparities of 20 and 22 is 0.3m. This difference skews depth as a metric for error and as such, we use disparity for error evaluation in this work.

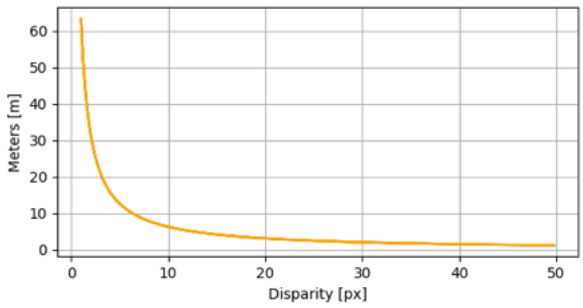


Figure 12.1: The depth-disparity curve for the system used in this work featuring a baseline of 12cm and a focal length of 531mm.

12.1 Qualitative Disparity Estimation Results

Qualitative analysis of the results gives insights which results in a table lack the dimension to represent. While the error metrics we present in the quantitative analysis provide a good understanding of the overall accuracy of the methods, the qualitative results presented in this section show the smoothness, completeness and appearance of the disparity estimations. The ideal output is one which provides smooth estimates while also maintaining information about small details in the image. The result should not have any missing data or erroneous artefacts. A densified projection of the lidar ground truth disparity is also presented alongside the estimates to give an approximation of what a sparser ground truth should yield.
All disparity images presented are color mapped such that the brighter colours are larger disparities and, therefore, part of the foreground. Conversely, the darker colours are further away from the camera.

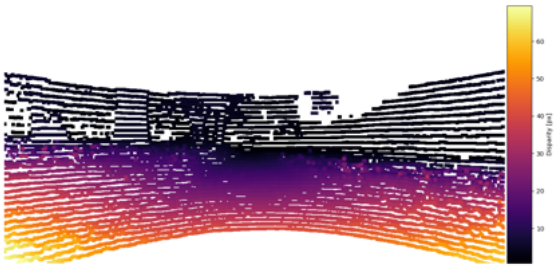
We present examples of the disparity estimates produced by the methods described in this chapter, both geometrically-driven and data-driven. The disparity maps for input images from the UCT Zoo as well as the Lindenhoff Farm are presented with the learned-methods having been trained on their respective datasets. While the two environments are similar in terms of being outdoors, their content is significantly different. Therefore, the selection of datasets are complementary and give different insights into the performance of the disparity estimation methods presented in this work.

12.1.1 Qualitative examples at the UCT Old Zoo

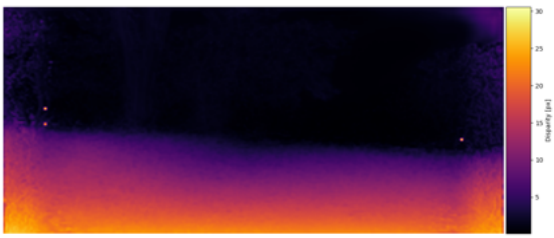
The first place the disparity estimation methods were tested was the UCT Old Zoo. This location provided an semi-structured environment so that the performance of the methods on organic as well as constructed geometries could be evaluated. More in-depth information about the dataset is given in Section 11.2.

For the purposes of disparity evaluation, a separate route was held back from training of the networks such that their performance could be fairly evaluated. This particular test set included many lens flares which occluded parts of the image, thus making disparity estimation particularly challenging. The reasoning for selecting this data for testing is to ensure that worst-case performance on this environment is known. Examples of estimated disparities for Frames 0 and 500 are shown in Figure 12.2 and Figure 12.4 respectively.

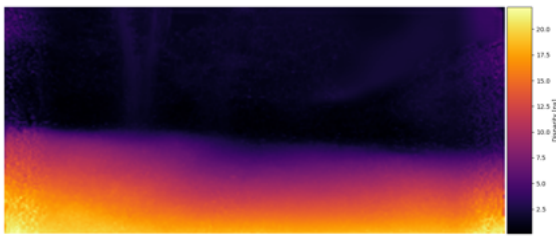
(a) Input image



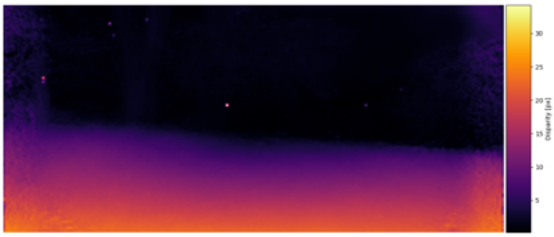
(b) Ground Truth from Lidar



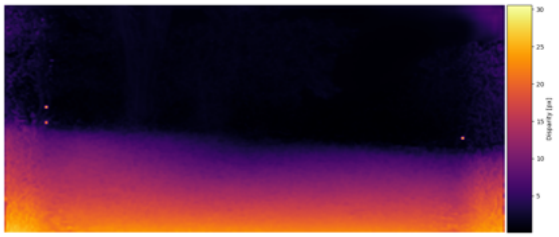
(c) Monodepth trained for 50 epochs



(d) Pydnet trained for 50 epochs

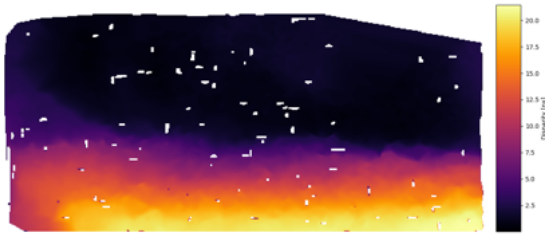


(e) Pydnet SGBM proxy supervised trained for 50 epochs

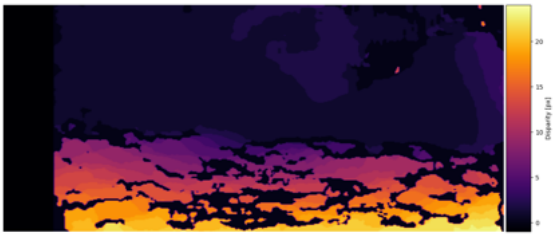


(f) Pydnet Mesh Disparity proxy supervised trained for 50 epochs

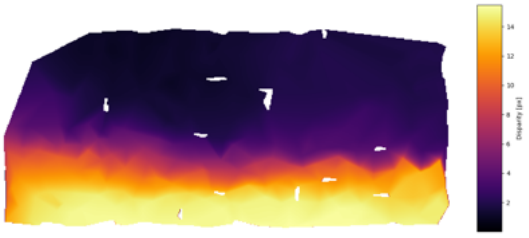
(g) SGBM stereo disparity on full-sized images



(h) Mesh regularised disparity on full-sized images



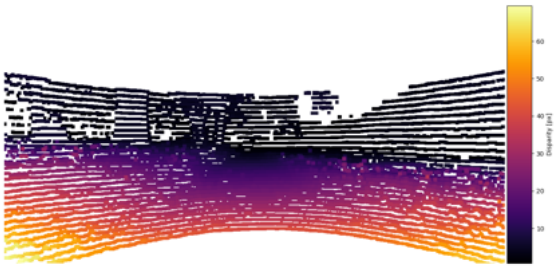
(i) SGBM stereo disparity on small images



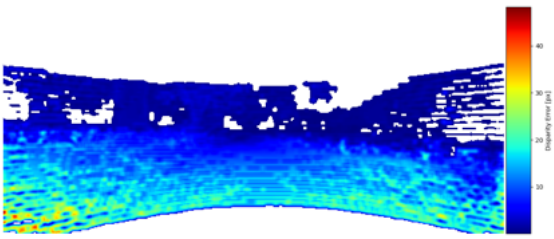
(j) Mesh regularised disparity on small images

Figure 12.2: Qualitative depth estimates at the UCT Old Zoo dataset for Frame 0.

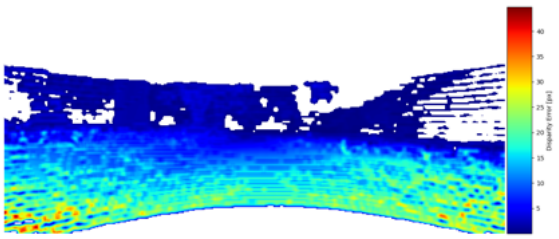
(a) Input image



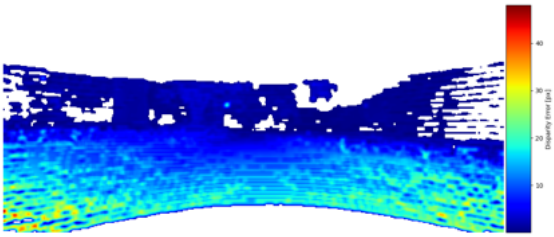
(b) Ground Truth from Lidar



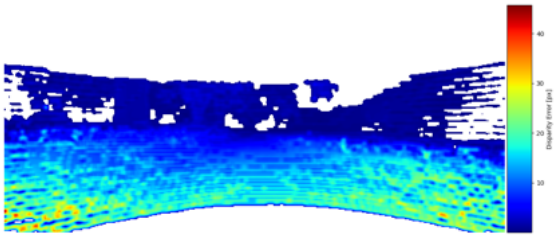
(c) Error Map: Monodepth trained for 50 epochs



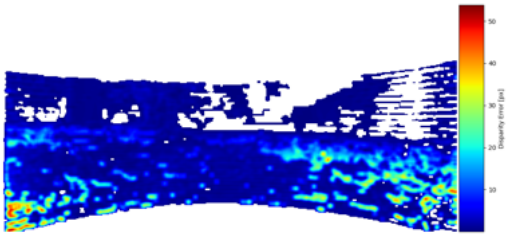
(d) Error Map: Pydnet trained for 50 epochs



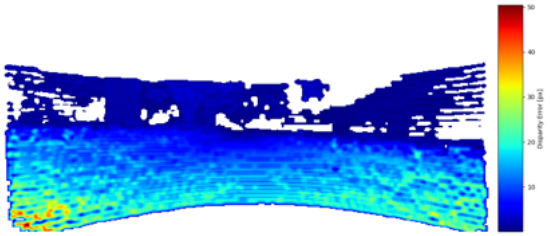
(e) Error Map: Pydnet SGBM proxy supervised trained for 50 epochs



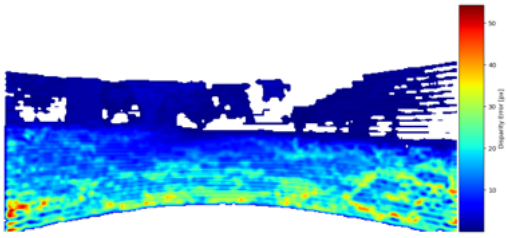
(f) Error Map: Pydnet Mesh Disparity proxy supervised trained for 50 epochs



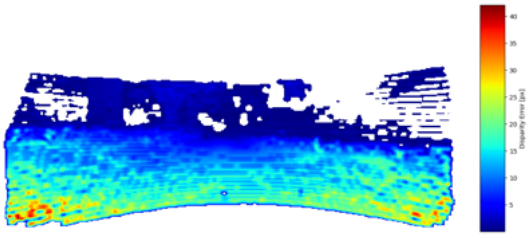
(g) Error Map: SGBM stereo disparity full-sized imgs



(h) Error Map: Mesh regularised disparity full-sized imgs



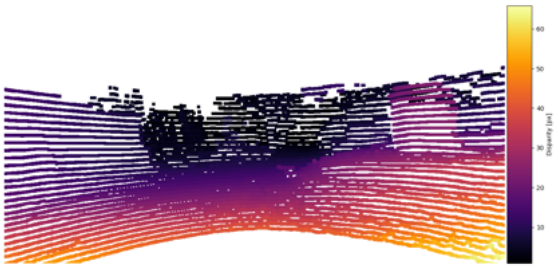
(i) Error Map: SGBM stereo disparity small imgs



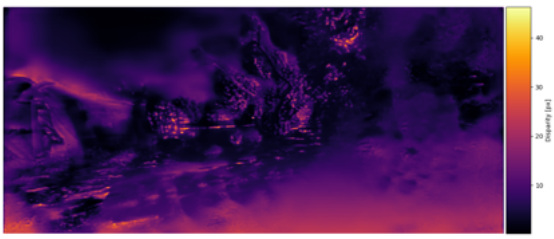
(j) Error Map: Mesh regularised disparity small imgs

Figure 12.3: Error Heatmaps for the disparities estimated on Frame 0 of the Old Zoo dataset. See Figure 12.2.

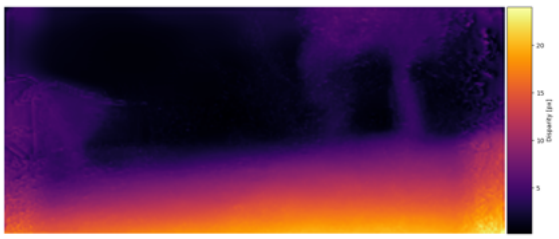
(a) Input image



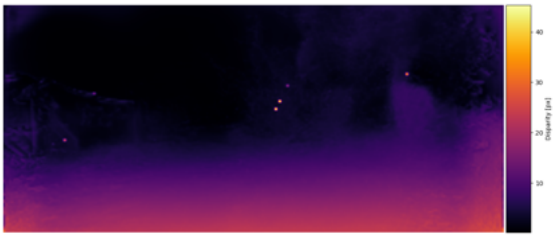
(b) Ground Truth from Lidar



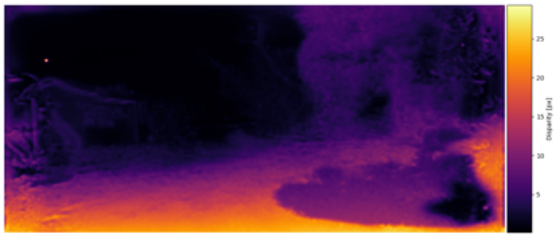
(c) Monodepth trained for 50 epochs



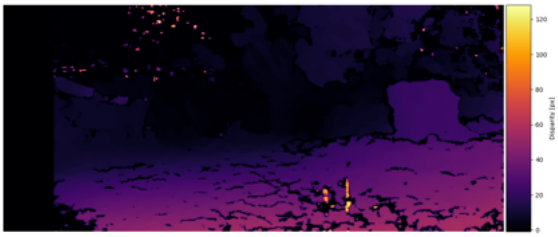
(d) Pydnet trained for 50 epochs



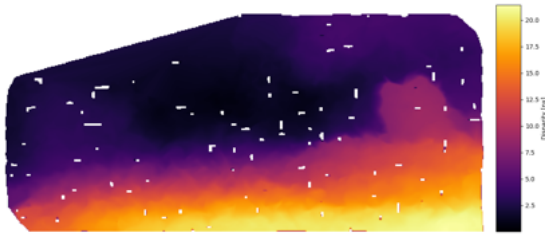
(e) Pydnet SGBM proxy supervised trained for 50 epochs



(f) Pydnet Mesh Disparity proxy supervised trained for 50 epochs

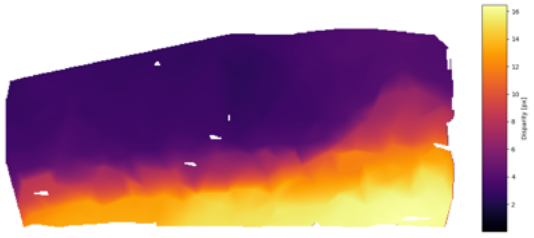


(g) SGBM stereo disparity on full-sized images



(h) Mesh regularised disparity on full-sized images

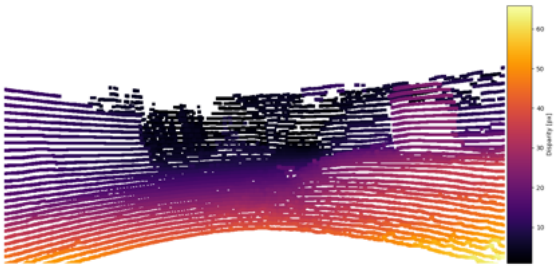
(i) SGBM stereo disparity on small images



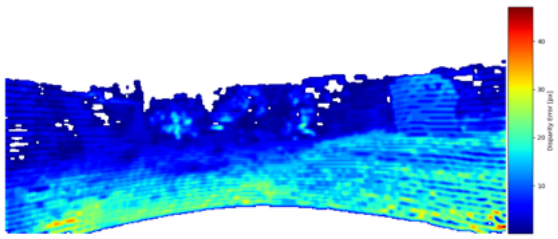
(j) Mesh regularised disparity on small images

Figure 12.4: Qualitative depth estimates for Frame 500 of the UCT Old Zoo dataset for each method. Best viewed on screen.

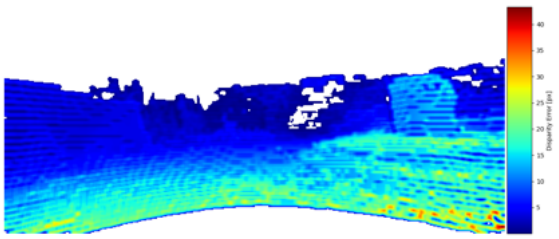
(a) Input image



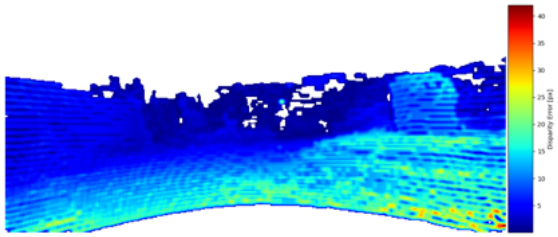
(b) Ground Truth from Lidar



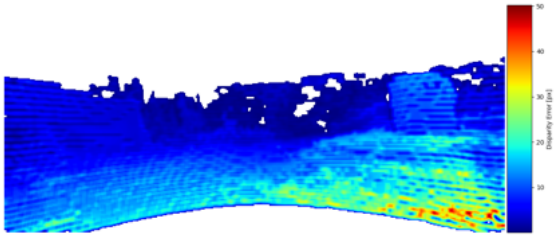
(c) Error Map: Monodepth trained for 50 epochs



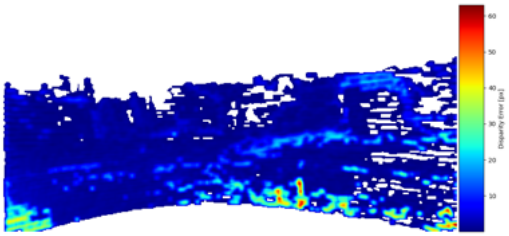
(d) Error Map: Pydnet trained for 50 epochs



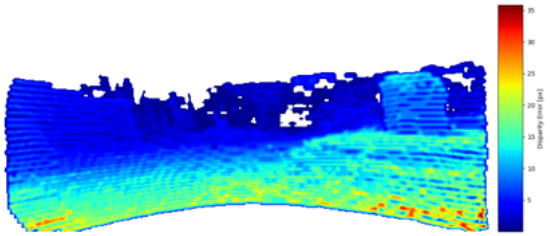
(e) Error Map: Pydnet SGBM proxy supervised trained for 50 epochs



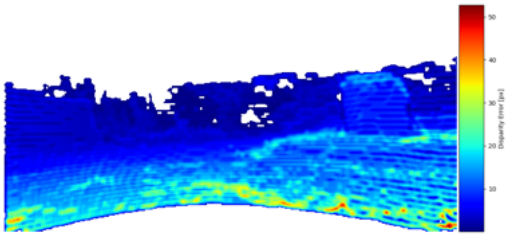
(f) Error Map: Pydnet Mesh Disparity proxy supervised trained for 50 epochs



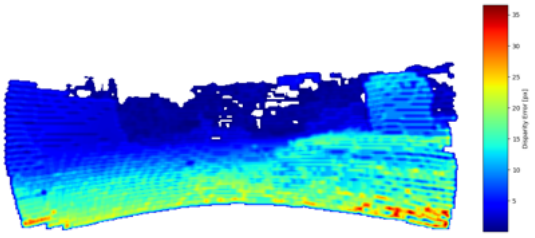
(g) Error Map: SGBM stereo disparity full-sized imgs



(h) Error Map: Mesh regularised disparity full-sized imgs



(i) Error Map: SGBM stereo disparity small imgs



(j) Error Map: Mesh regularised disparity small imgs

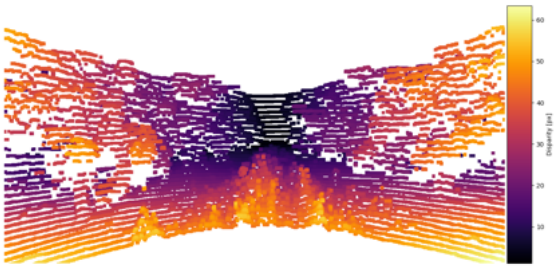
Figure 12.5: Error Heatmaps for the various methods of disparity estimation for the Old Zoo dataset at Frame 500. See Figure 12.4 for the disparity maps.

12.2 Qualitative examples at Lindenhoff Farm

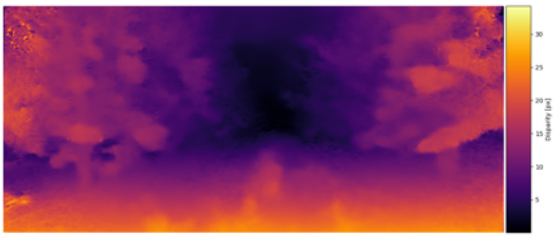
The environment at Lindenhoff farm that was used to evaluate disparity estimation consisted almost entirely of organic geometry. This provides a different test environment to that of the semi-structured environment of the UCT Old Zoo which contained built structures and paved surfaces. In comparison, the environment at Lindenhoff features floors that are covered in plants such as grasses as well as lines of trees growing in the orchards which would protrude into the path of the vehicle and the camera. While this gave a feature-rich environment, many of the features are repeating and therefore hard to track for global stereo-matching algorithms. The similarity between different patches of an image is also similar due to the similar nature of the trees in the images. The combination of these factors provides a different set of factors for disparity estimation methods.

After the performance of the mesh disparity on the Old Zoo dataset, it was no longer used as a proxy disparity signal for Pydnet on the Lindenhoff Dataset. Instead, an investigation was performed on the difference between a proxy weight of 0.3 and 0.6 for Pydnet proxy supervised with SGBM stereo disparity.

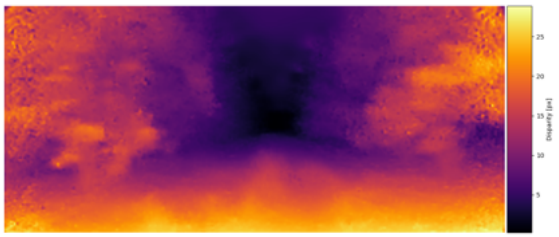
(a) Input image at Frame 1000



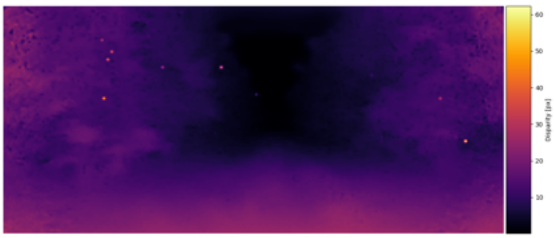
(b) Ground Truth from Lidar



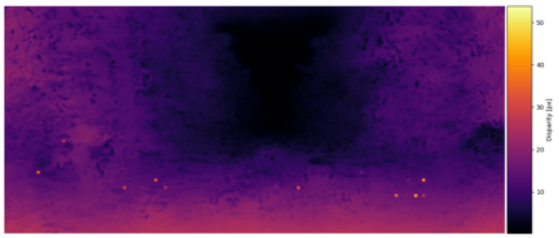
(c) Monodepth trained for 50 epochs



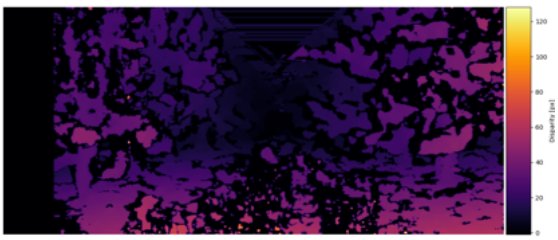
(d) Pydnet trained for 50 epochs



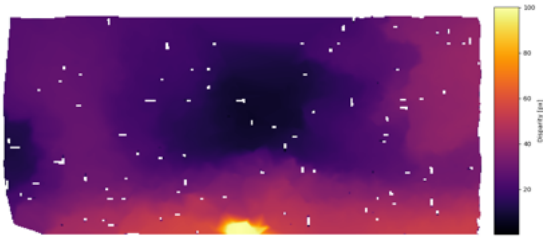
(e) Pydnet SGBM proxy supervised with α_{px}=0.3



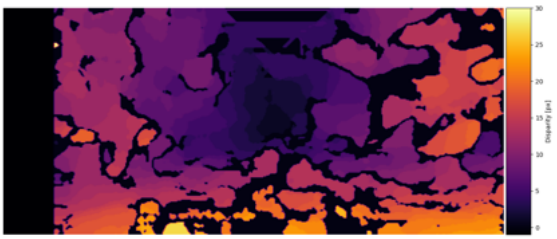
(f) Pydnet SGBM proxy supervised with α_{px}=0.6



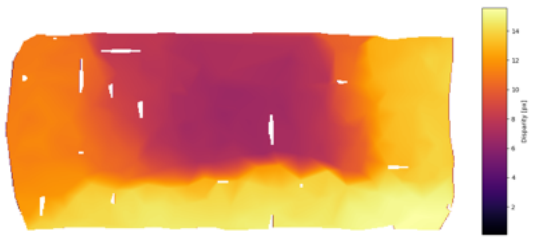
(g) SGBM stereo disparity on full-sized imgs



(h) Mesh regularised disparity on full-sized imgs



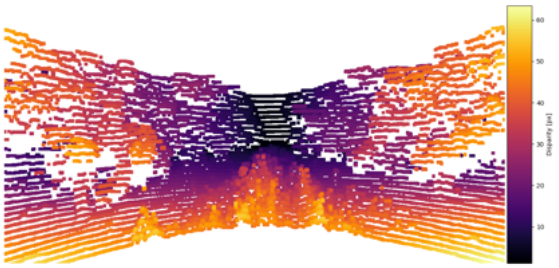
(i) SGBM stereo disparity on small imgs



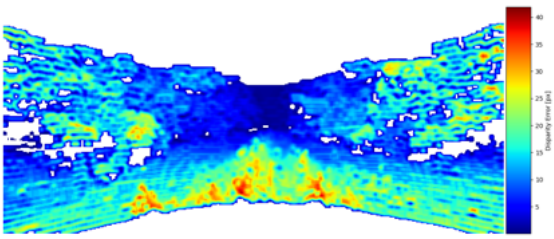
(j) Mesh regularised disparity on small imgs

Figure 12.6: Qualitative depth estimates for Frame 1000 of the Lindenhoff Farm data

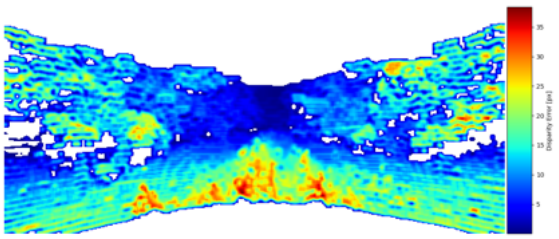
(a) Input image of Frame 1000



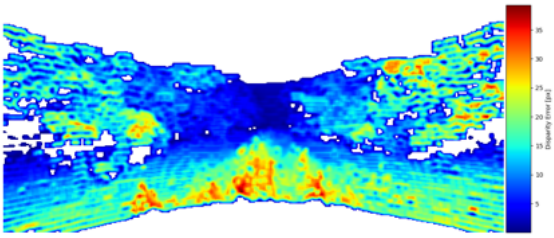
(b) Ground Truth from Lidar



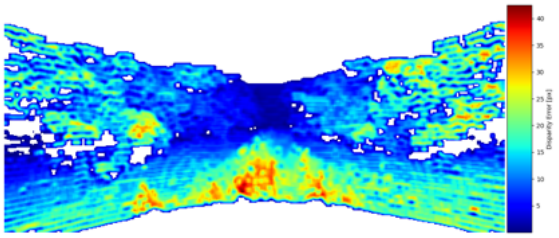
(c) Error Map: Monodepth trained for 50 epochs



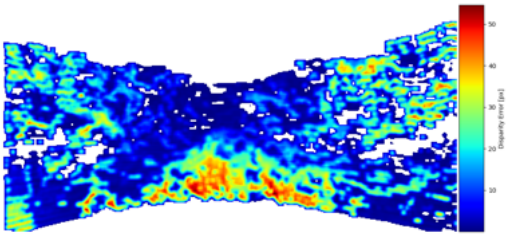
(d) Error Map: Pydnet trained for 50 epochs



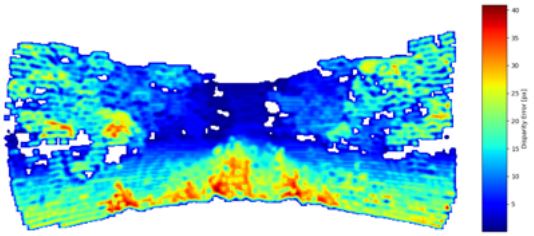
(e) Error Map: Pydnet SGBM disparity proxy supervised α_{px}=0.3



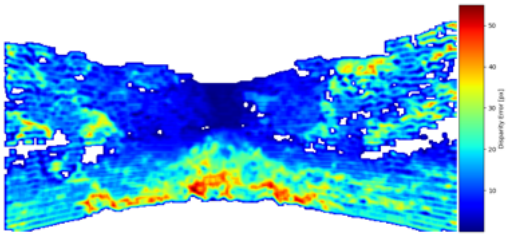
(f) Error Map: SGBM disparity proxy supervised α_{px}=0.6



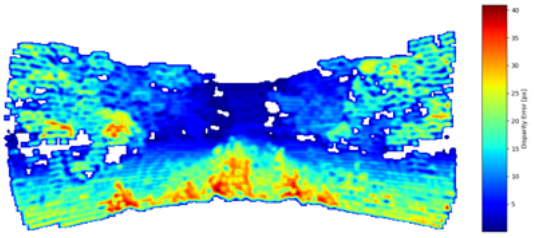
(g) Error Map: SGBM stereo disparity full-sized imgs



(h) Error Map: Mesh regularised disparity full-sized imgs



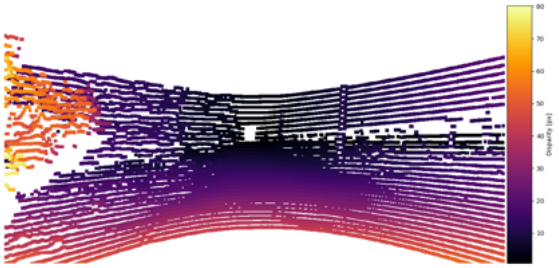
(i) Error Map: SGBM stereo disparity small imgs



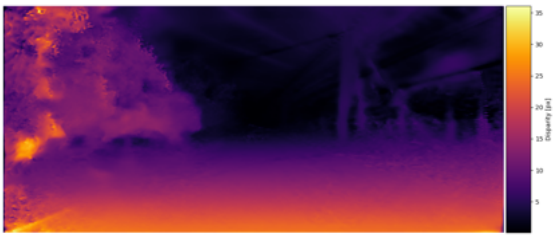
(j) Error Map: Mesh regularised disparity small imgs

Figure 12.7: Error Heatmaps for disparity estimates from Frame 1000 of the Lindenhoff dataset. See Figure 12.6 for the disparity maps.

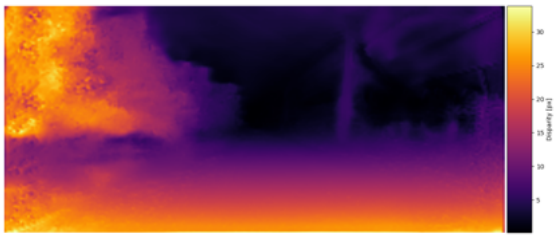
(a) Input image at Frame 2000



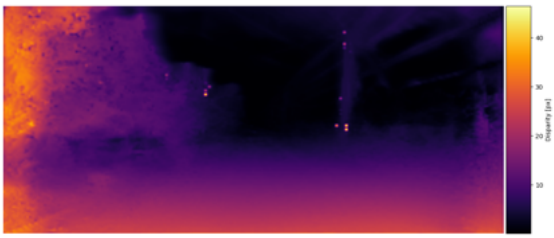
(b) Ground Truth from Lidar



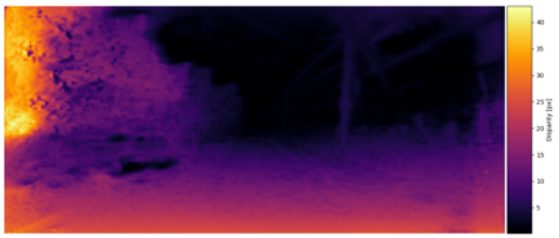
(c) Monodepth trained for 50 epochs



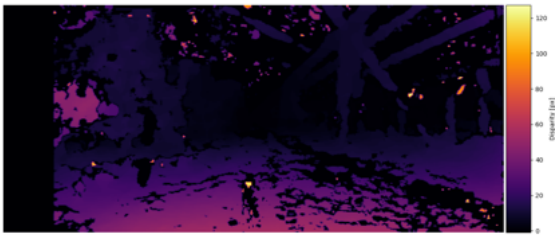
(d) Pydnet trained for 50 epochs



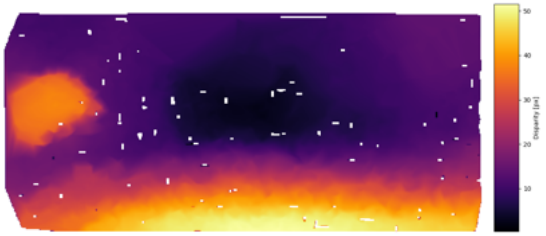
(e) Pydnet SGBM proxy supervised with α_{px}=0.3



(f) Pydnet SGBM proxy supervised with α_{px}=0.6

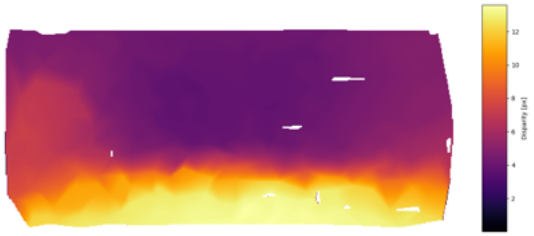


(g) SGBM stereo disparity on full-sized imgs



(h) Mesh regularised disparity on full-sized imgs

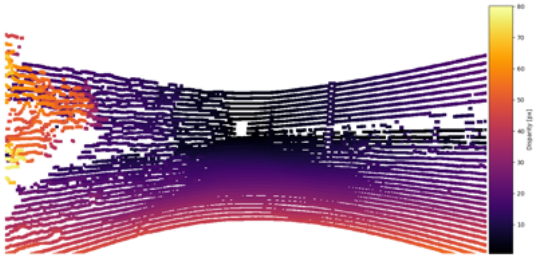
(i) SGBM stereo disparity on small imgs



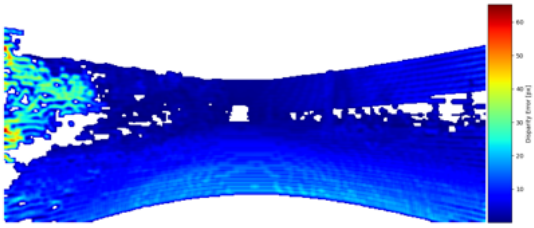
(j) Mesh regularised disparity on small imgs

Figure 12.8: Qualitative depth estimates for Frame 2000 of the Lindenhoff Farm data

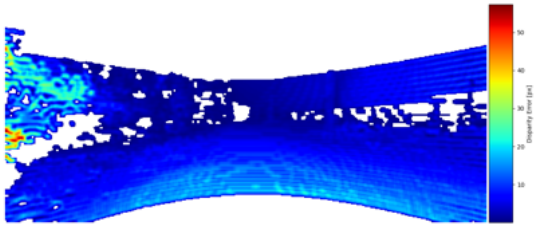
(a) Input image for Frame 2000



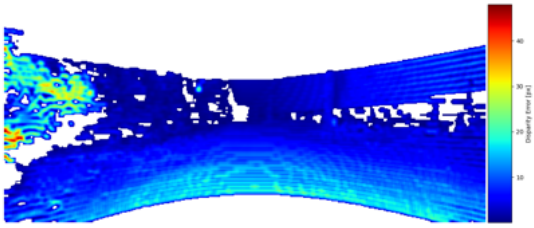
(b) Ground Truth from Lidar



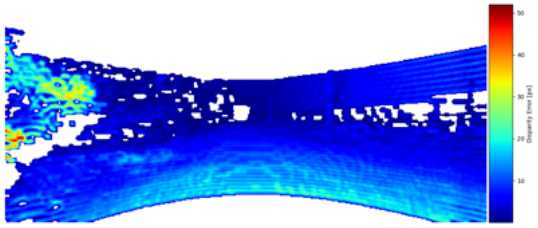
(c) Error Map: Monodepth trained for 50 epochs



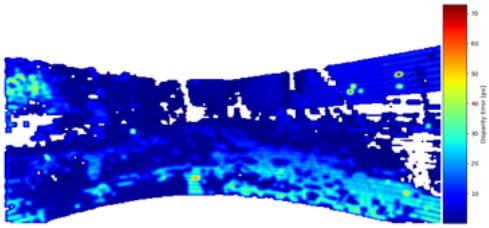
(d) Error Map: Pydnet trained for 50 epochs



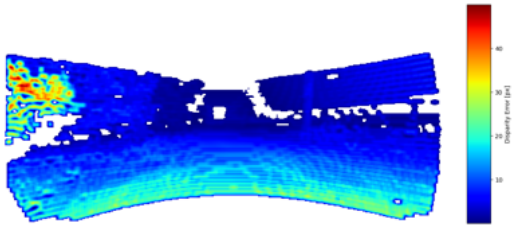
(e) Error Map: Pydnet SGBM disparity proxy supervised α_{px}=0.3



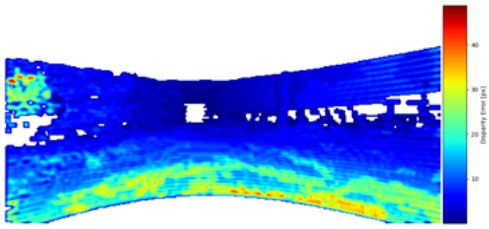
(f) Error Map: SGBM disparity proxy supervised α_{px}=0.6



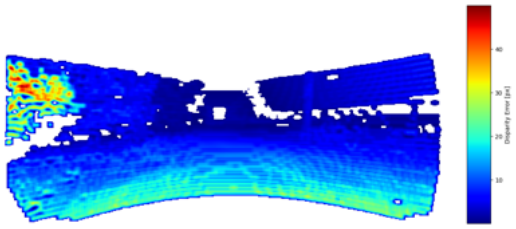
(g) Error Map: SGBM stereo disparity full-sized imgs



(h) Error Map: Mesh regularised disparity full-sized imgs



(i) Error Map: SGBM stereo disparity small imgs



(j) Error Map: Mesh regularised disparity small imgs

Figure 12.9: Error Heatmaps for disparity estimates from Frame 2000 of the Lindenhoff dataset. See Figure 12.8 for the disparity maps.

Qualitative Results Analysis

The images presented in Figure 12.2 to Figure 12.8, show the output disparities across all the different depth estimation methods are somewhat similar to the lidar ground truth in terms of colour distribution. The significant difference in density is immediately obvious, even with the 10x dilation of the lidar points in these figures. The estimations of all the methods follow the same trend of the foreground being closer, and therefore brighter in the colour mapping, while the background is further away. However, the disparity estimations all have slightly different appearances.

As a generalisation, the smoothness and completeness of the learned methods is better than that of the geometric stereo estimates. The disparity maps for SGBM are more discretised and piecewise in comparison to that of the networks. The learned methods are also able to provide disparity estimates across the entire image which is in contrast to geometric methods which are unable to utilise the area of the stereo image where there is no overlap. This gives the learned methods an advantage provided the depth estimates are accurate enough.

Looking at the error map figures, the performance of the full-sized SGBM stereo disparity on both datasets has the lowest amount of error for the example images. However, as soon as the input images are scaled down to the same size used on the learned methods (256x512px) the error maps closely resemble that of the other methods. This observation indicates the importance of high-resolution images for accurate depth estimate as much of the image data is lost during downscaling.

Mesh regularised disparity does produce good estimates of far-field objects on the full-sized images however the performance on smaller images seems to lose the good separation of far-field and near-field. There are very few gaps in the estimation as the colour-mapped disparity shows for both full-sized and small images. SGBM disparity on the other hand provides good separation of the foreground and background at all input sizes. However, this is at the cost of gaps in the disparity. While these gaps could be amended using post-processing steps, they represent missing data that will present a challenge for the VO as well as the reconstruction.

The performance of Pydnet and Monodepth are similar, both having smooth depth estimations and showing the difference in disparity even for far-field objects. The same is true for the proxy-supervised versions of Pydnet which show that the results of proxy supervision are qualitatively comparable to the regular Pydnet.

On the zoo dataset, Monodepth does produce artifacts in the prediction as shown in the example from Frame 500 (see fig. 12.4). This is contrary to the rule of thumb that a deeper network will produce smoother, more accurate results. In comparison, the predictions using the Lindenhoff dataset for training are cleaner and better defined. This implies that the zoo dataset is not large enough to train the Monodepth network sufficiently and, as a result, the network is overfitting to the training data.

In summary for the qualitative comparison, the methods are all producing disparity estimates that correctly classify far and near-field objects. Each of the different approaches shows different disparity estimates which reflects their respective underlying methodology. The best qualitative performance can be attributed to Monodepth, Pydnet, and stereo SGBM using full-sized images. However, the most accurate method is to be determined in the Section 12.3

12.3 Quantitative Disparity Estimation Evaluation

To quantitatively evaluate the performance of the disparity estimation, the results of the error between the disparity estimation and the projected lidar points in terms of disparity is shown in Table 12.1 and Table 12.2 for the Old Zoo and Lindenhoff Farm datasets respectively.

Table 12.1: Table of Disparity Error for the Old Zoo test set averaged over 1474 frames

Type	Method Name	Input Size [px]	MSE [px^2]	SSE [px^2]	ME [px]	σ [px]	Time [s]
Gemetric Disparity	Stereo SGBM Disparity	720 x1280	**169.34**	**2.854e+06**	**6.18**	10.81	0.294
	Stereo Mesh Disparity	720 x1280	287.57	4.957e+06	13.53	**9.97**	0.620
	Stereo SGBM Disparity	256x512	388.55	6.520e+06	15.29	12.17	0.174
	Stereo Mesh Disparity	256x512	282.51	4.628e+06	13.21	10.17	0.370
Learned Disparity	Monodepth	256x512	**301.27**	**5.382e+06**	**13.60**	10.43	0.324
	Pydnet	256x512	372.63	6.674e+06	15.33	11.22	0.048
	Pydnet Proxy SGBM Disp 0.3 Weight	256x512	307.04	5.496e+06	13.70	**10.40**	0.048
	Pydnet Mesh Disparity 0.8 Weight	256x512	356.07	6.381e+06	14.88	10.92	0.048

Table 12.2: Table of Disparity Error for the Lindenhoff Farm test set averaged over 4742 frames

Type	Method Name	Input Size [px]	MSE [px^2]	SSE [px^2]	ME [px]	σ [px]	time [s]
Gemetric Disparity	Stereo SGBM Disparity	720 x1280	**414.95**	**7.105+06**	**12.35**	15.06	0.294
	Stereo Mesh Disparity	720 x1280	585.82	9.658+06	19.51	14.31	0.620
	Stereo SGBM Disparity	256x512	555.89	9.526+06	19.25	**12.79**	0.174
	Stereo Mesh Disparity	256x512	673.04	11.657+06	21.46	14.56	0.370
Learned Disparity	Monodepth	256x512	381.24	6.939+06	15.91	10.41	0.345
	Pydnet	256x512	**359.11**	**6.537+06**	**15.25**	**10.17**	0.048
	Pydnet Proxy SGBM Disp 0.3 Weight	256x512	386.67	7.038+06	15.83	10.49	0.048
	Pydnet Proxy SGBM Disp 0.6 Weight	256x512	432.05	7.863+06	16.89	11.10	0.048

Analysis of Quantitative Results

In the quantitative accuracy results for the UCT Old Zoo dataset (see Table 12.1), the performance of the geometric disparity methods was significantly more spread out than that of the learned methods. Stereo SGBM disparity from full-sized images provided the most accurate method of all with a mean error of 6px, less than half the error of the closest competitor. However, when SGBM was used on the smaller images, its results were worse than almost all of the networks. This again emphasises the observation of the importance of image resolution that was made in the qualitative results section.

The poor performance of the learned methods compared to the geometric methods implies that the dataset was neither large enough or diverse enough for the networks to be sufficiently trained. Comparing the proxy-supervised networks' performance to that of Pydnet and Monodepth across the two datasets, the better relative performance of the proxy-supervised networks on the zoo data also provides evidence for the hypothesis of the dataset being too small. This is in agreement with the claims of [22, 5] stating that proxy-supervision increases the training speed of the network.

On the Lindenhoff Farm dataset, the results obtained by the learned methods outperformed the geometric methods in all metrics except Mean Error, where SGBM had a lower score. If we compare the methods on the same input size, the performance of the learned disparity estimation outperforms the stereo methods. Of the networks, Pydnet achieved the best results

for all metrics, surprisingly outperforming the deep Monodepth network. Given the performance of the geometric methods in comparison to the learned methods, the poor performance of the proxy-supervised networks continues the trend relative to the entirely unsupervised Pydnet and Monodepth. Increasing the proxy weighting also results in a worse-performing network which shows that there is a limit to the amount of supervision that can be provided by the proxy disparity before it becomes detrimental to the network's performance. As a result, the proxy weighting of 0.3 is used for the remainder of the evaluation of proxy-supervised networks.

12.3.1 Discussion of Results

From the analysis of the results, several conclusions can be drawn.

The first conclusion is the noticeable impact that the size and diversity of the training data have on these learned disparity estimation methods. Increasing the dataset from 20k to 75k images resulted in a significant improvement in the performance of the learned methods. From this, we can confirm performance of the learned methods is highly dependent on the size and diversity of the training dataset. This follows conventional wisdom that the more data that is available for training, the better the performance of the network at generalising to new, unseen data. This highlights the importance of the dataset used for training the network. The dataset must be large enough to capture the diversity of the environment that the network will be used in. For our application to agricultural regions, this means that the data should be collected from a variety of farms with different crops and different terrains, at different times of the day, in different seasons, and under different weather conditions to ensure that the network is able to generalise to the variety of conditions that it will be used in. For our application, the testing of the networks was performed on a dataset that was collected in a different area of the farm, in a different orchard.

The noteworthy improvement of Monodepth on the Lindenhoff dataset compared to the zoo dataset, especially when compared with the lighter Pydnet, further shows that the smaller network generalises better than the deeper network on our datasets. This implies that resources can be saved by using the smaller network while still achieving good results.

The second conclusion is that, when given enough high-quality data to train on, the learned methods outperform the geometric methods in an agricultural setting. This is in agreement with the findings of [67, 22, 5, 4] who found that their learned method outperformed the geometric methods. On the smaller Zoo dataset, the best performance was achieved using SGBM (with a MSE that was nearly half that of the leading learned method). However, on the larger Lindenhoff dataset, the performance of Pydnet was better than that of SGBM with a MSE reduction of $\sim 14\%$.

From this, we can conclude that the performance of a learned method will be better than a geometric method given a good training set. This, coupled with the lower weight, size, cost and power requirements associated with only requiring one camera, makes the learned methods the better choice for our application. However, if in a real-world application, there is insufficient time and data to perform a comprehensive data collection and training of the network, then the geometric methods will provide a good alternative.

Interestingly, the performance of the proxy-supervised networks was worse than that of the unsupervised networks. This is in contrast to the findings of [22, 5] which found that proxy-supervision improved the performance of their networks. This could be due to the fact that the unsupervised networks were able to find a better local minimum than the proxy-supervised

networks. Both of the proxy-supervised networks were unable to outperform the unsupervised networks on the Lindenhoff dataset. This implies that the proxy-supervision is not providing a significant benefit to the network when there is sufficient data available for training. However, on a small dataset such as the Old Zoo dataset, the proxy-supervised networks were able to outperform the unsupervised networks. This implies that proxy-supervision is beneficial when there is insufficient data available for training as it prevents the network from over-fitting to the training data.

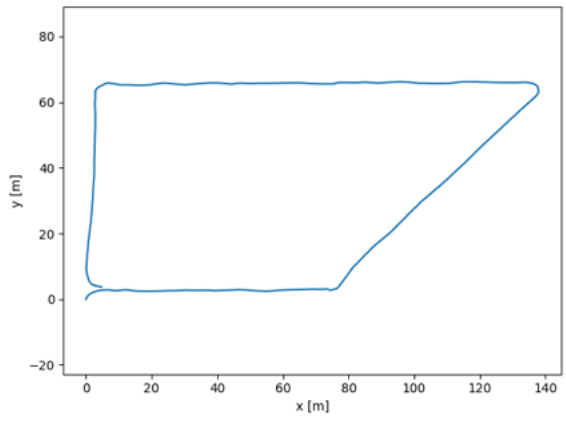
On reflection of the results for the geometric disparity, we can make the following conclusions. The performance of the geometric methods is highly dependent on the resolution of the input images. More information allows for better matching and as such, produces more accurate disparity maps. However, this is at the cost of a higher computational load. The performance of the SGBM is almost 2x as slow on the larger image.
Secondly, given the better results of mesh disparity regularisation in the semi-structured environment of the Old Zoo, the results suggest that it is useful in environments where there are planar surfaces such as walls and paved roads. However, in the organic environment of the Lindenhoff, mesh regularisation of the disparity was detrimental to the performance of the disparity estimation.

One of the key requirements for the reconstruction is for it to run in real time such that it can be used for navigation. As such, the time taken to perform the disparity estimation is an important factor. From the tables in Table 12.1 and Table 12.2 for the Old Zoo and Lindenhoff Farm datasets respectively, we see that the disparity estimation using the learned methods is significantly faster than the geometric methods. We see that the fastest learned methods (Pydnet-based) are approximately 3.6 times faster than the fastest geometric method on an image of the same size. This is a significant difference in time and is a key factor in the decision to use the learned methods for the reconstruction.

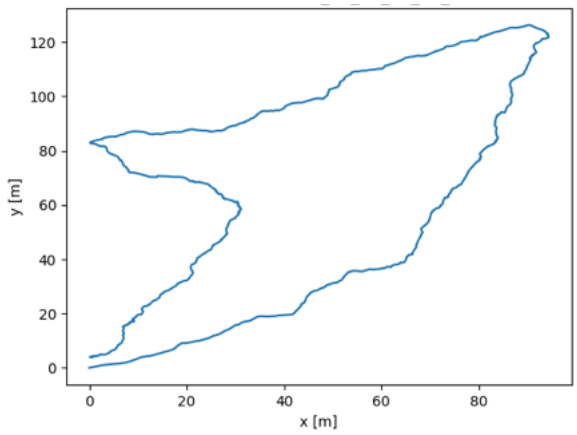
Therefore, considering the performance of the different disparity methods, we conclude that the best option for a lightweight reconstruction pipeline Pydnet. This is due to it having the best performance when trained with enough data, low computational load, and fast processing time. The Pydnet network is significantly smaller than the Monodepth network which makes it more suitable for deployment on a low-power embedded system. Additionally, the lack of extra weight or power required for a stereo camera makes the Pydnet network the best choice for our application.

Visual Odometry Methods

In an unstructured and agricultural setting, the environment that a field robot is placed in is rich in image features due to the organic objects in the surroundings. At the same time, these objects provide occlusions for GPS signal which in turn significantly increases the dilation of precision, resulting in inaccurate measurements. [38] show that the accuracy of consumer GPS is limited to an error of $\sim 5m$ in areas containing a large number of overhead trees. Additionally, they show that tall, dense trees such as Pine (*Pinus taeda*) add a further 0.6m of inaccuracy, even when using static measurements which is the best case for GPS. To illustrate how this affects real-world performance, in fig. 13.1 we present the plot of the GPS data obtained in an orchard on the Lindenhoff Farm dataset and compare the results to that of a ground truth lidar odometry obtained with LOAM



(a) Plot of the ground truth trajectory using the lidar points and LOAM from the Lindenhoff Farm Test Route



(b) Plot of the raw GPS data from the Lindenhoff Farm Test Route

Figure 13.1: Comparison of GPS to the highly accurate LOAM trajectory demonstrating why VO is needed for reconstruction.

While the same trees and objects obstruct the signal for GPS, they provide useful image features that can be used for visual odometry.

In this section, we explore the methods for constructing a VO pipeline as well as the impact of feature selection and disparity estimation methods on the overall accuracy in comparison to the ground truth trajectories.

13.1 Mathematical formulation of VO

13.1.1 Solving for the estimated Transform of motion

Given a set of features ($\mathcal{F}$), matched between the frame at time t_n and the frame at t_{n+1}, where the depth estimations are known for the first set of features $\mathcal{F}_n$. After calculating the 3D position of $\mathcal{F}_n$, we refer to the resulting set of 3D points as landmarks L_n.

The transformation (T_n^{n+1}) between the two images n and $n+1$ is then calculated by minimising the Euclidean distance between the location of features in the image at t_{n+1} and the location of the corresponding landmarks once they have been projected into the image frame using the estimated transform.

$$T_n^{n+1} = \min \sum_{i=0}^{len(F_{n+1})} |F_{n+1} - f(L_n, T)| \tag{13.1}$$

where $f(L_n, T)$ is the re-projection of points L_n into the image frame using the estimated transform T.

By iteratively minimising the re-projection error, we can obtain the best-fit transform between the landmarks at t_n and the features at t_{n+1}. Therefore, the accuracy of the estimated transform is determined by the accuracy of the input. Namely, the accuracy with which the features are matched as well as the accuracy of the 3D location of the landmarks.

In order to improve the accuracy of the visual odometry of our pipeline, we conduct an investigation to optimise the choice of features used as well as find which disparity estimate yields the most accurate VO.

13.2 Feature selection effects on VO

Image features are, by design, biased to selecting certain points in images which are more trackable. For example, in an outdoor setting, Shi-Tomasi/Harris corners and ORB would bias towards selecting features with corners such as the points of leaves as opposed to the centres of tree trunks. Therefore, when these features are used to build reconstructions, the impact of this bias on the accuracy of the visual odometry needs to be understood.

To analyse the influence that the choice of image features has on the geometric accuracy of the reconstruction, four different features are compared. Namely Harris Corners, SIFT [6], ORB [47], MOC and $\mathcal{BRISK}$[7] Points at the respective feature locations were sampled from a dense disparity formed using Semi-Global Block Matching to represent the standard benchmark for the geometric disparity. Additionally, another set of samples was obtained using Monodepth disparity which represented the benchmark for the learned disparity.

The initial approach of determining the impact of the feature selection on the Visual Odometry was to calculate an error between the benchmark disparity and an interpolated disparity formed from a dense sampling using the features. This involved interpolating the sampled points by first tessellating them to form a Delaunay triangulation and then interpolating the mesh using

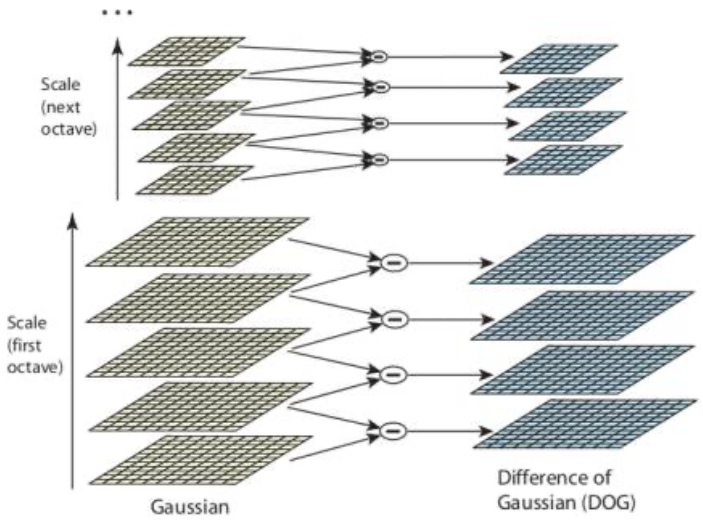


Figure 13.2: Difference of Gaussians at scales used by SIFT. Figure taken from [6]

disparity coordinates. These two dense disparities could then be compared using the euclidean distance between their disparity values.

This approach was initially used however, it is an indirect method of evaluating the effect on VO with a set of biases and errors introduced due to the interpolation and meshing. Instead, the final approach used was to simply calculate the VO trajectories on the geometric and learned disparity benchmarks. This result was then compared to a ground truth trajectory obtained using the highly accurate lidar odometry method of LOAM [13].
The results of this comparison are shown in Table 13.1 at the end of this chapter.

13.2.1 SIFT Features

Scale Invariant Feature Transform (SIFT) proposed by [6] which provides a solution to the problems posed by changes in intensity, rotation and affine transformations of the image data when matching features. SIFT can be summarised as a method of transforming image data into a set of feature coordinates that are scale-invariant relative to local features.

There are four main computational steps in generating the set of SIFT image features.
The first step is to search all scales (after reducing the image size iteratively over several octaves) and image locations for points of interest using a difference-of-Gaussian (DoG). This identifies the scale space extrema which will be invariant to changes in scale and orientation.

The second step is to compare a point of interest to the pixels in its immediate neighbourhood. By rejecting points of interest that have low contrast, the most stable candidate keypoints can be identified. Then, thirdly, each keypoint is assigned an orientation. The orientation is based on the gradient directions of the local region as opposed to the orientation relative to the image coordinate axis. Finally, the keypoint descriptors are generated using the direction and magnitude of the local image gradients after the local gradient is transformed relative to the orientation, location and scale. This transformation further reduces the impact of changes in illumination and local distortions (due to the object moving or the camera moving around the object).

SIFT has the advantage over other features in it being scale invariant as well as not relying on

image corners. This translates to being able to track features more accurately as they travel towards and away from the camera, causing size changes due to perspective. However, this comes at the cost of requiring an increased amount of time to compute the features.

13.2.2 ORB Features

Orientation FAST and Rotated Brief (ORB) [47] builds on the FAST [87] keypoint detector and a modification of the BRIEF [88] feature descriptor to produce a high-speed binary descriptor that is rotation invariant and more noise resistant than its parent methods.

FAST to get keypoints The first step of ORB is to use the FAST-9 algorithm to identify potential keypoints. The FAST-9 algorithm identifies points that have a high change in intensity from the centre pixel to those at the edge of a 9-pixel wide ring around the centre. The intensity threshold is set low enough to obtain more keypoints than the target number N.

After identifying the potential keypoints using FAST, the points are ordered according to their Harris corner measure [45]. The top N points are then chosen as keypoints.

The orientation of the detected corner of each point is calculated using the intensity centroid. By the angle of rotation of the vector from the centre of the corner descriptor patch to the corner intensity centroid, ORB calculates the corner's moment and thus its orientation.

BRIEF descriptors are a binary string computed using the difference in pixel intensities of a smoothed patch sampled in a set of locations. Since BRIEF descriptors perform poorly if there is any in-plane rotation of anything more than a few degrees. The orientation of the keypoint is used to calculate a rotation matrix which is then used to steer the BRIEF descriptors. This results in the Rotated BRIEF descriptors and thus the ORB features are calculated.

ORB features are matched and tracked using the methods outlined for BREIF descriptors [89]. Using the hamming distance between BRIEF descriptors, the closest match in the next image can be easily determined.

13.2.3 BRISK Features

The Binary Robust Invariant Scalable Keypoints (BRISK) [7] algorithm is a low computational cost method of detecting, describing and matching keypoints. The method applies a scale-space variation of a FAST detector and a pre-defined sampling strategy to produce binary bit-string descriptors from intensity comparisons.

To identify keypoints, BRISK uses a similar FAST-based approach to ORB (as described above). However, points of interest are identified in image dimensions as well as in scaled layers of an image pyramid downsized by half over several (normally 4) layers as well as layers in between. This is a similar scale space to SIFT, however, the method uses FAST instead of Difference of Gaussians.

Keypoint descriptors are formed using a sampling pattern of concentric circles (see Figure 13.3). The descriptor patch is first smoothed using Gaussian smoothing where the standard deviation of the kernel (shown in dashed red) is proportional to the location of the point in the pattern (shown in blue).

Two sets of pairs of points are used from the sampling pattern. A short-distance pairings $\mathcal{S}$ for

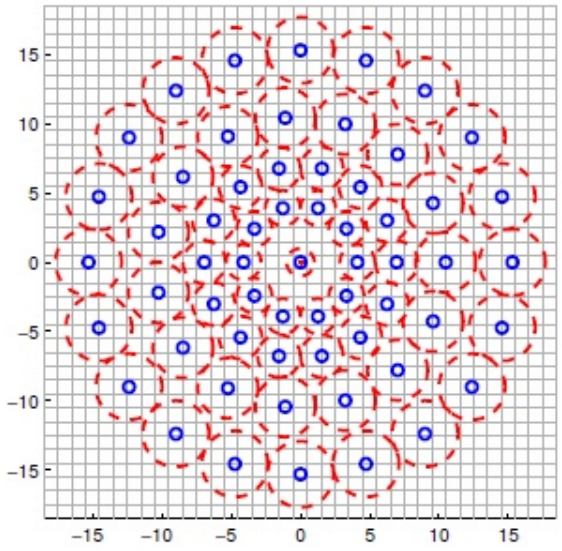


Figure 13.3: Sampling pattern used by BRISK with 60 points.The pattern is shown with a scale $s = 1$ as presented in [7]

building the descriptor and a long-distance set $\mathcal{L}$ for determining the orientation.
Using all the points in $\mathcal{L}$, rotation is estimated by taking the angle of the vector formed by average in the x and y directions for a point-to-point local gradient estimation function. The rotation angle is used to rotate the set $\mathcal{S}$ around the feature centre. Then, by using the sampled greyscale intensities of the rotated pairs of points (similarly to BRIEF), a binary descriptor string is formed with the concatenated results of a comparison test. Feature tracking is performed by matching descriptors using the Hamming distance between respective bit strings.

13.2.4 MOC Features

Originally developed for visual tracking and dense mapping, Maximum of Curvature (MOC) as a feature descriptor was proposed in [30] and provides an extremely dense set of image features well distributed across the image. By setting the size of a windowing function, the distribution and density of features can be controlled. This feature extractor can track points without the need for corners (unlike ORB, BRISK and to an extent SIFT).

The MOC features use the local extrema of the curvature function

$$\mathcal{K} = \delta_y^2 \delta_{xx} - 2\delta_x \delta_y \delta_{xy} + \delta_x^2 \delta_{yy} \tag{13.2}$$

Where δ_x is the partial derivative of the image in the x direction using a Sobel operator or similar method.

Matching of features is performed by matching the current extrema of curvature with the extrema of curvature in a small region around the projected location of the feature in the next image. The flow between two images is estimated using a small set of BRIEF image features which are then used to fit an affine transformation. A distance-regularised evaluation function is then maximised using an iterative hill-climbing method until the best matching feature is located.

By not utilising image feature descriptors, changes in the appearance of an object/feature does not influence the tracking of the points significantly. As a result, provided that the image data

surrounding the feature stays relatively constant, tracking of a feature under large illumination and appearance changes is not an issue.

13.3 Evaluation of Visual Odometry

Having established the method used to calculate the visual odometry as well as the methods for obtaining the image features, an investigation of the effect of the different methods on the visual odometry can be conducted.

To calculate the error, we adopt the metrics used in the KITTI Visual Odometry benchmark [90]. The metrics that we find most applicable are the relative translational error (t_{rel}) and the relative rotational error (r_{rel}). In addition, we present the Absolute Trajectory Error (ATE) for completeness.

Relative Pose Error (RPE) is calculated by comparing the transforms from one frame to the next. The errors for each axis are then added for translation and rotation. This error is calculated over the entire trajectory and is a measure of the differential accuracy of the VO. A large RPE indicates that there will be significant drift present as the error accumulates over the trajectory. While it is useful to look at the RPE in absolute terms, it is also useful to look at the RPE relative to the distance travelled. This is calculated by dividing the RPE by the distance travelled between the two frames. This metric is referred to as the relative translational error (t_{rel}) and the relative rotational error (r_{rel}).

For this analysis, the framerate of the camera is 10Hz while the robot is moving at a speed of 1.0m/s. As a point of reference, the algorithm VISO2-M [91], is considered the benchmark for monocular VO on the Kitti dataset. This algorithm achieves an average t_{rel} of 25% and r_{rel} of 12.5% [$^o/50m$] at similar speeds to the robot used in this work [90]. These results are achieved using a pipeline that is optimised for city driving and is significantly more complex than the pipeline used for evaluation in this work. The offroad environment we test our pipeline in introduces vibrations and rotations due to the uneven terrain which are not present in the Kitti dataset. Therefore, the results presented in this work are not directly comparable to the results from the Kitti dataset.
However, they do provide a point of reference for the performance of the VO pipeline presented in this work even though the environments are fundamentally different.

Absolute Trajectory Error (ATE) measures the RMS error between the estimated camera pose and the ground truth pose relative to the global coordinate frame origin. This metric is useful for comparing the accuracy of the VO to the ground truth trajectory. However, it is not as useful for comparing the accuracy of different VO methods as the error is dependent on the length of the trajectory. This is because the error is accumulated over the entire trajectory. In other words, a longer trajectory will have a larger ATE than a shorter trajectory if we hold the relative error constant in both cases.

13.3.1 Evaluation of Impact of Feature Choice on VO

To compare the impact of the choice of feature, we performed VO using each type of image feature to track the points. For the source of disparity, we used the benchmarks for each of the geometric disparity and the learned disparity, Stereo SGBM and Monodepth respectively. This choice also shows if there is a bias towards different features for each approach to disparity

estimation, especially considering the qualitative difference as discussed in the evaluation of the disparity estimation methods (see Section 12.1).

For this evaluation, the error in the relative translation between one pose and the next was used as the metric. This choice of metric allowed the trajectories from the LOAM ground truth to be compared with the trajectory for each type of feature and the results are presented in Table 13.1

Table 13.1: Table of Relative Trajectory Errors for VO on Old Zoo dataset

Type	Features	Depth Model	Image Size	RTE [m]	σ_{RTE}	MSE RTE
Gemetric Disparity	Brisk	Mesh Stereo	256x512	0.221	0.375	0.596
	MOC	SGBM Stereo	256x512	0.223	0.379	**0.582**
	ORB	SGBM Stereo	256x512	**0.212**	**0.372**	0.598
	SIFT	SGBM Stereo	256x512	0.358	0.484	0.697
Learned Disparity	MOC	Monodepth	256x512	0.203	0.373	0.539
	ORB	Monodepth	256x512	**0.197**	**0.336**	**0.456**
	SIFT	Monodepth	256x512	0.212	0.374	0.554

From these results, we see that the choice of feature selection does indeed have an effect on the accuracy of the visual odometry, confirming our hypothesis.

The results from the Old Zoo dataset indicate that the best visual odometry feature for the geometric disparity methods is closely matched between Brisk, MOC and ORB with all of them returning similar errors. ORB has a lower average error in terms of mean RTE while Brisk has a lower mean squared error. The poor performance of SIFT on the SGBM disparity was in contrast to its good performance on the Lindenhoff Farm dataset where it was competitive for the position of the best feature.

From these evaluations, the most robust feature for use in VO is ORB. While not factored into this decision, ORB has an additional advantage as it is the fastest feature to calculate the methods that are presented in this work. Therefore, the selection of ORB as the feature of choice further aids in making the reconstruction pipeline more computationally lightweight.

13.3.2 Impact of Disparity Estimation Method on VO

With a feature selector chosen, the last variable of the VO pipeline to be decided is which method of disparity estimation provides the most accurate and reliable visual odometry. Each of the depth estimation methods was switched into the VO pipeline and the trajectories were obtained. This was then compared with the ground truth trajectory obtained using LOAM and the errors were calculated. The results are displayed in Table 13.1.

Table 13.2: Table of Relative Translational Errors for VO on UCT Old Zoo Dataset

Type	Features	Depth Model	Img Size	RTE [m]	σ RTE [m]	MSE RTE
Gemetric Disparity	ORB	SGBM Stereo Disparity	720x1280	**0.571**	**1.092**	**1.518**
	ORB	Mesh Stereo Disparity	720x1280	0.693	1.217	1.963
	ORB	SGBM Stereo Disparity	256x512	0.640	1.177	1.796
	ORB	Mesh Stereo Disparity	256x512	0.619	1.156	1.722
Learned Disparity	ORB	Monodepth	256x512	0.611	**1.116**	1.619
	ORB	Pydnet	256x512	0.628	1.133	1.679
	ORB	Pydnet Mesh 0.8 Proxy	256x512	0.638	1.160	1.754
	ORB	Pydnet SGBM 0.3 Proxy	256x512	**0.586**	1.122	**1.604**

Table 13.3: Table of Absolute Trajectory Errors for VO on the Lindenhoff Farm dataset

Type	Features	Depth Model	Img Size [px]	ATE [m]	σ ATE [m]	MSE ATE	Normalised ATE [%]
Gemetric Disparity	ORB	SGBM Stereo Disparity	720x1280	0.491	0.820	0.914	168.54
	ORB	Mesh Stereo Disparity	720x1280	0.482	0.735	0.773	**75.67**
	ORB	SGBM Stereo Disparity	256x512	**0.556**	0.778	0.915	176.91
	ORB	Mesh Stereo Disparity	256x512	0.476	**0.712**	**0.734**	86.21
Learned Disparity	ORB	MonodepthFarm	256x512	0.477	0.673	0.681	100.84
	ORB	Pydnet	256x512	0.459	0.678	0.671	121.97
	ORB	Pydnet SGBM 0.3 Proxy	256x512	**0.441**	**0.630**	**0.592**	102.62
	ORB	Pydnet SGBM 0.6 Proxy	256x512	0.458	0.667	0.655	**48.59**

From the results presented, we can see that the results from the Old Zoo do not initially appear to select the same disparity method as the results from the Lindenhoff Farm dataset. In the results of the evaluation of the disparity estimation, it was noted that the geometric disparity methods outperformed the learned methods on the Old Zoo dataset while the opposite was concluded for Lindenhoff. This difference in disparity estimation performance is reflected in the results for the different VO errors.

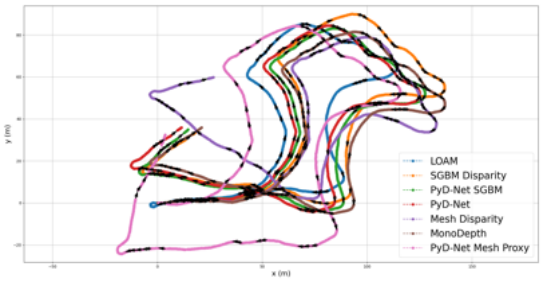
The better disparity estimate from the geometric methods on the Old Zoo dataset is reflected in their better performance in the VO. Stereo SGBM provided the best results of all the methods when the input was full-resolution images. However, the performance of SGBM with training-size images was worse than that of the networks. The best-performing network on the Old Zoo data was the Pydnet SGBM proxy supervised.
On the Lindenhoff dataset, the best-performing results were obtained with the learned disparity methods, which again reflects the better quality disparity estimations. Of the networks, the best performing was Pydnet SGBM proxy supervised with a weight of 0.3 which is actually in agreement with the results from the Old Zoo despite initial impressions.

From this evaluation of the effect of the different choices of disparity estimation methods, it can thus be concluded that the method that provides the best visual odometry results across

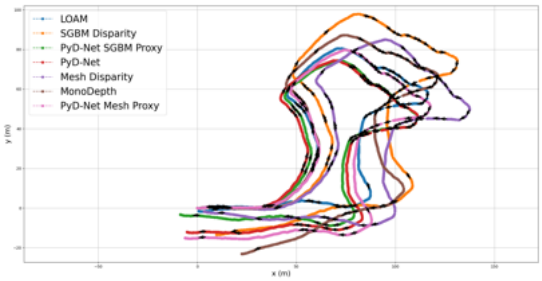
the different datasets is Pydnet SGBM proxy supervised with $\alpha_{px} = 0.3$ which does not reflect the result for the best disparity estimation method. However, the use of SGBM as a proxy supervision signal has improved the VO performance over the original Pydnet.

13.3.3 Evaluation of the final VO Pipeline

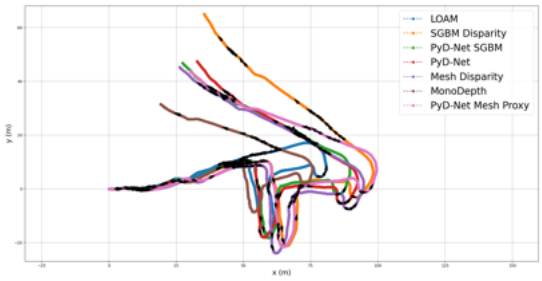
The evaluation of the final VO pipeline was performed on the Old Zoo dataset. Three routes were used to evaluate the performance in slightly different environments. For each route, the VO was obtained with a different disparity estimation method to evaluate their impact on the VO. The results are as shown in Figure 13.4



(a) VO trajectories on Old Zoo Route A



(b) VO trajectories on Old Zoo Route B

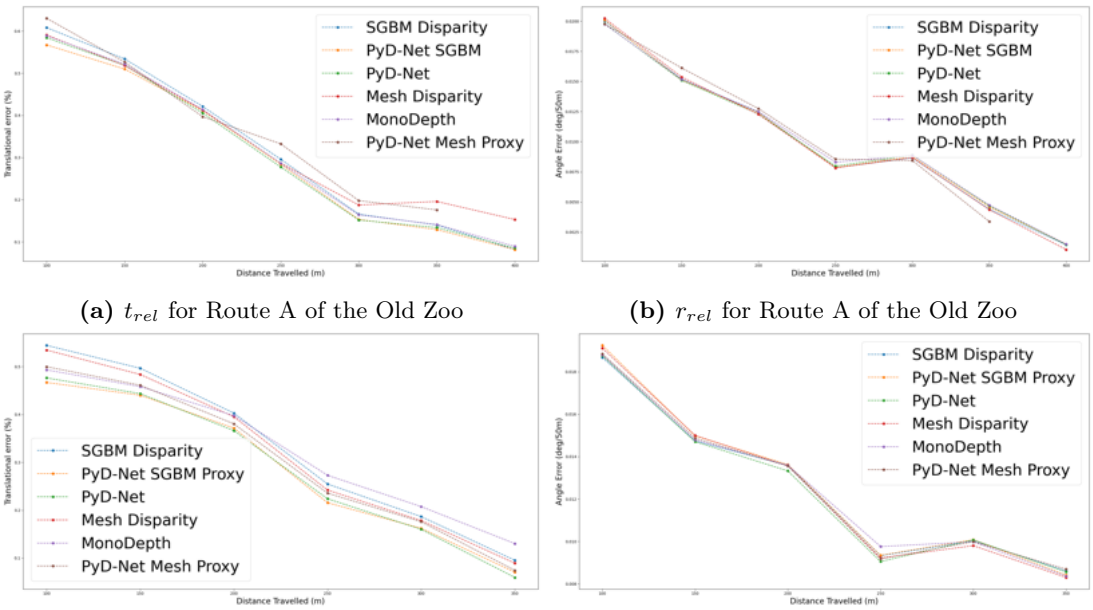


(c) VO trajectories on Old Zoo Test Route

Figure 13.4: Odometry Performance: a) and b) shows the trajectories included in the training of the network. c) shows the trajectory that wasn't included in our network training. The proxy supervision shows compelling results on the validation and testing sets.

As can be seen from these figures, drift plays a large impact on the resultant trajectory over the course of the route. None of the methods were able to complete the loops as demonstrated by the ground truth LOAM trajectory. However, they all were able to keep an approximation of the shape of the path with some distortion. To quantify the amount of error, the translational and rotational errors are plotted against the distance travelled in Figure 13.5. The quantitative values for the figures for the relative translational and rotational errors in Figure 13.5 are shown in Table 13.4

From these results, we can determine that the learned methods have a significant time advantage over the geometric methods of disparity estimation. While Monodepth and SGBM are comparable, the Pydnet based methods are around 8 times faster than SGBM, giving them a significant advantage. The performance of the learned methods as a group is also better than that of the geometric methods both on validation and test data. On the validation data, Pydnet proxy supervised by SGBM disparity featured the best performance. On the test data, Monodepth generalised the best of the learned methods, resulting in the best performance. However, the performance of Pydnet proxy supervised with SGBM was in second place, not too far behind.



(a) t_{rel} for Route A of the Old Zoo

(b) r_{rel} for Route A of the Old Zoo

(c) t_{rel} for Route B of the Old Zoo

(d) r_{rel} for Route B of the Old Zoo

Figure 13.5: Comparing the relative translational and rotational errors for the different disparity estimation methods on routes at the UCT Old Zoo

Table 13.4: Comparison of VO Methods on the UCT Old Zoo

	Experiments	Validation					Test					Time
	VO Method	t_{rel} [%]	r_{rel} [$^o/50m$]	ATE [m]	RPE [m]	RPE [deg]	t_{rel} [%]	r_{rel} [$^o/50m$]	ATE [m]	RPE [m]	RPE [deg]	CPU only [s]
Geometric	SGBM Disparity Stereo Odometry	42.911	80.504	19.254	**0.209**	**2.643**	50.050	117.194	25.889	0.225	**2.124**	0.370
	Mesh Disparity Stereo Odometry	42.324	81.095	24.155	0.288	2.646	44.343	115.554	19.612	0.225	2.137	0.174
Learned	Monodepth [4]	41.556	80.871	20.083	0.402	2.644	**37.937**	118.573	**12.273**	**0.201**	2.136	0.324
	Pydnet [67]	39.839	**80.093**	28.976	0.396	2.651	44.781	117.905	19.591	0.218	2.136	**0.048**
	Pydnet Proxy Mesh Disparity	42.862	82.743	18.504	0.405	2.624	45.350	**115.375**	20.808	0.228	2.146	**0.048**
	Pydnet Proxy SGBM Disparity	**39.149**	81.240	**14.639**	0.395	**2.643**	43.802	116.472	18.056	0.218	2.141	**0.048**

With this evaluation, and factoring in the goal of this work to produce a lightweight reconstruction pipeline, the method of choice for the pipeline is Pydnet proxy supervised with SGBM. The accuracy achieved has shown that the performance of the learned methods outperforms that of the stereo methods which in turn results in our pipeline being more lightweight in terms of size, weight and power since only a monocular camera is required as opposed to a stereo camera.

Reconstruction Evaluation

After evaluating the methods of disparity estimation and the pipeline for the visual odometry in the previous chapters, the best approaches for each respective task have been identified. This chapter first evaluates the impact of different choices of disparity estimation methods on the final reconstruction after the disparity has been used to create a pointcloud which is then regularised as a mesh surface and placed in the reconstruction.
Finally, the final pipeline can be assembled and evaluated to determine the performance of this lightweight reconstruction method on the target end user in an agricultural setting.

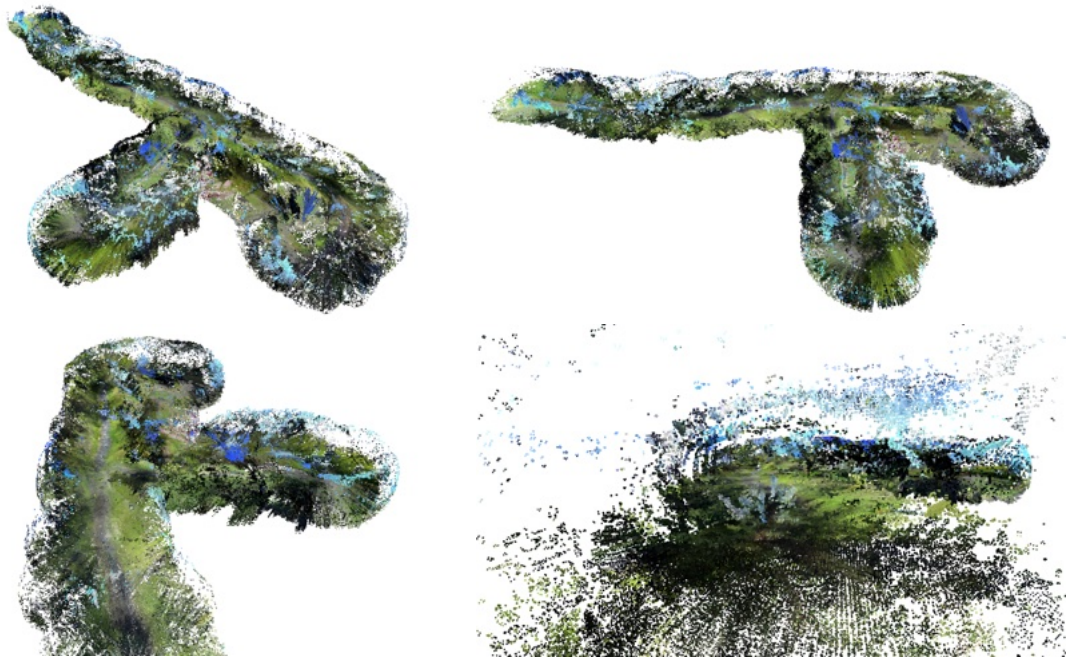


Figure 14.1: Reconstruction on the Zoo using ground truth coordinates and Monodepth as the disparity estimation

14.1 Qualitative Evaluation of the Choice of Disparity Method

The first qualitative evaluation was to use the benchmark disparity methods, Monodepth and stereo SGBM - see Figure 14.1 and Figure 14.2 respectively, to evaluate the impact of a smooth vs piecewise disparity estimate on the reconstruction. The hypothesis was as follows, if the disparity estimations were reliable and smooth enough then they would not be removed by the

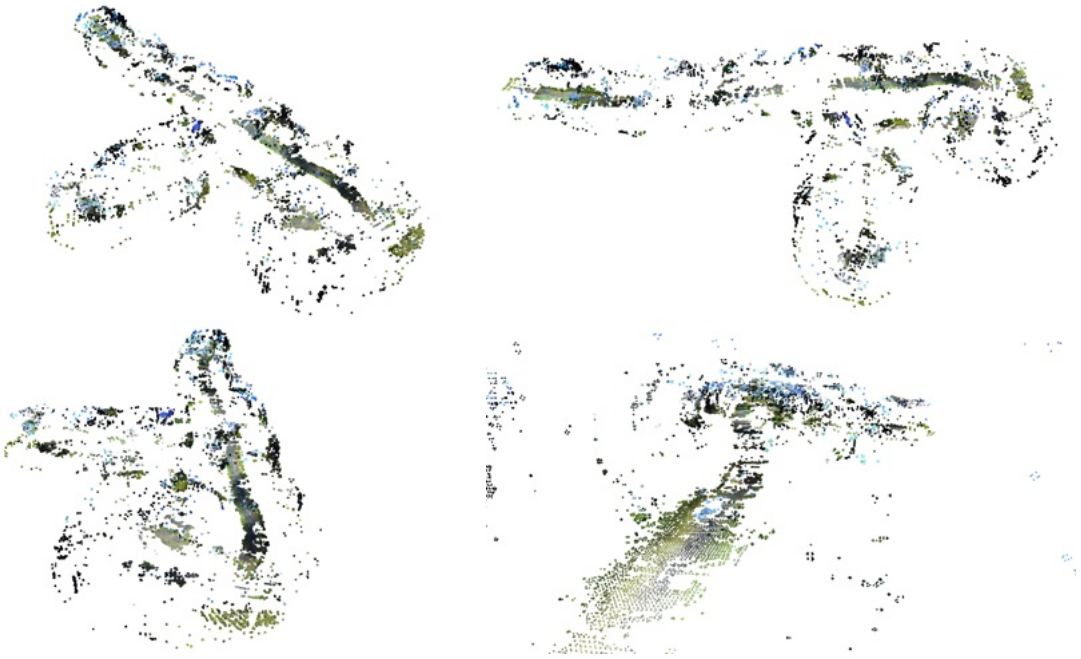


Figure 14.2: Reconstruction on the Zoo using ground truth coordinates and Stereo SGBM as the disparity estimation

regulator when meshing the pointcloud before adding to the TSDF of the reconstruction. If the disparity estimate was not smooth or unreliable, the reconstruction would be sparse or have sections missing. Conversely, a smooth, reliable disparity should produce dense reconstructions.

From the reconstructions in Figure 14.1 and Figure 14.2, the reconstruction created using Monodepth is significantly denser than the one created using stereo SGBM. While this comparison is only evaluating the performance of the depth estimation method on a ground truth VO, it indicates that the smoother disparity estimate obtained by the learned disparity estimation methods is better suited to reconstruction than that of stereo SGBM.

To further test that the driving factor to the sparseness is smoothness, another reconstruction was created using the proposed Mesh Disparity method with the same set of ground truth VO coordinates. The results from this were significantly denser as seen in Figure 14.3, confirming the hypothesis.

To compare these different disparity estimation methods to that of the chosen disparity estimation, Pydnet SGBM with $\alpha_{px} = 0.3$, it was reconstructed using the same set of ground truth coordinates and the results are shown in Figure 14.4.

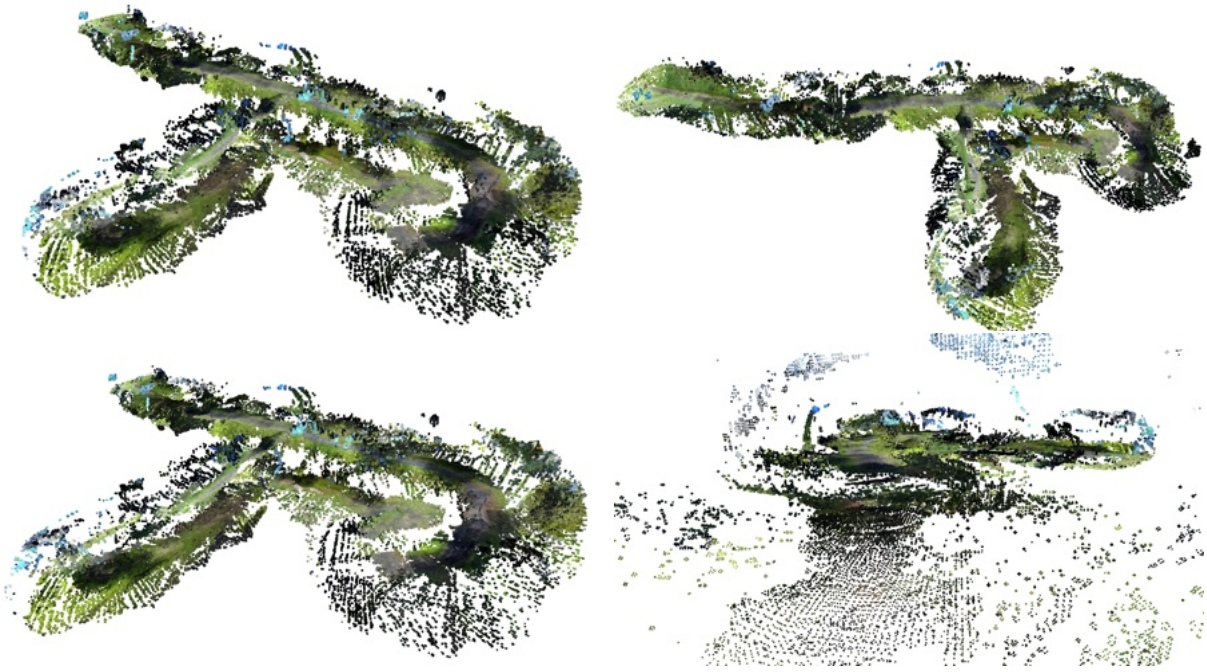


Figure 14.3: Reconstruction on the Zoo using ground truth coordinates and mesh regularised stereo disparity as the method of disparity estimation

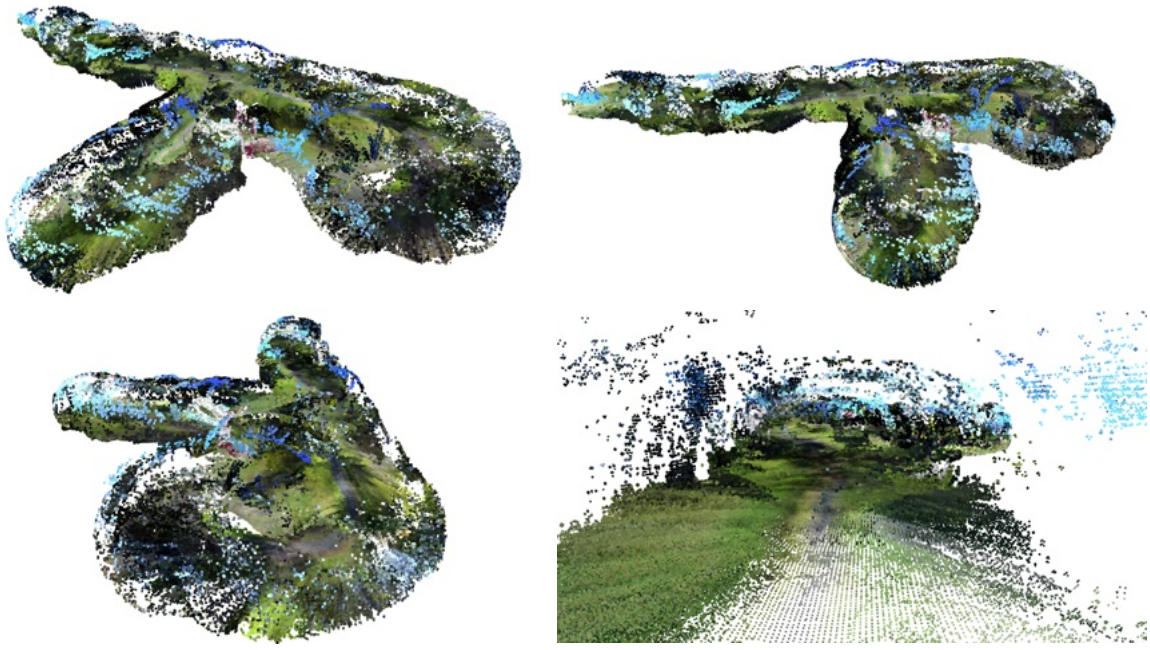


Figure 14.4: Reconstruction on the Zoo using ground truth coordinates and Pydnet SGBM proxy supervised with $\alpha_{px} = 0.3$ as the method of disparity estimation

14.1.1 Qualitative Analysis of the Impact of Disparity Estimation Methods on Reconstruction

From the different reconstructions of the Old Zoo test route, there are clear differences between smooth disparities and those that are not smooth. The reconstruction from SGBM is very sparse in comparison to the others. Smoothing the mesh disparity has increased the density of the reconstruction significantly, however, it is still substantially more sparse than the reconstructions made with Monodepth and Pydnet SGBM. This result from the mesh disparity shows that the accuracy and reliability of the disparity estimate are required for the pointclouds at each frame to be added to the TSDF of the reconstruction without conflict.

Comparing the reconstructions obtained from the two learned depth estimation methods, they are very similar with no immediate differences. Both are extremely dense with the solid reconstruction of the ground plane however, the floor of the Pydnet SGBM proxy method is marginally more detailed and complete. Where the network has given incorrect disparity estimations for the sky, normally due to soft boundaries on the edges of silhouetted objects, it shows clearly as blue streaks in the reconstruction. Using the amount of blue as a qualitative proxy measurement for the amount of correctly classified data, both networks are classifying most points correctly with little blurring on the edges of objects. However, there are still blue voxels in both reconstructions so there is room for improvement.

One limitation that is clear from the reconstructions is that there is little information other than the path directly in front of the vehicle. In future research, this could be improved upon by using side-facing cameras to produce a multi-view pointcloud of the front as well as the sides of the vehicle. This would significantly increase the density of points on the sides of the path while also giving a series of different perspectives of an object as the vehicle passes an object. However, it is noted that this would increase the computational, power and size requirements for the pipeline and therefore classifying this multi-view pipeline as lightweight would be optimistic.

14.2 Quantitative Evaluation of the Choice of Disparity Method

As seen in the previous section, the choice of disparity estimation method has a significant impact on the appearance of the end reconstruction. However, the accuracy of this reconstruction needs to be investigated. To qualitatively compare the effects of the different methods of disparity estimation on the reconstructions, we utilise CloudCompare to align and quantify the error between the reconstruction pointcloud obtained for each disparity method compared to a ground truth pointcloud created using the pointclouds from the lidar. In order for the different reconstructions to be compared, they need to be constructed using the same trajectory in order to avoid differences that might occur from warping and misalignment. For the shared trajectory points, we use LOAM to calculate a ground truth trajectory using the Velodyne points, thereby ensuring that the reconstructions can be aligned for comparison.

After matching the pointclouds, the distances between the corresponding best matching points were computed and placed in a histogram for comparison. A pointcloud was also made with the matched points and coloured by the Euclidean distance between the matches and is shown alongside the error histogram.

Looking at the coloured error maps, the floor of all the reconstructions has little error while the complex geometry on the sides of the path is where the error is concentrated. The majority of the error is concentrated where there are turns in the path. Comparing the error maps for the

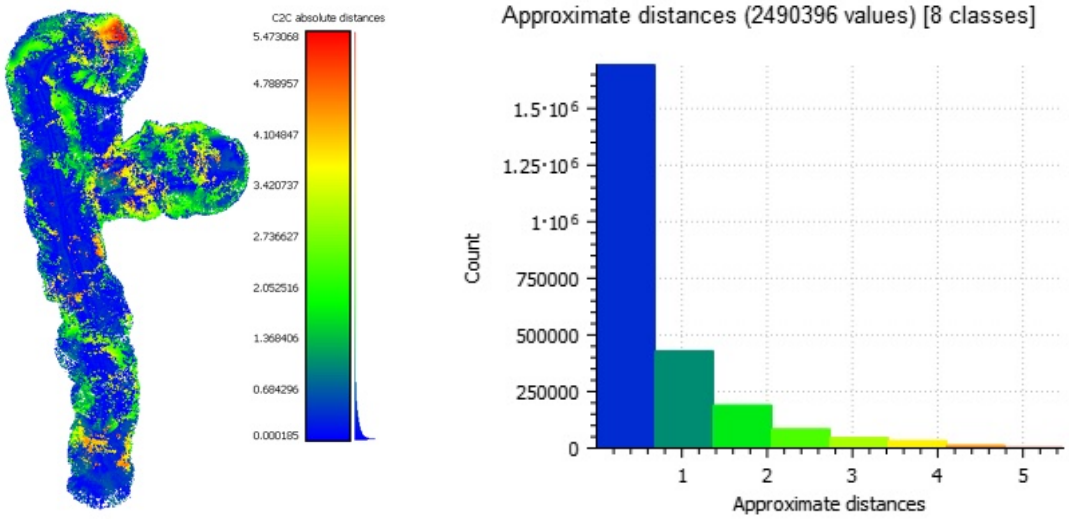


Figure 14.5: Reconstruction Errors on the Old Zoo dataset using MonoDepth for disparity

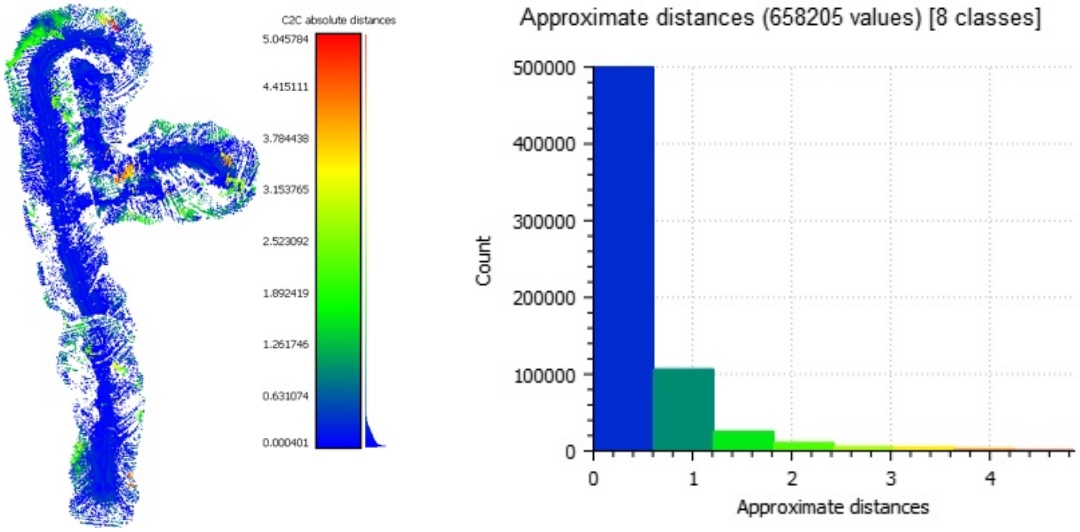


Figure 14.6: Reconstruction Errors on the Old Zoo dataset using Stereo Mesh-Disparity

different methods, stereo SGBM has the best ratio between points with an error <0.5m and the number of points with an error higher than that. Monodepth and Pydnet SGBM proxy are comparable however Monodepth has slightly more points with low error.

Based on the quantitative findings, it is evident that there is minimal distinction in performance between Monodepth and Pydnet SGBM proxy supervised. The main difference is their respective frame rates of 1.2Hz and 9Hz. Consequently, when accounting for the qualitative analysis, it can be concluded that Pydnet SGBM emerges as the superior method for lightweight reconstruction. This assertion is grounded in its equitable performance to Monodepth, coupled with the notable advantages of being nearly 9x faster to compute while possessing a lighter memory footprint.

With these results and confidence in the decision of the disparity estimation method, the final reconstruction pipeline is chosen. Using Pydnet SGBM proxy with $\alpha_{px} = 0.3$ and ORB as the feature selector of the Visual Odometry, the performance of the pipeline can now be evaluated

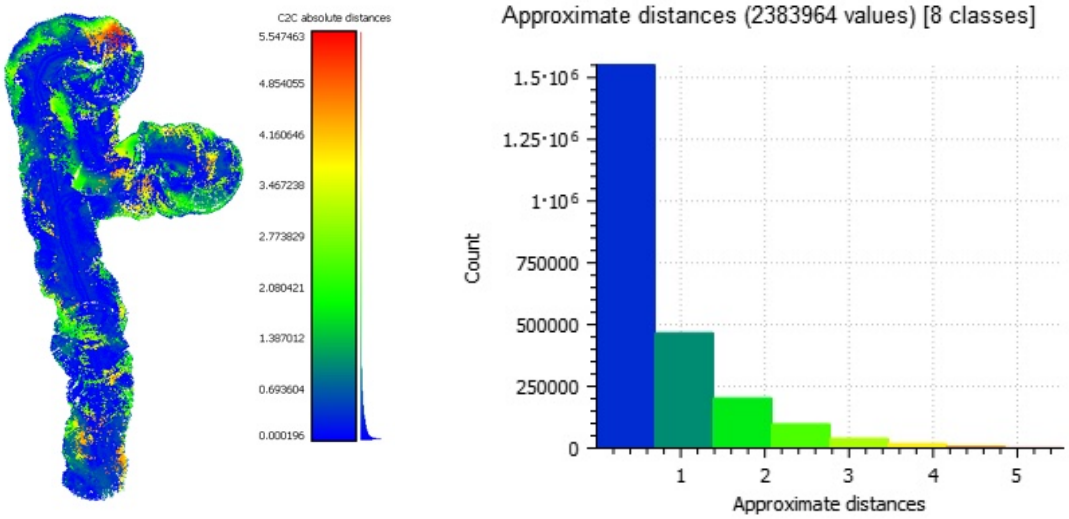


Figure 14.7: Reconstruction Errors on the Old Zoo dataset using Pydnet SGBM Proxy with $\alpha_{px} = 0.3$

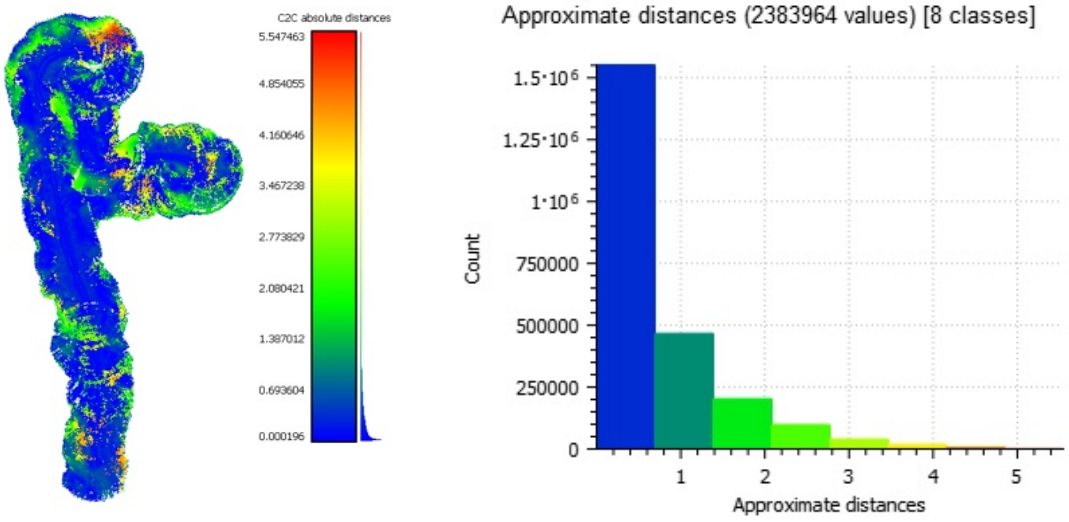


Figure 14.8: Reconstruction Errors on the Old Zoo dataset using Pydnet SGBM Proxy with $\alpha_{px} = 0.3$

on the Lindenhoff Farm to determine its performance in an agricultural environment.

14.3 Performance of the reconstruction pipeline

As for the performance of the final pipeline utilising ORB features and Pydnet SGBM proxy with $\alpha_{px} = 0.3$, a short reconstruction was generated on the Lindenhoff Farm test set. This reconstruction pipeline took 1.59s per iteration from obtaining the images to updating the reconstruction. While this does not achieve the goal of near real-time performance, a large portion of the implementation was done in Python, and as such, significant improvements in speed could be achieved by implementing in C++ or an equivalent compiled language.

The reconstruction was created using 200 image frames at 5hz using pre-computed disparity estimates and the result is shown in Figure 14.9 and Figure 14.10.

Figure 14.9: Detail image showing the test reconstruction pointcloud at the Lindenhoff Farm compared to a reference photo

The components which calculated the Visual Odometry from the tracked 3D points using RANSAC as well as the placement of the 3D points in the TSDF were the most computationally expensive. These components were implemented in C++ and as such, the speed of

these components was not a limiting factor in the overall speed of the pipeline. The limiting factor in this implementation was the disparity estimation (whether using a learned or geometric method), feature extraction and tracking of points. These components were implemented in Python and as such, the speed of these components was significantly slower than the C++ components. Of these, the only component that is difficult to implement in C++ would be disparity estimation using the learned methods. However, if the rest of the pipeline was implemented in C++, the speed of the pipeline would be significantly improved from 0.6Hz to 5Hz end-to-end.
Additionally, if the disparity estimation network could be offloaded to a small GPU it would operate at 40+ Hz (on a Jetson TX2) Therefore, the speed of the pipeline would be further improved to an expected 20Hz. This would be sufficient for the pipeline to be used in real-time on a high-speed robot at a cost of 10-30W depending on load and Jetson variant. [92]

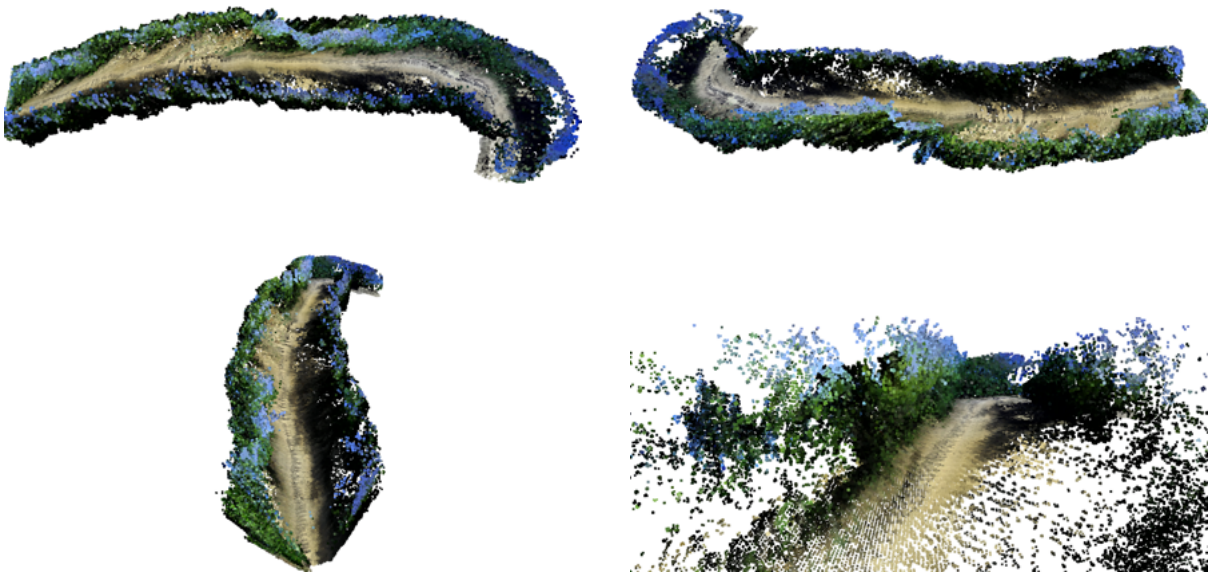


Figure 14.10: Reconstruction on the Lindenhoff Farm using the entire VO pipeline with ORB and Pydnet SGBM Proxy supervised with a weight of 0.3

The images in Figure 14.10 show that the reconstruction is capable of accurately capturing the structure of the environment along the path travelled by the vehicle. The reconstruction is significantly dense, showing the shapes of individual trees, plants, and other features in the environment. Colour is accurately captured from the images and the amount of points from the sky that are included in the reconstruction (indicating inaccurate depths) is relatively low, being comparable to that of the UCT Old Zoo.

While the macro details are accurately captured, the reconstruction is not detailed enough to identify a specific individual plant among the many in the orchard. This is a limitation of the reconstruction and merits further investigation. However, the reconstruction is more than accurate enough to identify spaces between trees, identify where signs are located, and identify the location of the path without any difficulty. This is sufficient for the target end user of the reconstruction, the agricultural robot, as it can identify the path and the spaces between trees to navigate through the orchard.

Identifying individual plants from a reconstruction is however a difficult task and would require a much more detailed reconstruction. For lightweight agricultural robotics, such an expenditure of resources would be wasteful when the use of object recognition on the camera images would

outperform identification from a reconstruction without the need for the additional computational and memory load that a detailed reconstruction would require. Considering the end user and goal of this research, the reconstructions produced are more than adequate to produce a useful reconstruction of the path travelled for use by both human operators as well as other robots. While not exhaustive in capturing minute details, these capabilities underscore the practical utility of the reconstruction in most navigation scenarios that would be required of a robot in an agricultural scenario.

Despite the visual odometry's weakness evident in the curvature of Figure 14.10, the pipeline's commendable performance over short stretches implies practical applicability. The identified room for improvement, particularly in mitigating drift over longer distances, points towards the need to integrate additional information sources. Further research into this area would be required to determine the best method of integration and the most appropriate methodologies to apply. However, a promising avenue of research would be the addition of inertial measurement or wheel odometry to the visual odometry pipeline. This would allow for a more robust visual odometry pipeline that would be able to handle longer distances and more complex environments while reducing the uncertainty that results in the drift that is currently present.

In conclusion, given the constraints of a 256x512px input image size, the absence of additional sensors, and the intentional optimization for lightweight efficiency, the quality of the demonstrated reconstructions stands as a testament to the pipeline's potential to deliver a dense, detailed, and accurate representation of agricultural paths.

Part III

Conclusions

14.4 Reflection on the Research

At the inception of this research, our primary objective was to establish a robust, accurate and lightweight reconstruction pipeline tailored for deployment in the challenging context of an unstructured agricultural environment. Given the increased difficulties in navigating such environments—namely rugged terrains, harsh conditions, as well as increased demands for independence and self-sufficiency—we recognized the necessity for a specialized solution.

In response, we developed a pipeline designed to operate solely on image inputs, conducting in-depth analysis by comparing the performance of Stereo and Monocular sensor setups. Our focus extended to evaluating both geometric and learned depth estimation methods within this context.

The key goals guiding our efforts were:

- Creation of a reconstruction pipeline optimized for unstructured environments, aligning with the core challenge of autonomous navigation in agricultural settings.
- Tuning the pipeline to meet the constraints placed by mobile robots, necessitating optimisation of computational requirements, power consumption, size, and cost.
- Striving for a resultant reconstruction that is not only dense but also presented such that a human operator can immediately recognise the location of a portion of the scene within the overall reconstruction.
- Ensuring that the pipeline is as lightweight as possible while still being geometrically accurate such that it caters to the imperative requirements of autonomous navigation and path planning.

As we delve into the conclusions and summaries of our research, these initial goals and considerations provide a foundational framework for reflecting on the efficacy and implications of our presented reconstruction pipeline in the context of unstructured agricultural robotics.

14.5 Key Insights

Reflection on Visual Odometry (VO)

Our exploration of Visual Odometry (VO) highlighted its potential on a local scale, where it showed that small that small movement estimation resolution is significantly superior to GPS ($\pm$0.2-0.3m vs $\pm$3-5m accuracy). In addition, we demonstrate that VO can be estimated using learned depth estimates in a much simpler VO system yet still provide similar performances to benchmark VO pipelines such as VISO2 [91]. This is a key result as it shows that physically minimalist systems with light computational capabilities can still provide accurate VO estimates when using learned depth estimation methods. However, on a global scale, the trajectories exhibited a divergence from the ground truth, emphasizing the need for a more proficient VO pipeline to enhance the precision of the reconstructed paths by reducing drift. This finding is particularly relevant in the context of agricultural robotics, where the ability to navigate autonomously and accurately is imperative.

Our investigation extended to various feature tracking methods, unveiling their impact on the VO pipeline's performance. We showed that by selecting ORB as the feature tracking method, there is a 17% decrease in the MSE of the relative transform error when compared to the industry standard of SIFT. Furthermore, the integration of learned depth estimation methods over geometric stereo methods showcased their potential to enhance VO accuracy, warranting further exploration into their applicability across diverse VO methodologies.

Reflection on Depth Estimation

Our comprehensive analysis of depth estimation methods, both geometric and learned, yielded new insights into their relative performance. The in-depth analysis of depth estimation methods shows a contrast between geometric and learned approaches. Learned methods consistently outperform their geometric counterparts, showcasing their ability to generate more accurate disparity estimations. Learned methods had a MSE 14.5% lower than geometric methods when comparing the best performing in each category on the Lindenhoff dataset. This is a significant improvement and shows that the use of learned methods is justified in this context. Furthermore, we show that the disparity smoothness renders them particularly suitable for dense reconstructions through an increased VO performance as well as less leakage of depth estimation onto other objects or features such as the sky.

Further investigations into the impact of proxy supervision on the performance of learned depth estimation methods revealed that the use of proxy supervision does not improve the performance of the depth estimation methods in this environment to the same extent that it does in structured environments. However, it did decrease the speed of convergence of the training process which is as expected.

Additionally, we showed that while the geometric methods have lower computational requirements, their inferior performance in terms of accuracy and disparity smoothness renders them less suitable for use in the reconstruction pipeline. Coupled with the increased space, power and weight requirements of stereo cameras, we reason that the use of stereo cameras in a lightweight reconstruction pipeline is not justified.

We also show that the integration of lightweight pyramidal networks presents a resource-efficient alternative to deep networks, performing admirably in unstructured environments. Their ability to generalise to a wide range of environments and their low computational requirements make them a promising candidate for use in lightweight reconstruction pipelines.

Reflection on the Reconstruction Pipeline

In the final section of the results, we demonstrate that the reconstructions output by our proposed pipeline are able to provide a dense reconstruction that is not only geometrically accurate but also easy for a human operator to interperate. This achieves the goal of this research as these reconstructions are able to be used for path planning and navigation in unstructured environments.

On reflection of the goals towards the size, power consumption and cost of the pipeline, we show that the strategic choices in sensor, depth estimation, feature selection, visual odometry and TSDF pointcloud fusion have led to a result that meets the requirements.

In terms of the computational requirements, we identify two main areas that require further

investigation. With Pydnet being the biggest use of resources, 200MB of RAM on at least two cores of the processor, the computational requirements of the disparity estimation are high. However, the disparity estimation is also the most computationally demanding aspect of the pipeline and therefore, reducing these demands will aid the performance towards the goal of real-time onboard reconstruction. We achieve the framerates stated in this work by utilising the Husky platform's i5 processor, however, the computational requirements can be dramatically reduced further if the input resolution to the network is reduced.
The TSDF fusion is also a large consumer of resources, accounting for the second largest use of CPU time. This can be reduced by either reducing the number of tracked points or by computing the TSDF offline after the VO and depth estimation have been completed and the Robot is no longer in use. Offline compute is a viable option if the reconstruction is only used for visualization purposes or by use on other robots and is not required for the operation of the robot in real time. Ultimately, the computational requirements of the pipeline are low enough to run on a low to mid-tier consumer CPU which enables the use of this pipeline on a wide range of robots in their standard, out the factory configurations, without the need for additional hardware.

By strategically opting for a monocular camera as the sole sensor, the pipeline's potential use on a broad spectrum of robots was increased as the only requirements, a monocular camera and a mid-tier cpu, are commonplace and already present. This choice not only reduced system costs ($< $300 for a camera and compute module) but also minimized power consumption to 30-40W and size of the pipeline to a volume of less than 1ℓ. In addition, the system's compact footprint extended its utility to diverse industries beyond the realm of robotics. Industries such as construction, mining, and surveying can all utilise the lightweight, low-cost, and low-power solution it provides in their unstructured or semi-structured environments. Additionally, the choice of sensor facilitates easy integration into pre-existing platforms and vehicles, eliminating the need for custom-built solutions.

In conclusion, our research has not only shed further light on the relationship between all the different components and design decisions of a reconstruction pipeline but has also provided a comprehensive analysis of the performance of the pipeline in an unstructured environment. The strategic choices in sensor selection and computational optimizations position this pipeline as a versatile and cost-effective solution for diverse applications in a variety of unstructured environments.

14.6 Opportunities For Future Research

While the results of this research are promising, there are still many avenues for further investigation and improvement. As observed in the findings, the Visual Odometry aspect of the pipeline is the weak point. While it was initially kept as simple as possible to allow us to study the effects of the depth estimation methods and feature selection, ultimately it proved to be too simplified. The drift at global scale observed when path lengths were increased highlights the need for more robust Visual Odometry (VO) algorithms which can integrate more advanced methods such as loop closures and bundle adjustment. Additionally, exploring sensor fusion techniques, combining visual data with other sensor modalities such as inertial measurement units (IMUs), could contribute to improving global localization accuracy.

While learned depth estimation methods have shown superiority over geometric counterparts, further research can delve into optimizing these methods for specific agricultural scenarios. Investigating the impact of varying environmental conditions, such as different lighting and

weather conditions, on the performance of learned depth estimation models can enhance their robustness. Moreover, exploring real-time adaptation strategies for depth estimation in dynamically changing environments can be pivotal for maintaining accuracy during agricultural operations. This could be as simple as using a more diverse array of times and seasons when assembling a training dataset or as complex as using image generative models to synthesise changes to the environment and then using these synthetic images to train the depth estimation network. Nevertheless, the ability to adapt to changing environments is a key requirement for any depth estimation method to be used in agricultural settings. In addition, since the advent of this research, new generations of disparity estimation networks have been developed which utilise transformer and diffuser architectures. When used for disparity estimation, these networks show promise in terms of accuracy and generalisation and therefore, future research could focus on their implementation in the pipeline.

Alternatively, while IR based depth sensors such as the Kinect camera are not suitable for use in outdoor environments, further research into sparse, few-beam lidars and image-pointcloud fusion could be explored for applications where cost is not a constraint however, accuracy and robustness are. The combination of sensors would provide a more robust and accurate depth estimation than a single camera, and as such, would be more suitable for high accuracy applications.

The computational requirements, particularly for disparity estimation and pointcloud fusion, present further opportunities for efficiency enhancements. Future work could explore model architecture optimizations for the disparity estimation network (and potentially pointcloud fusion), potentially leveraging small hardware accelerators such as TPUs or GPU boards such as the Nvidia Jetson for improved real-time performance at the cost of weight and power. These would allow for higher resolution depth maps to be generated which, in turn, would allow for denser reconstructions at higher frame-rates.
By offloading this compute to a separate board, the main CPU would be free to perform other tasks such as path planning and navigation. This would allow for the pipeline to be used in conjunction with other processes, in real-time on a working robot, if it was implemented as a ROS node or similar.

The pipeline's adaptability to different unstructured environments beyond agriculture presents opportunities for cross-industry applications. Future research could explore its effectiveness in construction, mining, and surveying applications. This research would involve studying the impact as well as tailoring the system for specific industry requirements and expanding its utility where necessary.

In summary, while this research provides a solid foundation, continuous exploration and refinement are essential for the practical implementation of any proccess. The findings of this research have highlighted the potential for this technology to be applied in a variety of unstructured environments. Additionally, we have shown that expensive, heavy or power demanding sensors are not required for the creation of a dense reconstruction. This pushes the boundaries of where modern robotics can be applied to solve everyday problems for people around the world without prohibitive costs. However, as we have shown, there is still much work to be done to improve the accuracy and robustness before this technology can be fully autonomous and reliable in the field.

Bibliography

[1] B. Niemans, "'simple, robust, unlikely to break': a space-inspired earth rover robot," Oct 2018. [Online]. Available: https://www.theguardian.com/environment/2018/oct/20/space-robots-lasers-rise-robot-farmer

[2] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2366–2374, 6 2014. [Online]. Available: https://arxiv.org/abs/1406.2283v1

[4] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," 2017, pp. 270–279.

[5] M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia, "Real-time self-supervised monocular depth estimation without gpu," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[7] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," 2011, pp. 2548–2555.

[8] A. Robotics, "Autonomous mobile robots (amrs) - abb amr/agv robots," May 2023. [Online]. Available: https://new.abb.com/products/robotics/robots/autonomous-mobile-robots

[9] C. Jarrett, "Introducing titan, amazon's new mobile robot that can lift up to 2,500 pounds," Nov 2023. [Online]. Available: https://www.aboutamazon.com/news/operations/amazon-unveils-titan-fulfillment-center-robot

[10] S. Technologies, "Starship technologies: Autonomous robot delivery," Dec 2021. [Online]. Available: https://www.starship.xyz/

[11] P. USA, "Raspberry pi hq camera cs," 2022. [Online]. Available: https://www.pishop.us/product/raspberry-pi-hq-camera-cs

[12] "Puck sales, clearpath — store.clearpathrobotics.com," https://store.clearpathrobotics.com/products/puck, [Accessed 18-11-2023].

[13] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," 2014.

[14] N. J. Wade, "Ocular equivocation: The rivalry between wheatstone and brewster," *Vision*, vol. 3, no. 2, p. 26, 2019.

[15] V. Nityananda and J. C. A. Read, "Stereopsis in animals: evolution, function and mechanisms," *Journal of Experimental Biology*, vol. 220, no. 14, pp. 2502–2512, 07 2017. [Online]. Available: https://doi.org/10.1242/jeb.143883

[16] J. Hecht, "Beam: The race to make the laser," *Opt. Photon. News*, vol. 16, no. 7, pp. 24–29, Jul 2005. [Online]. Available: https://www.optica-opn.org/abstract.cfm?URI=opn-16-7-24

[17] X. Sun, "Space-based lidar systems," in *2012 Conference on Lasers and Electro-Optics (CLEO)*, 2012, pp. 1–2.

[18] D. E. Smith, M. T. Zuber, G. A. Neumann, and F. G. Lemoine, "Topography of the moon from the clementine lidar," *Journal of Geophysical Research: Planets*, vol. 102, no. E1, pp. 1591–1611, 1997.

[19] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, pp. 1147–1163, 2015.

[20] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam," *IEEE Transactions on Robotics*, vol. 37, pp. 1874–1890, 7 2020. [Online]. Available: http://arxiv.org/abs/2007.11898http://dx.doi.org/10.1109/TRO.2021.3075644

[21] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," 2015, pp. 1935–1942.

[22] W. N. Greene and N. Roy, "Metrically-scaled monocular slam using learned scale factors," 2020, pp. 43–50.

[23] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," 2014, pp. 834–849.

[24] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system," 11 2020. [Online]. Available: http://arxiv.org/abs/2011.11357

[25] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," 2014, pp. 477–491.

[26] W. N. Greene, K. Ok, P. Lommel, and N. Roy, "Multi-level mapping: Real-time dense monocular slam," 2016, pp. 833–840.

[27] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, pp. 1052–1067, 2007.

[28] A. Rosinol, "Densifying sparse vio: a mesh-based approach using structural regularities," 10 2018. [Online]. Available: https://www.research-collection.ethz.ch/handle/20.500.11850/297645

[29] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: from slam to spatial perception with 3d dynamic scene graphs," *International Journal of Robotics Research*, vol. 40, pp. 1510–1546, 1 2021. [Online]. Available: https://arxiv.org/abs/2101.06894v3

[30] M. Yokozuka, S. Oishi, T. Simon, and A. Banno, "Vitamin-e: Visual tracking and mapping with extremely dense feature points," 2019. [Online]. Available: https://youtu.be/yfKccCmmMsM

[31] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6243–6252.

[32] F. Bartoccioni, Éloi Zablocki, P. Pérez, M. Cord, and K. Alahari, "Lidartouch: Monocular metric depth estimation with a few-beam lidar," 9 2021. [Online]. Available: http://arxiv.org/abs/2109.03569

[33] D. Doria and R. J. Radke, "Filling large holes in lidar data by inpainting depth gradients," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2012, pp. 65–72.

[34] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," 9 2017. [Online]. Available: http://arxiv.org/abs/1709.07492

[35] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6713–6719.

[36] H. Durrant-Whyte and T. Bailey, ""simultaneous localisation and mapping (slam): Part i the essential algorithms"," *Robotics and Automation Magazine*, vol. 13, 01 2006.

[37] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the atlas framework," *The International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[38] P. Bettinger, K. L. Merry *et al.*, "Influence of the juxtaposition of trees on consumer-grade gps position quality." *Math. Comput. For. Nat. Resour. Sci.*, vol. 4, no. 2, pp. 81–91, 2012.

[39] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., September 1980.

[40] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004.

[41] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.

[42] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 1403–1410, 2003.

[43] H. Jin, P. Favaro, and S. Soatto, "Real-time 3-d motion and structure of point features: a front-end system for vision-based control and interaction," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 778–779, 2000.

[44] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20184

[45] C. Harris and M. Stephens, "A combined corner and edge detector," 1988.

[46] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," 2006, pp. 404–417.

[47] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571, 2011.

[48] R. Wang, M. Schworer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3903–3911.

[49] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[50] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: from software to hardware," *International Journal of Optomechatronics*, vol. 2, pp. 435–462, 2008.

[51] M. Bleyer and C. Breiteneder, *Stereo matching—State-of-the-art and research challenges*. Springer, 2013.

[52] M. Bleyer and M. Gelautz, "A layered stereo matching algorithm using image segmentation and global visibility constraints," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 59, pp. 128–150, 5 2005.

[53] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," 2007, pp. 1–8.

[54] J. Zbontar and Y. Lecun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, pp. 1–32, 2016.

[55] W. N. Greene and N. Roy, "Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs," 2017, pp. 4696–4704.

[56] J. Pilbrough and P. Amayo, "What's best for my mesh? convex or non-convex regularisation for mesh optimisation," 2021.

[57] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *Journal of Sensors*, vol. 2016, 2016.

[58] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," *Advances in Neural Information Processing Systems*, vol. 18, 2005.

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

[60] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," 2017, pp. 1851–1858.

[61] B. Li, Y. Dai, H. Chen, and M. He, "Single image depth estimation by dilated deep residual convolutional neural network and soft-weight-sum inference," *arXiv preprint arXiv:1705.00534*, 2017.

[62] C. Godard, O. M. Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," 2019, pp. 3828–3838.

[63] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2024–2039, 2 2015. [Online]. Available: http://arxiv.org/abs/1502.07411http://dx.doi.org/10.1109/TPAMI.2015.2505283

[64] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," 2015. [Online]. Available: https://www.blender.org/

[65] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[66] Y. Zhang and T. Funkhouser, "Deep depth completion of a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 175–185.

[67] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "Towards real-time unsupervised monocular depth estimation on cpu," 2018, pp. 5848–5854.

[68] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[69] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," 2018, pp. 7286–7291.

[70] S. L. Phung and A. Bouzerdoum, "A pyramidal neural network for visual pattern recognition," *IEEE transactions on neural networks*, vol. 18, no. 2, pp. 329–343, 2007.

[71] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935.

[72] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.

[73] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *Ieee Access*, vol. 9, pp. 82 031–82 057, 2021.

[74] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," 2020.

[75] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.

[76] "ZED 2 - AI Stereo Camera — stereolabs.com," https://www.stereolabs.com/zed-2/, [Accessed 18-11-2023].

[77] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, pp. 259–268, 1992.

[78] A. Chambolle and P. L. Lions, "Image recovery via total variation minimization and related problems," *Numerische Mathematik*, vol. 76, pp. 167–188, 1997. [Online]. Available: https://link.springer.com/article/10.1007/s002110050258

[79] R. Ranftl, K. Bredies, and T. Pock, "Non-local total generalized variation for optical flow estimation," 2014, pp. 439–454.

[80] D. T. Lee and B. J. Schachter, "Two algorithms for constructing a delaunay triangulation 1," *International Journal of Computer and Information Sciences*, vol. 9, 1980.

[81] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015.

[82] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: learning 3d scene structure from a single still image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, pp. 824–840, 2009. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/19299858/

[83] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," 12 2018. [Online]. Available: http://arxiv.org/abs/1812.11941

[84] A. B. Owen, "A robust hybrid of lasso and ridge regression," *Contemporary Mathematics*, vol. 443, pp. 59–72, 2007.

[85] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, and G. Research, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 3 2016. [Online]. Available: https://arxiv.org/abs/1603.04467v2

[86] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[87] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," 2006, pp. 430–443.

[88] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," pp. 778–792, 2010. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-15561-1_56

[89] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1281–1298, 2011.

[90] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," 2012.

[91] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 963–968.

[92] H. V. Pham, T. G. Tran, C. D. Le, A. D. Le, and H. B. Vo, "Benchmarking jetson edge devices with an end-to-end video-based anomaly detection system," 2023.

Appendices

Dataset Example Images

Examples of the dataset collected at Lindenhoff farm.

Figure A.1: Set of images sampled from the Lindenhoff training and validation dataset. This set was used for training the network and validating performance before testing on the test dataset.

Figure A.2: Images of the test dataset from Lindenhoff farm. This dataset was used to evaluate the performance of the methods presented in this work.

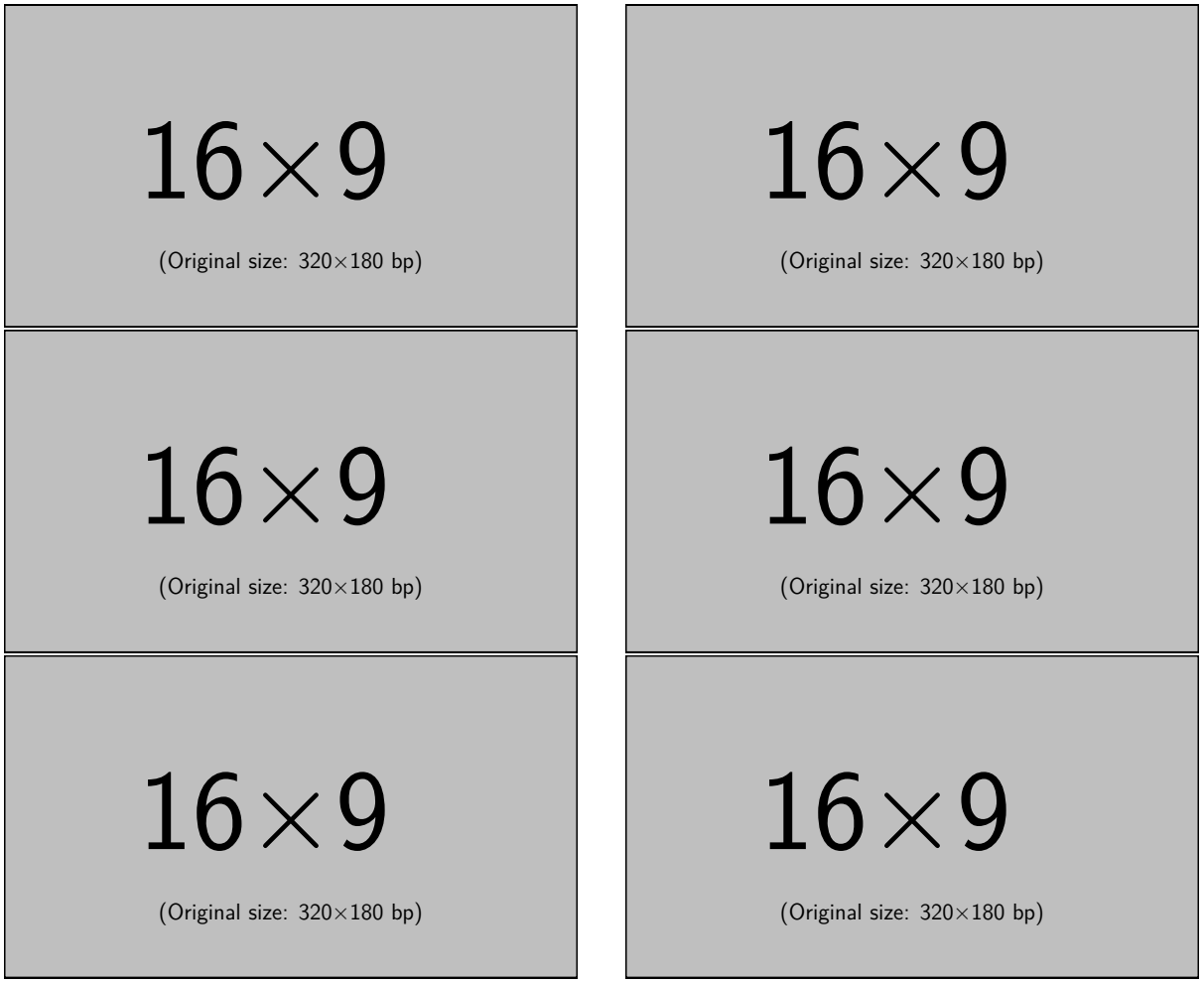


Figure A.3: Set of images from Lindenhoff farm used to test the generalisation of the algorithms to other crop types. Vineyards were used for this dataset.

Full Visual Odometry Study Results

Results for all the combinations of VO runs are shown in the following tables

Table B.1: Table of Absolute Trajectory Errors for VO on UCT Old Zoo Dataset

Features	Depth Model Mean ATE [m]	σ ATE [m]	MSE ATE	Mean Normalised ATE [%]	σ Normalised ATE [%]	
Brisk	meshDisp FullSized	0.573095	0.903212	1.144229	-408.169759	13085.225740
Brisk	meshDisp	0.664421	1.126169	1.709712	-1376.909434	86625.136421
MOC	meshDisparityFullSize	0.631340	1.061967	1.526363	-732.142680	31475.501848
MOC	meshDisparityTrainingSize	2.754691	4.877062	31.374058	-14452.350742	931244.078677
MOC	MonodepthZoo50e	0.591839	1.010194	1.370766	-208.766300	10390.492365
MOC	PydnetMeshDisp08WeightZoo50e	0.663102	1.121662	1.697830	-981.974713	35368.424713
MOC	PydnetSGBMDisp03WeightZoo50e	0.655860	1.111427	1.665423	-1384.281059	65147.822607
MOC	PydnetZoo50e	0.657014	1.107912	1.659138	-781.864970	34790.223232
MOC	sgbmDisparityFullSize	0.598316	1.084081	1.533213	-618.756404	29456.904853
MOC	sgbmDisparityTrainingSize	0.670073	1.139308	1.747021	-2770.864142	151569.098870
ORB	meshDisparityFullSize	0.693426	1.217809	1.963898	-732.882268	37191.318389
ORB	meshDisparityTrainingSize	0.619995	1.156748	1.722460	-358.474145	18140.783362
ORB	MonodepthZoo50e	0.611456	1.116020	1.619379	-90.553320	7088.119379
ORB	PydnetMeshDisp08WeightZoo50e	0.638448	1.160386	1.754113	-633.695532	40596.590310
ORB	PydnetSGBMDisp03WeightZoo50e	0.586027	1.122974	1.604498	-209.834633	20800.833214
ORB	PydnetZoo50e	0.628325	1.133534	1.679692	-451.658291	28224.917015
ORB	sgbmDisparityFullSize	0.571350	1.092002	1.518910	-322.623486	26216.091608
ORB	sgbmDisparityTrainingSize	0.640536	1.177319	1.796366	-354.504654	15986.224656
SIFT	meshDisparityFullSize	1.002716	1.472267	3.173009	-1078.824967	55745.327080
SIFT	meshDisparityTrainingSize	2.576067	5.192442	33.597574	-5607.834855	216658.676111
SIFT	MonodepthZoo50e	0.635387	1.122186	1.663020	-319.059079	10042.366076
SIFT	PydnetMeshDisp08WeightZoo50e	0.906661	1.324302	2.575809	-1031.507340	44532.395749
SIFT	PydnetSGBMDisp03WeightZoo50e	0.904003	1.462790	2.956976	-1123.915507	44814.243483
SIFT	PydnetZoo50e	0.802981	1.222921	2.140315	-741.730021	26707.312683
SIFT	sgbmDisparityFullSize	0.586481	1.031069	1.407064	-499.431794	21435.189330
SIFT	sgbmDisparityTrainingSize	1.007603	1.617847	3.632693	-4366.493674	252418.460595

Table B.2: Table of Absolute Trajectory Errors for VO on the Lindenhoff Farm dataset

Features	Depth Model	ATE [m]	σ ATE [m]	MSE ATE	Normalised ATE [%]	σ Normalised ATE [%]
Brisk	meshDisp	0.752780	1.100127	1.776958	615.643084	35182.728386
Brisk	meshDisp	0.738010	0.994035	1.532765	208.989012	17357.219176
MOC	PydnetFarm50e	0.638502	0.788921	1.030082	482.495207	38267.820433
MOC	PydnetSGBM03Proxy50e	0.613081	0.727161	0.904630	469.827060	33083.193656
MOC	PydnetSGBM06Proxy50e	0.732771	0.917472	1.378709	319.426101	23502.115712
MOC	meshDisparityFullSize	0.489592	0.696152	0.724329	70.966317	4633.027768
MOC	meshDisparityTrainingSize	0.920928	1.169870	2.216704	1163.123753	84199.941644
MOC	Monodepth50eFarm	0.890749	0.993652	1.780778	781.475819	55249.864406
MOC	sgbmDisparityFullSize	5.645116	9.827377	128.444669	2161.470293	183788.604088
MOC	sgbmDisparityTrainingSize	0.671581	0.751701	1.016075	318.782279	10589.065201
ORB	PydnetFarm50e	0.459822	0.678396	0.671657	121.972882	10760.089040
ORB	PydnetSGBM03Proxy50e	0.458459	0.667416	0.655628	-48.599497	10440.203696
ORB	PydnetSGBM06Proxy50e	0.441595	0.630775	0.592884	102.628537	6673.775457
ORB	meshDisparityFullSize	0.482803	0.735208	0.773629	75.675670	2077.963671
ORB	meshDisparityTrainingSize	0.476300	0.712557	0.734598	86.214540	3536.023659
ORB	Monodepth50eFarm	0.477588	0.673182	0.681264	100.840612	6758.961514
ORB	sgbmDisparityFullSize	0.491502	0.820027	0.914018	168.549041	2304.093658
ORB	sgbmDisparityTrainingSize	0.556740	0.778407	0.915876	176.915260	5323.694079
SIFT	PydnetFarm50e	0.961134	1.173568	2.301042	386.582979	41793.526751
SIFT	PydnetSGBM03Proxy50e	0.862368	1.042895	1.831308	603.593942	48269.164106
SIFT	PydnetSGBM06Proxy50e	0.987748	1.109438	2.206497	207.013153	29606.628539
SIFT	meshDisparityFullSize	0.551177	0.761895	0.884279	171.923515	9516.616579
SIFT	meshDisparityTrainingSize	1.407393	2.832700	10.004946	1171.872554	101592.762123
SIFT	Monodepth50eFarm	0.923999	1.041794	1.939108	101.391810	19385.629422
SIFT	sgbmDisparityFullSize	0.504119	0.748537	0.814445	262.433491	13663.597044
SIFT	sgbmDisparityTrainingSize	1.022438	1.207992	2.504624	255.155311	11968.106494