3D Privacy Masking using Monocular Depth Estimation

Mattias Lundström

Jakob Pettersson

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6168
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2022

Abstract

This thesis strives to dive deeper within the area of Monocular Depth Estimation, approximating distance information from one single image using deep neural networks. It introduces a thorough evaluation and analysis of state-of-the-art depth estimation models regarding proposed aspects of relevance for downstream video applications, specifically in a surveillance domain. This leads to three custom data sets where two include ground truth depth data. Results on accuracy, temporal inconsistency, and range resolution is presented and analysed utilising the collected data sets, with selected metrics. It is concluded that the accuracy performance of the models, even though impressive, is also highly scene dependant. Regarding temporal inconsistency, which causes apparent video instability, it is concluded to be a prominent concern for typical downstream video applications that calls for further attention. This leads to a proposed minor post-processing step, with promising results.

Furthermore, this thesis also presents a novel end-to-end algorithm referred to as ”3D Privacy Masking”. Privacy masking is a typical task in camera surveillance, where a certain region of the image scene needs to be anonymised. This functionality is here extended by including depth, such as that from monocular depth estimation, resulting in a depth aware privacy mask. Thereby, events in front of the mask as seen by the camera can still be observable. The suggested algorithm and proof-of-concept application also includes a stabilising technique to account for sub-perfect depth data. Conclusively, this thesis showcases the potential of monocular depth estimation in downstream computer vision tasks, like that of 3D privacy masking, and proposes continued directions forward.

Acknowledgements

We would want to direct our thanks and gratitude to people that have helped and guided us along the way. Without any of you, the journey would have been much less enjoyable and the final result of much less quality. First off, we would like to direct a big thank you to our faculty advisers Anders Robertsson and Björn Olofsson, from the Department of Automatic Control at Lund University. You have guided us in decision makings as well as asserting that we had an academic and scientific reasoning throughout the thesis. Additionally a big thank you for your proof-reading of the report and all the rewarding discussions we have had along the way. Secondly, we direct our thanks to the examiner of the thesis, Karl-Erik Årzén.

Moreover, many thanks to all the people at Axis Communications, and particularly the people at Core Tech Graphics department, that have made certain that we feel welcome and shown great support and interest in our work and this thesis. Special thanks goes to our supervisor, Joakim Ericson, who have provided us with guidance, connections and resources. Additionally, a big thank you to Mikael Lindberg, Yuan Song, Joel Sjöbom and Jiandan Chen, who have regularly stayed connected with us throughout the thesis and guided us with valuable input and expertise. Lastly, a thank you to Niklas Hansson, for being helpful and providing us with fancy tools and equipment that made this thesis possible.

Contents

List of Acronyms

CNN	**c**onvolutional **n**eural **n**etwork
CPU	**c**entral **p**rocessing unit
FPS	**f**rames **p**er **s**econd
GDPR	**g**eneral **d**ata **p**rotection **r**egulation
GPU	**g**raphics **p**rocessing **u**nit
LiDAR	**l**ight **d**etection **a**nd **r**anging
MDE	**m**onocular **d**epth **e**stimation
MDEs	**m**onocular **d**epth **e**stimators
POV	**p**oint **o**f **v**iew
RGB	**r**ed **g**reen **b**lue
RNN	**r**ecurrent **n**eural **n**etwork
SfM	**s**tructure **f**rom **m**otion
ViT	**v**ision **t**ransformer

1

Introduction

This chapter serves to introduce the topics of the project. By describing where current research and industry is, regarding Monocular Depth Estimation and privacy masking, and why this thesis is suggested, the reader should by the end of this chapter be able to comprehend the material and the ultimate purpose of this thesis. Firstly, we introduce the background and purpose, discussing the prospects of utilising resent research of Monocular Depth Estimation, and why it would be favourable to include distance information in surveillance video applications, such as privacy masking functionality. Secondly, we present the problem formulation, by listing what we aim to accomplish from the development work and analysis. This is with the consideration of the defined project limitations listed in the subsequent section. Thirdly, a description of the report outline is included. Finally, references to related work are reviewed, as well as connecting this to how the thesis contributes scientifically.

1.1 Background and Purpose

Acquiring distance to objects in an image has long been a task that typically requires either external sensors, stereo cameras, or careful examination of visual cues. In computer vision, this task is termed *depth estimation*, which is a crucial step towards inferring scene geometry [Vasiljevic et al., 2019]. There is extensive literature on depth estimation using methods relying on multiple viewpoints, controlled lighting conditions [Forsyth and Ponce, 2012], and even controlled focal length exploiting depth cues from the appeared image focus [Nayar and Nakagawa, 1994]. However, to the best of our knowledge no such method have satisfactory result utilising only a single monocular camera.

Stemming from the groundbreaking result and milestone of AlexNet in 2012, and later ResNet in 2015 [Alom et al., 2018], deep-learning approaches have become the *de facto standard* solution for complex image classification tasks. Since then, there has been a surge of research addressing the task of pixel-wise depth estimation

from monocular images with deep-learning techniques [Godard et al., 2017; Casser et al., 2019; Ranftl et al., 2020; Yin et al., 2019], with progressively better and more impressive results. This research has also to a large extent been fueled by the automotive industry race towards autonomous capabilities, where high-quality depth prediction would add valuable redundancy to depth sensors such as radars and LiDARs, or even eliminate the need for expensive sensors completely [Godard et al., 2019].

Although recent progress in the field of Monocular Depth Estimation is very notable, in many cases producing outputs very closely resembling the output you would expect from expensive depth sensors, state-of-the-art depth estimator models are still very fragile and can sometimes fail completely to discern object depth [Ranftl et al., 2020].

Two such identified issues are the temporal inconsistency between frames when these models are tasked to continuously process inputs from a video stream, and the fact that small variations in the image scene can induce highly different depth outputs. A third issue, which is present in most learning based methods in computer vision, is the inherent vulnerability to unseen scene characteristics and contexts.

Furthermore, while most auxiliary tasks of monocular depth estimation currently fall into the domain of self-driving cars, robotics and AR-composition from handheld devices, little research has been carried out in the surveillance domain. This means that the effectiveness of state-of-the-art depth estimators on image characteristics typical for security cameras remains largely unexplored, and use-cases combining security imaging with pixel-wise depth information are still in its infancy.

Today, one typical task in camera surveillance is to perform privacy masking, where a certain area of the image scene needs to be anonymised. This is normally achieved by defining a polygon in image space and hide its pixel content, for example by overlaying a blurred or blank image. However, one large drawback of such classic 2D privacy mask is that all information between the camera and the anonymised object or area is lost. For example, consider a department store that wants to monitor its entrance and exit, but due to privacy and integrity reasons cannot film outside its property boundary. The department store must then either compromise with viewing areas or implement a 2D privacy mask. However, with a classic 2D privacy mask a person or object in front of the exit is then also anonymised, resulting in a loss of valuable information. Because of this, it is very compelling to be able to have a system that can measure depth and the full scene geometry in order to perform masking in 3D world space instead of solely on the image plane. In the example mentioned, the person or object that is located inside the store would remain visible, while objects outside would be masked. We refer to this technique as *3D privacy masking*. See Figures 1.1 and 1.2 for illustrations of 2D masking compared to 3D

masking, respectively.

In this thesis, we explore the prospects of developing this new type of privacy masking in 3D space, utilising the surge of recent work in the area of monocular depth estimation. We also analyse current state-of-the-art models and compare them on metrics such as accuracy, stability, depth range resolution and how different scene properties may affect the quality of the produced depth map. This is done from a surveillance perspective since, to our knowledge, this combination of monocular depth estimation using deep-learning is mostly an unexplored area. It is an area where many use-cases and applications could prosper in the near future, with more edge devices now being able to host and power heavy AI and machine learning applications.

Use Cases

There are many use-cases for 3D masking functionality that warrant such an invention. Integrity and privacy are becoming more relevant and important along the swift progress of technologies, with more and more surveillance cameras entering the market. Today, a typical use case for a regular privacy mask is to hide sensitive information. This could be information that is illegal or intricate to film because of laws and regulations, or it could be signs, logos, or sensitive information in a scene that you do not want to capture on a recording. Furthermore, it might be allowed to film in a certain area, but it could come with regulations or certain conditions that render it ill-suited/non-profitable (such as handling sensitive personal data, certain storage requirements, access restrictions, etc.). Below is a short list of imaginable use-cases that are possible with technique that allows a privacy mask to be depth aware.

- **Window masking**: Allow windows, for example in residential buildings or ground-level boutiques, to be masked in public places without loosing valuable information. Keeps the integrity but allows the capture of events in front of the windows.
- **Entree monitoring**: Allow a public-facing camera to be placed more freely, for example from inside a retail store/commercial building or a parking garage. Allows capture of events that happen close to the entree, but keeps the integrity of the public outside.
- **Property monitoring**: Allows better monitoring of property, for example with a wide-angle, public facing camera, without risking integrity violations. Areas and events that occur out of the property border are masked, but events inside the property are visible, even if in front of the masked region.
- **Video Conference Privacy**: Allow a webcam to blur/hide background content based on its distance. Today, typical webcam filters blur every pixel not

deemed belonging to a person. This hides objects that a user may want to remain visible, for example showcasing a drawing/object during a video conference.

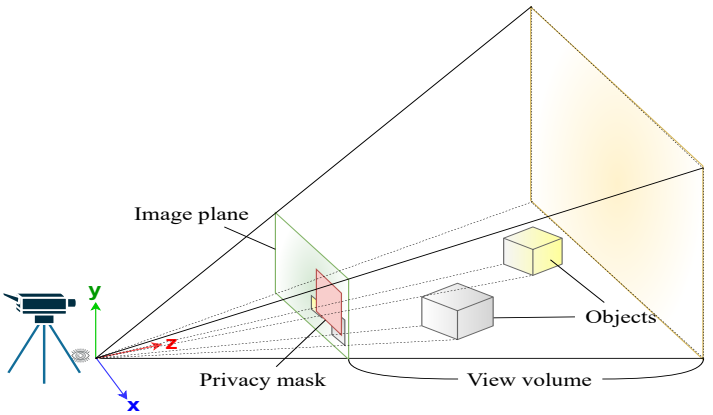


Figure 1.1 Illustration of regular 2D masking. Notice that the red privacy mask in the image plane is in front of both objects.

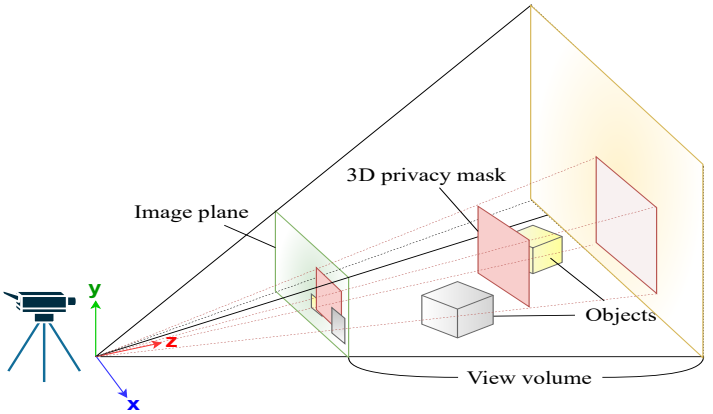


Figure 1.2 Illustration of 3D masking. Notice that the projected 2D mask in the image plane is in front of the yellow object, but behind the other.

1.2 Problem Formulation

The main goal of this thesis is to investigate if, and how, current monocular depth estimators can be used to create 3D masking functionality on surveillance cameras with satisfactory performance.

This translates into several sub-goals. Firstly, this thesis aims to propose a complete end-to-end 3D masking algorithm, taking a depth map and a colour image (RGBD) as input, that is able to perform in close to real-time, i.e., 30 frames per second (FPS), on a desktop device. This algorithm should be robust and perform satisfactory even on sub-perfect depth data, as long as sufficient depth resolution is available. Robustness and satisfactory performance is here defined as ”hide-rather-than-show”, i.e., privacy should not be compromised, while any event in front of the mask should be observable and understandable.

Secondly, the thesis also aims to evaluate performance of state-of-the-art depth estimators and identify domains and properties that result in accurate depth maps as well as weaknesses and domains that result in inaccuracies. In other words, we aim to find the vital functionality, with 3D privacy masking in mind, and propose quantitative evaluation of these functionalities.

To conclude, this thesis aims to demonstrate 3D masking capabilities by showcasing real examples from various environments together with a proposed algorithm. It will, by presenting graphs and data in various forms, try to highlight differences between the depth models themselves. It will also highlight issues, and suggest a mitigation strategy, that may arise when performing monocular depth estimation on continuous frames, so that such issues can be avoided, mitigated and/or improved upon.

1.3 Project Limitations

This thesis has an extensive problem description, and therefore, several limitations in the project scope have been made in order to keep the project clear and concise and without compromising the project deadline. Another important aspect for limiting the project scope is to be able to spend more time on evaluating and analysing interesting phenomena more thoroughly. These limitations are related to model training, weather effects, GDPR, and hardware utilisation on edge devices.

Firstly, the thesis will not fine-tune the models by training on self-collected data sets. While it would be pleasant to be able improve the accuracy on specific types of surveillance scenes by model-training, our work will instead focus on general evaluation and analysis of these methods, and leave model fine-tuning as a subsequent future task. It is already known that these kind of regression models can to some degree be fine-tuned on specific domains, introducing scene biases improving the accuracy on that domain slightly. Also, since the field of monocular depth estimation is rapidly progressing and state-of-the-art models annually being replaced or updated, we reason it unnecessary to fine-tune models until it is ready for actual deployment, and this is not the end-goal of this thesis.

This thesis will also not perform metrics evaluation regarding various weather effects, such as rain, snow or fog. Instead, it targets environments in their natural state such as outdoor scenes during midday and indoor scenes with realistic lighting.

Due to GDPR, General Data Protection Regulation, our dataset used for evaluation and development will contain only limited scenes in controlled environment, where the subjects only include the thesis authors and other notified subjects. While typical surveillance scenes such as town squares or streets, inside department stores, gated property and more, would be of interest for assessment, those kind of scenes were deemed too difficult to collect due to integrity reasons and/or project scope. Instead scenes that have similar point of view in a controlled simulated surveillance environment has been collected.

Furthermore, while the thesis aims to showcase the possibility of using monocular depth estimation models and use-cases such as the proposed 3D privacy mask on edge devices, e.g., on embedded camera hardware, the research and final demonstrations will be performed or recorded on a desktop PC with modern GPU and CPU capabilities. The ambition is that the reached result, performance-wise, will be able to be an approximated benchmark or reference for other embedded devices, each with different compute capabilities, requirements and limitations.

1.4 Report Outline

The project process in this thesis can be divided into three distinct phases. These phases are largely represented in the report structure.

The first phase, **planning and prestudy**, is contained in Chapter 2 and Chapter 3. Relevant theory of both classical and modern depth estimation is first presented in Chapter 2, followed by an in-depth presentation and motivation of selected models in Chapter 3.

The next phase, **masking development**, is contained in Chapter 4 where the concept and design specification of 3D masking are first introduced. Then various techniques and ideas are explored to iteratively develop a robust algorithm to perform 3D privacy masking.

Finally, the concluding phase is **evaluation and assessment** which is presented in Chapter 5. Here evaluation of own-recorded data sets is performed using multiple metrics, with the assistance of depth sensors such as LiDAR and a stereo-camera, as well as the presentation and evaluation of the proposed stabilisation post-processing technique.

Everything is then summarised in Chapter 6 into a conclusion and a brief outlook

of future work.

1.5 Related Work

Inferring 3D geometry information from single-view depth, where monocular depth estimation being one part of it, has historically proven useful in different computer vision tasks and applications. This includes the following: 2D to 3D conversion to display images or video in 3D using VR or 3D glasses [Xie et al., 2016], room re-construction estimation in computer-aided design (CAD) [Izadinia et al., 2017], converting video into 3D in cinematography [Phan and Androutsos, 2014], augmented reality [Liu et al., 2018], and in, maybe most prominently, robot navigation [Mancini et al., 2018] and autonomous driving [Godard et al., 2017].

The work on static privacy masking of a region or area to keep private is usually solved by defining a polygon in the image, in which all internal pixels are masked out [Axis Communications, 2021]. Including depth in privacy masking is not as well established, yet not completely unexplored. Depth data from monocular depth estimations have been used as a way of reconstructing a scene in 3D, in order to keep the overall information about scene flow and characteristics, but at the same time keep all personal information private [Conde Moreno, 2019]. The idea of including depth in a static privacy mask, as a way of anonymising regions covered by a mask in 3D, has conceptually been thought of in various patent applications [Takeyuki Goto et al., 2009; Berlic and Strandevall, 2012].

Evaluation and comparison of monocular depth estimation (MDE) models is as well not unexplored. The authors of [Koch et al., 2018] released the iBims-1 data set in 2018. This enables, in addition to standardised per-pixel depth error evaluation, for: distance related assessment (DRA), meaning to evaluate how well a model performs in different depth ranges, planarity error so as to evaluate how well planar surfaces is reproduced, depth boundaries evaluation where you can determine the accuracy of locating depth boundaries, and directed depth error showing if the estimated depth is predicted too far or too short [Koch et al., 2020]. In addition, [Zhao et al., 2020] provides a paper on an overview of monocular depth estimators (MDEs) and how they function in general terms. In terms of using the MDEs on video data, or a sequence of frames, some work has been made on utilising temporal information in order to acquire temporal consistent depth estimations. Google Research published a paper regarding this, that functions off-line with moving cameras [Zhang et al., 2021], and [Zhang et al., 2019] strives to accomplish this in a real-time realisation.

1.6 Scientific Contribution

This thesis' work aims on continue the exploration of monocular depth estimation models. It provides a careful evaluation and analysis of some currently state-of-the-art models, in a surveillance domain, with 3D privacy masking in mind. This includes quantitative evaluation of performance, as well as some qualitative assessment continuing to explore in what situations they function or not, leading to an increased understanding of MDE. In addition to reaching conclusions about their abilities and limitations, we aim to raise awareness on how the performance can be evaluated, and what aspects to raise when comparing MDEs with one another. This will be additional work to papers such as [Koch et al., 2018] and [Zhao et al., 2020], as we will continue to explore evaluation of performance and model comparison, with the main difference that our work will be in the surveillance domain, with 3D masking and video applications in mind. The thesis also explores the issue of temporal inconsistency mentioned by [Zhang et al., 2019] and propose an alternative non-complicated post-processing method to mitigate the effect of this.

The thesis also contributes with a 3D masking algorithm technique, demonstrating how depth information can be utilised in terms of privacy masking, as an improvement relative to the static 2D mask [Axis Communications, 2021]. We are utilising depth in order to anonymise a scene without losing desired information, as have been investigated in terms of 3D reconstruction of an entire scene for anonymisation purposes [Conde Moreno, 2019], but doing this using a static live-mask instead, realising the prospects proposed in approved patent registrations [Takeyuki Goto et al., 2009; Berlic and Strandevall, 2012]. We want to, with this application, manifest the use of monocular recovered depth in auxiliary computer vision or video applications, extending the list of applications presented in Section 1.5.

1.7 Individual Contributions

The authors of this thesis have contributed all-in-all equally. Both authors have played its part in all individual topics. The background research have been studied in parallel by both authors. For the development of the algorithm, there have been an on-going discussion of the conceptual design, and for all intermediate techniques that have been tried out, this work has been alternating between the authors, so to have both authors invested and inducted in the proposed algorithm. The same goes for the evaluation and analysis of the chosen models, where the subsequent tasks have been divided internally for a fair distribution of work load. For example, Mattias have focused on the accuracy evaluation on the self-captured stereo data set, while Jakob did the same analysis on the LiDAR data set. To repeat ourselves, even though one of the authors have been lead responsible for a given task, this does not mean one have been solely invested in this task, and the work load and decision making have been divided equally.

2

Theory

This chapter aims to give a brief introduction to terminology and concepts in terms of acquiring depth from images. Firstly, classical ways of inferring depth using photogrammetry are introduced. This includes a description of image formation and how one can acquire depth from stereo. Subsequently, terminology connecting to the heart of this thesis, namely depth estimation using deep-learning, is reviewed. Here, concepts such as U-net architecture, encoders, relevant loss functions, etc. are described. Lastly, the metrics used for evaluation of the monocular depth estimation models are reviewed.

2.1 Depth Perception using Photogrammetry

A core and long standing problem in Computer Vision and Robotics is estimating distance to scene objects and their geometrical properties. When images are involved, distance estimation is typically referred to as depth estimation. The results of this is preferably a pixel-wise depth map representation, where each pixel has a corresponding depth value. In computer vision, this is known as a *depth map* or a *dense depth map* [Forsyth and Ponce, 2012].

Classical methods to infer depth in an image typically fits under the larger umbrella term *Photogrammetry*. We will in this section briefly explain the concept of image formation and touch on some of these more classical methods of acquiring depth, and leave methods involving Machine Learning to the next section.

Photogrammetry

Photogrammetry is an extensive existing field where the aim is to retrieve three-dimensional information about real-world objects from one or multiple images. Applications were early mostly related to cartography and military intelligence, but has evolved along computers and numerical analysis into analytical photogrammetry using methods from fields like optics and projective geometry [Forsyth and Ponce, 2012].

Image Formation

This thesis will use the *pinhole camera model*. It is commonly used due to its simplicity, and while there exist more complicated camera models describing intricate lens and light interactions [Forsyth and Ponce, 2012], the accuracy gained is not in proportion to the added complexity in this thesis.

Throughout this thesis we will use uppercase letters to denote points in world space with the camera centre at the origin. Small letters will denote points in camera space. From the pinhole camera model, the world space point $P = (X, Y, Z)$ as seen from the camera coordinate frame is represented in camera pixel space by projection point $p = (x, y, 1)$. This mapping is done with the intrinsic matrix K by $\lambda p = KP$, mapping real-world points to pixel coordinates, with λ describing the existing scale ambiguity. The intrinsic matrix is referred as the inner parameters of the camera, which encodes properties such as focal length (f) along the x- and y direction, skew (s) and optical centre (x_0, y_0). The relation is given by

$$\lambda p = \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.1}$$

where λ is the result of the fact that infinitely many points in world space can be mapped to the same point in image space, making the inverse operation impossible without losing depth information.

In this thesis we assume the effects of skew (s) to be negligible for our purpose, and project and transform points using known or estimated focal lengths. This can mathematically be described as

$$\lambda p = \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{2.2}$$

Depth from Stereo

A more direct way to perceive depth of a scene is to capture and fuse the information from multiple images, much like our human eyes. This is called *Depth from Stereo* or *Stereopsis* [Forsyth and Ponce, 2012].

Using pixel correspondence from a stereo pair images and calculating the pixel displacement, or disparity, results in a disparity map. This disparity, d, is inversely proportional to depth according to [Szeliski, 2011]

$$d(x,y) = f\frac{B}{Z} \tag{2.3}$$

where f is the focal length in pixels, B the baseline in meters (distance between the cameras), and Z the 3D depth in meters. The (horizontal) disparity $d(x,y)$ relationship between pixel coordinates in a stereo pair is [Szeliski, 2011]

$$x' = x + d(x,y), \quad y' = y \tag{2.4}$$

in a typical stereo setup where the cameras are identical, calibrated and placed parallel with a known baseline distance. In (2.4), x' and y' are the displaced pixel coordinates due to stereo disparity. Since disparity is the (horizontal) pixel displacement, this means that object that are close have large disparity, while objects far away have none, or minuscule disparity. As such, the reconstruction from disparity to a depth map is relatively straight forward, and finding the pixel correspondences in stereo pairs is the more challenging step. Especially since there often are millions of pixels involved.

Finding pixel correspondence is a large topic by itself and today's stereo-camera products and solutions employ various algorithms to perform accurate and fast stereo matching. General problems involve limited depth range, occlusion, and stereo matching when there are low amounts of features such as a complete white wall. A modern consumer stereo camera such as ZED stereo can perceive depth up to 15–20 meters [Stereolabs, 2022] with decreasing accuracy. This is related to the baseline length, as well as the image resolution which is because of the increased difficulty to find stereo pairs on objects far away, since they are represented by fewer pixels, adding uncertainty.

2.2 Depth Perception using Deep Learning

This section will underline important aspects of depth estimation using deep-learning. More specifically, these topics will be introduced in regard to the models of choice in this thesis, namely Monodepth2 [Godard et al., 2019], MiDaS [Ranftl et al., 2020], DPT [Ranftl et al., 2021] and AdelaiDepth [Yin et al., 2021]. A comparison and elaboration of these models will be presented in Chapter 3, but for now, we introduce some more general terminology in order to understand the foundation.

First off, the popular U-net architecture is introduced. This enables acquiring both high and low-level features of an image when performing dense prediction. Secondly, encoders, and relevant variants of these, are presented. Thirdly, the term vision transforms is introduced, as this is the foundation of DPT [Ranftl et al., 2021],

further described in Section 3.3. We also describe scale- and shift image transformation as this is an important step for calculating error metrics, as well as a key factor for determining a loss function effective on a mixed data set. Lastly, the theory behind some proposed loss functions and their functionality is described, as introduced in selected model papers.

U-net

Since depth estimation of monocular images is a dense image prediction task, it is important to acquire both low and high-level features of the image. It is desirable to output both the coarse prediction at the higher feature levels, without losing the fine-grained predictions at lower feature levels. This is typically solved using an encoder–decoder network architecture, with skip-connections, much like the published U-net paper in 2015 [Ronneberger et al., 2015]. Here, the encoder part consists of convolutional layers, downsampling the input image into more and more isolated regions of features. The decoder part of the architecture performs de-convolutional steps and concatenates this with the convoluted output at the corresponding level from the encoder, i.e., *skip-connection*. The skip connections are what enables a global receptive field, and resolves to a fine-grained, dense prediction. An illustrative figure of the U-net architecture is presented in Figure 2.1.

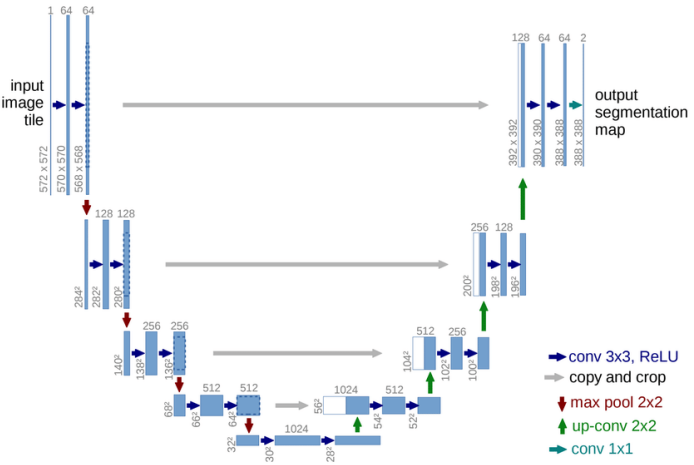


Figure 2.1 Illustration of the U-net architecture [Taghanaki et al., 2020] under license [CC BY 4.0].

Encoders

The encoding part in a U-net architectural network is important in order to acquire a good classification. The size of the network strictly relates to the quality of the prediction. However, a too deep network also quickly leads to cumbersome training, as well as long inference time. Therefore the need for a compromise between

these two aspects is needed, and monocular depth estimation is no exception. In addition, it is also highly beneficial to utilise pre-trained encoders, in order to save on total training time, as well as being able to keep the depth of the networks in order to attain high accuracy. The latter has previously proved to be problematic, since simply stacking more layers on top of each other soon lead to a saturation, and even a degradation, of accuracy, i.e., the vanishing or exploding gradient problem [Bengio et al., 1994; Glorot and Bengio, 2010]. Here is where ResNet [He et al., 2016] comes in.

The ResNet [He et al., 2016] solution of the vanishing/exploding gradient problem was to introduce *deep residual networks*. The baseline here is a regular feed-forward network, with the addition of skip-connections past one or more layers, in order to solve for the residual mapping instead. In the original paper, the ResNet was trained on ImageNet [Russakovsky et al., 2015], an object classification data set with hundreds of categories and millions of images. The following deep-learning models that we will present more in depth later on, are, with a few exceptions, based on a ResNet encoder pre-trained on ImageNet.

Alternatively, large depth prediction models also tend to use the further developed ResNeXt-101 [Xie et al., 2017]. This network is formed by a set of building blocks, each with internal ResNet layers, adding a dimension (*cardinality*) to the network. This leads to fewer hyper parameters to set. The smaller model of MiDaS [Ranftl et al., 2020], made for mobile platforms, is using the EfficientNet-Lite3 [Tan and Le, 2019]. Tan and Le are here looking at scaling up the dimensions of a neural network, without for that sake necessarily increasing inference time.

Vision Transforms

As previously mentioned, DPT [Ranftl et al., 2021] is a depth estimation model that utilises vision transformers (ViT) [Dosovitskiy et al., 2021] as backbone to their encoder–decoder network architecture. The ViTs are based on the work with transformers for natural language processing, for self-attention based learning [Vaswani et al., 2017]. Using transformers in self-attention learning is a way of computing a representation of a sequence of data points, or words in the original paper by Vaswani et al., in order to find global dependencies between input and output, without for that matter use sequential RNN or convolutions. This leads to better scalability, and reduces the need of processing input in a sequential manner, as the global context of the input is inferred by self-attention. Lastly, Vaswani et al. implemented *multi-headed self-attention*, which basically refers to implementing a multiple of self-attention layers in parallel. With this, the model can extract information from different subset representations at different positions at once [Vaswani et al., 2017].

The ViT model follows the original design of transformers closely. As the transformer architecture takes a 1D-input array, the 2D images need to be represented as

such. The idea is to divide the input image into squared, non-overlaying, patches, and the pixel data of these patches are flattened and embedded by linear projection. In addition, in order to reduce the input data size, the input 1D-array can consist of feature maps from an initial convolutional neural network (CNN) instead of the raw pixel data. This is the case for DPT Hybrid [Ranftl et al., 2021], that we will include in our evaluation. Conclusively, using ViT as a backbone architecture, is another way of attaining a dense prediction model, with fine-grained prediction using the global receptive field that comes with ViT.

Scale and Shift Alignment

For depth estimation models, the perspective depth prediction is often only to be noted as relative depth, as it in general terms is a frame-by-frame prediction with no prior knowledge about the scene or the camera setup. In addition, some of these models are trained on data with depth up to an unknown scale and shift. In other words, all we can trust is the internal relative depth of the scene. This needs to be addressed in two prominent situations. Firstly, when a depth prediction model is trained on multiple data sets, with different depth sources (some output relative depth, some metric depth), but without separating the loss functions for each data set, and secondly, when performing evaluation of these data sets. As the output from the models are internally relative, and many ground truth validation data sets come in the form of metric depth, the output needs to be aligned with the ground truth data in order to make a fair comparison. These two situations are handled somewhat differently, and we will present the background to each of them respectively.

During training with multiple data sets, it is desirable to remove the scale and shift of the ground truth and predicted values. This is to be able to use the same loss function on fundamentally different sources of data. This was presented by MiDaS [Ranftl et al., 2020], and results in the zero-translated and unit scaled disparity map ($\hat{\mathbf{d}}$) of the input disparity map ($\mathbf{d}$),

$$\hat{\mathbf{d}} = \frac{\mathbf{d} - t(\mathbf{d})}{s(\mathbf{d})} \tag{2.5}$$

where the shift, $t(\mathbf{d})$, is defined as the median of the data. The scale, $s(\mathbf{d})$, is given by

$$s(\mathbf{d}) = \frac{1}{M} \sum_{i=1}^{M} |\mathbf{d}_i - t(\mathbf{d}_i)| \tag{2.6}$$

where M denotes the number of pixels in the given image [Ranftl et al., 2020].

The other aspect of this was, as mentioned, to align a predicted output disparity/depth map with the ground truth, for fair evaluation. This was as well presented by [Ranftl et al., 2020], and adopted by AdelaiDepth [Yin et al., 2021] for their evaluation. The idea is to align the predicted output with the ground truth in accordance with least-squares principle. This is read out as scaling (s) and translating (t) the input depth map, $\mathbf{d}$, such that it is as close to the ground truth, $\mathbf{d}^*$, as possible according to least-squares. This is mathematically described as

$$(s,t) = \arg\min_{s,t} \sum_{i=1}^{M} (s\mathbf{d}_i + t - \mathbf{d}_i^*)^2 \tag{2.7}$$

Referencing (2.7), the prediction is then transformed as

$$\hat{\mathbf{d}} = s\mathbf{d} + t \tag{2.8}$$

and the ground truth is kept in its original form.

With the notations, $\vec{\mathbf{d}}_i = (\mathbf{d}_i, 1)^\top$ and $\mathbf{h} = (s,t)^\top$, the objective of (2.7) can be rewritten as

$$\mathbf{h}^{opt} = \arg\min_{\mathbf{h}} \sum_{i=1}^{M} \left(\vec{\mathbf{d}}_i{}^\top \mathbf{h} - \mathbf{d}_i^* \right)^2 \tag{2.9}$$

which in turn is in closed-form solved by

$$\mathbf{h}^{opt} = \left(\sum_{i=1}^{M} \vec{\mathbf{d}}_i \vec{\mathbf{d}}_i{}^\top \right)^{-1} \left(\sum_{i=1}^{M} \vec{\mathbf{d}}_i \mathbf{d}_i^* \right) \tag{2.10}$$

This will be used, and implemented by us when performing depth metrics evaluation of chosen models, in Chapter 5.

Loss Functions

Another important aspect of depth estimation models is the loss function. How do you determine in the best suited manner how well the model has actually been able to predict the depth, compared to a ground truth depth or disparity? Here, we introduce some intermediate loss functions, that individually play an important role, for different reasons, in the complete loss functions of the models that we will evaluate later on.

Firstly, we have the photometric reprojection error, used by Monodepth2 [Godard et al., 2019],

$$\mathcal{L}_p = \sum_{t'} pe(I_t, I_{t'\to t}) \tag{2.11}$$

This describes the L1 distance, as well as the Structural SIMilarity (SSIM) [Wang et al., 2004] of the reprojected image, $I_{t'\to t}$, compared to the target image, I_t. This is referred to as pe in (2.11), and is given by

$$pe(I_a, I_b) = \alpha/2(1 - SSIM(I_a, I_b)) + (1-\alpha)\|I_a - I_b\|_1 \tag{2.12}$$

Secondly, the edge-aware smoothness loss is given by

$$\mathcal{L}_s = |\partial_x d_t^*| e^{-|\partial_x I_t|} + |\partial_y d_t^*| e^{-|\partial_y I_t|} \tag{2.13}$$

where d_t^* is the mean-normalised inverse depth and I_t is the target image [Godard et al., 2019]. This loss function encourages smoothing the resulting depth map by computing the disparity gradients, penalising sharp contours, weighting this cost by an edge-aware term using the image gradients [Godard et al., 2017].

Further, the multi-scale, scale-invariant gradient matching term, introduced by [Li and Snavely, 2018], is used by MiDaS [Ranftl et al., 2020] as a way of accomplishing sharp discontinuities that are aligned with discontinuities of the ground truth. This is achieved with

$$\mathcal{L}_{reg}(\hat{\mathbf{d}}, \hat{\mathbf{d}}^*) = \frac{1}{M} \sum_{k=1}^{K} \sum_{i=1}^{M} \left(\left| \nabla_x R_i^k \right| + \left| \nabla_y R_i^k \right| \right) \tag{2.14}$$

where $\hat{\mathbf{d}}$ and $\hat{\mathbf{d}}^*$ are the predicted disparity map and ground truth, respectively, being shifted and scaled accordingly with (2.5). Further, M is the number of pixels in the image, $R_i = \hat{\mathbf{d}} - \hat{\mathbf{d}}^*$ and k is the scale to apply on the disparity map difference, R. Similarities between this loss and the one given in (2.13) is that they both try to accomplish contour refinement, by using gradient values.

Additionally, the scale and shift-invariant loss by MiDaS [Ranftl et al., 2020], is computed as the absolute difference error,

$$\mathcal{L}_{ssi}(\hat{\mathbf{d}}, \hat{\mathbf{d}}^*) = \frac{1}{2M} \sum_{i=1}^{U_m} \left| \hat{\mathbf{d}}_i - \hat{\mathbf{d}}_i^* \right| \tag{2.15}$$

The notation for number of pixels (M), scale and shift-invariant ground truth ($\hat{\mathbf{d}}^*$) and predicted disparity ($\hat{\mathbf{d}}$) are the same as for (2.14). In addition, $U_m = \alpha M$, which is referring to the down-scaled ($\alpha < 1$) set of image pixels, being sorted on size. This means that the top-value outliers are disregarded during training.

An alternative way of countering scale and shift ambiguities connected to varied sources of data, contrary to $\mathcal{L}_{ssi}$ in (2.15), is the image-level normalised regression loss, presented by AdelaiDepth [Yin et al., 2021]. The idea is to normalise the ground truth data in a robust manner, so as to exclude its scale and shift. As regular min-max-normalisations generally are prominent to be affected by outliers and long-tail residuals, they produced a combination of a tanh normalisation [Singh and Singh, 2020] and a trimmed Z-score normalisation. The loss function is presented as

$$\mathcal{L}_{ILNR}(\mathbf{d}, \mathbf{d}^*) = \frac{1}{M} \sum_{i=1}^{M} \left| \mathbf{d}_i - \overline{\mathbf{d}}_i^* \right| + \left| \tanh(\mathbf{d}_i/100) - \tanh(\overline{\mathbf{d}}_i^*/100) \right| \tag{2.16}$$

Here, $\overline{\mathbf{d}}_i^* = (\mathbf{d}_i^* - \mu_{trim})/\sigma_{trim}$, where μ_{trim} and σ_{trim} are the mean and standard deviation of a trimmed version of the ground truth, respectively. A trimmed version means in this context that the top percentage of pixels is removed (both high and low depth values).

Lastly, as a way of enforcing geometric constraints of a predicted scene, [Yin et al., 2021] introduces a pair-wise normal loss, given by

$$\mathcal{L}_{PWN}((n_A, n_B), (n_A^*, n_B^*)) = \frac{1}{N} \sum_{i=1}^{N} |n_{Ai} \cdot n_{Bi} - n_{Ai}^* \cdot n_{Bi}^*| \tag{2.17}$$

Here, the surface normal pairs, (n_A, n_B), and its corresponding ground truth (marked with a $(^*)$) is sampled based on the image edges, so as to sample points on both sides of every image edge, accordingly with [Xian et al., 2020]. This gives geometric constraints on a local level, and in addition, points are sampled globally as well. The total number of sampled points is then given by N [Yin et al., 2021]. This loss function targets improving local and global geometric features.

2.3 Evaluation

In this section, metrics related to the quantitative evaluation in Chapter 5 are introduced. Firstly, the metrics relating to depth accuracy evaluation are inspected. Secondly, the metrics related to temporal stability evaluation are reviewed.

Depth Accuracy

In order to determine the per-pixel accuracy of a given depth prediction, two existing measurements of accuracy are adopted. Both of these metrics were included as two of six error metrics in [Eigen et al., 2014], which was early on the task regarding MDE. These metrics have since then become the de-facto-standard when models are to be evaluated relative to ground truth depth data, e.g., Monodepth2 [Godard et al., 2019]. The first metric presented here is the average absolute relative difference (*AbsRel*) [Eigen et al., 2014],

$$AbsRel = \frac{1}{|T|} \sum_{y \in T} |y - y^*| / y^* \tag{2.18}$$

where y is the predicted value, y^* is the corresponding ground truth, and T marks the number of predictions. This is a common metric for data sets that include a per-pixel ground truth depth value, and gives a value of how far off a prediction is, relative to the ground truth.

The second metric that we will include in our evaluation is the delta-threshold metric (δ_{err}), also given in [Eigen et al., 2014]. This gives a percentage of pixels that are more than a threshold off relative to the ground truth. The per-pixel relative error, δ_i, is given by the maximum value of $\frac{y_i}{y_i^*}$ and $\frac{y_i^*}{y_i}$,

$$\max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) = \delta_i \tag{2.19}$$

The final metric is then given by the percentage of δ_i that is higher than a threshold, thr,

$$\delta_{err} = \frac{1}{N} \sum_{i=1}^{N} \delta_i > thr \tag{2.20}$$

where N is the number of predictions. Normally, thr is a power of 1.25, e.g., $thr = 1.25^2$. As the predicted depth is often up to an unknown scale- and shift, the predictions will be aligned to the ground truth according to least-squares (2.7)

as first presented by [Ranftl et al., 2020] and later followed by [Yin et al., 2021]. For both of the metrics presented here, lower is better.

Stability

When evaluating how well a MDE model performs in terms of temporal consistency, investigated in Chapter 5, a quantitative result on this stability concern is desirable. In order to achieve this, an updated video frame's background pixels are compared to the corresponding pixels of a reference frame, using three metrics to be described briefly in the following.

The first evaluation metric to be presented is the *Structural Similarity Index* measurement (SSIM) [Wang et al., 2004]. The SSIM is the result of Wang et al. trying to construct an error measurement to be in agreement with how the human visual system (HVS) perceives image quality. It consists of the combination of three image properties that represent object structure in a scene, disregarding the average image luminance and contrast. The three properties are

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = f(l(\mathbf{x},\mathbf{y}), c(\mathbf{x},\mathbf{y}), s(\mathbf{x},\mathbf{y})) \tag{2.21}$$

where $l(\mathbf{x},\mathbf{y})$ refers to a function assessing the (local) luminance of the two images, $\mathbf{x}$ and $\mathbf{y}$, $c(\mathbf{x},\mathbf{y})$ refers to local contrast computation and, lastly, $s(\mathbf{x},\mathbf{y})$ refers to a structure comparison of the two images. These are combined with the function $f(\cdot)$. The similarity measurement is assembled so as to fulfil the following constraints: being symmetric ($S(\mathbf{x},\mathbf{y}) = S(\mathbf{y},\mathbf{x})$), bounded by one, and having a uniquely defined maximum, s.t. $S(\mathbf{x},\mathbf{y}) = 1 \Leftrightarrow \mathbf{x} = \mathbf{y}$. This results in the final closed-form expression,

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.22}$$

where C_1 and C_2 are constants, decomposable into $C_i = (K_i L)^2$, which in turn denotes the pixel value dynamic range, L, and an arbitrary constant, $K_i \ll 1$. Further, μ_x and μ_y are the mean, σ_x and σ_y are the standard deviation, of image $\mathbf{x}$ and $\mathbf{y}$, respectively, and σ_{xy} is defined as

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \tag{2.23}$$

The SSIM is then evaluated on sub-regions of the image, on a local level, and the final result is produced by the mean of these sub-regions, over the entire image.

Lastly, since it is the dissimilarity of the images that is of interest here, we instead perform the following operation,

$$\mathrm{DSSIM}(\mathbf{x},\mathbf{y}) = \frac{1-\mathrm{SSIM}(\mathbf{x},\mathbf{y})}{2} \tag{2.24}$$

resulting in a structural *dissimilarity* index (DSSIM), where lower is better. In addition to the DSSIM, two per-pixel error metrics are included for reference. Firstly, the mean absolute error (MAE), as represented by

$$\mathrm{MAE}(\mathbf{x},\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i| \tag{2.25}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the images, and x_i and y_i are indexed pixels, totalling N number of pixels. Secondly, the root mean squared error (RMSE), as given by

$$\mathrm{RMSE}(\mathbf{x},\mathbf{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2} \tag{2.26}$$

is used.

3

Model Overview and Comparison

This chapter serves to give a thorough insight into the chosen models that are further investigated and evaluated in Chapter 5. Firstly, we describe Monodepth2, a semi-supervised model originally targeting the automotive industry of autonomous driving (AD). Secondly, MiDaS is reviewed, that opts Monocular Depth Estimation with good generalisation abilities. Following these is DPT, a further development of MiDaS. Lastly, we discuss AdelaiDepth, from Adelaide University, a model that targets good geometric consistency in its output. The introduction of these models all include a background, the specific loss function of each model, and some general information about its architecture and training technique. This chapter is finished by putting the chosen models next to each other in a table, highlighting their differences and similarities.

Monodepth [Godard et al., 2017] followed by Monodepth2 [Godard et al., 2019], tackled the problem of inferring depth from single images in 2017 and 2019, respectively. They considered it problematic acquiring vast amount of ground truth data, so instead of approaching this as a supervised regression problem, they solved it in a semi-supervised manner. Even further down the road, MiDaS [Ranftl et al., 2020] and AdelaiDepth [Yin et al., 2020] referred to the task as a regression problem, but addressed the difficulties in acquiring a large amount of ground truth depth data by training on a mixture of data sets. In addition to these model, we have researched and briefly looked at several additional models, such as [Lee et al., 2019], [Casser et al., 2019], [Alhashim and Wonka, 2018], but concluded on the chosen models, Monodepth2, MiDaS, DPT, and AdelaiDepth, as they were considered representative models of different techniques and approaches, as well as achieving state-of-the-art results.

3.1 Monodepth2

Monodepth2 [Godard et al., 2019] is the further development of the 2017 Monodepth model [Godard et al., 2017]. These models are self-supervised, meaning that no actual ground truth depth is needed during training. The Monodepth depth model treats the problem at hand as an image reconstruction task, instead of a direct depth prediction task. In other words, what the model actually predicts, is the correspondence field, d^r, such that when applied to a left image of a rectified stereo pair, would result in the reconstructed right image. If the two images are rectified, the image pair disparity is given by d.

Given the predicted disparity, along with the stereo camera's baseline and focal length, which was used to capture the training data, the depth can be retrieved by reformulating (2.3) so as to acquire the distance, Z. In addition, Monodepth relies on what they refer to as "left-right consistency cost", meaning that during training, both the left-to-right, as well as the right-to-left disparity maps are produced, and forced to be consistent. At run-time, the left image is the sole input, and both disparities are predicted and forced to be consistent [Godard et al., 2017].

Monodepth2 continues on this work, and brings it further by exploring training the models on monocular video data, utilising structure from motion (SfM) in order to acquire self-supervised ground truth depth, in addition to keep exploring the stereo self-supervision. Their work resulted in state-of-the art results, compared to competing self-supervised approaches. As Monodepth2 is the only version of the two models that will be considered hereon after, it will for simplicity reasons be referred to as Monodepth.

Loss Function

The resulting loss function, presented by Monodepth [Godard et al., 2019] is given by

$$\mathcal{L} = \mu \mathcal{L}_p + \lambda \mathcal{L}_s \tag{3.1}$$

This is averaged over each pixel, scale and batch. In (3.1), L_p refers to the photometric reprojection error, computed by (2.11) and (2.12). This loss function refers to the L1 distance and SSIM between the reprojected and target image. Likewise, L_s refers to the edge-aware smoothness loss in (2.13), in order to smooth the output depth where there is no detected edge.

The naive approach to compute the per-pixel reprojection error is to average this metric over all source images for each pixel. This becomes problematic for pixels that are visible in the target image but not in the source image (because of occlu-

sion or out-of-view pixels between frames). According to the authors of Monodepth [Godard et al., 2019], this case leads to the model over-penalising the unseen pixel values. Instead, they suggest choosing the *minimum* per-pixel photometric reprojection error.

During the self-supervised monocular training on SfM data, it is generally assumed that the camera is moving and captures a static scene. In cases when this is not the case, i.e., the camera stands still, or scene objects are moving, the predicted depth is not to be trusted and often results in infinite depth prediction. To account for this, [Godard et al., 2019] adds a filter to exclude pixels that do not change appearance from one frame to another, as motion is therefore not encountered.

Architecture and Implementation

The Monodepth model takes the renowned *U-net* architecture, taking on skip-connections so as to acquire both low and high-level features. The encoder is a ResNet18 [He et al., 2016] network, being pre-trained on the ImageNet [Russakovsky et al., 2015] data set.

The decoder consists of deconvolutional layers back to the original resolution, with skip connections from the encoders' activation blocks [Godard et al., 2017]. At the output, the sigmoid activation function is used, and for all other decoder layers, ELU nonlinearities are used [Godard et al., 2019]. For pixels outside the source image boundaries, reflection padding is performed. Image augmentation is performed, such as horizontal flips, random brightness, contrast saturation and hue jitter, each with a 50% chance. The Adam optimiser [Kingma and Ba, 2015] was used and input and output resolution is 640x192 by default.

3.2 MiDaS

MiDaS depth estimation model was developed by Intel Labs in 2020. MiDaS is, in contrast to Monodepth, a supervised deep neural network model. In other words, MiDaS is in need of ground truth depth in order to improve the model. To acquire such data is cumbersome, and especially so collecting a larger set with a *diverse* set of scenes and environments. Monodepth's solution to this was to take on a self-supervised approach, removing the need of expensive distance-measuring sensors, like a LiDAR. MiDaS approach is, on the other hand, to use a mixture of data sets and consider depth estimation as a multi-task learning problem, enabling mixing of data sets, that may be annotated differently and come from a variety of sources [Ranftl et al., 2020].

Additionally, in order to further address the difficulties that come with mixing data sets, it requires novel loss-functions. Originally, the model was trained on four publicly available data sets, and in addition, the authors of MiDaS [Ranftl et al., 2020]

put together a new image-depth data set making use of 3D movies. In a later version of the model, the model was extended to in total 10 data sets. It was later evaluated on six never-seen-during-training data sets, to prove for its generalisation performance. Conclusively, MiDaS work resulted in, at the time, a state-of-the art depth estimation method, with great generalisability [Ranftl et al., 2020].

Loss Function

Some data sets contain given ground truth in metric depth, while others contain the ground truth depth up to an unknown scale, displaying only the relative depth of that given scene. This requires a loss functions that is invariant to different kinds of depth annotation. This can be concluded with three types of challenges, that the loss function needs to adapt to. These three challenges are: different depth representation (depth or disparity), unknown depth scale (relative depth, unlike metric depth), and unknown shift (horizontal shift of the principal points).

These targets result in a loss function invariant to the scale and shift problematics, and a function that performs predictions in inverse depth space, i.e., disparity prediction. The final loss function is given by

$$\mathcal{L}_l = \frac{1}{N_l} \sum_{n=1}^{N_l} \mathcal{L}_{ssi}(\hat{\mathbf{d}}^n, (\hat{\mathbf{d}}^*)^n) + \alpha \mathcal{L}_{reg}(\hat{\mathbf{d}}^n, (\hat{\mathbf{d}}^*)^n) \tag{3.2}$$

where N_l refers to the training set size, α is a constant set to 0.5, and $\hat{\mathbf{d}}_i$ and $\hat{\mathbf{d}}_i^*$ are the predicted and ground truth disparity, respectively, for pixel i, being scaled and shifted [Ranftl et al., 2020]. The scale and shift operation of the disparity data is performed according to (2.5), where the same function is applied to the ground truth disparity, $\mathbf{d}^*$.

In (3.2), $\mathcal{L}_{reg}$ refers to the multi-scale gradient matching term given in (2.14). This serves to accomplish high-contrast discontinuities, aligned with discontinuities of the ground truth. During training, MiDaS uses four scale levels ($K = 4$), and halves the image resolution for each scale.

The function $\mathcal{L}_{ssi}$ in (3.2) is referring to the scale and shift-invariant loss function presented in (2.15). Here, U_m is set to $0.8M$ and refers to not training for the top 20% residuals, so as to exclude outliers of the ground truth data. This function serves to pushing the predicted depth closer to that of the ground truth.

Architecture and Implementation

Having the loss function invariant to different types of data sets, originating from different sources and taking different ground truth depth annotation is one step on

the way of having a depth estimation model trained on a mixture of data sets. The other step is to determine how to choose the number of samples from each data set, and decide on how to define optimum weights given all data sets during training. This is solved by MiDaS as a multi-task learning problem, defining depth estimation of each individual data set as a separate task. This is done in accordance with [Sener and Koltun, 2018]. Optimum is here reached when *the loss can not be decreased for any given task, without it being increased for at least one of the other tasks*.

The backbone of MiDaS is based on the ResNet [He et al., 2016] architecture, pre-trained on the ImageNet [Russakovsky et al., 2015] data set. Specifically, the large MiDaS model runs with ResNeXt-101 [Xie et al., 2017], while their smaller model, developed for embedded devices, runs on EfficientNet-Lite3 [Tan and Le, 2019]. Furthermore, they adopt the multi-scale architecture of [Xian et al., 2018]. This is a U-net architecture with skip-connections, enabling capturing both high and low-level semantic features in the dense depth predictions.

The Adam optimiser [Kingma and Ba, 2015] was used by MiDaS. Augmentation wise, training data are horizontally flipped by 50% chance, and randomly resized to 384x384. All ground truth data are given in inverse depth space (disparity) up to an unknown scale and shift [Ranftl et al., 2020].

3.3 DPT

DPT stands for Dense Prediction Transformer, and is the product of the dense prediction research performed by Intel Labs and how vision transformers (ViT) [Dosovitskiy et al., 2021] can be used for the benefit of dense prediction tasks, like depth estimation or image segmentation [Ranftl et al., 2021]. Considering the depth estimation task, DPT is the natural successor of MiDaS. In fact, the depth prediction model using DPT is all in all, almost the same as MiDaS, i.e., the same loss function, trained on the same data set (the mixture of 10 data sets), using the same optimiser, etc. What separates the two Intel depth estimation models are then the architecture, and namely the encoder part, where DPT utilises vision transforms instead of a convolutional encoder like ResNet [He et al., 2016]. This results in, according to [Ranftl et al., 2021], a more fine-grained and gloablly coherent dense depth prediction.

Vision Transforms

In order to employ vision transforms [Dosovitskiy et al., 2021], the input images need to be represented as a ”bag-of-words” [Ranftl et al., 2021]. These bag-of-words is referred to as *tokens*. This embedding is done by dividing the image into squared, non-overlapping patches. These pixels are flattened, and embedded using linear projection. An alternative way of doing this is to specify the image token embedding as the pixel features from a ResNet50 [He et al., 2016] encoder. This

model variant is referred to as DPT Hybrid. In addition, a separate token is added, called the *readout* token, as is customary in natural language processing. This token is used for classification.

3.4 AdelaiDepth

AdelaiDepth is a toolbox with collected work within monocular depth estimation from the University of Adelaide. Their work consists of a series of papers, all relating to depth and scene geometry reconstruction. They started off by introducing their work on single-view depth estimation by using convolutional neural fields [Liu et al., 2016]. Following that, in 2019, they started focusing on introducing geometric constraints in the depth estimation model, enabling not only better prediction results, but also allowing better 3D reconstruction and point cloud transformation [Yin et al., 2019]. The idea behind this is to include a loss using the virtual normal, the normal of the plane that is made of three randomly sampled points from a reconstructed point cloud, when training the MDE. Further, a new data set was presented, DiverseDepth [Yin et al., 2020], and they trained the model on learning depth up to an affine transformation, in contrast to learning either the metric or relative depth. In other words, when training, they allow for the prediction to be off, relative to the ground truth, as long as the prediction is off only by a translation and scale transformation. This produced good generalisation and sharp geometric shapes.

This work is then further improved in [Yin et al., 2021]. Here they continue to explore the DiverseData set, accomplishing geometric constraints from normal losses and affine-invariant prediction. In addition, [Yin et al., 2021] includes a 3D point cloud reconstruction network, that learns to predict the unknown scale and shift for better 3D reconstruction, something that will not be further investigated in this thesis.

Loss Function

The loss function of the latest MDE model from AdelaiDepth toolbox uses an image-level normalised regression loss (ILNR), a pair-wise normal loss (PWN), and a multi-scale gradient loss (MSG) [Yin et al., 2021],

$$\mathcal{L} = \mathcal{L}_{PWN} + \lambda_a \mathcal{L}_{ILNR} + \lambda_g \mathcal{L}_{MSG} \tag{3.3}$$

Here, λ_a and λ_g are set to 1 and 0.5, respectively. The multi-scale gradient loss, $\mathcal{L}_{MSG}$, originates from [Li and Snavely, 2018], and is presented in (2.14), with the slight change that AdelaiDepth enforces Z-score normalisation on their ground truth depth map. It is, conclusively, the same loss function as used by MiDaS and serves

to acquire sharp depth discontinuities aligned with discontinuities in the ground truth.

The image-level normalised regression loss, $\mathcal{L}_{ILNR}$, is shown in (2.16) and serves to normalise the ground truth to address scale and shift ambiguities that comes with mixing data sets. In their experiments, they remove the highest and lowest 10% depth value pixels when computing the mean and standard deviation for Z-score normalisation.

The pair-wise normal loss, $\mathcal{L}_{PWN}$, is given in (2.17) and strives to enforce geometric consistency on both a global and local level, on the contrary of the proposed virtual normal loss [Yin et al., 2019], which only proclaimed geometric constraints on a global level [Yin et al., 2021]. Before calculating this loss, and the predicted surface normals, the prediction is aligned in scale and shift with its ground truth according to the least-squares (2.7).

Architecture and Implementation

The latest model from Adelaide University is trained on a mixture of five different data sets [Yin et al., 2021]. In contrast to MiDaS multi-task learning approach using pareto-optimal optimisation [Ranftl et al., 2020], AdelaiDepth trains the model on diverse data by utilising *multi-curriculum learning*, according to [Yin et al., 2020]. In conclusion, this technique is based on sorting the training data on increasing difficulty, and samples mini-batches of training data from this sorted set of data.

The architecture of AdelaiDepth follows [Xian et al., 2020]. As a backbone, it consists of either a ResNet50 [He et al., 2016] *or* ResNeXt101 [Xie et al., 2017], resulting in a large and a small model, respectively. This is followed by a backbone. The model is trained using stochastic gradient descent (SGD) and a learning rate of 0.02, with the decay of 0.1. Images are resized to 418x418 pixels, and flipped horizontally by 50% chance.

3.5 Model Comparison

To summarise this chpater, the MDE models that we have investigated are in many senses similar. They include a loss function consisting of one part that enforces the predictions to be close to the ground truth values. Commonly, they are also including a kind of loss function that enforces contour refinement and discontinuity alignment. All models are using an encoder–decoder network architecture. The models strive to achieve a global receptive field for a more fine-grained prediction. Augmentation wise, horizontal flip is consistently included.

However, there are also some implementation choices that differ for each model, compared to the others. What must be the most prominent distinction is between

Table 3.1 Comparison table of "large" monocular depth estimation models.

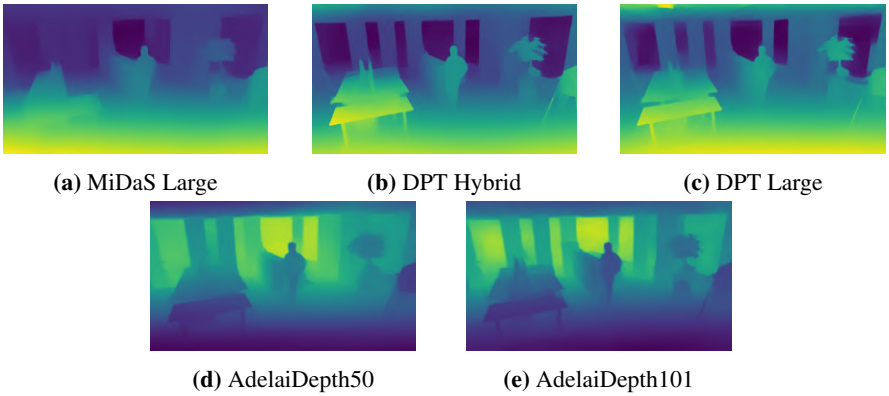
	MiDaS Large	DPT Hybrid/Large	AdelaiDepth 50/101
Year	2020	2021	2021
Encoder	ResNeXt-101	ResNet50+ViT/ViT	ResNet50/ResNeXt101
Training Data	10 mix	10 mix	5 mix
Resolution	224x384	384x672	448x448
Inference (FPS) *	6	1/0.4	15/8
Output Space	Inverse depth	Inverse depth	Depth

* This measurement is when run on GTX1650 GPU, with no tensor accelerating device. This is only to be considered as a relative comparison, and in no way a de-facto performance assessment.

Monodepth and the other models. Monodepth is self-supervised, and is based on an image reconstruction task, from which depth is inferred. In addition, it is trained solely on the automotive targeted data set KITTI [Geiger et al., 2013]. Moreover, MiDaS is separated from the others by introducing their own data set consisting of a collection of 3D movies, training on 10 different data sets and introducing training on a mixture of data sets as a multi-task learning objective. Additionally, DPT was ground braking by introducing vision transformers in order to acquire a global receptive field. AdelaiDepth is unique by focusing on geometric properties, and looking further into 3D reconstruction using single-view depth.

A comparison table of the larger models (MiDaS Large, DPT and AdelaiDepth), with more demanding computer processing at run-time, is given in Table 3.1. Here, we present the year of release, the backbone network, specifying training data (mix is here referred to being trained on a mixture of different data sets, e.g., 10 mix means being trained on a mixture of 10 different data sets), input and output resolution, inference time on desktop devices, and the output space. It is worth mentioning here that the DPT models show low update frequency during inference, in terms of frames-per-second (FPS), compared to the other models, when run on a desktop device with GTX1650 GPU. It is also noted that AdelaiDepth is the only model that has its output in depth space, while the others have their output in inverse depth space (similar to disparity), which also can be seen in Figure 3.1, where example output is presented for the five models. This is a key aspect when working with MDEs, and in order to utilise scene depth in an application, an inverse transformation is needed if the output is given in inverse depth space originally.

Similarly as for the larger models, we present a comparison table for the smaller models *MiDaS Small* and *Monodepth*. This is shown in Table 3.2. The same information is included as in Table 3.1, i.e., year of release, encoder, training data, res-



(a) MiDaS Large **(b)** DPT Hybrid **(c)** DPT Large

(d) AdelaiDepth50 **(e)** AdelaiDepth101

Figure 3.1 Example output of ”large” MDEs.

Table 3.2 Comparison table of ”small” monocular depth estimation models, proposed for embedded systems.

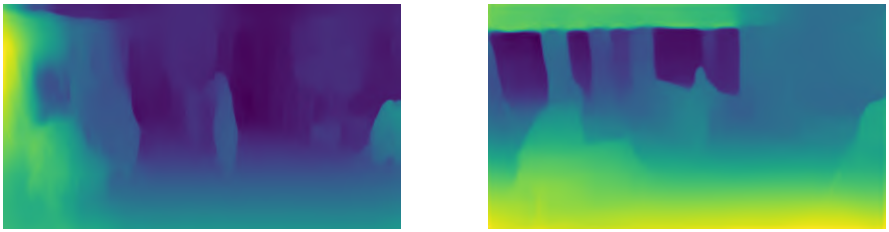
	Monodepth	MiDaS Small
Year	2019	2020
Encoder	ResNet-18	EfficientNet-Lite3
Training Data	KITTI	10 mix
Resolution	192x640	128x256
Inference (FPS) *	49	50
Output Space	Inverse depth	Inverse depth

* This measurement is when run on GTX1650 GPU, with no tensor accelerating device. This is only to be considered as a relative comparison, and in no way a de-facto performance assessment.

olution, inference time, and output space. Here, it is noticeable how inference time on our desktop device differ relative to the larger models, as these models run in approximately 50 FPS, compared to 15 FPS as best performing model in Table 3.1. This distinction is why we will, in various evaluation analysis, separate these set of models, for a fair comparison. The example output of these smaller models is presented in Figure 3.2.

Depth Accuracy Metrics on Existing Data Sets

For depth accuracy evaluation of these models, we follow the evaluation procedure of MiDaS [Ranftl et al., 2020], that was in turn followed by AdelaiDepth [Yin et al., 2021]. This includes a scale and shift alignment of the predictions with the ground truth in accordance with least-squares criterion (2.7), as the output is up to an unknown scale and shift (relative depth). This alignment procedure is done in



(a) Monodepth **(b)** MiDaS Small

Figure 3.2 Example output of "small" MDEs.

the *model output* domain, i.e., the inverse transform is performed on the ground truth depth, if the given model's output is in inverse depth space. After aligning the predictions, this is transformed into the *ground truth* domain, if needed, and the *AbsRel* (2.18) and δ_{err} (2.20) metrics are calculated for each image, where the threshold is set to 1.25 for δ_{err}. The final score is given by the average over all frames of the data set. We do this on three existing data sets, which will briefly be introduced in the following sections, and we finish this chapter with a comparison table of the models performance.

NYUv2

The data set NYUv2 presented in [Nathan Silberman and Fergus, 2012], consists of a handheld stereo camera, from various indoor scenes. Here, we will use their densely labelled data set, where missing depth pixel values have been filled using the colourisation technique in [Levin et al., 2004]. This data set consists of 1449 RGBD frames, containing 464 scenes [Nathan Silberman and Fergus, 2012]. We set a depth cap when evaluating on this data set to 10 m.

ETH3D

ETH3D is a data set with both indoor and outdoor scenes. The ground truth is acquired with a laser scanner [Schöps et al., 2017]. Hence, this data set figures static scenes, with high accuracy in the ground truth values. In this thesis, we present results with their multi-view high resolution data set, containing 13 scenes and 454 frames [Schöps et al., 2017]. We set a depth cap to 72 m for our evaluation on ETH3D.

iBims-1

The data set iBims-1 is also a high resolution RGBD data set, where the ground truth is captured by a 3D scanner, as was the case for ETH3D. In addition of providing ground truth depth, for accuracy evaluation, iBims-1 also includes various metrics for geometric reproduction, such as planarity error, how they perform at various depth ranges, and locating depth boundaries. The iBims-1 data set consists of 100 frames in various indoor scenes, and provides depth up to 50 m [Koch et al., 2018].

Results

The achieved results (AbsRel and delta-error) on the three data sets, of the seven models of choice are given in Table 3.3. The best performing model, for each data set and each metric (for every column), is marked with **bold**. The second best performing model is marked with underline. The table is split such that the upper part represents the "larger" models, and the two last rows present the "smaller" models. If the same metrics have been published for a given model, on the same set of data, this is presented in parenthesis, for reference.

From Table 3.3, it is concluded that, considering the smaller models, MiDaS small [Ranftl et al., 2020] outperforms Monodepth [Godard et al., 2019], by a margin. For the larger models, we see that the two models from AdelaiDepth [Yin et al., 2020] are performing best, with both metrics, on the labelled NYUv2 data set. On the laser scanned data set with static indoor and outdoor scenes, ETH3D, it is instead the DPT models [Ranftl et al., 2021] that consistently produce the best results, for the two metrics. On the smaller data set, also with 3D-scanned data, iBims-1, it is noted that AdelaiDepth101 is a top performer. However, it produces only second best on the delta-threshold error, where DPT Large instead seems to outperform AdelaiDepth, when some room for error is allowed.

Table 3.3 Depth accuracy score on three data sets (NYUv2 [Nathan Silberman and Fergus, 2012], ETH3D [Schöps et al., 2017] and iBims-1 [Koch et al., 2018]), presenting both the AbsRel error as well as the delta-threshold error. For models, data set and metrics where corresponding results have been published, this is included in parenthesis.

	NYUv2*		ETH3D**		iBims-1***	
Online Data Sets	*AbsRel*	δ >1.25	*AbsRel*	δ >1.25	*AbsRel*	δ>1.25
AdelaiDepth101	**0.104**	**0.112**	0.119	0.130	**0.075** (0.079)	0.060
AdelaiDepth50	0.105	0.116	0.127	0.144	0.080	0.073
DPT Large	0.117	0.138	**0.089** (0.0888)	**0.070**	0.080	**0.057**
DPT Hybrid	0.132	0.170	0.094 (0.0934)	0.072	0.084	0.068
MiDaS Large	0.115	0.130	0.116 (0.1155)	0.117	0.093	0.080
MiDaS Small	**0.134**	**0.180**	**0.135** (0.1344)	**0.161**	**0.102**	**0.106**
Monodepth	0.222	0.397	0.185	0.279	0.201	0.333

* [Nathan Silberman and Fergus, 2012]
** [Schöps et al., 2017]
*** [Koch et al., 2018]

4

Depth Aware Privacy Mask

This chapter presents a method for what is referred to as 3D Privacy Masking, a depth aware privacy mask. It serves to introduce an end-to-end algorithm implementing this functionality as a downstream application from Monocular Depth Estimation. Additionally, extensions to mitigate the effect of the inherent instability produced by temporal inconsistency between frames are also introduced. Finally, we present results from selected models on multiple scenes to investigate if current state-of-the-art monocular depth estimation can produce adequate depth maps for such downstream application to satisfactory results.

4.1 Concept

Regular privacy masks on surveillance cameras are typically implemented by manually defining a polygon on the target area and replacing that region of the image stream with either a uniform colour, or a blurred or pixelated version, see Figure 4.1 for an example.

With added depth information of each pixel, this concept can be extended into the third dimension. Utilising depth information together with a mathematical model of the camera view, this privacy region can be placed arbitrary in space, effectively shielding a private area without losing visibility of objects outside this region. This concept is illustrated in Figure 4.2.

Algorithm Specifications

The following requirements were defined and considered during development of the 3D masking algorithm. While proof-of-concept and analysis were the primary goal, these requirements give the project realistic constraints that affected all design decisions.

Figure 4.1 Example of a regular privacy mask, here represented with a green uniform colour.

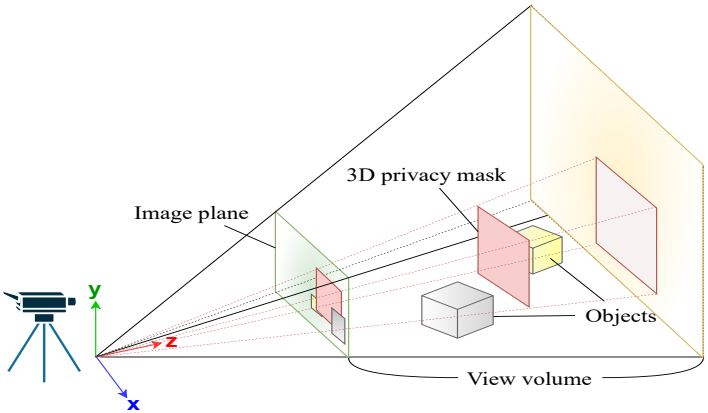


Figure 4.2 Illustration of 3D masking. Notice that the projected 2D mask in the image plane is in front of the yellow object, but behind the other.

1. **Real-time compatible:** The overall algorithm, containing depth model inference and masking computations, needs to be real-time compatible. While development was performed on desktop machines, and not in an embedded environment, the latency of the system needs to be kept as low as possible. This is in order to motivate a realistic use-case on a camera with modern computational capabilities.

2. **Robust:** It needs to be robust to minor inaccuracies and instabilities that may arise when performing frame-wise depth prediction. This means no masking artefacts or erroneous masking, and any event in front of the mask should be

observable and understandable.

3. **Hide-rather-than-show:** The design needs to promote false-positives over false-negatives when performing masking. It always needs to mask the area that is supposed to be masked in order to not compromise privacy.

4. **Intuitive user interface:** Any user interface should be easy and intuitive to use. The user should preferably be able to define a 3D privacy mask by creating a polygon on a reference image, just as in regular 2D privacy masking without any 3D tools, and the result needs be consistent and predictable. This is important since defining a 3D area from a 2D view is in general ambiguous and the mask can be situated arbitrarily in 3D space, see Figure 4.2 for an illustration.

Euclidean Space Transformation

The generated depth or disparity map from MDEs is what enables the transition from image space to Euclidean space. Since each pixel for each frame receives an independent depth value, it is possible to, together with the mathematical model (2.2) of the camera that captured the frame, transform image points into world-space points $P = (X, Y, Z)$. Utilising the image-point and world-point relation in (2.2) we can derive the inverse operation, or *back projection*, if we have depth Z, by

$$\begin{aligned} x &= f_x \frac{X}{Z} + x_o, & X &= \frac{x - x_o}{f_x} Z \\ y &= f_x \frac{Y}{Z} + y_o, & Y &= \frac{y - y_o}{f_y} Z. \end{aligned} \tag{4.1}$$

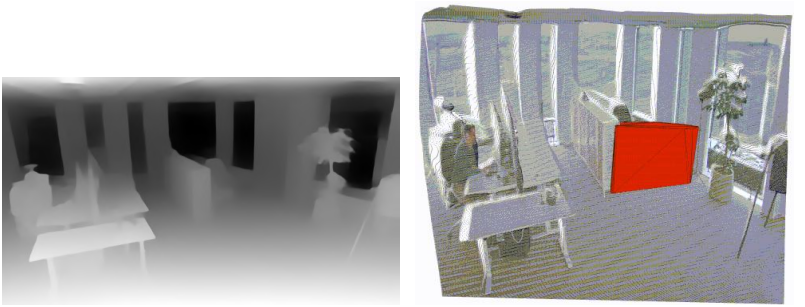
Here, the scale ambiguity can be ignored since depth is known. Performing this operation for every pixel results in a point cloud that can be coloured with the corresponding RGB values. An example of this is illustrated in Figure 4.3.

4.2 Development Phases

The development process of the 3D masking functionality was performed iteratively and in an offline environment with pre-recorded scenes. From these iterations, *four* masking concepts were developed, starting with the simplest approach we could think of and progressively making the algorithm more refined and tuned to the specifications in Section 4.1. In general, the most discriminating feature was the approach on how to define the actual 3D privacy region in world space.

Privacy Region Creation

The four different investigated techniques is here listed. It briefly explains the conceptual idea, as well as illustrating the functionality using figures. Reasons for why



(a) Depth map.

(b) Point cloud of (**a**) in a 3D viewer. The red mesh defines the masking region.

Figure 4.3 Resulting coloured point cloud (**b**) from back projection of depth map (**a**).

a given technique was deemed to not fulfil the algorithm requirements is as well included.

1. **Plane**
   In the first iteration the privacy region was defined as an (infinite) plane. The plane was constructed by the user picking three points on the image plane, subsequently picking three points in world space using the corresponding depth map. Using these points, a unique plane can be defined on the form $ax+by+cz=d$. Using the selected points P, Q and R, we can find $\vec{n}=(a,b,c)$ by

   $$\vec{n} = \vec{PQ} \times \vec{PR} \tag{4.2}$$

   and d defined by

   $$d = \vec{p} \cdot P. \tag{4.3}$$

   The depth of the plane can be varied by scaling d. The defined plane is used to calculate inliers and outliers, which are then projected to image space as a binary mask. See Figure 4.4 for an illustration.

   However, getting a consistent (plane normal) result by picking three points was sometimes difficult and somewhat unintuitive.

2. **Bounding Box**
   Next, we explored ways to define a 3D region by utilising bounding boxes and minimum spanning volume algorithms. This is constructed by picking an arbitrary amount of points (minimum four points) of the object that should be masked. Then, a bounding box encapsulating the minimum volume of these

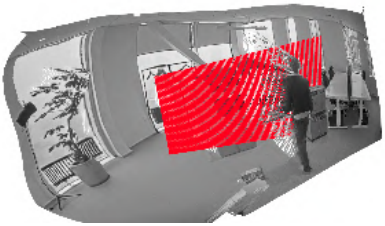
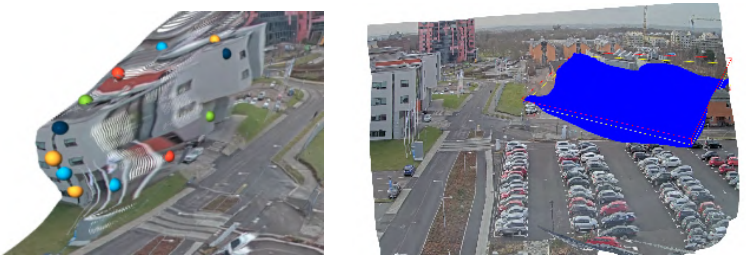


Figure 4.4 Plane. Observe that the plane illustrated is not infinite for illustration purposes.

points is calculated. This bounding box can then be resized in all three dimensions (width, height, depth), as well as rotated and translated. The bounding box is then used to calculate inliers and outliers, and projected to image space as a binary mask is then used. See Figure 4.5 for an illustration.

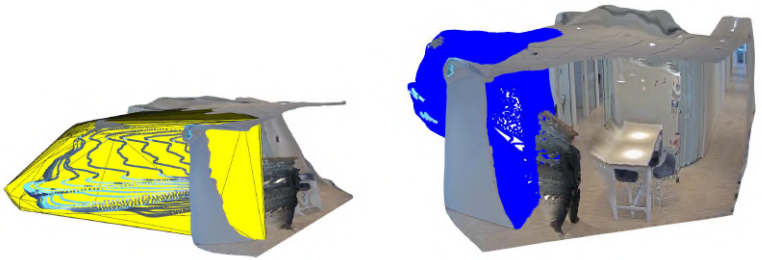


(a) Picked points **(b)** Bounding box

Figure 4.5 Bounding box technique. (a) Selected points. **(b)** The bounding box of these selected points. Interior points marked with blue resulted from ray-casting. These points correspond to the masked pixels.

This technique did also not produce a consistent enough result, and a 3D viewer was deemed necessary to always validate the created region. Finally, we moved onto mask definition techniques that proceeded from a regular 2D privacy mask selection using polygons.

3. **Convex Hull**
   In the third iteration, only the pixels inside the marked polygon were transformed into a sub-point cloud. This is constructed by first defining a mask in image space, similarly as for the regular case of 2D-masking, and then extracting a sub-point cloud of that region of the image. A *convex hull* [Boyd and Vandenberghe, 2004], the shape of the smallest convex set, is created around this point cloud that functions as a definition of the 3D-mask. New input frames are converted into point cloud in a similar manner. Using, for example, ray-casting from the camera point of view, these points are determined to be behind or in front of the mask, and each pixel is masked thereafter. See Figure 4.6 for an illustration.



(a) Convex hull mesh **(b)** Inside points

Figure 4.6 Convex Hull Masking Technique. (a) Defined convex hull mesh surrounding the defined masking point cloud. **(b)** Interior points marked with blue resulted from ray-casting. These points corresponds to the masked pixels.

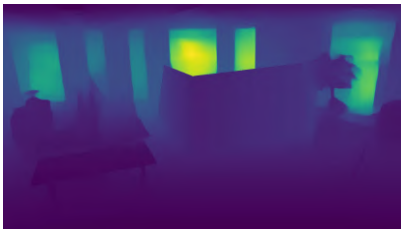
This solution met the requirement of an intuitive user interface. However, the problem with this approach was for masks defined by a ground boundary line, rather than a defined object. This led to the fourth iteration, referred to as *folding*.

4. **Folding**
   This approach exploits the fact that realistic scenes typically have increasing depth in a bottom to top manner. Here we assert the same depth value column-wise, or a subset of the mask image columns, ”folding” up an imaginary depth wall. This was converted into a 3D-point cloud, where a volume/surface was defined around the point cloud. In addition to improving the generalisability of the application, this solution also lead to a more robust approach. This follows since it results in locating a larger set of points to the mask volume side closest to the camera, acting re-enforcingly when computing the volume shape. As a final upgrade to this algorithm, the folding technique was updated so as to not having to convert the data into 3D space, in an attempt to increase run-time performance.

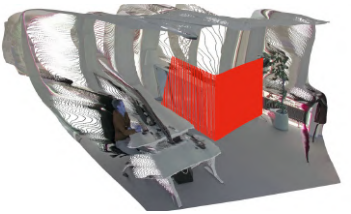
   See Figure 4.7 for an illustration. Implementation details follows in Section 4.3.

4.3 Implementation

The masking implementation was developed, tested and evaluated on high-end desktop hardware. More specifically, a GeForce GTX 1650 GPU and AMD Ryzen 9 3900x 12-core CPU was used. Development was done in the Python programming language utilising multiple libraries, with key tools being OpenCV [Bradski, 2000] and Open3D [Zhou et al., 2018] for optimised image- and 3D processing and visualisation tools.



(a) ”Folding” Depth Map



(b) ”Folding” Point Cloud

Figure 4.7 Folding Masking Technique. (a) The folded mask depth map. **(b)** The folded mask wall illustrated in Open3D.

In Figure 4.8, the application flowchart is illustrated. It consists of two parts. In the upper part, the mask definition is presented. This takes an initial RGBD frame (RGB image and depth map). Firstly, the user marks a polygon, as is done for regular 2D masking. With the folding technique, the polygon baseline is sought for, as the depth data at these pixels is what is used to *fold* up the masking wall. This is done in the *Find Baseline*-block, in the flowchart in Figure 4.8. With the baseline defined, this is passed as input to the *Folding*-block. This is where the mask is defined. For every pixel coordinate along the x-direction, i.e., image column wise, the depth value for the whole column is replaced by the depth value of the baseline. Lastly, the 2D-polygon mask is applied, and this functions as the depth aware privacy mask.

In the bottom part of the flowchart in Figure 4.8, the continuous masking algorithm is presented. This takes a sequence of incoming RGBD-frames, and masks out pixels within the 2D-polygon that have a depth value being larger than that of the mask, and shows pixels with depth values smaller than those in the 3D-mask.

The pseudo code for the algorithm finding the baseline vertices, from the input polygon vertices, is presented in Algorithm 1. The algorithm assumes polygon vertices sorted in a clockwise or counter-clockwise order. The first step taken is to find the leftmost polygon vertex, meaning the polygon vertex with the minimum column coordinate value. If two vertices were to have the same column values, the vertex further down is chosen, i.e., the maximum row coordinate value. This is the baseline starting vertex. Next, the two neighbouring polygon vertices are evaluated, and the one with the maximum row coordinate value is chosen, i.e., choosing the bottom polygon line as baseline. If this were the previous or succeeding vertex relative to the starting vertex, we choose to iterate counter-clockwise or clockwise, respectively. When this is determined, the list with polygon vertices is iterated through, and new baseline vertices are added, as long as they are to the right of the previous baseline vertex, i.e., with a higher image column coordinate. Once the baseline vertices are found, these are used to produce a line, retrieving the baseline pixel coordinates. The result from this algorithm on an example is presented in Figure

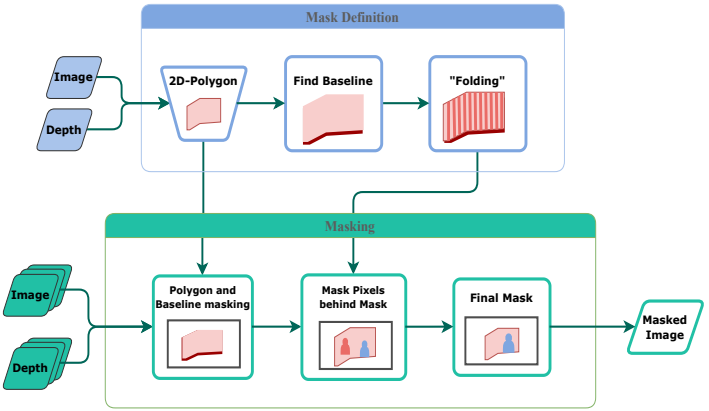


Figure 4.8 Illustration of the 3D masking algorithm. The upper part shows how the mask is defined from RGBD input, making a 2D-polygon mask from user input, applying the ”folding”-technique to define the mask. In the bottom part, the general procedure is illustrated masking incoming frames, extracting region of interest from the 2D-mask, find inside points and mask pixels behind mask wall.

4.9.

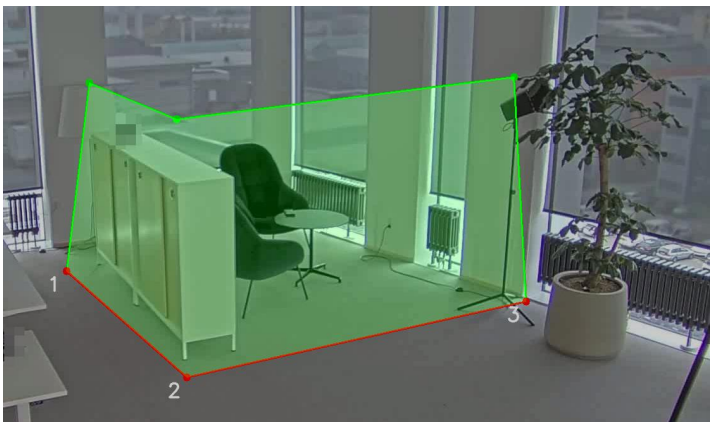


Figure 4.9 The final result from the baseline algorithm. All points in the image are taken as user input and the polygon 2D-mask is illustrated by the faded green colour. Red vertices are the baseline vertices, found by Algorithm 1, and the numbers refer to the order of which they are found. The remaining vertices are marked with green.

The pseudo code for the Folding-algorithm is presented in Algorithm 2. This takes a number of inputs: a depth map image, the polygon 2D-mask, the pixel coordinates of the baseline, and a step size, marking the column-wise resolution of the baseline to extract depth from. The output is the folded depth aware mask. Firstly, the

Algorithm 1 Find Baseline

Input: $polygon \leftarrow [(c_1, r_1), (c_2, r_2), ..., (c_i, r_i)]$
Output: $baseline \leftarrow [(c'_1, r'_1), (c'_2, r'_2), ..., (c'_j, r'_j)]$
$i_0 \leftarrow \arg\min_i(polygon(c_i))$
if i_0 not unique **then**
$i_0 \leftarrow \arg\max_{i \in i_0}(polygon(r_i))$
end if
$baseline(start) \leftarrow polygon(i_0)$
$i_{next} \leftarrow \arg\max_{i \in [i_0+1, i_0-1]}(polygon(r_i))$
if i_{next} is $i+1$ **then**
$s \leftarrow +1$
else if i_{next} is $i-1$ **then**
$s \leftarrow -1$
end if
repeat
$baseline(end) \leftarrow polygon(i_{next})$
$i_{next} \leftarrow (i_{next} + s) \bmod len(polygon)$
until $polygon(c_{i_{next}}) < baseline(c_{end})$

baseline pixels are split into chunks of *stepSize*, e.g., 10 for extracting every 10th column depth, and asserting this depth to chunks of 10 columns in the final depth aware privacy mask. This split is iterated through, in order to extract the actual depth at these pixels. Lastly, the depth map is updated, so as to assert the baseline depth values for all the rows, in the appropriate columns, building a depth wall. The final mask is then computed as bit-wise masking the folded depth map with the input 2D-polygon mask.

Temporal Instability

A depth aware privacy masking application requires both consistent depth prediction for high-dynamic scenes together with low latency, fulfilling requirements (**1**) and (**2**) in Section 4.1. The previously presented algorithm implementation in Section 4.3 specifically assumes consistent depth for sequential frames, so that a placed 3D masking region in Euclidean space stays fixed in relation to the perceived scene in world space. However, this depth consistency has been observed to not be guaranteed, as illustrated in Figure 4.10.

Temporal instability is an identified issue that may arise when wanting to utilise monocular depth estimation models in video context. This is because monocular depth estimation models by definition only infer depth from a single RGB frame as input, which is called for in order to meet requirements such as low latency for real-

Algorithm 2 Folding

Input: $depthMap, mask2D, baseline, stepSize$
Output: $mask3D$

```
baselineSplit ← baseline split into chunks of stepSize
for split in baselineSplit do
    idx ← arg max_i (split(r_i))
    coord ← split(idx)
    baselineDepth(end) ← depth(coord)
end for
col ← baseline(c_0)
for depth in baselineDepth do
    nextCol ← (col + stepSize)
    if nextCol ≤ c_end then
        depthMap(r, col : nextCol) ← depth
    else if nextCol > c_end then
        depthMap(r, col : c_end) ← depth
    end if
    col ← nextCol
end for
mask3D ← (depthMap and mask2D)
```

(a) Initial frame point cloud. **(b)** Subsequent frame point cloud.

Figure 4.10 Issue of Temporal Instability Illustrative point cloud of how temporal inconsistency of MDEs can cause instabilities when applied to downstream tasks, such as depth aware privacy masking.

time applications. We have found that this single-shot approach in general predicts consistent depth for multiple frames for *static* scenes, i.e., scenes without any dynamics or moving objects. However, for scenes that include movement, e.g., moving people or vehicles, the perceived depth may change on a frame-to-frame basis, creating *temporal instability*. This phenomenon is further investigated in Chapter 5 and specifically Section 5.3.

Stabilisation Implementation — Recurrent Mask Update

In Figure 4.11 the proposed technique for masking stabilisation is illustrated. It contains a new *Stabilisation* part marked in yellow. The overall idea is to continuously updating the mask and mask definition, which we refer to as *Recurrent Mask Update*. Defining and instantiate a static mask, as in Figure 4.8, that is fixed for the rest of the scene duration causes issues when this temporal instability is present. A mask defined from frame *1* is then not guaranteed to be correctly placed relative to the scene and the desired privacy region, in for example frame *10*. An illustration of this is seen in Figure 4.10.

In order to dynamically update the mask frame by frame, it is needed to consider moving objects that may be present in the masking region. More specifically, when using the *Folding* method, objects that are covering the *baseline* need to be considered. Updating the baseline depths when a *dynamic* object that was not present when the 2D masking polygon was defined is covering the baseline, as seen in the *Foreground/Background*-block, will result in a faulty mask and privacy region. We propose handling this issue with a continuously updated *depth buffer*.

The stabilisation part of the flowchart in yellow consists of four blocks. The first block performs *background subtraction* on each incoming frame to identify moving objects (foreground). Any background subtraction method can be applied, as long as it is sufficiently fast. We experimented with various existing background subtractors such as MOG2 [Zivkovic, 2004] as implemented by the OpenCV library, and morphology operations such as dilate and erode for post-processing. In the subsequent step the depth buffer is updated by means of *weighted moving average* in non-moving regions, and the current frame's baseline is updated. Since we cannot use the corresponding depth values in detected foreground regions, we instead fetch depth values from the available depth buffer.

Also, since foreground and background are now available from the *Foreground/Background*-block, we can utilise this to optimise run-time by only performing the full masking algorithm (as well as inference on the depth estimation network) when a moving object is detected in the 2D polygon region, and only update the depth buffer otherwise. Furthermore, we can with detected foreground in the *Mask Pixels behind Mask*-block, force background regions in the mask area to behave as a regular 2D privacy mask, and only perform "3D-masking" in detected

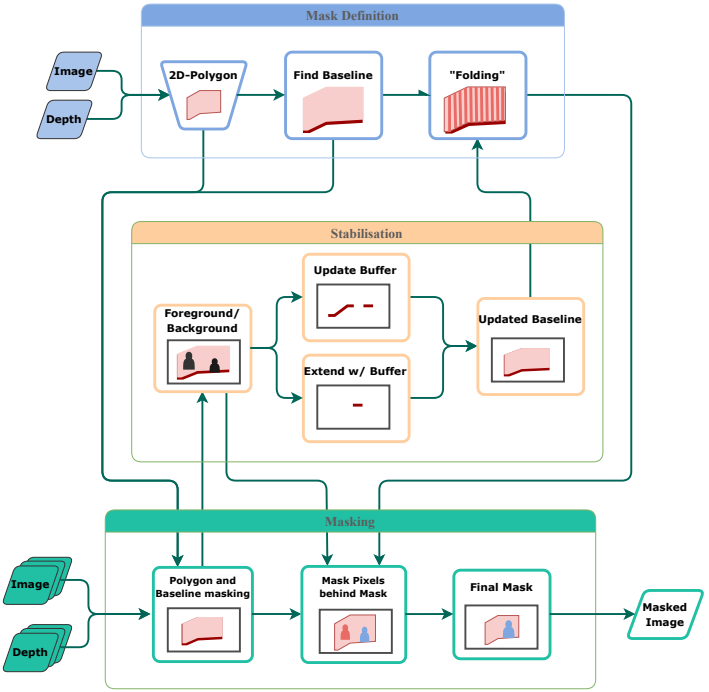


Figure 4.11 Illustration of the extended 3D masking algorithm, including the stabilisation algorithm. The upper part shows how the mask is defined from RGBD input. In the bottom part, the general procedure is illustrated masking incoming frames. The mask is re-created for every frame, where the new frame depth is used, except where foreground is detected, where the depth is replaced by a depth buffer.

foreground regions. This further enforces the algorithm requirements (**2**) Robust, and (**3**) hide-rather-than-show principle.

4.4 Results

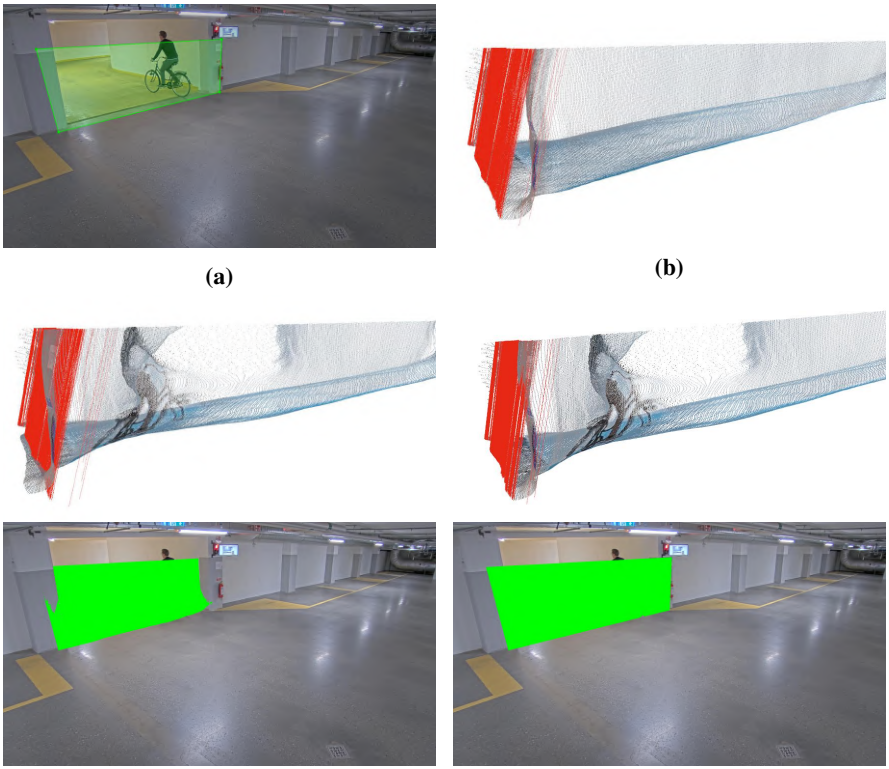
In this section, qualitative assessment and analysis will be performed of the proposed depth aware privacy masking application. The full algorithm, including the stabilisation part in Figure 4.11, is running on recorded scenes at approximately 25-30 FPS using the specified setup in section 4.3. This is in accordance with the thesis goal, where the proposed application, to be perceived as a proof-of-concept, targeted a run-time performance around 30 FPS on a local desktop machine. This result is with limited amount of work put in optimisation of the implementation, and all the code being written in Python. It is important to note that this is expected to run even faster using a more low-level language such as C/C++. Additionally, the most exhaustive part of the algorithm is the stabilisation part, in Figure 4.11, which takes about 22 ms. This timing is also highly dependent on the chosen background model, which is important to be aware of. The raw implementation, Figure 4.8, runs at approximately 85-90 FPS.

In the subsequent sections, we will present examples on qualitative performance of the algorithm. Firstly, this includes an evaluation of the recurrent mask-updating technique (Figure 4.11), that achieves a more robust and stable privacy mask, relative to the raw implementation (Figure 4.8). Secondly, the importance of depth range is exemplified. Thirdly, some situations are presented that illustrate when the privacy mask does not succeed to anonymise a given region, in terms of stability issues and depth accuracy. Lastly, some show-casing examples are provided illustrating scenes where the privacy mask fulfils all requirements, i.e., it manage to keep events in front of the mask *observable and understandable*, without breaching privacy behind the mask.

Recurrent Mask Update vs Static Mask

In Figure 4.12, a comparison is presented between the raw implementation and the implementation with the stabilisation part. Figure 4.12a shows the input RGB image, with the displayed polygon mask. Figure 4.12b shows the masked area of the first frame, and the folded wall, visualised as a point cloud. Figure 4.12c and Figure 4.12d shows the results from the raw and recurrent mask-updating implementation, respectively.The produced point cloud from these two implementations, it is noted that the point cloud has shifted since the first frame, and the original mask is therefore not entirely accurate relative to the updated frame. It is also perceived that, even for the implementation with the stabilisation part, in Figure 4.12d, some parts of the wall are leaking through the mask, but this erroneous mask artefact is mitigated by always masking the background pixels from the foreground–background detection model, resulting in a more *robust* mask.

Another illustration of how the further developed implementation outperforms the raw version, in terms of robustness, is illustrated in Figure 4.13. A subsequent frame



(a) **(b)** **(c)** **(d)**

Figure 4.12 Garage Scene (DPT Large): Illustrating the instability issue, and how the recurrent mask-updating technique accounts for this. **(a)** RGB image with defined mask **(b)** Masked area visualised with Open3D **(c)** Static mask point cloud and resulting masked image **(d)** Recurrent masking point cloud and resulting masked image.

point cloud is illustrated in Figure 4.13c using the static mask. Comparing this with the point cloud illustrating the initial frame, in Figure 4.13b indicates that the room has shifted, since the mask was first defined. Figure 4.13d illustrates how the recurrent mask-updating technique accounts for this, as the mask is updated every frame. As the mask is updated, and background pixels are continuously masked, the improved result is visualised in Figure 4.13f, compared to the breach of privacy for the static mask that is distinguished in Figure 4.13e.

Lastly, in Figure 4.14, it is visible how the proposed improved implementation, compared to the raw, enacts on the ”hide-rather-than-show” specification. There exist inaccuracies and stability issues of the input depth information, i.e., the arm of the person is perceived as part of the staircase in the background, and the room has

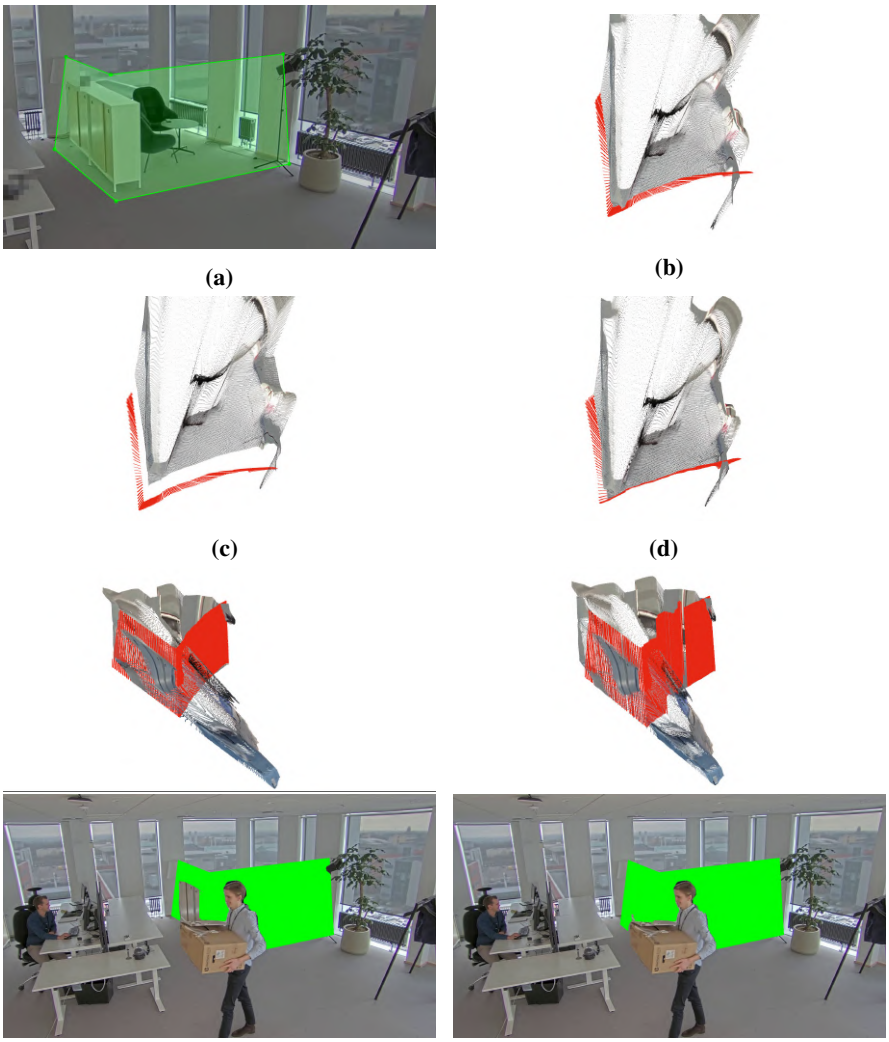
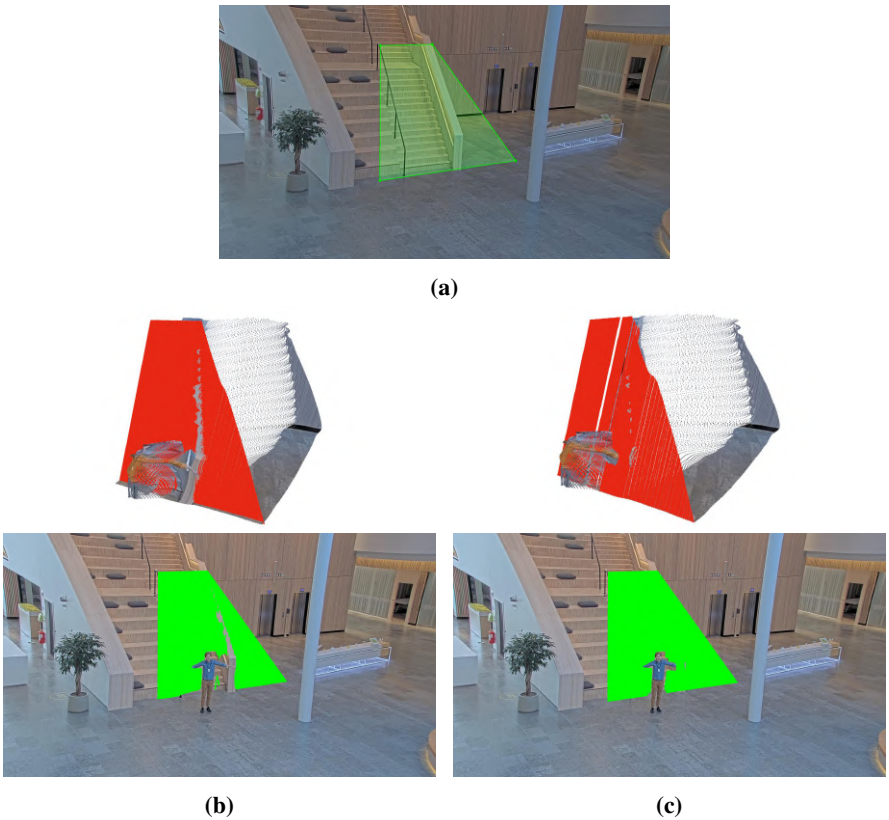


Figure 4.13 Office Scene (AdelaiDepth50): Illustrating the instability and depth accuracy issues, and how the recurrent mask-updating technique accounts for this. **(a)** RGB image with 2D-mask **(b)** First frame point cloud and defined mask **(c)** subsequent frame with no foreground in Open3D using *static* mask **(d)** subsequent frame with no foreground in Open3D using *recurrent updated* mask **(e)** Static mask: current frame in Open3D, and resulting masked image **(f)** Recurrent masking: current frame in Open3D, and resulting masked image.

shifted relative to the initial scene frame, Figure 4.14b. The stabilisation technique here acts on the ”hide-rather-than-show” specification, as well as demonstrating increased robustness, as seen in Figure 4.14c.



(a) **(b)** **(c)**

Figure 4.14 Staircase Scene (AdelaiDepth50): Illustrating the increased robustness and enacting on ”hide-rather-than-show” specification. **(a)** RGB image with 2D-mask **(b)** Static mask: masked area visualised in Open3D, resulting masked image **(c)** Dynamic mask: masked area visualised in Open3D, resulting masked image.

In Figure 4.15, a problematic scene in terms of the stabilised masking application is illustrated. Figure 4.15b presents images, from top to bottom, displaying the scene point cloud a couple of frames before the person breaches the mask, the point cloud of when the person breaches the mask, and lastly the resulting mask. Figure 4.15c shows the same frames, but instead with the recurrent mask-updating technique. In the second point cloud of Figure 4.15c, it is clear that the room has shifted considerably during the time that the area has been foreground. Because of this the applied depth buffer is no longer prevailing. The depth according to the depth buffer is not

consistent with that of the updated frame depth, which violates privacy on the border of the privacy region. This concern would be mitigated by using a surface or volume reconstruction, e.g., computing a convex hull around the masked points, for example using Open3D, and mask inside pixels using ray casting. This technique, as mentioned, has been explored. An illustration of how this would solve a situation like this is presented in Figure 4.16. This shows that the convex hull volume has enclosed the entire mask point cloud, so the interruption of the mask is reinforced.

Depth Range Issues

Another important aspect when working with depth for auxiliary computer vision tasks, such as depth aware privacy masking, is the range of the perceived depth. More precisely, it is the ability to distinguish relative depth difference that plays an important role here. In Figure 4.17a, a privacy mask is illustrated of a scene with a long distance to the mask. In the second image from the top, the person walking is not visible, even though the feet mark that the event is in fact much in front of the mask. In other words, there is no depth awareness to this given mask. This is further illustrated by examining the point cloud, where the person is unified by the floor. In Figure 4.17b, the same scene is captured using the same MDE model. Here, it can be concluded that the depth is distinguished to a higher degree. Conclusively, depth-range issues can be mitigated by applying a zoom on the input image, as the depth range is highly connected with resolution.

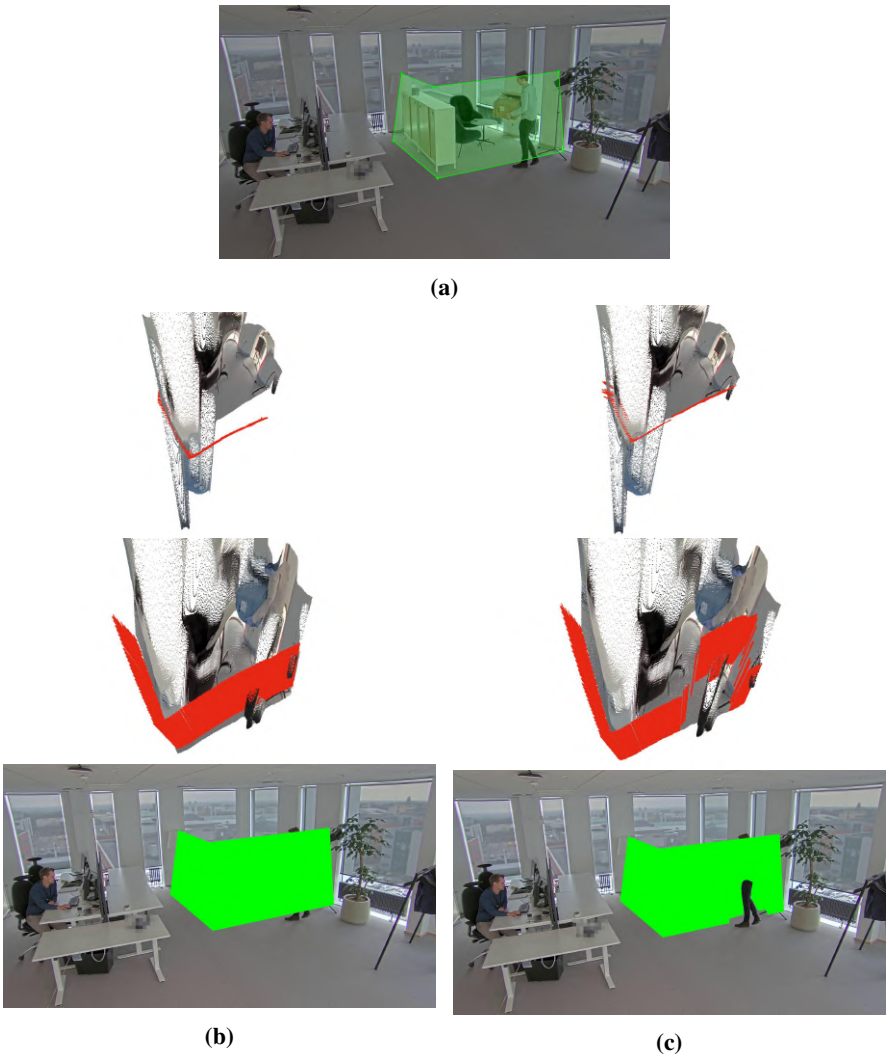
Failure Cases

In Figure 4.18a, another case of depth failure is seen. The person on the bicycle is visible even though the cycling is taking place *behind* the produced privacy mask. Inspecting the produced depth map and resulting point cloud closer, in Figure 4.18b and Figure 4.18c, it is concluded that the source of depth is inaccurate and inconsistent. That is, the model itself is having trouble perceiving the correct relative distance to pixels in the scene, and it is additionally inconsistent of predicting one object's depth.

Presented in Figure 4.19, a sequence of resulting privacy masks are presented, originating from neighbouring video frames. In the middle frame, it is noted a slight breach of privacy, showing a piece of furniture close to the persons face. This is as well a result of inaccurate depth from the source. What is worth noting here is that the error is isolated to a single frame so even though the privacy is breached, this appears only shortly for a streaming video in real-time. However, if the video data were to be recorded a viewer might have the option to pause the video which could be problematic depending on the requirements.

Showcases

In this part, example scenes are presented where the masking application performs depth aware privacy masking according to the design specification. In these scenes,



(a) **(b)** **(c)**

Figure 4.15 Office Scene (DPT Large): An illustration of erroneous depth buffer and breaking of privacy region. **(a)** RGB image with 2D-mask **(b)** Static mask: subsequent frame with no foreground in Open3D, frame with foreground in Open3D, and resulting masked image **(c)** Recurrent masking: subsequent frame with no foreground in Open3D, frame with foreground in Open3D, and resulting masked image.

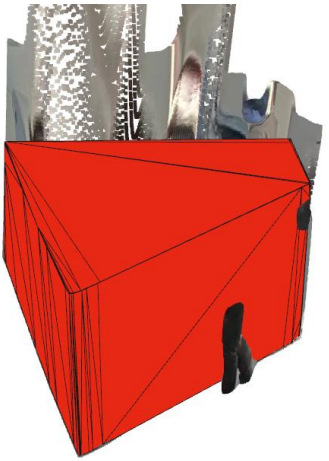
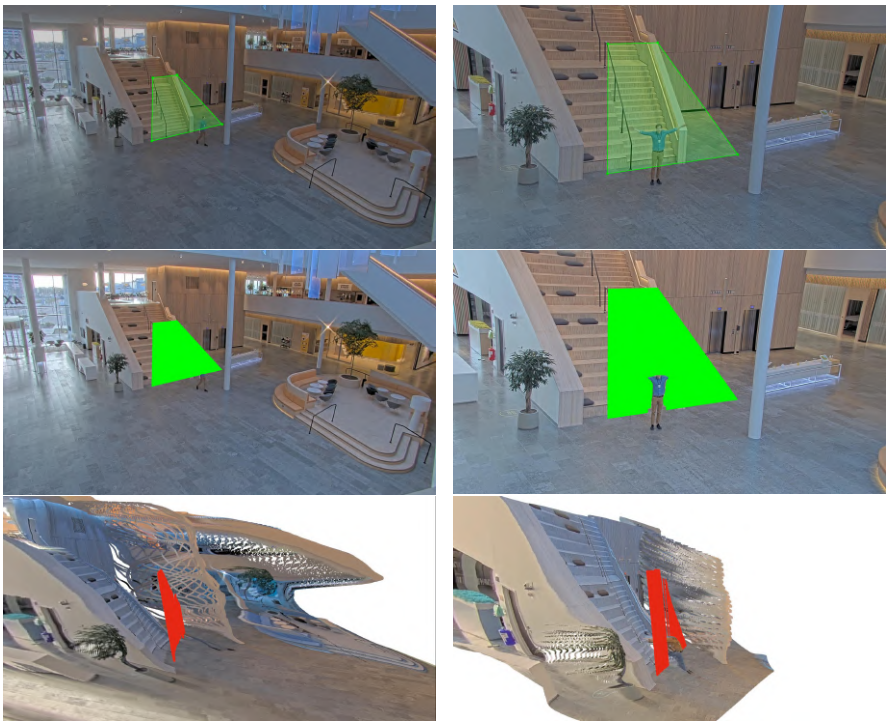


Figure 4.16 Office Scene (DPT Large): An illustration of masking area as a point cloud, and how a convex hull volume reconstruction would mitigate the disrupted mask in Figure 4.15.

any event taking place in front of the mask is visualised, while privacy remains intact behind the mask at hand. Figure 4.20, together with video (a) in Table 4.1, illustrates an indoor scene displaying a corridor and two masks, where one event is a person walking past the first mask, and the other event is the person walking through the second mask. Figure 4.21, together with video (b) in Table 4.1, is displaying an outdoor scene outside a building's entrance. A depth aware mask is placed, illustrating covering an area that, for certain reasons, might not be suitable to record. The example frames show one person walking in front of the mask, another person walking far behind the mask and lastly a frame where the first person has gone behind the mask. Both these scenes also exemplify another style of masking, namely the pixelated mask. Figure 4.22 illustrated an outdoor scene where the windows of a building are to be shielded. This is accomplished simultaneously as a bus is driving past the mask. Furthermore, two more demonstrations are available in Table 4.1, picturing a bike in a parking garage from both a point cloud perspective as well as the end result.

4.5 Discussion

The results of the proposed algorithm look promising and reach, to a large extent, the preset goals: running close to real-time on a desktop machine, robust, hide-rather-than-show, and intuitive user interface. The improved algorithm, with recurrent mask-updating enabled, shows improved robustness, as is seen by Figures 4.12, 4.13 and 4.14. In these figures, temporal consistency, or scene stability, of the depth information is noted to be of high importance. Even though the recurrent mask-



(a) **(b)**

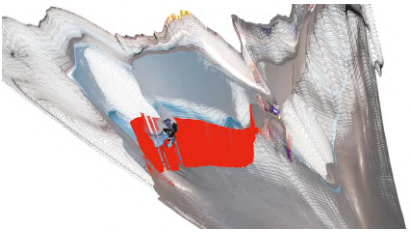
Figure 4.17 Staircase Scene (MiDaS Large): Illustrating the importance of depth range. **(a)** Staircase scene long range: polygon mask, failed depth aware privacy mask, resulting point cloud **(b)** Staircase scene with zoom: polygon mask, depth aware privacy mask, resulting point cloud.

Table 4.1 Videos — 3D Privacy Masking Examples. Video visualising 3D masking functionality.

Scene	Video Description
Corridor	(a) DPT Large - Pixelated mask
Entrance	(b) DPT Large - Pixelated mask
Garage Bike	(c) AdelaiDepth101 - Green mask
Garage Bike	(c) AdelaiDepth101 - Point cloud perspective

(a)

(b)



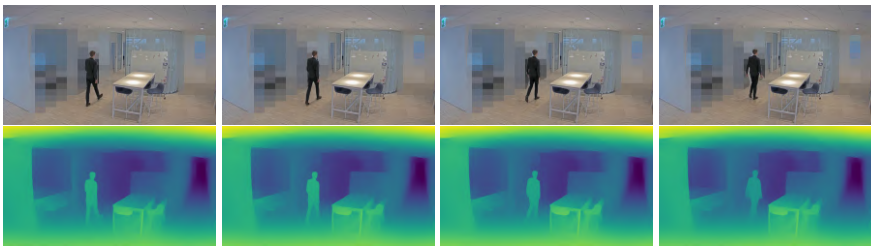
(c)

Figure 4.18 Garage Scene (Monodepth): Case of failure, due to inaccurate depth map. **(a)** Resulting privacy mask **(b)** Monodepth depth map **(b)** Illustration of point cloud.

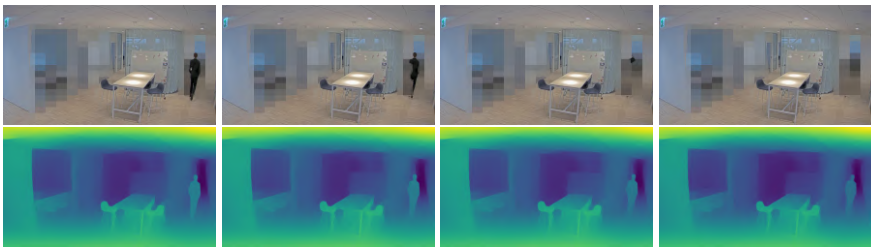
Figure 4.19 Office Scene (DPT Large): Single frame masking error for a sequence of three frames.

updating technique accounts for this, Figure 4.15 exemplifies that this is no certainty, and depth-map instabilities can still play an important role. Continuing, the algorithm is shown to possibly be more robust still. This is achieved by defining the mask as a convex hull, as illustrated in Figure 4.16. This leads to further increased robustness when the current algorithm fails in doing so. An idea, that no time was found to further investigate, is to, instead of keeping a weighted sum depth buffer, interpolate/extrapolate depth values over missing values on the baseline based on the current frame's background depth values. In other words, this would result in a similar mask reinforcement as the convex-hull technique without the need of 3D computational operations.

Moreover, it was proven that depth accuracy and object depth consistency are of great importance. More importantly is that the relative depth between objects is



(a) Event in front of mask.



(b) Person going through the mask.

Figure 4.20 Corridor Scene (DPT Large): Show-casing scene.

Figure 4.21 Entrance Scene (DPT Large): Show-casing scene.

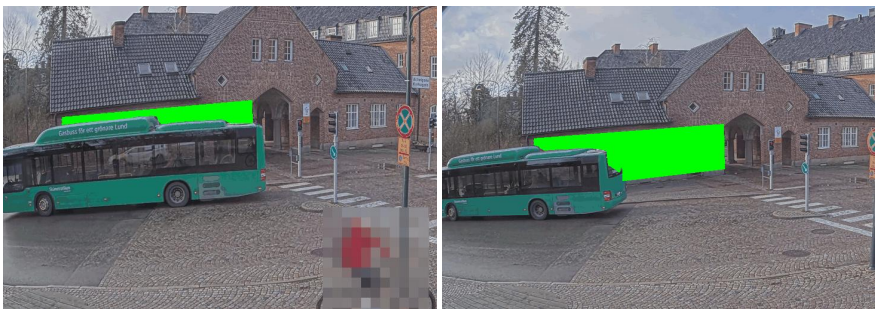


Figure 4.22 Bus Scene: Show-casing scene.

accurate and consistent. Depth-accuracy problematics is best illustrated by the point cloud in Figure 4.15, where the shelf is seen to blend together with the person walking in front of it to some degree. Additionally, a failure case due to inaccurate and inconsistent frame depth was presented in Figure 4.18. The importance of range resolution, in terms of utilising depth information for depth aware privacy masking, is highlighted in Figure 4.17.

Conclusively, the application looks promising even though not completely robust in every situation. However, as demonstrated in Figure 4.19, erroneous masking is often isolated to single frames, so it is arguable that this would not lead to a breach of privacy to any severe degree. However, this would depend on the application and its requirements. It should also be noted that the masking algorithm is no better than its source of depth. We have concluded on the three most important aspects in terms of depth quality, when it comes to depth aware privacy masking. These are listed as: accuracy, scene depth stability, and depth range resolution. These aspects are further examined in Chapter 5.

Depth Source Comparison

It is worth noting that this algorithm and feature in practice would be achievable with any depth sensor or product with depth functionality, such as a LiDAR or a high resolution stereo camera, but could come with other cons. The main issue with a LiDAR would be the added sensor cost, limited frame rate, as well as the natural decrease in resolution as the depth increases because of the fixed angular resolution. Since a LiDAR will also not give pixel-wise depth, other methods such as depth interpolation and/or point-cloud based semantic segmentation [Biasutti et al., 2019] to differentiate object locations in relation to a defined 3D mask, would likely have to be supplementary used as well. A stereo camera would enable high-resolution depth maps, but be limited to indoor environments, given the typical max range of around 20 m. It is also worth noting that stereo cameras may require high computing capabilities of the target device, depending on output quality requirements, as well as the cost of handling two simultaneous image streams. A commonly used consumer stereo camera, which we also used to acquire indoor ground truth for accuracy evaluation in Chapter 5, lists its SDK requirements to render real-time depth as Dual-core 2.3GHz CPU, 4 GB RAM, and a NVIDIA's GPU with computing capability > 3.0 [Stereolabs, 2022], which currently would be very high specifications for an embedded device.

5

Model Evaluation

This chapter present a thorough evaluation of the selected Monocular Depth Estimation models in the emulated surveillance domain. We perform quantitative and qualitative evaluation and analysis with regards to model accuracy, output stability and range resolution with three self-captured data sets. We also present a post-processing method to mitigate the effect of output instability. Lastly, the chapter includes a conclusive discussion regarding the achieved results.

5.1 Data Sets

In this thesis we utilise three compositions of self-captured data sets that are captured to mimic surveillance domain scenes. The main characteristic that separates these data sets from existing online data sets is that it is from a static point of view as of that of a wall-mounted security camera. Another surveillance domain attribute present is mostly static scenes with repeated minor movement in distinct regions of the scene. Additionally, when choosing scenes to capture, we have, based on our inherent judgement, chosen scenes that are realistic for an installed surveillance camera. Two of the produced data sets are for depth accuracy evaluation, with ground truth depth available for a subset of the image pixels. The third data set is produced for frame-by-frame stability evaluation, investigating temporal instability in the context of utilising depth prediction on real-time video. The data sets each include multiple different scenes with varying length captured as a 30 FPS video stream and then converted into a sequence of frames at a rate of 10 FPS. A summary of the data sets is presented in Table 5.1.

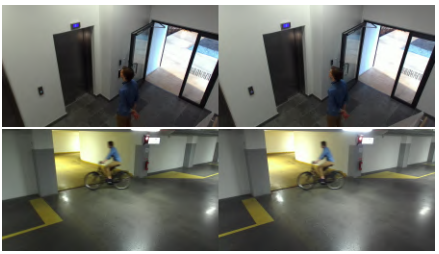
Table 5.1 Data sets. Self captured data sets used for evaluation in the surveillance domain.

Data set	Ground Truth	Dense	# Scenes	# Frames	# Samples
Accuracy — indoor	Stereo	✓	14	4382	2100
Accuracy — outdoor	Lidar	(✓)	4	1656	750
Stability	—*	—	3	998	998

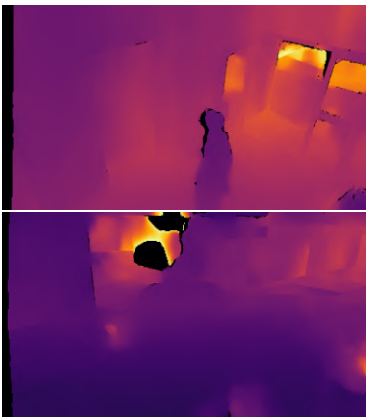
* For Stability Data set, the "ground truth/target" is defined as a reference (predicted) frame, here the first frame in each scene.

Accuracy — Indoor Data Set

To collect a data set for indoor scenes with limited range a stereo camera, ZED Stereo Camera [Stereolabs, 2022], was used. A stereo camera such as ZED Stereo Camera can produce a dense depth map by capturing and fusing information from multiple images, see more detailed theory in Section 2.1. It is important to note that the quality of the produced depth map can vary depending on available stereo-matching features. However, for our purposes the quality was deemed high enough to perform a fair and meaningful evaluation and comparison. In order to process captured stereo images and to acquire ground truth depth maps, StereoLabs' ZED SDK 3.7 was used.



(a) Stereo pairs from two scenes in the Accuracy — Indoor data set.



(b) Produced depth maps from stereo camera to be used as ground truth in indoor accuracy evaluation.

Figure 5.1 Depth from Stereo (a) Stereo pair as captured from a ZED Stereo Camera **(b)** Produced ground truth depth map from (a). Notice that the produced depth maps are not fully complete, and here contain small erroneous regions under strong light-sources.

Scene	# Frames
Conference 1	200
Conference 2	220
Elevator 1	190
Elevator 2	210
Garage Bike 1	167
Garage Bike 2	258
Kitchen	248
Office	359
Stairway	412
Garage Car 1	359
Garage Car 2	430
Garage Car Exit	630
Corridor	291
Lounge	406

Table 5.2 Accuracy — Indoor data set. Self-captured data set with a stereo camera as ground truth for accuracy evaluation. The set contains 14 scenes, in 9 unique locations.

Accuracy — Outdoor Data Set

To collect a data set with outdoor scenes, with in general longer range, a LiDAR sensor of the type Cepton Vista-P60 [Cepton, 2022] was used. The data set is divided into four outdoor scenes, each with characteristics of typical surveillance scenes, meaning from a tilted point of view, a mostly static scene, and monitoring areas with movement of people. The disposition of the data set is provided in Table 5.3. Example images from these scenes are shown in Figure 5.2, along with the corresponding depth map.

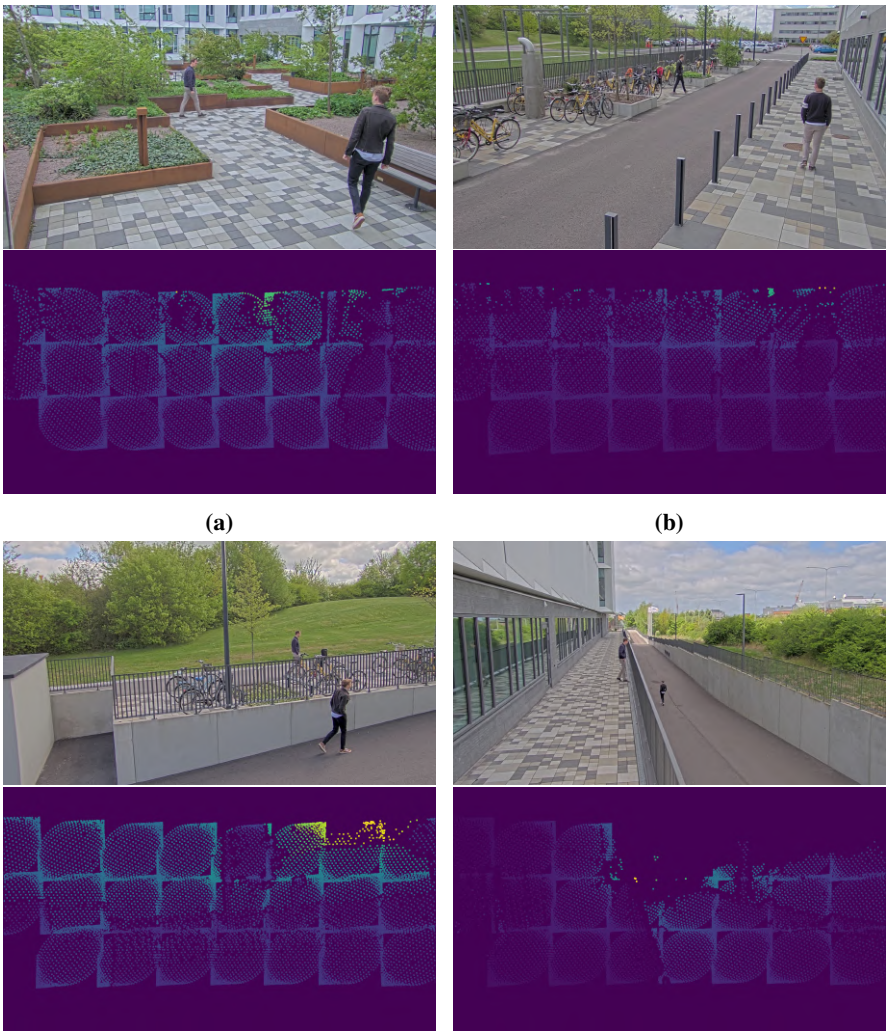
Table 5.3 Disposition of scenes for outdoor depth accuracy data set, with ground truth from a LiDAR sensor.

Scene	# Frames
Courtyard	536
Entrance	272
Bicycle lane	319
Loading Area	529

To translate one frame of LiDAR measurement points, to a depth map synchronised with a corresponding RGB frame, is not completely trivial. This requires three principal processing steps, that will be further introduced in the proceeding sections: calibration, synchronisation, and depth map interpolation.

Calibration

In order to use the depth retrieved from the LiDAR, in the form of a point cloud, these points are transformed into image space, using image formation of a pinhole camera model, described by (2.2). To employ this, for a projection of world



(a) **(b)**

(c) **(d)**

Figure 5.2 **Depth from LiDAR (a)** Courtyard **(b)** Entrance **(c)** Bicycle Lane **(d)** Loading Area.

points, (X,Y,Z), into image space, (x,y), the intrinsic parameters of the given camera's focal length, (f_x, f_y), and origin, (x_0, y_0), are needed. These were retrieved by OpenCV's [Bradski, 2000] library `calibrateCamera`, and a printed checkerboard pattern, according to the "Camera Calibration" documentation [OpenCV, 2022]. Illustration of this procedure is presented in Figure 5.3. In Figure 5.3a, we see where OpenCV have found corner locations on the chess board, which then acts as the foundation of calibrating the intrinsic camera parameters. This is possible since the size of the chess board is known. To the right, in Figure 5.3b, the result is presented where the projected LiDAR points are mapped on top of the corresponding RGB image.

(a) Intrinsic Calibration **(b)** Projected Points

Figure 5.3 Intrinsic Calibration (a) Illustration of intrinsic camera calibration using a chess board. **(b)** Projected points into image space, mapped on top of the original image.

As is seen in Figure 5.3b, the intrinsic camera calibration is not enough in order to acquire an accurate depth map, and it is noticed that the points are shifted in relation to their corresponding objects. So in addition to determining the intrinsic parameters, there is also a need of finding the extrinsic parameters. The camera and LiDAR sensor are mounted on a plate next to each other, as is shown in Figure 5.4. Hence, it is clear that there will be a slight translation of points along the camera's x-direction, measured to approximately 12 cm. In addition, the exact location of each sensor origin in z-direction was unknown. However, for the LiDAR, this was estimated by measuring the distance to an object both by hand, and examining the LiDAR-measured depth value. This way, the distance along z-direction, to the LiDAR origin, was estimated to 3 cm. Lastly, the translation along y-direction was approximated to zero.

Assuming these small translations being constant and correct, the calibration is still not visually accurate. Even though the sensors are mounted steadily next to each other, there is room for some slight rotational transformations, where small rotations show great impact on the accuracy of the projections. These rotations where estimated by a trial-and-error procedure. This is done by tweaking the rotations about the x-, y-, and z-directions, so as to align the static objects with the projected LiDAR point cloud. The result from the extrinsic and intrinsic calibration put to-

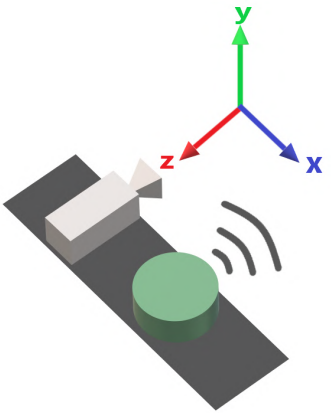


Figure 5.4 Illustration of the camera and LiDAR setup, and the camera coordinate system. The camera is positioned to the left, and the LiDAR sensor is positioned to the right.

gether is presented in Figure 5.5

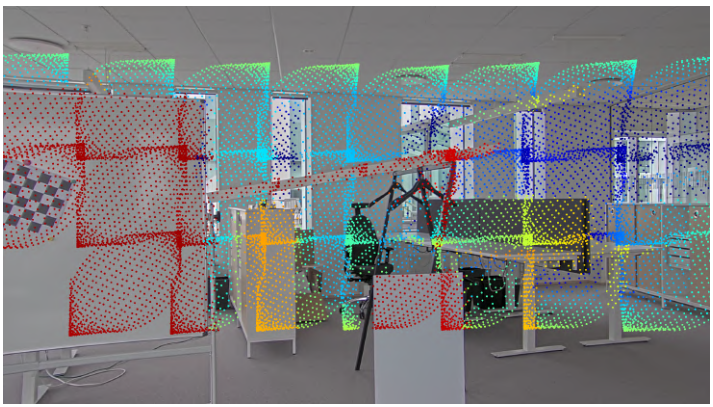


Figure 5.5 The LiDAR point cloud projected into image space after intrinsic and extrinsic camera calibration.

The attentive reader can, from examining the image in Figure 5.5, notice that there are leaking points onto objects that should cover these points. This is, for example, seen by the right-most edge of the white board where blue points (further away) are shown together with the red points marking the distance to the white board. This is the result of points being visible to the LiDAR while corresponding pixels are not visible for the camera, because of the translation along the x-direction, i.e., the parallax effect. This leads to erroneous depth values in these regions and is needed to account for. To solve this, the LiDAR point cloud was projected into 3D-

space with Open3D [Zhou et al., 2018]. It is desired to determine which points that are non-visible, considering the location of the camera, and the point cloud itself. This is done with Open3D's implementation of "hidden point removal", a method presented by [Katz et al., 2007]. The final results from the calibration are presented in Figure 5.6.

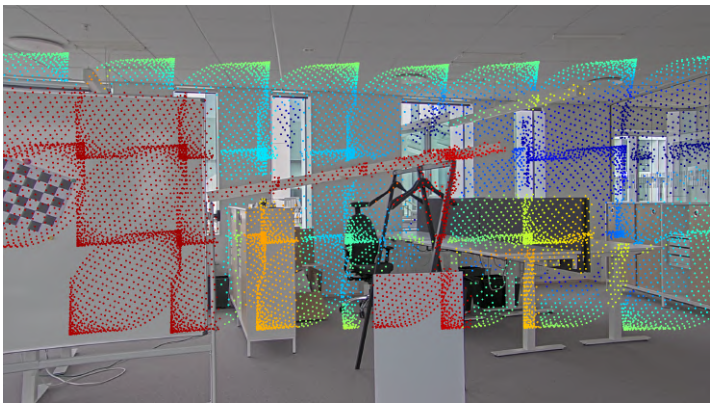


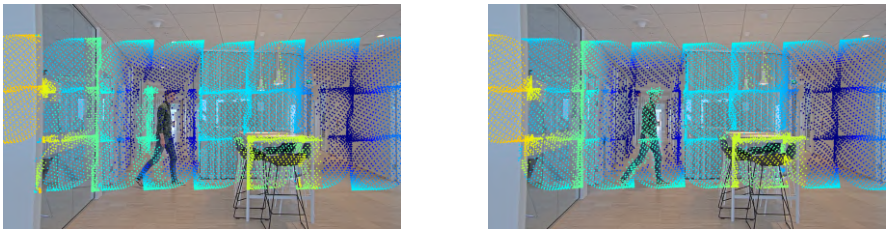
Figure 5.6 The final result from the camera and LiDAR calibration showing LiDAR points projected onto the corresponding image frame.

Synchronisation

In order to produce an accurate RGBD-data set, the two sensors also need to be synchronised. This is achieved by grabbing the most recent camera frame, on every incoming LiDAR frame, assuming that both sensors are connected. This should, with our reasoning, lead to the minimum time error between a LiDAR and camera frame possible, without looking into software or hardware triggers of the sensors. Yet, investigating the result, it is clear that there is still a distinct synchronisation error. Hypothesising that this is because of an unknown delay from the time of capture to the time of receiving the two frames, and that this is constant, the aim is to find this offset and synchronise the frames thereafter. In other words, all needed is to synchronise on one frame, and subsequently pair camera frames with LiDAR frames based on this offset. This is exemplified by Figure 5.7, where Figure 5.7a shows the overlaying point cloud of the first LiDAR frame, on the corresponding image frame. In Figure 5.7b, the LiDAR frames have been traversed, finding the frame that aligns with the person in the image.

Interpolation

As can be seen in Figures 5.5 and 5.6, the LiDAR provides only a sparse point cloud. It is desirable to attain a more dense depth map in order to increase the amount of data for every frame. Therefore, as a last data processing step, the point cloud is interpolated. To extend the depth map, a MATLAB script provided by [Balcilar,



(a) First frames **(b)** Synced frames

Figure 5.7 Synchronisation (a) The first two frames showcasing the offset in captured camera and LiDAR frames. **(b)** Subsequent LiDAR frame, aligning with the person of the camera frame.

2022] is used. This script interpolates the depth by extracting depth to neighbouring pixels by taking a weighted sum of the known depth in every grid region, where the weight is determined by the distance from the grid centre point. That is, the further away a pixel is from the known depth at the grid centre, the lower the summation weight becomes. The final depth map, to be used as ground truth in depth accuracy evaluation, is presented in Figure 5.8.

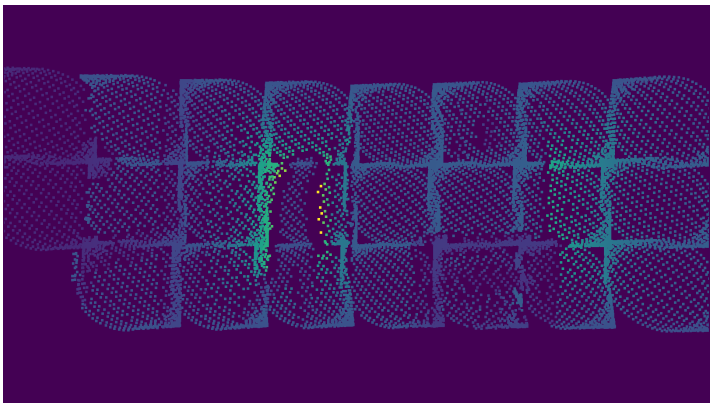


Figure 5.8 Example image of a processed depth map to be used as ground truth for depth accuracy evaluation.

Stability Data Set

The data set for stability evaluation contains three scenes with three levels of dynamics or scene movements. They were captured with a regular surveillance camera from a fixed surveillance perspective. The levels of dynamics are referred to as *High Dynamics*, *Low Dynamics* and *No Dynamics/Static*. High dynamics scenes involve walking and moving around in the scene. The low dynamics scenes involve standing or sitting still with only minor movements. Static scenes involve no moving objects

at all and the frames are identical except for tiny dynamics such as brightness fluctuations, shadows and wind, etc.

Table 5.4 Stability data set. The stability data set divided into three subset for different level of dynamics.

Scene	High Dynamics # Frames	Low Dynamics # Frames	Static # Frames
Entrance	181	80	30
Kitchen	121	145	79
Roof top bar	140	101	82
Total	442	365	191

5.2 Depth Accuracy

The depth accuracy evaluation on our custom data sets have been evaluated on all models. This is done in a similar manner as presented in Section 3.5. To summarise, the predictions are aligned with the ground-truth, in the prediction's space, by performing a scale and shift alignment in accordance with least-squares (2.7), and in turn solved by (2.10). After the alignment, the predictions are evaluated in ground truth space on the metrics *AbsRel* (2.18) and δ_{err} (2.20). This is averaged over all frames of the data set.

The result on the indoor data set, captured with a ZED stereo camera and the outdoor data set, captured with a camera and LiDAR sensor, will be presented in each of the subsequent subsections.

Indoor

For evaluation on the indoor data set, a depth cap of 10, 15, and 20 m was applied, meaning ground truth values above the depth cap is ignored during evaluation. Here 20 m represents the upper maximum range the stereo camera is specified to be accurate. For each of the 14 scenes in Table 5.2, we randomly sample 150 frames, resulting in a total of 2100 sampled frames. This is in order to not introduce unnecessary scene-bias due to varying length. The set of randomly sampled scenes is the same for each metric, model and depth range cap.

Table 5.5 shows the results of accuracy evaluation. MiDaS Large marginally outperforms both DPT Large and DPT Hybrid in both metrics, and the accuracy trend is consistent over all depth range caps. For the delta metric, AdelaiDepth101 marginally scores higher than both DPT models. For the smaller models, MiDaS Small outperforms Monodepth by a large margin in both *AbsRel* and δ_{err} metric.

Table 5.5 Depth Accuracy — Indoor data set. Accuracy evaluation on indoor data set, with varying depth cap of 10, 15 and 20 m.

Model	$AbsRel_{10m}$	$AbsRel_{15m}$	$AbsRel_{20m}$	$\delta_{10m>1.25}$	$\delta_{15m>1.25}$	$\delta_{20m>1.25}$
AdelaiDepth50	0.131	0.142	0.149	0.154	0.173	0.183
AdelaiDepth101	0.128	0.137	0.141	0.148	0.164	0.169
DPT Hybrid	0.125	0.132	0.135	0.157	0.172	0.178
DPT Large	0.124	**0.130**	**0.133**	0.152	0.165	0.170
MiDaS Large	**0.123**	**0.130**	0.134	**0.143**	**0.162**	**0.167**
MiDaS Small	0.140	0.148	0.151	0.182	0.199	0.205
Monodepth	0.199	0.213	0.217	0.364	0.389	0.397

Outdoor

For evaluation on the outdoor data set, a depth cap of 80 meter was applied. The full data set contains 250 randomly sampled frames from each of the four scenes presented in Table 5.3, totalling 1000 frames. The scatter plot of this result, for the AdelaiDepth101 model, is presented in Figure 5.9. What is essential to note here is that the average error is very high compared to previously analysed data sets. Looking closer at the plot, it is one scene specifically that the model fails to even moderately infer depth from, namely the ”Loading Area”-scene. Similar patterns are shown for all models, both on *AbsRel* as well as δ_{err} metric. Conclusively, a scene has been found that all models are unable to handle, in terms of establishing accurate depth predictions.

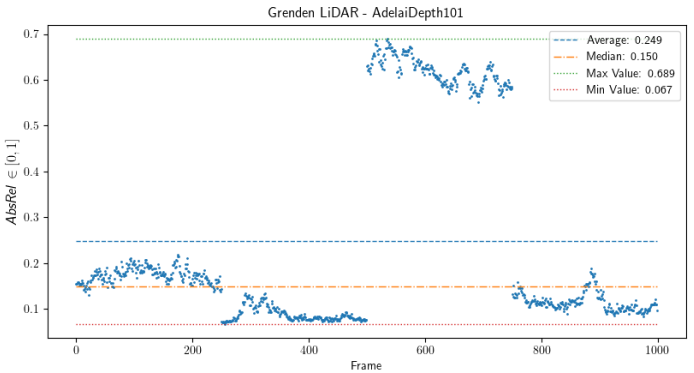
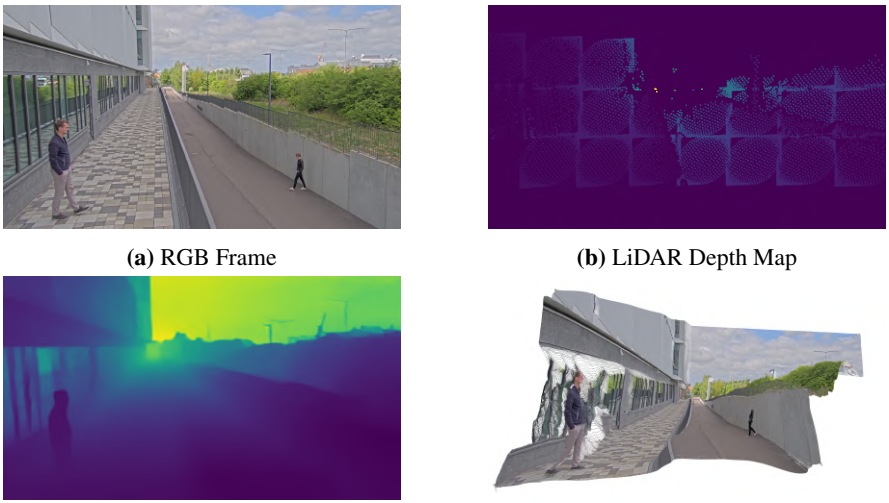


Figure 5.9 Scatter plot of each frame's *AbsRel* metric on AdelaiDepth101 predictions, evaluated on the outdoor data set containing 250 frames from four scenes.

Example frame data of this scene are given in Figure 5.10. Figure 5.10a displays the captured RGB frame, showing the *Loading Area* scenery. Examining this, it is concluded that the scene in itself appears almost as an optical illusion perceived by the human eyes. The scene in itself actually contains two separate levels of height,

to either side of the railing. This is distinguishable in the LiDAR depth map, Figure 5.10b, where the points to the right of the railing shows larger depth values. However, looking at the predicted depth map from AdelaiDepth101 in Figure 5.10c, it does not appear to have captured this depth difference. Similarly as we are having trouble differentiating the depth of this scene, without additional information about the scene at hand, the MDE models do so too. This is further verified by investigating the resulting point cloud in Figure 5.10d, where the right handed road is visually appearing to be on the same level as the pavement to the left of the railing. Because of this, we withdraw this scene from the outdoor data set from the general analysis, and present results on this subset of the data separately.



(a) RGB Frame

(b) LiDAR Depth Map

(c) AdelaiDepth Depth Map

(d) AdelaiDepth Point Cloud

Figure 5.10 Loading Area Example Frame

The total outdoor data set contains 250 frames from three scenes, excluding the ”Loading Area”-scene, totalling 750 frames. The result is presented in Table 5.6. For the *AbsRel* metric, DPT Large shows the best performance, whereas MiDaS Large is second best. This result differs slightly considering the δ_{err} metric, where DPT Hybrid is the strongest contestant and DPT Large at second place. For the smaller models, it is once again concluded on an exclusive difference, favouring MiDaS Small over Monodepth.

The scene-specific results from the outdoor LiDAR evaluation data set are presented in Table 5.7. Firstly, it is clear that any DPT version is a top performer for every scene and either MiDaS or the corresponding DPT version is second best. This, with the exception of AdelaiDepth101 on the ”Bicycle Lane”-scene, evaluated on the δ_{err} metric. For the smaller models, MiDaS Small is once again shown to outperform

Table 5.6 Depth Accuracy - Full Outdoor data set. Depth accuracy score on custom outdoor, long range, data set, presenting both the *AbsRel* error as well as the delta-threshold error ($\delta_{>1.25}$).

Model	*AbsRel*[*]	$\delta_{>1.25}$[*]
AdelaiDepth101	0.127	0.129
AdelaiDepth50	0.137	0.129
DPT Large	**0.115**	0.128
DPT Hybrid	0.117	**0.114**
MiDaS Large	0.116	0.139
MiDaS Small	**0.165**	**0.240**
Monodepth	0.270	0.589

[*] 250 sampled frames from the first three scenes of Table 5.3, excluding "Loading Area"-scene, totalling 750 frames.

Monodepth uniformly. Further, looking at each individual scene as presented in Table 5.7, it is concluded that every model performs its inherent top and worst scores on the same data sets. That is, looking at the three data sets, excluding "Loading Area", all models perform their best result on the "Entrance" scene and consistently the worst result on the "Bicycle Lane" scene.

Table 5.7 Depth Accuracy — Indivudual scenes. Depth accuracy score on custom outdoor, long range, data set, presenting both the AbsRel error as well as the delta-threshold error.

	Bicycle Lane		Entrance		Courtyard		Loading Area[*]	
Model	*AbsRel*	$\delta>1.25$	*AbsRel*	$\delta>1.25$	*AbsRel*	$\delta>1.25$	*AbsRel*	$\delta>1.25$
AdelaiDepth101	0.173	0.186	0.088	0.081	0.109	0.099	0.618	0.800
AdelaiDepth50	0.178	0.192	0.104	0.059	0.121	0.118	0.458	0.741
DPT Large	0.175	0.231	**0.063**	**0.037**	**0.085**	**0.079**	**0.334**	**0.666**
DPT Hybrid	**0.144**	**0.157**	0.088	0.046	0.101	0.097	0.405	0.717
MiDaS Large	0.161	0.241	0.075	0.059	0.102	0.094	0.444	0.724
MiDaS Small	**0.190**	**0.264**	**0.162**	**0.252**	**0.136**	**0.183**	**0.393**	**0.666**
Monodepth	0.345	0.839	0.246	0.504	0.212	0.422	0.415	0.853

[*] The "optical illusion" scene where all models more or less fail to predict the depth, see Figure 5.10 for an illustration.

5.3 Stability

As introduced during the 3D masking development in Section 4.3 under *Temporal Instability*, predicted depth values are not consistent in time domain even if the scene view is fixed. One cause for this is emphasised in Section 3.4, where it was described that AdelaiDepth learns depth up to an *affine transformation*, meaning depth up to an unknown *scale* and *shift*. Naturally, during a video sequence, if these unknown scale and shift vary throughout the scene, the consequence is temporal instability. This issue is present for all investigated models.

Temporal Instability can be illustrated by Figure 5.11 which shows the coefficient of variation ($CV_i = \frac{\sigma_i}{\mu_i}$) during the scene, where i corresponds to pixel location. Here, some regions that are supposed to be consistent, such as floor and walls, manifest unwanted high variation during predictions. The only region expected to be bright is the bottom right.

Evaluation Method

In order to perform quantitative evaluation of model stability, three different metrics were utilised. These were DSSIM (structural dissimilarity) (2.24), MAE (mean absolute error) (2.25) and RMSE (root mean squared error) (2.26). As stated in Section 5.1, the data set used is divided into three categories, *high dynamics*, *low dynamics* and *static*. For ground truth, or reference frame, we use the first frame in each scene, for each respective model. Thus, since the camera position is fixed, the metrics will describe how well each succeeding frame matches the reference/ground truth frame.

In order to perform stability evaluation fairly across all models and scenes, three pre-processing steps are called for. These are *static region bit-mask*, *depth alignment* and *mean normalisation*.

In order to not introduce errors from dynamic regions, a scene-specific static bit-mask, representing the pixels to be excluded during the evaluation, was used. This bit-mask is calculated by utilising an existing foreground–background detector, similar to the background detector used in Section 4.3. An example of this bit-masking technique is illustrated in Figure 5.13.

Next, depth alignment into realistic depth and mean normalisation is performed. This is desired since the ground truth, or reference frame, is unique for every model.

Since the raw depth output is of unknown scale and shift, there exists a *global* scale and shift that transform and align the depth map into realistic metric depth. To clarify, for models that output disparity, the output scale may differ in the range of 10^{-5} and 10^5 depending on scene and model. Since the relation between disparity and depth, i.e., the baseline and focal length in a stereo-camera setup, see Section 2.1, is

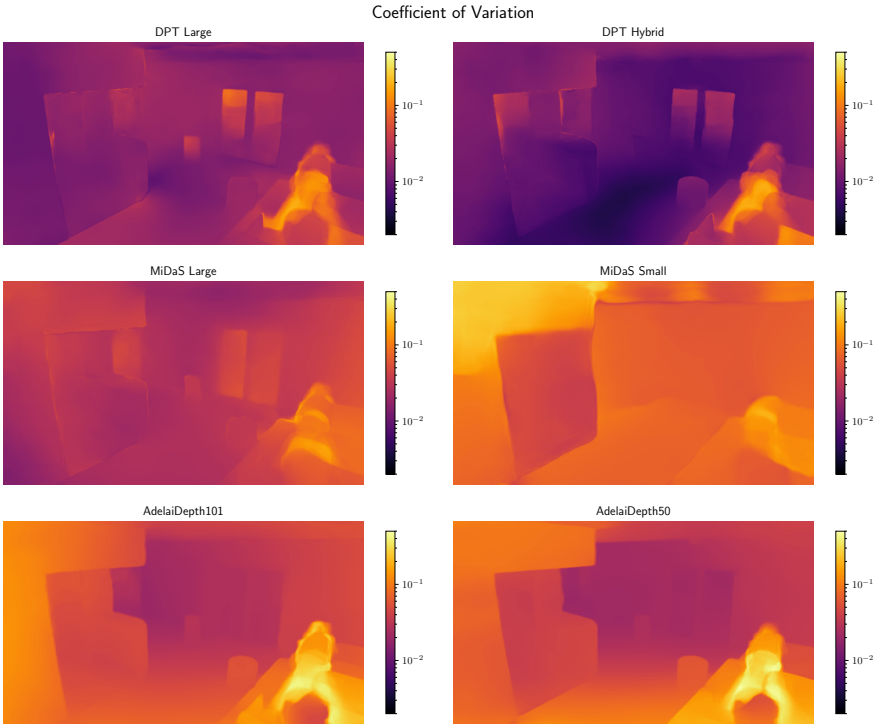
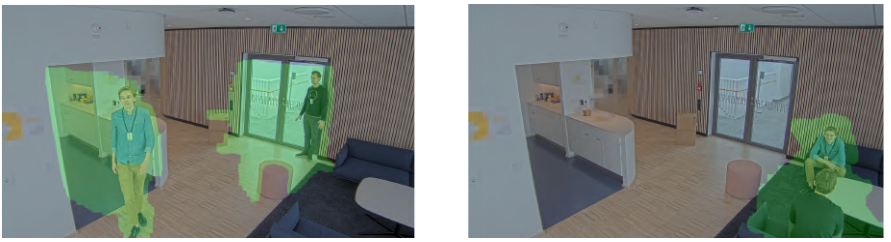


Figure 5.11 Coefficient of Variation. Coefficient of Variation in a logarithmic scale (for visualisation purposes) from a sequence of 121 frames of the *Low Dynamic* kitchen scene. Darker regions correspond to high stability, while brighter regions correspond to instability. The equivalent RGB-image is shown in Figure 5.12.

Figure 5.12 Kitchen — Low Dynamic. Image from low dynamic kitchen scene. Minor movement is only present in the bottom right corner.



(a) Resulting bit-mask for *High Dynamic - Kitchen* scene.

(b) Resulting bit-mask for *Low Dynamic - Kitchen* scene.

Figure 5.13 Dynamic region bit-mask. Resulting bit-masks used to ignore true dynamic regions during stability evaluation.

unknown, the focal length multiplied with the baseline can be asserted to be equal to 1 and the transformation from disparity to depth becomes

$$d = f\frac{B}{Z} = \frac{1}{Z} \tag{5.1}$$

where d is disparity and Z is depth. In order to end up in realistic depth space, for example in the range of 2 to 20 m in the case of the kitchen scene in Figure 5.12, a scale and shift transformation is found using least-squares fit, as first described in (2.7), by

$$(s,t) = \arg\min_{s,t} \sum_{i=1}^{2} (sd_i + t - d_i^*)^2 \tag{5.2}$$

and solve for scale and shift, s and t, respectively, when d^* is asserted to 2 and 20, and d is the minimum and maximum depth map output. This *global* shift and scale is computed once for the reference frame, and then applied to every sequential frame, which uniformly transforms each frame into realistic depth space:

$$d_{raw} \xrightarrow{s,t} d_{\text{2-20m}}.$$

The metrics are then calculated frame-wise and averaged into a final DSSIM, MAE and RMSE measurement for each model. The MAE and RMSE are mean normalised by the interquartile range, with $\frac{\text{MAE}}{Q_3 - Q_1}$ and $\frac{\text{RMSE}}{Q_3 - Q_1}$, respectively.

Stabilisation Technique — Static Area Alignment

We propose a post-processing stabilisation technique applicable in real-time in order to reduce the effect of temporal instability for static camera installations, common in video surveillance.

Acknowledging that the models only learn depth up to an *affine transformation*, and inspired by the established use of scale and shift for ground truth alignment during

accuracy evaluation, we suggest using the same technique in order to align to a previous predicted frame, i.e., a *reference* frame. We motivate this by the observation that large parts of the pixel content of a fixed surveillance camera are *static*, such as fixed ground, walls and stationary objects, etc., at which depths should be constant. Hence, by continuously aligning, in terms of scale and shift, incoming predicted depths to a reference frame, the overall scene stability should improve.

However, in order to not introduce alignment issues when dynamic objects are involved, only static pixels in the current frame and reference frame should be included when solving for the alignment coefficients with least-squares. Hence, similar to (2.7), this is mathematically described as

$$(s,t) = \arg\min_{s,t} \sum_{i=1}^{M^s} \left(s\mathbf{d}_i + t - \mathbf{d}_i^R\right)^2 \tag{5.3}$$

where M^s, $\mathbf{d}$, and $\mathbf{d}^R$ are the number of *static* pixels, the current depth map, and the defined reference depth map, respectively. The static pixels for each frame are, as previously, identified by using an existing foreground–background detector out-of-the-box.

The proposed post-processing stabilisation method is evaluated with similar methodology as described in Section 5.3, and the overall transformations scheme can be described as

$$d_{raw} \xrightarrow{\hat{s},\hat{t}} d_{aligned} \xrightarrow{s,t} d_{aligned_{2-20m}} \tag{5.4}$$

where $\hat{s}$ and $\hat{t}$ are the coefficients for the reference frame alignment.

This technique do add some latency. Most of the latency can be attributed to the foreground–background detector used, as well as solving for the alignment coefficients. However, this should very much still be applicable for real-time usage since optimised foreground–background detectors often are present in high performance surveillance cameras. Also, the alignment coefficients can be solved for fast and efficiently, since the closed-form solution of linear least-squares (2.10) can be utilised.

Finally, this technique does not have impact on the accuracy metrics *AbsRel* and δ_{err}. This is because both accuracy evaluation metrics utilise the same alignment technique targeting the ground truth values. This essentially reverses the operations in (5.4).

Results

The results of the full stability evaluation for each level of dynamics are presented in Tables 5.8, 5.9 and 5.10. For *High Dynamics*, the DPT models consistently scores high in all three metrics. Interestingly, the more compact AdelaiDepth50 model, while having the same input size, consistently achieves higher stability than AdelaiDepth101, and even outperforms both DPT models in RMSE. This is because

AdelaiDepth50 performs remarkably well in the Roof Top scene, as shown in Figure 5.14, while the other models fall short. Similarly, for *Low Dynamics*, DPT Hybrid performs the best in the DSSIM and MAE category, while AdelaiDepth50 performs best in RMSE. For *Static*, all models perform better than in Low Dynamics. Surprisingly, the winner on the static dataset is Monodepth, which is generally the less accurate model.

Furthermore, while the average of each stability metric is the main measurement presented, the locality, spread and outliers are also important aspects. This is presented for MAE for High Dynamics as a box-and-whisker plot in Figure 5.15. Interestingly, while DPT Large have marginally lower median- and mean MAE than AdelaiDepth50, the latter have much lower spread, or variation, which may be preferable.

Table 5.8 High Dynamics Results from *High Dynamics* scenes. Lower is better in all categories.

Model	DSSIM $\times 10^{-3}$	DSSIM w. stabilise $\times 10^{-3}$	MAE $\times 10^{-2}$	MAE w. stabilise $\times 10^{-2}$	RMSE $\times 10^{-2}$	RMSE w. stabilise $\times 10^{-2}$
AdelaiDepth101	7.41	6.30 (-15.0%)	4.30	3.53 (-17.9%)	7.83	6.20 (-20.8%)
AdelaiDepth50	4.47	5.51 (+23.3%)	2.48	2.08 (-16.1%)	**4.31**	**3.56** (-17.3%)
DPT Large	3.41	**1.18** (-65.4%)	**2.43**	**1.86** (-23.5%)	6.03	5.78 (-4.0%)
DPT Hybrid	**2.97**	1.62 (-45.5%)	2.76	1.99 (-28.0%)	7.22	5.36 (-25.8%)
MiDaS Large	4.09	2.65 (-35.2%)	4.00	3.44 (-14.1%)	8.75	7.73 (-11.7%)
MiDas Small	11.71	8.14 (-30.5%)	7.64	6.81 (-10.9%)	16.50	15.92 (-3.5%)
Monodepth	9.03	8.52 (-5.6%)	4.42	5.57 (+26%)	10.81	11.68 (+8%)

Table 5.9 Low Dynamics Results from *Low Dynamics* scenes. Lower is better in all categories.

Model	DSSIM $\times 10^{-3}$	DSSIM w. stabilise $\times 10^{-3}$	MAE $\times 10^{-2}$	MAE w. stabilise $\times 10^{-2}$	RMSE $\times 10^{-2}$	RMSE w. stabilise $\times 10^{-2}$
AdelaiDepth101	4.59	5.27 (+14.8%)	5.91	3.43 (-41.9%)	7.80	5.00 (-35.8%)
AdelaiDepth50	2.14	1.33 (-37.9%)	3.68	2.46 (-33.2%)	**4.91**	**3.67** (-25.3%)
DPT Large	1.09	0.75 (-31.2%)	2.66	2.41 (-9.6%)	6.85	6.84 (-0.1%)
DPT Hybrid	**0.64**	**0.53** (-17.2%)	**2.33**	**2.04** (-12.4%)	5.55	5.57 (+0.5%)
MiDaS Large	2.00	1.10 (-45.0%)	3.61	2.85 (-21.1%)	6.07	5.16 (-15%)
MiDas Small	9.76	5.38 (-44.9%)	10.38	8.79 (-15.4%)	19.89	15.65 (-21.3%)
Monodepth	6.68	5.94 (-11.1%)	5.32	7.66 (+44.0%)	11.92	12.80 (+7.4%)

There exists a clear trend relating scene dynamics to overall scene stability. Nearly all models progressively perform worse stability-wise from high to static scene data set, indicating the issue with temporal inconsistency. By inspecting the captured footage as an animated point cloud, there is correlation between movement in one part of the scene, to large depth changes of static background areas in other parts of the scene, see videos (a) and (d) in Table 5.11. In video (d), the movement of people walking slightly distorts the overall scene and appears to slightly shake, stretch, and

Table 5.10 **Static** Results from *Static* scenes. Lower is better in all categories.

Model	DSSIM $\times 10^{-3}$	DSSIM w. stabilise $\times 10^{-3}$	MAE $\times 10^{-2}$	MAE w. stabilise $\times 10^{-2}$	RMSE $\times 10^{-2}$	RMSE w. stabilise $\times 10^{-2}$
AdelaiDepth101	2.74	0.71 (-74.1%)	3.03	1.80 (-40.7%)	3.87	2.61 (-32.7%)
AdelaiDepth50	0.97	1.20 (+24.3%)	2.23	1.89 (-15.1%)	3.17	2.79 (-11.9%)
DPT Large	**0.26**	0.22 (-15.4%)	1.43	1.40 (-2.2%)	4.55	4.57 (+0.5%)
DPT Hybrid	0.30	**0.22** (-26.7%)	1.58	1.46 (-7.5%)	3.64	3.77 (+3.5%)
MiDaS Large	0.59	0.46 (-22.0%)	1.95	1.79 (-8.2%)	3.55	3.45 (-2.7%)
MiDas Small	2.54	1.14 (-55.1%)	3.79	3.20 (-15.6%)	6.28	6.18 (-1.7%)
Monodepth	0.40	0.38 (-5.0%)	**1.06**	**1.08** (+1.8%)	**2.43**	**2.29** (-5.8%)

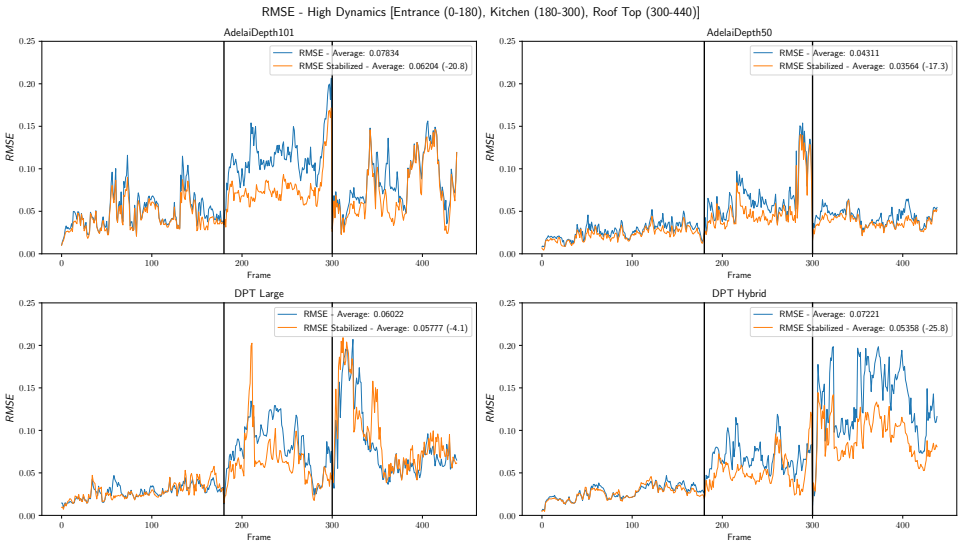


Figure 5.14 **RMSE — High Dynamics** RMSE timeseries of AdelaiDepth and DPT models in the High Dynamics dataset. Top left: AdelaiDepth101. Top right: AdelaiDepth50. Bottom left: DPT Large. Bottom right: DPT Hybrid. The vertical bar represents change of scene. Frame 0 to 180 is the Entrance scene. Frame 180 to 300 is the Kitchen scene. Frame 300 to 440 is the Roof Top scene.

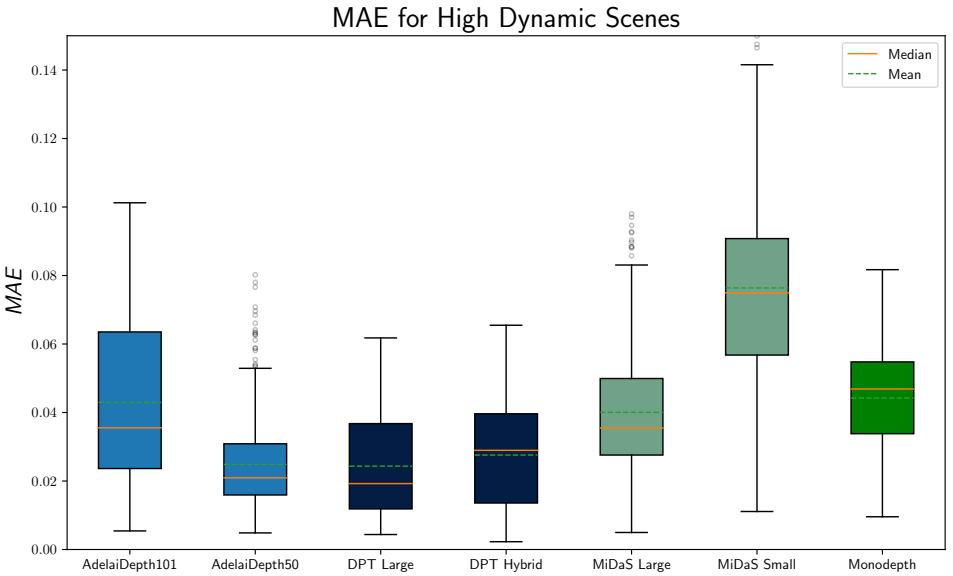
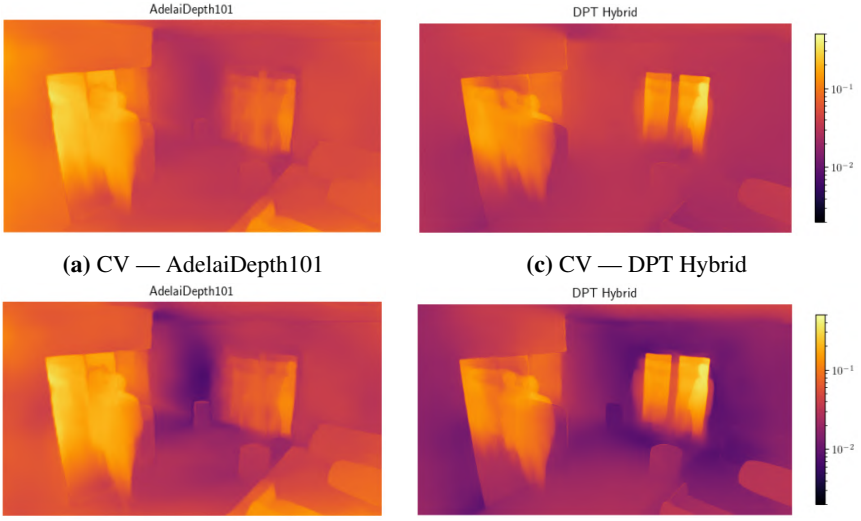


Figure 5.15 MAE — High Dynamics MAE for High Dynamics visualised at box plots. While mean MAE is the main stability measurement, the locality, spread and outliers are also important aspects.

shift back and forth in various regions. This is further illustrated in Figure 5.16, demonstrating that instability levels varies over different regions of the scene.

The stability issue is visually very prominent in MiDaS Small, see video (b). This is also the model that overall performs worst on the stability metrics. Compared to DPT Large in video (a), which generally has very high results across all stability metrics, the difference in very notable. As stated previously, surprisingly AdelaiDepth50 with the shallower ResNet50 encoder clearly outperforms the deeper AdelaiDepth101 on all scenes and on all metrics. The difference, while subtle, is visualised in video (c). The difference between AdelaiDepth50 and DPT Hybrid on High Dynamic scenes is visualised in videos (e) and (f). The overall stability is relatively similar for both the Roof Top and the Kitchen scenes. However, a difference is how they both handle the transparent windows. DPT predicts, though unstably, more accurate window depths, while AdelaiDepth predicts a more constant depth, even though it is less accurate. Inspecting both DPT Large and Monodepth on the static Roof Top Scene manually, shows that while DPT has arguably more accurate room projection than Monodepth, it picks up the movement of small cars outside the large window. This phenomenon is illustrated in video (g).

The stabilisation technique proposed, *Static Area Alignment*, shows in general sig-



(a) CV — AdelaiDepth101 **(c)** CV — DPT Hybrid

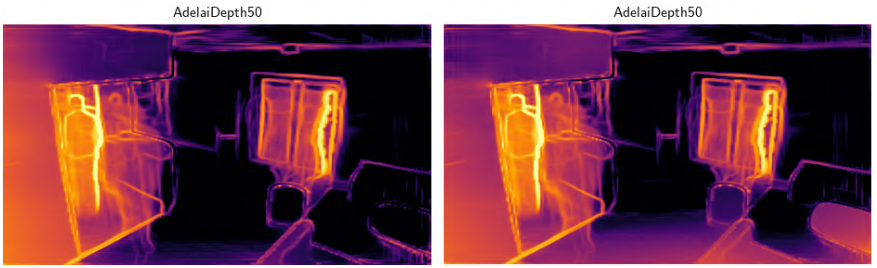
(b) CV Stabilised — AdelaiDepth101 **(d)** CV Stabilised — DPT Hybrid

Figure 5.16 Coefficient of Variant (CV) — High Dynamics. Coefficient of Variant ($CV_i = \frac{\sigma_i}{\mu_i}$) for Kitchen scene on AdelaiDepth101 and DPT Hybrid in logarithmic scale. Darker regions corresponds to high stability, while brighter regions corresponds to high instability.

nificant improvement at all levels of dynamics and all three dynamics. For high dynamic scenes the DPT models have the most notable improvements. This improvement is illustrated in Figure 5.16 as well as visualised in video (h). For low and static scenes the results varies, but generally results in increased stability. AdelaiDepth50 and AdelaiDepth101 show a slight increase in DSSIM, i.e., a decrease in stability, for high and low dynamics. This could be attributed to the fact that DSSIM is a more complex measurement, i.e., perceived change in structural information. Finding a global alignment, by globally minimising according to least-squares principles, does not necessarily lead to better overall such measurement. However, it seems that in general, DSSIM is often also improved when MAE and RMSE are improved. This can be visualised in video (i), and Figure 5.17, and showcase the difficulty at finding a global alignment that results in a more structurally stable scene overall. The MiDaS models have consistent gain in stability on all metrics and all levels of dynamics. The difference, while subtle, is visualised in video (j).

Table 5.11 Videos — Stabilty Evaluation. Video visualising generated point clouds as animation. The models are mentioned in left to right order, when viewing side-by-side video. For footage containing stabilised point cloud, the raw unstabilised point cloud is marked in red.

Scene	Video Description
Kitchen	(a) DPT Large - High and Low Dynamics
Kitchen	(b) Midas Small - High and Low Dynamics
Kitchen	(c) AdelaiDepth101 and AdelaiDepth50 — High Dynamics
Entrance	(d) AdelaiDepth101 — High Dynamics
Kitchen	(e) DPT Hybrid and AdelaiDepth50 — High Dynamics
Roof Top	(f) DPT Hybrid and AdelaiDepth50 — High Dynamics
Roof Top	(g) DPT Large and Monodepth — Static
Kitchen	(h) DPT Large w. stabilise — High Dynamics
Kitchen	(i) AdelaiDepth50 w. stabilise — High Dynamics
Roof Top	(j) MiDaS Large w. stabilise — High Dynamics



(a) DSSIM **(b)** DSSIM Stabilised

Figure 5.17 DSSIM AdelaiDepth50 — Stabilise Failure. DSSIM for Kitchen High Dynamics scene on AdelaiDepth50 in logarithmic scale. Notice that DSSIM increases in the bottom region for the stabilised variant. Also notice DSSIM is invariant to changes in the far wall, which generally is more uniformly varied, i.e., retaining the structural information.

5.4 Depth Resolution

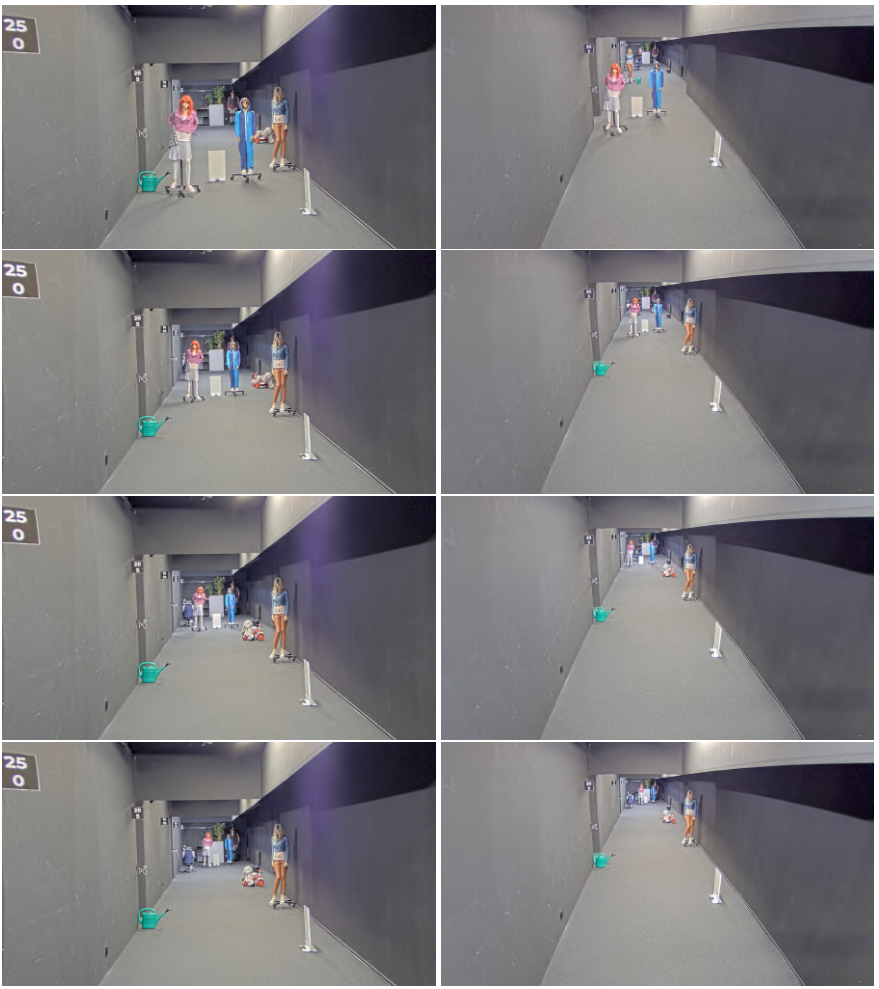
As was concluded in Section 4.5, an important aspect of a MDE model is its depth range, i.e., the range at which different object depths are clearly discernible. This section will therefore compare the seven models that have been evaluated so far in terms of depth-range using a simple experiment setup. This depth range evaluation consist of one scene, captured from two different points of view (POV). The first POV is with the camera mounted with a straight perspective to the scene at hand. The second scene is with a camera mounted from with a tilted perspective, more similar to that of a typical surveillance camera installation. The scene is a corridor, with mannequins placed at known distances. Importantly, one of the mannequins is placed as front object, and the other is placed 1 m behind the first. The objective is then to capture one frame with the mannequins positioned at 5, 10, 15 and 20 m from the camera, and in the subsequent analysis part, determine at what ranges each model is able to distinguish this difference in depth. The setup is presented in Figure 5.18. What is also worth mentioning is that the setup with straight POV is also captured with optical zoom (increased focal length), while the tilted POV setup was captured with lower focal length. However, the cameras were placed so that they both could visualise everything behind the zero meter mark.

For each of the frames, in each camera setup, the front mannequin (pink shirt), and the mannequin behind (dressed in blue), are manually annotated. The average of these pixels' depth values marks the object depth. The relative depth difference between these objects is then calculated as

$$\Delta_{rel} = \frac{d_A - d_B}{d_A} \tag{5.5}$$

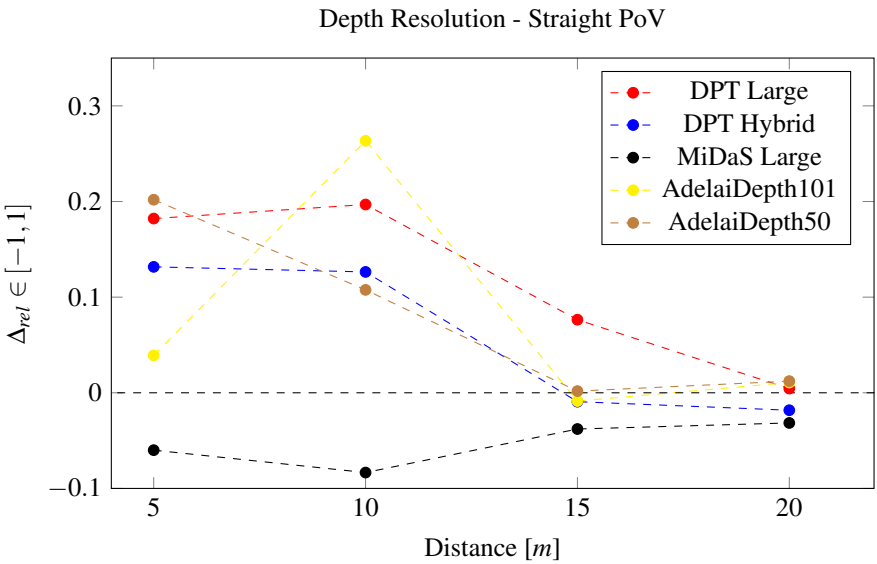
where Δ_{rel} is the relative depth difference, d_A marks the depth of the frontmost mannequin, and d_B marks the depth of the mannequin behind. That is, a Δ_{rel} value of zero marks that a model is unable to distinguish the depth difference between the objects. The result is presented as a plot, marking the Δ_{rel} for each model, at each range. The result for the straight POV are given in Figure 5.19. Figure 5.19a shows this result for the larger models. Firstly, it is noted that MiDaS Large fails completely on this scene. Secondly, all models are able to distinguish depth up to 10 m, with AdelaiDepth101 having some issues already at 5 m. No model except DPT Large is able to distinguish the objects in depth at 15 m. For the smaller models, in Figure 5.19b, MiDaS Small is the only one that succeeds, and that up to 10 m.

In Figure 5.20, the result obtained for the setup with the camera mounted at a tilted POV are shown. Figure 5.20b shows the result for the small models. Here, it is only at 5 m the models are able to perceive the depth difference. Figure 5.20a shows the result for the larger models. As is noted from this plot, almost all models remarks

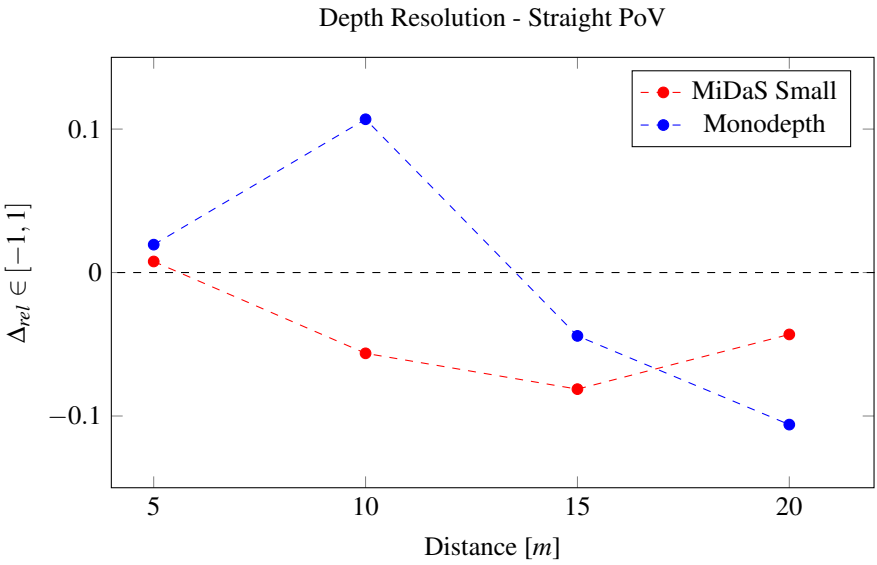


(a) Straight POV **(b)** Tilted POV

Figure 5.18 Depth Range Scene



(a) Large Models



(b) Small Models

Figure 5.19 Depth Resolution Evaluation: Straight POV.

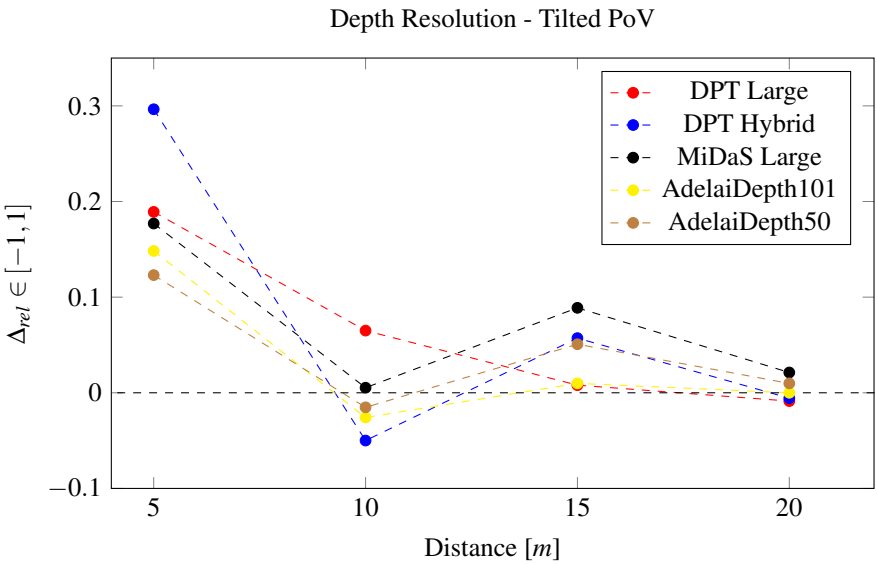
a peak relative distance at 15 m. This is somewhat baffling. In Figure 5.21, the relative difference in depth is plotted for the front most object and the annotated background. Looking closer at the model's values at 15 m, it is noted that they all are very close to zero. In other words, they have almost faded together with the background. Because of this, the Δ_{rel} yields positive result at 15 m, as is seen in Figure 5.20a, simply because both objects are part of the background, and the behind mannequin is blended together with the background, which happens to be slightly ahead of the background behind the front-most mannequin. Conclusively, it is only DPT Large that is able to distinguish depth at 10 , and all models fail on doing so on 15 m and longer.

5.5 Discussion

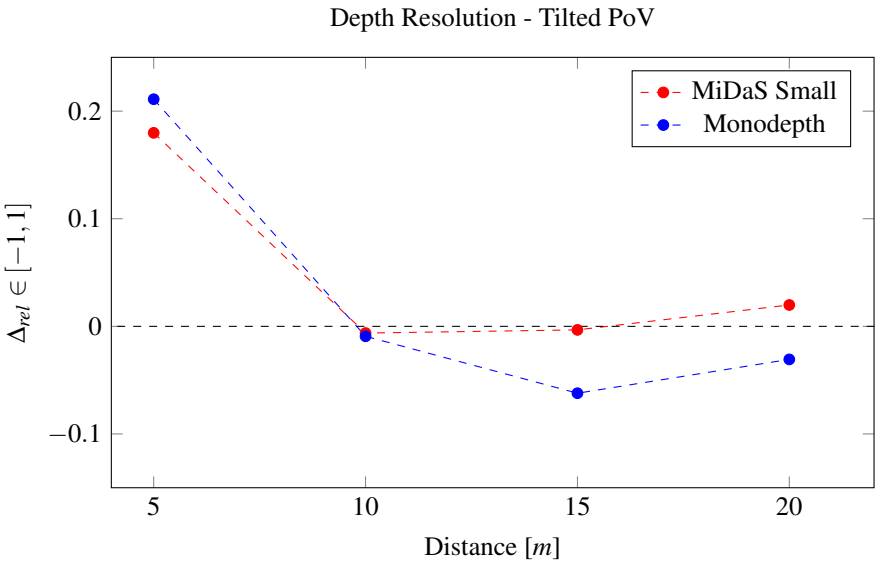
As discussed in Section 3.5, the models that have been investigated and evaluated here are in many senses similar. All models apply a U-net architecture, in the process of acquiring a dense prediction. They include similar loss functions, regarding accuracy loss, and contour refinement loss. Every model, with the exception of Monodepth, is fully supervised and trained on a mixture of data sets, for improved generalisation performance. There are minor differences that separate these models from each other. AdelaiDepth toolbox models, including AdelaiDepth50 and AdelaiDepth101, are intending to achieve consistent geometries, for accurate geometric reconstruction, in addition of acquiring accurate depth estimations. DPT and MiDaS, developed by Intel Labs, have instead focused fully on accuracy and on improved generalisation by training on ten data sets, compared to five data sets for the AdelaiDepth models. This is reflected in the results as the models perform similarly but is still distinguishable when it comes to qualitative assessment, such as accuracy and stability.

Accuracy

Comparing the accuracy results, from the outdoor and the indoor data sets, it is concluded that the DPT and/or MiDaS models come out the strongest, relative to AdelaiDepth, even though the difference is small in some cases. What is worth mentioning though is that the difference between the Intel models versus AdelaiDepth models are bigger in terms of absolute relative error (AbsRel). In other words, when allowed for some depth accuracy errors, i.e., allowing predictions being not more than 1.25 off from the ground truth (delta-error), the performance difference is not as compelling between the DPT/MiDaS and the AdelaiDepth models, see Table 5.5 and Table 5.6 for reference. Additionally, it is concluded that the accuracy performance of the smaller models are significantly worse than that of the larger models. Furthermore, MiDaS Small uniformly outperforms Monodepth in terms of accuracy. MiDaS generalisation performance appears distinctly advantageous relative to Monodepth, being trained solely on data targeting autonomous driving.



(a) Large Models



(b) Small Models

Figure 5.20 Depth Resolution Evaluation: Tilted POV.

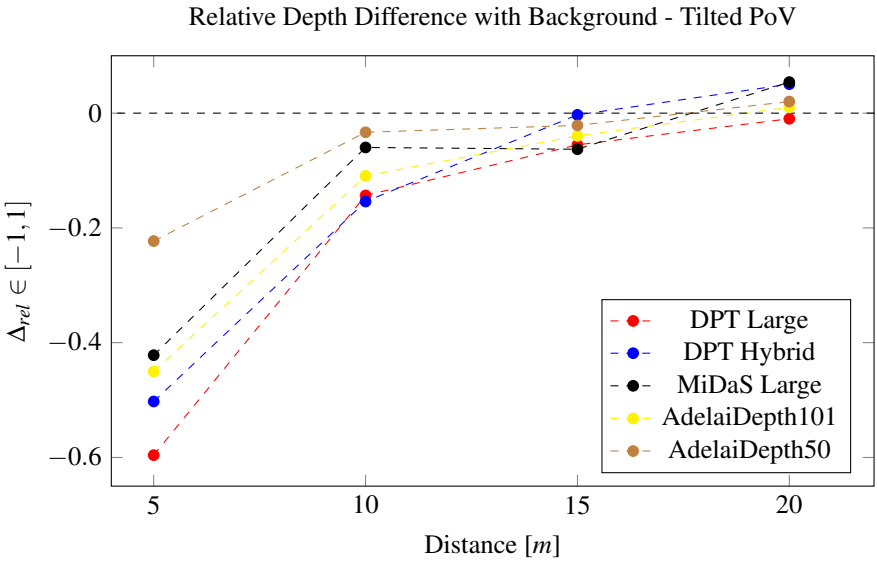


Figure 5.21 Relative Depth Difference with Background: Tilted POV.

Moreover, it can be concluded that the performance of these models is highly scene dependent. Additionally, connecting to the previous conclusion that the models are in many senses similar, it appears that all models predict depth more easily on the same type of scenes. To support this statement, we refer back to Table 5.7. This table showed, as already stated, that the models achieve their best and worst results respectively on the same scenes. That is, every model achieves its best, second best, and so on, result for the same scene. By also looking closer at the sample from the ”Loading Area”-scene, containing the *optical illusion* of depth in Figure 5.10, it proves as a good example of this scene dependency, affecting the performance of the depth predictions. With this in mind, we find it as a compelling argument for fine tuning on surveillance data. In fact, we suggest the possibility of utilising scene specific fine tuning, due to the fixed installation nature of surveillance cameras, i.e., a model can be fine tuned on, e.g., indoor boutique data, for increased accuracy and performance on these specific categories of scenes.

Stability

All models experience increased temporal instability when scene dynamics are introduced. The level of instability can also be concluded to be highly scene dependent, referring to Figure 5.14. Furthermore, it seems linked to the visibility and occlusion of *depth cues* in the frame. From our recorded material, important depth cues seem to be geometrical shapes such as corners, wall posts, windows, doorframes, etc. If viewing the problem of performing 3D reconstruction of a scene as a

function with the input being numerous such depth cues, it is not unreasonable that the predicted depth reconstruction slightly varies as a person is walking through the scene and occluding various depth cues. This effect was especially prominent for MiDaS Small in the kitchen scene, as seen in video (b) in Table 5.11.

Furthermore, it was noticed that even in static scenes, tiny movements such as almost indiscernible outside traffic through a window, could introduce small instabilities, as seen in video (g) in Table 5.11. This temporal stability is an important aspect when developing real-time downstream applications, but is seldom evaluated in the papers that accompany these models. Instead, the main focus lies only on reaching high performance on known accuracy benchmark data sets. We think stability evaluation, targeting the issue of temporal inconsistency, would benefit the research and development in the field of Monocular Depth Estimation, and that our proposed evaluation method and stabilisation technique is called-for. Moreover, we can conclude from Figure 5.16 that this instability is not uniform across the image, indicating that a single ”global” scale and shift alignment can be improved upon. Instead, one idea that was out-of-scope for this thesis was to perform alignment in local patches (eg. split into 8x8 sections) for each frame; however, this would decrease real-time performance and would not necessarily preserve accuracy. In fact, during the writing of this thesis, the authors of AdelaiDepth released a paper [Xu et al., 2022] aiming to address the issue of temporal inconsistency.

They identified the same overall issue regarding depth inconsistency and proposed a scale and shift alignment module with a locally weighted linear regression method, with scale and shift recovered by sparse ground truth anchor points. This is conceptually similar to our proposed stabilisation method, although arguably more sophisticated. Their proposed technique even resulted in significant gain in accuracy performance, further strengthening our conclusion that the field of Monocular Depth Estimation would benefit with more consideration to temporal consistency and output stability.

To conclude, our proposed stabilisation technique was successful on most models and data sets by consistently reducing the chosen stability metrics, while not having an impact *AbsRel* and δ_{err} accuracy metrics. It is also worth mentioning that our proposed stabilisation technique has promising practicability, as it only requires a fast foreground–background model. Also, the alignment coefficients can be solved for fast and efficiently, since it utilises the closed-form solution of a linear least-squares approximation. In our work, we use all available depth values as input to the optimisation problem, but we believe, for example, a sparse sampling of depth values could be used to enhance performance with similar results.

Depth Range

That the resolution of distinguishing depth becomes harder at larger ranges of depth has already been concluded. It is as well an important aspects when utilising MDE in computer vision applications. According to our evaluation in Section 5.4, DPT Large is showing the best performance, as it is the sole model that is able to distinguish depth differences at 15 m and 10 m for the straight POV scene and tilted POV scene, respectively. It is also concluded that the smaller models perform worse than the larger models in this aspect. As distinguishing depth at long ranges is strongly connected to the input pixel resolution, this comes as no surprise. This could also be what is the underlying factor to why DPT Large outperforms the other models, as it has the largest input image size, in addition to proving for great depth accuracy results. However, as have been stated, the performance of these models are scene dependent, and the depth range evaluation has so far only been evaluated on one single scene. Therefore, these conclusions are not certainly applicable in other contexts. Referencing the depth range plots in Figures 5.19 and 5.20, it appears as that the depth range of the models is more constrained for the scene captured from a tilted POV. We see two possible answers to this observation. Firstly, all models are trained on data sets that are mostly captured from a more straight-on perspective, and not as a wall-mounted security camera, with the typical tilted perspective. Secondly, as the straight POV scene was captured with an optical zoom applied, this could also be a reason for the different performance.

Summary

Bringing the evaluation results together, we can establish that the DPT models and/or MiDaS are all-in-all the best performing models on our custom data sets and evaluation metrics. This is in terms of both accuracy, stability and depth range aspects, even though the performance difference is modest, and also not consistent over multiple measurements and metrics in many cases. The favouring of MiDaS/DPT models on our custom data sets could be derived to these models being implemented with isolated focus on depth accuracy and generalisation performance, compared to AdelaiDepth models that additionally acquire accurate scene geometry that has not been evaluated on in this thesis. The average accuracy results of all models on our custom data sets compared to the existing data sets in Section 3.5, are generally higher. Here, the indoor data set captured with a stereo camera is compared to the NYUv2 data set. The outdoor LiDAR data set is compared to the longer range iBims-1 and ETH3D data sets. This, along with the hypothesis of decrease of depth range for a tilted point-of-view, could be an indication that the models have not been trained on data targeting a surveillance domain. Of course, there might be other underlying explanations to these results as well, but we believe it to be a fair assumption that fine tuning these models on typical surveillance scenes should increase the performance result on the custom data sets and metrics provided in this thesis. Additionally, we stress the importance of temporal stability. Our naive pro-

posed solution indicates that this can be mitigated, and our stability evaluation can evince for such improvements. Lastly, it should be put forward that the larger models that achieve the highest results here are not as of now ready to be launched on a *constrained* embedded system. Especially, the DPT-models show high inference time on desktop GPUs without tensor accelerating hardware.

6

Conclusion and Future Work

6.1 Conclusion

Referencing the predefined goals defined in Section 1.2, they can be listed as: develop a robust proof-of-concept privacy mask even with sub-perfect depth data running close to real time, determine the most vital functionality aspects for MDE used in depth aware privacy masking, and perform benchmarking and analysis on metrics related to these aspects in order to conclude on alternative solutions to improve on such metrics. It can be concluded that these set goals have been met to a predominant degree. Firstly, we have presented a complete end-to-end depth aware privacy masking algorithm, to be referenced to as a proof-of-concept. This algorithm is designed for robustness and ”hide-rather-than-show”-criteria, which is achieved by the stabilisation algorithm. This is principally fulfilled, especially when assuming a more or less accurate and temporally stable depth input. For situations where the MDEs do fall short, it has been determined to be sourced back to one or more of the following aspects: depth accuracy, depth range resolution, and temporal stability. Therefore, we have presented self-captured evaluation data and metrics to quantitatively evaluate performance relating to these aspects, targeting a surveillance domain. This includes two custom data sets with pixel-aligned ground truth depth data. We have also proposed a minor post-processing method to mitigate the effect of temporal instability, without sacrificing accuracy, with promising results.

To summarise, with this thesis we hope to have given a thorough introduction to what is referred to as monocular depth estimation utilising deep-learning. Our hope is that this is a step forward towards utilising the potential of monocular depth estimation in downstream computer vision applications, such as depth aware privacy masking. We suggest doing so, by referencing the proposed future work outlined in Section 6.2.

6.2 Future Work

To utilise the depth estimators considered in this thesis, it is firstly concluded that some sort of fine-tuning or transfer learning is desirable in order to increase the performance on typical surveillance data. Additionally, to account for the temporal stability issues prominent in video usage, there are research from two papers we see fit to reference. Firstly, as mentioned in Section 1.5, the authors of [Zhang et al., 2019] implement a convolutional long-short-term-memory (CLSTM) framework in addition to the depth estimation network. With this, they include an additional temporal loss in order to strive for consistent depth with previous frames. Secondly, as mentioned in *Related Work*, Section 5.5, AdelaiDepth have published a new paper during the time of writing this thesis [Xu et al., 2022]. This paper investigates applying a pixel-wise scale and shift operation of the output depth map based on a sampled set of ground truth pixels and Gaussian smoothing. Conclusively, any model chosen for utilising MDE in a downstream video application would benefit being fine tuned on appropriate ground truth data, and account for temporal instability by introducing a CLSTM appendage and/or follow up on the scale- and shift alignment post processing technique.

Considering that the proposed masking algorithm is a proof-of-concept, it should not by any means be considered as a ready-to-use application. Firstly, it needs to be tested on more scenes, and preferably realistic ”in-the-wild” captured scenes. This is to ensure that it fulfils the set requirements for an arbitrary surveillance scene. Secondly, it should be refactored and rewritten to the purpose of embedded runtime. In addition, using a fine-tuned MDE model on scene-specific data, that is also accounted for temporally unstable depth data should ultimately increase the general robustness. Lastly, as pointed on in Section 4.4 in connection with Figure 4.15, there are cases when the depth buffer fails to establish a continuous depth mask. In order to mitigate this disruption in the mask wall seen in Figure 4.15, this can be mitigated as illustrated in Figure 4.16, by defining the mask as a convex hull. As also suggested, we believe it is feasible to instead implementing a linear interpolation/extrapolation of depth values over the missing frame depth in the baseline.

Bibliography

Alhashim, I. and P. Wonka (2018). "High quality monocular depth estimation via transfer learning". *CoRR* **abs/1812.11941**. arXiv: 1812.11941. URL: http://arxiv.org/abs/1812.11941.

Alom, M. Z., T. Taha, C. Yakopcic, S. Westberg, M. Hasan, B. Esesn, A. Awwal, and V. Asari (2018). "The history began from AlexNet: a comprehensive survey on deep learning approaches". *CoRR* **abs/1803.01164**.

Axis Communications (2021). *Privacy in video surveillance*. URL: https://www.axis.com/solutions-by-application/privacy-in-video-surveillance (visited on 2022-04-14).

Balcilar, M. (2022). *Densedepthmap*. URL: https://github.com/balcilar/DenseDepthMap (visited on 2022-05-19).

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". *IEEE Transactions on Neural Networks* **5**:2, pp. 157–166. DOI: 10.1109/72.279181.

Berlic, J. and J. Strandevall (2012). "Selective viewing of a scene". Patent 8311275. URL: https://patents.google.com/patent/US8311275.

Biasutti, P., A. Bugeau, J. Aujol, and M. Brédif (2019). "Riu-net: embarrassingly simple semantic segmentation of 3D [lidar] point cloud". *CoRR* **abs/1905.08748**. arXiv: 1905.08748. URL: http://arxiv.org/abs/1905.08748.

Boyd, S. and L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press. DOI: 10.1017/CBO9780511804441.

Bradski, G. (2000). "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools*. Vol. 25, pp. 120–123.

Casser, V., S. Pirk, R. Mahjourian, and A. Angelova (2019). "Depth prediction without the sensors: leveraging structure for unsupervised learning from monocular videos". In: *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*. Vol. 33. 01, pp. 8001–8008. DOI: 10.1609/aaai.v33i01.33018001.

Cepton (2022). *Cepton vista-p*. URL: https://www.cepton.com/products/vista-p (visited on 2022-05-16).

Conde Moreno, L. (2019). *Automated Privacy-Preserving Video Processing through Anonymized 3D Scene Reconstruction*. MA thesis. Utrecht University.

Creative Commons Attribution 4.0 International (n.d.). Version 4. Creative Commons. URL: https://creativecommons.org/licenses/by/4.0/.

Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby (2021). "An image is worth 16x16 words: transformers for image recognition at scale". In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=YicbFdNTTy.

Eigen, D., C. Puhrsch, and R. Fergus (2014). "Depth map prediction from a single image using a multi-scale deep network." In: Ghahramani, Z. et al. (Eds.). *NIPS*, pp. 2366–2374. URL: http://dblp.uni-trier.de/db/conf/nips/nips2014.html#EigenPF14.

Forsyth, D. A. and J. Ponce (2012). *Computer Vision: A Modern Approach*. 2nd ed. Pearson Education.

Geiger, A., P. Lenz, C. Stiller, and R. Urtasun (2013). "Vision meets robotics: the KITTI dataset". *The International Journal of Robotics Research* **32**:11, pp. 1231–1237. DOI: 10.1177/0278364913491297.

Glorot, X. and Y. Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. Proceedings of Machine Learning Research. PMLR, pp. 249–256.

Godard, C., O. M. Aodha, and G. J. Brostow (2017). "Unsupervised monocular depth estimation with left-right consistency". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6602–6611. DOI: 10.1109/CVPR.2017.699.

Godard, C., O. M. Aodha, M. Firman, and G. J. Brostow (2019). "Digging into self-supervised monocular depth estimation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3827–3837. DOI: 10.1109/ICCV.2019.00393.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep residual learning for image recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Izadinia, H., Q. Shan, and S. M. Seitz (2017). "Im2cad". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2422–2431. DOI: 10.1109/CVPR.2017.260.

Katz, S., A. Tal, and R. Basri (2007). "Direct visibility of point sets". In: *ACM Transactions on Graphics*. Vol. 26. DOI: 10.1145/1275808.1276407.

Kingma, D. P. and J. Ba (2015). "Adam: A method for stochastic optimization". In: *3rd International Conference on Learning Representations (ICLR 2015)*.

Koch, T., L. Liebel, F. Fraundorfer, and M. Körner (2018). "Evaluation of CNN-based single-image depth estimation methods". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

Koch, T., L. Liebel, M. Körner, and F. Fraundorfer (2020). "Comparison of monocular depth estimation methods using geometrically relevant metrics on the ibims-1 dataset". *Computer Vision and Image Understanding* **191**, p. 102877. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2019.102877.

Lee, J. H., M. Han, D. W. Ko, and I. H. Suh (2019). "From big to small: multi-scale local planar guidance for monocular depth estimation". *CoRR* **abs/1907.10326**. arXiv: 1907.10326. URL: http://arxiv.org/abs/1907.10326.

Levin, A., D. Lischinski, and Y. Weiss (2004). "Colorization using optimization". In: *ACM Transactions on Graphics*. Vol. 23. DOI: 10.1145/1015706.1015780.

Li, Z. and N. Snavely (2018). "Megadepth: learning single-view depth prediction from internet photos". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2041–2050. DOI: 10.1109/CVPR.2018.00218.

Liu, C., J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa (2018). "Planenet: piecewise planar reconstruction from a single RGB image". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2579–2588. DOI: 10.1109/CVPR.2018.00273.

Liu, F., C. Shen, G. Lin, and I. Reid (2016). "Learning depth from single monocular images using deep convolutional neural fields". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **38**:10, pp. 2024–2039. DOI: 10.1109/TPAMI.2015.2505283.

Mancini, M., G. Costante, P. Valigi, and T. A. Ciarfuglia (2018). "J-mod^2: joint monocular obstacle detection and depth estimation". *IEEE Robotics and Automation Letters* **3**:3, pp. 1490–1497. DOI: 10.1109/LRA.2018.2800083.

Nathan Silberman Derek Hoiem, P. K. and R. Fergus (2012). "Indoor segmentation and support inference from RGBD images". In: *12th European Conference on Computer Vision (ECCV 2012)*. PART 5, pp. 746–760. DOI: 10.1007/978-3-642-33715-4_54.

Nayar, S. and Y. Nakagawa (1994). "Shape from focus". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 16. 8, pp. 824–831. DOI: 10.1109/34.308479.

OpenCV (2022). *Camera calibration*. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (visited on 2022-05-16).

Phan, R. and D. Androutsos (2014). "Robust semi-automatic depth map generation in unconstrained images and video sequences for 2D to stereoscopic 3D conversion". *IEEE Transactions on Multimedia* **16**:1, pp. 122–136. DOI: 10.1109/TMM.2013.2283451.

Ranftl, R., K. Lasinger, D. Hafner, and V. Koltun (2020). "Towards robust monocular depth estimation: mixing datasets for zero-shot cross-dataset transfer". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PP**, pp. 1623–1637. DOI: 10.1109/TPAMI.2020.3019967.

Ranftl, R., A. Bochkovskiy, and V. Koltun (2021). "Vision transformers for dense prediction". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12159–12168. DOI: 10.1109/ICCV48922.2021.01196.

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-net: convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)*. Vol. 115. 3, pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

Schöps, T., J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger (2017). "A multi-view stereo benchmark with high-resolution images and multi-camera videos". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2538–2547. DOI: 10.1109/CVPR.2017.272.

Sener, O. and V. Koltun (2018). "Multi-task learning as multi-objective optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31. Curran Associates, Inc.

Singh, D. and B. Singh (2020). "Investigating the impact of data normalization on classification performance". In: *Applied Soft Computing*. Vol. 97, p. 105524. DOI: https://doi.org/10.1016/j.asoc.2019.105524.

Stereolabs (2022). *Zed stereo camera*. visited on 2022-04-28. URL: https://www.stereolabs.com/zed/.

Szeliski, R. (2011). *Computer vision algorithms and applications*. URL: http://dx.doi.org/10.1007/978-1-84882-935-0.

Taghanaki, S. A., K. Abhishek, J. P. Cohen, J. Cohen-Adad, and G. Hamarneh (2020). "Deep semantic segmentation of natural and medical images: A review". In: *Artificial Intelligence Review*. Vol. 54, pp. 137–178.

Takeyuki Goto, Y., E. Takashi Maruyama, and C. Makoto Kikuchi (2009). "Camera and image processor". Patent 0128632. URL: https://www.google.com/patents/US174465.

Tan, M. and Q. V. Le (2019). "Efficientnet: rethinking model scaling for convolutional neural networks". *CoRR* **1905.11946**. arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

Vasiljevic, I., N. Kolkin, S. Zhang, R. Luo, H. Wang, F. Dai, A. Daniele, M. Mostajabi, S. Basart, M. Walter, and G. Shakhnarovich (2019). "DIODE: A Dense Indoor and Outdoor DEpth Dataset". *CoRR* **abs/1908.00463**. URL: http://arxiv.org/abs/1908.00463.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). "Attention is all you need". In: Guyon, I. et al. (Eds.). *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, Z., A. Bovik, H. Sheikh, and E. Simoncelli (2004). "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing*. Vol. 13. 4, pp. 600–612. DOI: 10.1109/TIP.2003.819861.

Xian, K., C. Shen, Z. Cao, H. Lu, Y. Xiao, R. Li, and Z. Luo (2018). "Monocular relative depth perception with web stereo data supervision". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 311–320. DOI: 10.1109/CVPR.2018.00040.

Xian, K., J. Zhang, O. Wang, L. Mai, Z. Lin, and Z. Cao (2020). "Structure-guided ranking loss for single image depth prediction". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 608–617. DOI: 10.1109/CVPR42600.2020.00069.

Xie, J., R. B. Girshick, and A. Farhadi (2016). "Deep3D: fully automatic 2D-to-3D video conversion with deep convolutional neural networks". *CoRR* **abs/1604.03650**.

Xie, S., R. Girshick, P. Dollár, Z. Tu, and K. He (2017). "Aggregated residual transformations for deep neural networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995. DOI: 10.1109/CVPR.2017.634.

Xu, G., W. Yin, H. Chen, K. Cheng, F. Zhao, and C. Shen (2022). "Towards 3D scene reconstruction from locally scale-aligned monocular video depth". *arXiv*. DOI: 10.48550/ARXIV.2202.01470. URL: https://arxiv.org/abs/2202.01470.

Yin, W., Y. Liu, C. Shen, and Y. Yan (2019). "Enforcing geometric constraints of virtual normal for depth prediction". In: *The IEEE International Conference on Computer Vision (ICCV)*.

Yin, W., X. Wang, C. Shen, Y. Liu, Z. Tian, S. Xu, C. Sun, and D. Renyin (2020). "Diversedepth: affine-invariant depth prediction using diverse data". *arXiv* **abs/2002.00569**.

Yin, W., J. Zhang, O. Wang, S. Niklaus, L. Mai, S. Chen, and C. Shen (2021). "Learning to recover 3D scene shape from a single image". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (CVPR)*.

Zhang, H., Y. Li, Y. Cao, Y. Liu, C. Shen, and Y. Yan (2019). "Exploiting temporal consistency for real-time video depth estimation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1725–1734. DOI: 10.1109/ICCV.2019.00181.

Zhang, Z., F. Cole, R. Tucker, W. Freeman, and T. Dekel (2021). “Consistent depth of moving objects in video”. In: *ACM Transactions on Graphics*. Vol. 40, pp. 1–12. DOI: 10.1145/3476576.3476728.

Zhao, C., Q. Sun, C. Zhang, Y. Tang, and F. Qian (2020). “Monocular depth estimation based on deep learning: an overview”. In: *Science China Technological Sciences*. Vol. 63. DOI: 10.1007/s11431-020-1582-8.

Zhou, Q.-Y., J. Park, and V. Koltun (2018). “Open3D: A modern library for 3D data processing”. *arXiv* **abs/1801.09847**.

Zivkovic, Z. (2004). “Improved adaptive Gaussian mixture model for background subtraction”. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 2, 28–31 Vol.2. DOI: 10.1109/ICPR.2004.1333992.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2022

Document Number
TFRT-6168

Author(s)
Mattias Lundström
Jakob Pettersson

Supervisor
Björn Olofsson, Dept. of Automatic Control, Lund University, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Karl-Erik Årzen, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
3D Privacy Masking using Monocular Depth Estimation

Abstract

This thesis strives to dive deeper within the area of Monocular Depth Estimation, approximating distance information from one single image using deep neural networks. It introduces a thorough evaluation and analysis of state-of-the-art depth estimation models regarding proposed aspects of relevance for downstream video applications, specifically in a surveillance domain. This leads to three custom data sets where two include ground truth depth data. Results on accuracy, temporal inconsistency, and range resolution is presented and analysed utilising the collected data sets, with selected metrics. It is concluded that the accuracy performance of the models, even though impressive, is also highly scene dependant. Regarding temporal inconsistency, which causes apparent video instability, it is concluded to be a prominent concern for typical downstream video applications that calls for further attention. This leads to a proposed minor post-processing step, with promising results.
Furthermore, this thesis also presents a novel end-to-end algorithm referred to as ”3D Privacy Masking”. Privacy masking is a typical task in camera surveillance, where a certain region of the image scene needs to be anonymised. This functionality is here extended by including depth, such as that from monocular depth estimation, resulting in a depth aware privacy mask. Thereby, events in front of the mask as seen by the camera can still be observable. The suggested algorithm and proof-ofconcept application also includes a stabilising technique to account for sub-perfect depth data. Conclusively, this thesis showcases the potential of monocular depth estimation in downstream computer vision tasks, like that of 3D privacy masking, and proposes continued directions forward.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-101

Recipient's notes

Security classification