Self-Supervised Monocular Depth Estimation

of Untextured Indoor Rotated Scenes

Benjamin Keltjens

Delft University of Technology

Self-Supervised Monocular Depth Estimation

of Untextured Indoor Rotated Scenes

by

Benjamin Keltjens

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on July 7, 2021 at 9:30.

Student number:	4551311	
Project duration:	September 14, 2020 – July 7, 2021	
Thesis committee:	Dr. G. C. H. E. de Croon,	TU Delft, chair, supervisor
	Ir. T. van Dijk,	TU Delft, supervisor
	Dr. J. C. van Gemert	TU Delft, external examiner
	Dr. ir. E. J. J. Smeur,	TU Delft

TU Delft

Preface

The application of robotics in modern-day environments are becoming more widespread and complex. Enabling them to better interpret their surrounding world allows them to become more helpful and safe. This work extends recent machine learning advances for estimating monocular depth in a self-supervised manner to more complex domains. Specifically, this research enables the use of self-supervised learning for monocular depth estimation in untextured indoor environments under rotation. The code for this work is made available to all for application and improvements [1].

This paper marks the end of a long journey in Delft, full of good memories and important people in my life. Even though there have been difficult moments, especially considering a year like this one, the people around me have helped me persevere. First, I would like to express my gratitude to my supervisors Tom van Dijk and Dr. Guido de Croon for their continual support and understanding in a year like this. Their contributions and feedback have been essential to navigate and shape this thesis. I am thankful for the strong support and care I have received from my friends, old and new, whose presence in my life has been invaluable. I feel lucky to have such special friends and family in my life. Finally, thank you to my parents and my brother in exploring the world with me and guiding me on my way. None of these years would be possible without their constant support and love.

Benjamin Keltjens
Delft, July 2021

[1]https://github.com/tudelft/filled-disparity-monodepth

Contents

List of Figures

List of Tables

List of Symbols

d Depth of point in scene.

D Disparity of point between two images.

α Learning Rate.

θ Network Parameters.

List of Acronyms

CNN Convolutional Neural Network.

CPU Computational Processing Unit.

EWC Elastic Weight Consolidation.

FPS Frames per Second.

GAN Generative Adversarial Network.

GPU Graphics Processing Unit.

IMU Inertial Measurement Unit.

LiDAR Light Detection and Ranging.

LSTM Long Short Term Memory.

MAML Model Agnostic Meta Learning.

MAV Micro Air Vehicle.

RAM Random Access Memory.

ReLU Rectified Linear Unit.

RGB Red Green Blue.

RMSE Root Mean Squared Error.

SGBM Semi-Global Block Matching.

SGM Semi-Global Matching.

1
Introduction

1.1. Background

The application of robotics in modern-day environments are becoming more widespread and complex. Enabling them to better interpret their surrounding world allows them to become more helpful and safe. The ability to estimate depth in a scene is an essential component of many 3D computer vision tasks. It is extremely useful for many applications, such as scene reconstruction, autonomous navigation of indoor environments, augmented reality and image augmentation.

Initially, depth estimation was done through classical techniques such as stereo matching [20]. More recently, with the advent of machine learning, deep learning methods with monocular cameras have been used to tackle the problem. Deep learning methods leverage a statistical learning ability to recognise and make use of contextual cues in different scenes [52].

Many existing deep learning methods for monocular depth estimation use supervised learning where data must be labelled with ground truth data [9] [24]. These methods are popular as they have high accuracy. However, the data collection methods for these are quite cumbersome, often requiring heavier sensors such as LIDAR [14].

The development of self-supervised methods for monocular depth estimation has allowed for more flexible data collection methods. The use of stereo training pairs was proposed where disparity was estimated for the left image and a warped generated right pair was compared to the captured right pair [13]. This method has been considerably improved upon since its release, making self-supervised methods competitive [15].

Although this method has opened up data collection in many more domains, it has only been demonstrated on a select few domains. A large problem it runs into is that warping images in textureless regions is not affected by the difference of small or large disparity shifts. Consequently, errors in image reconstruction, as a consequences of disparity shifts, for these regions can not be penalised and learned. This leads to poor performance in untextured scenes that frequently occur in indoor and darker environments.

Another unexplored domain for training in this way is for rotated scenes in pitch and roll. Popular datasets, such as KITTI [14], maintain a fixed angle as data is collected on a car. This leads to issues when tested on images that are rotated [52].

This paper extends self-supervised learning of monocular depth to these two domains. First, a novel loss term, Filled Disparity Loss, is proposed to guide the learning of disparity in untextured regions of an image. Secondly, it is shown that having representative rotations, in both pitch and roll, in the training dataset is sufficient to improve the estimation over the range of rotations. These two advances allow for a broader use of self-supervised learning in more complex domains.

1.2. Research Aim

The research objective and questions of this thesis are shown below. These do not match those outlined in the literature review as the subject of the paper changed focus. The objective of this paper is

to improve self-supervised monocular depth estimation in untextured rotated environments by means of developing a self-supervised learning method trained for a rotated indoor domain

This objective led to the following research questions

Q1 How can self-supervised monocular depth estimation be improved for untextured regions in images?

Q1.1 To what extent is accuracy in untextured regions improved compared to previous methods?

Q1.2 Is there a trade-off in performance in textured regions with previous methods?

Q2 How can self-supervised monocular depth estimation be improved for rotations in pitch and roll?

Q2.1 Is the improvement in accuracy different for rotations in pitch and roll?

Q2.2 How well does the improvement in accuracy generalise over the entire range of rotation?

1.3. Thesis Structure

This report is structured in three parts. Part I contains the scientific article detailing the outcomes of this research. An overview of related work is given in section 2.2. This is followed by a description of the proposed method in section 2.3. The main results of the work are shown in section 2.4. Finally, the conclusions are presented in section 2.5.

Part II contains the preliminary research done for this thesis. The reader should keep in mind that the subject of the thesis was altered slightly after the literature study. First, various depth estimation methods are discussed in section 3.3. Next different domain adaption techniques are detailed in section 3.4. This is followed by an examination of the different methods to improve computational speed in section 3.5. Some details for the platforms for implementation are given in section 3.6 followed by the conclusions of the report in section 3.7.

The thesis is finished in Part III with a discussion of conclusions and recommendations for further work.

I

Scientific Article

Self-Supervised Monocular Depth Estimation of Untextured Indoor Rotated Scenes

Benjamin Keltjens
benjaminkeltjens@gmail.com
Tom van Dijk
J.C.vanDijk-1@tudelft.nl
Guido C.H.E. de Croon
g.c.h.e.decroon@tudelft.nl

MAVLab
Delft University of Technology
Delft, Netherlands

Abstract

Self-supervised deep learning methods have leveraged stereo images for training monocular depth estimation. Although these methods show strong results on outdoor datasets such as KITTI, they do not match performance of supervised methods on indoor environments with camera rotation. Indoor, rotated scenes are common for less constrained applications and pose problems for two reasons: abundance of low texture regions and increased complexity of depth cues for images under rotation. In an effort to extend self-supervised learning to more generalised environments we propose two additions. First, we propose a novel Filled Disparity Loss term that corrects for ambiguity of image reconstruction error loss in textureless regions. Specifically, we interpolate disparity in untextured regions, using the estimated disparity from surrounding textured areas, and use L1 loss to correct the original estimation. Our experiments show that depth estimation is substantially improved on low-texture scenes, without any loss on textured scenes, when compared to Monodepth by Godard et al. Secondly, we show that training with an application's representative rotations, in both pitch and roll, is sufficient to significantly improve performance over the entire range of expected rotation. We demonstrate that depth estimation is successfully generalised as performance is not lost when evaluated on test sets with no camera rotation. Together these developments enable a broader use of self-supervised learning of monocular depth estimation for complex environments.

1 Introduction

The ability to estimate depth in a scene is an essential component of many 3D computer vision tasks. It is extremely useful for many applications, such as scene reconstruction, autonomous navigation of indoor environments, augmented reality and image augmentation.

Initially, depth estimation was done through classical techniques such as stereo matching [12] [3] and Structure from Motion (SfM) [22], amongst others. More recently, methods using Convolutional Neural Networks (CNNs) have been developed for the purpose of estimating depth from a single image [5] [7] [10] [11] [25], opening the possibility to more lightweight and flexible robots and devices having this capability. This was first achieved using ground-truth labels for depth to train in a supervised fashion [5] [14]. The accuracy of supervised methods is impressive on datasets like KITTI [9], featuring outdoor car scenes, and NYUDepthV2 [16], showing indoor scenes. However, supervised learning requires complex and cumbersome data collection methods, such as heavy LIDAR systems.

To combat this, self-supervised learning of monocular depth has emerged to guide learning using stereo cameras [7] [10] or monocular sequences of images [25], where training data is more easily collected. These newer methods are promising for the application of online-learning and show strong performance on similar outdoor domains as supervised learning. However, stereo pair methods are generally applied to domains quite fixed in content and motion. We would like such networks to work in more generic settings, for example in indoor environments or when the camera is under rotation. These conditions still cause problems that hamper their widespread applicability [23].

For less physically constrained applications on dynamic platforms, such as hand-held phones or autonomous Micro Air Vehicles (MAVs), indoor scenes under rotation are common. Indoor domains often feature untextured scenes with structures such as walls and ceilings. Self-supervised learning with image-reconstruction is based on estimating the disparity shift between stereo pairs to synthesise the alternate view and compare it to ground truth view of the other camera. This method is ill-posed for textureless regions as small or large translations of pixels are indistinguishable in the generated, alternate, view.

Additionally, these platforms must handle a wide range of difficult rotations which are challenging. Van Dijk and De Croon show that Monodepth [10] and other networks have difficulties estimating depth under pitch and roll rotation when trained on KITTI [23]. This is explained by the dependence on the vertical position of objects as a depth cue for a camera with a fixed pose.

To improve the application of self-supervised learning in these more complex domains, we present two additions to the current state-of-the-art. First, we propose a novel loss term, Filled Disparity Loss, to learn correct disparity for textureless regions. We make use of an assumption, inspired by stereo matching disparity refinement [8], that disparity can be interpolated between edges, where the image-reconstruction loss is better posed. Our method improves depth estimation considerably when compared to previous image-reconstruction based self-supervised networks. Secondly, we show that building a dataset with representative rotations expected in the network's deployment significantly improves performance without requiring any changes to the network's structure. These two findings allow self-supervised networks to be deployed in the studied complex, but common, domains.

2 Related Work

Several advances have been made to estimate depth with monocular vision. Initially, handcrafted features combined with probabilistic methods were developed, such as Saxena *et al.*'s method which used Markov Random Fields [20]. However, more recently deep learning methods have come to the forefront to tackle the problem [5] [7] [10] [11] [25] .

Supervised Monocular Depth Estimation Eigen *et al.* [5] were the first to use deep learning to leverage learned features for monocular depth estimation. Their network used a two-stage process with coarse and refined depth estimation. Since this work, additions such as GANs [15] and multi-task learning have improved the performance of supervised methods. Some networks leverage semantic and surface normal labels to aid in depth estimation [4].

Performance is strong in various environments with datasets such as KITTI [9] for outdoor scenes for driving cars and NYU-Depth V2 [16] that has a variety of indoor scenes. Supervised methods are able to handle textureless surfaces in indoor environments [6] as depth is directly learned. To an extent, supervised methods have also been able to handle scenes with camera rotation, such as NYUDepth-V2 that contains slightly pitched images. Recently, Zhao *et al*. [24] use an encoding of the camera pose as an additional input to significantly improve performance on rotated datasets.

Self-Supervised Monocular Depth Estimation In order to avoid the vast amount of required labelled data for supervised methods, various self-supervised methods have been developed using both stereo images and monocular video sequences. Garg *et al*. developed the first self-supervised method using image reconstruction with stereo pairs [7]. Instead of estimating depth directly, their proposed method estimates disparity, which is then used to warp the input left image to synthesise the right image of the stereo pair. This warped version is compared to the true right image, and a loss based on the difference between the two (the image reconstruction loss) is used to guide learning. Disparity, d, is converted to depth, D, by $D = \frac{bf}{d}$ where b is the baseline distance and f is focal length in pixels. Godard *et al*. [10] improved on this method with the addition of a fully differentiable warping method as well as estimating the disparity for both the left and right image and defining a loss for consistency between them. There have been various improvements to these methods with diverse network structures as well as novel loss terms [11] [18] [17]. In addition to self-supervision with stereo-images, methods have also been developed to leverage monocular videos as training material. Zhou *et al*. [25] proposed a method that simultaneously estimates pose and depth, allowing images to be warped between neighbouring frames and used for an image reconstruction loss.

Contrary to supervised methods, not much has been done to evaluate self-supervised learning with stereo images when training in indoor environments under rotation; the KITTI dataset is used most often for training and evaluation [7] [10] [17]. This leads to degradation of performance when testing on rotated images and datasets [23]. Moreover, to our knowledge, the ill-posed nature of the image reconstruction loss with image warping in low-texture environments is also yet to be addressed in the literature.

3 Proposed Method

In order to estimate depth in indoor, rotated environments, our proposed method entails both a novel loss term using filled disparity maps as well as the use of representative datasets to learn expected rotations in environments. First, the losses and network are presented and then a description of the rotated datasets used for training is given.

3.1 Filled Disparity Loss

As described in section 2, self-supervised learning of monocular depth using stereo images infers disparity maps from a single RGB image. However, the image reconstruction loss in textureless regions of the scene is ill-posed due to small and large disparity maps producing the same result, and therefore loss, in those regions. The effect can be seen for the state-of-the-art Monodepth method [10] in the second column of Figure 4. We propose an additional loss term, called Filled Disparity Loss, which uses filled disparity maps in regions of low texture as an additional loss for indoor scenes. This is inspired by the interpolation of sparse of disparity estimates in ELAS [8]. This new loss is used together with the losses for stereo self-supervised learning used by Godard *et al.* in Monodepth [10] as can be seen in the network overview in Figure 1.

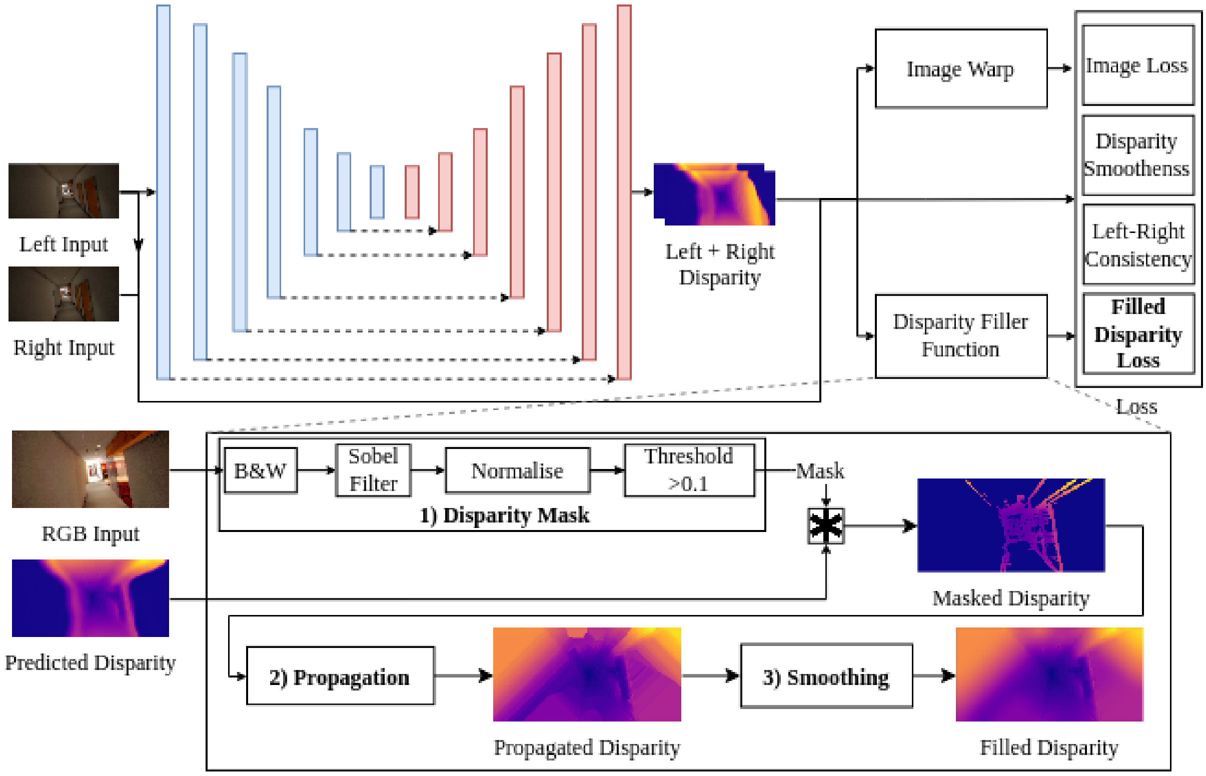

Figure 1: Structure of our network and the Disparity Filler Function

The Disparity Filler Function generates an interpolated disparity map as seen in the bottom of Figure 1. This function is split into three main parts.

Disparity Mask First, a sparse disparity mask is generated by identifying significant edges where the reconstruction loss is well posed. This is done by means of a Sobel filter of size 7 on the grayscale input image. A large filter is used to avoid detecting smaller changes due to noise, that may appear in regions of lower texture. Feature detection methods such as ORB [19] and SURF [2] were considered, however, they produce far too few, scattered points for the following steps. Additionally, Sobel filters perform well in the Tensorflow framework [1]. The gradient image is then normalised to the interval $[0,1]$ and then an "active" mask of textured pixels is defined as values greater than 0.1.

Propagation Secondly, the disparity is propagated outwards to other edges. A while loop is used where pixels are filled by averaging the immediately neighbouring active pixels as

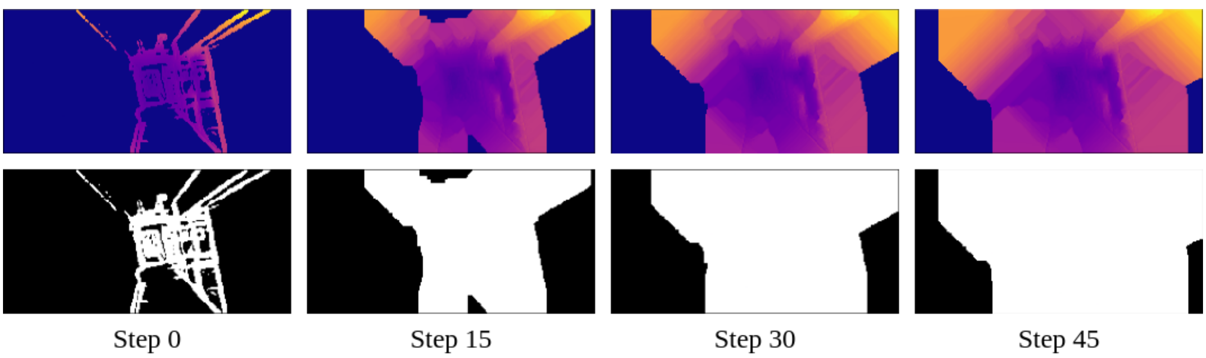

Figure 2: Progression of the Propagation step of the Disparity Filler Function to fill the sparse disparity map (top) and active pixel mask (bottom)

can be seen in Figure 2. This loop continues until the entire image has been filled with active pixels. This method allows for interpolation between edges as well as interpolation to the border of the image where there might be no active pixels.

Smoothing In the final step a 5x5 smoothing kernel, using inverse distance and 1 at the centre (normalised), is used to smoothen the new disparities in textureless regions. Textureless pixels are found using the inverse of the initial mask of active pixels.

Overall, the Disparity Filler Function is able to generate more accurate disparity estimates in the textureless regions. However, from Figure 2 it is apparent that there are some undesired effects such as filled disparities being propagated perpendicular to detected edges. This can be improved by methods such as used by Geiger *et al.* [8] with interpolation over Delauny Triangulation between points. However, this process is not differentiable and performs slower on the CPU with large quantities of active pixels. The new map is compared to the originally estimated disparity map using an L1 loss.

3.2 Losses

The loss function for this network is based on the work done by Godard *et al.* [10] and consists of 4 terms, resolved at multiple scales s, as seen in Equation 1. The total loss is the sum of losses at 4 different scales $L = \sum_{s=1}^{4} L_s$.

$$L_s = \alpha_{ap}\left(L_{ir}^l + L_{ir}^r\right) + \alpha_{ds}\left(L_{ds}^l + L_{ds}^r\right) + \alpha_{lr}\left(L_{lr}^l + L_{lr}^r\right) + \alpha_{fd}\left(L_{fd}^l + L_{fd}^r\right) \tag{1}$$

L_{ir} evaluates the loss of image reconstruction of input images, L_{ds} evaluates the smoothness of the disparity map, L_{lr} evaluates the consistency of the left and right disparity maps and L_{fd} evaluates the similarity of the estimated disparity map and the filled disparity map. Disparities are predicted for both the left and right images; the superscript l and r indicate which image the loss is calculated for. The losses are shown in detail for the left images case.

Image Reconstruction Loss To learn the correct disparity map, both the left and right images are warped to match the opposite direction and are compared to the input images as described in section 2. Similar to [10] the imaging warping is done using bilinear sampling [13]. The reconstruction loss is measured by both L1 and SSIM, comparing input image I_{ij}^l with the reconstruction $\tilde{I}_{ij}^l$, where N is number of pixels and $\alpha = 0.85$ is the weighting term:

$$L_{ir}^l = \frac{1}{N}\sum_{ij} \alpha \frac{1 - SSIM(I_{ij}^l, \tilde{I}_{ij}^l)}{2} + (1-\alpha)\left|I_{ij}^l - \tilde{I}_{ij}^l\right| \tag{2}$$

Disparity Smoothness This loss term encourages local smoothness by minimising gradients ∂d in the disparity map. The loss takes account of large gradients at edges in the input image by reducing the loss with the image gradients ∂I.

$$L_{ds}^l = \frac{1}{N}\sum_{ij} \left|\partial_x d_{ij}^l\right| e^{-\left|\partial_x I_{ij}^l\right|} + \left|\partial_y d_{ij}^l\right| e^{-\left|\partial_y I_{ij}^l\right|} \tag{3}$$

Left-Right Consistency As the network outputs disparities for both the left and right images, this loss term ensures consistency between them. It uses the L1 loss between the left disparity map d_{ij}^l and the right disparity map projected onto the left view $d_{ij+d_{ij}^l}$ [10].

$$L_{lr}^l = \frac{1}{N}\sum_{ij} \left|d_{ij}^l - d_{ij+d_{ij}^l}\right| \tag{4}$$

Filled Disparity Loss Our proposed loss is formulated as an L1 loss between the estimated disparity map d_{ij}^{l} and the filled disparity map $\tilde{d}_{ij}^{l}$. The loss only applies to untextured regions as the Disparity Filling Function does not change disparity in textured regions of an image.

$$L_{fd} = \frac{1}{N} \sum_{ij} \left| d_{ij}^{l} - \tilde{d}_{ij}^{l} \right| \tag{5}$$

3.3 Datasets

This work makes use of the AirSim Building_99 drone simulation environment developed by Microsoft [21] to generate training data. A more detailed description of the collection method can be found in Appendix A. Various indoor scenes are captured with left and right images as well as ground truth disparity. The environment contains a range of scenes ranging from long narrow hallways to open spaces as seen in first column of Figure 4. Flying on a trajectory through the entire building, 18000 images (128x256) are collected for training and 2000 for testing. Depth is saturated on the boundaries 0-80 [m] as done in [10].

Four datasets are generated with pitch and roll motions enabled or disabled, as listed in Table 1. The angles are normally distributed in pitch and roll with a standard deviation of 10 degrees about 0 degrees.

Dataset ID	Dataset Description
N	Nominal with no rotations
R	Rotation in roll with standard deviation of 10 degrees
P	Rotation in pitch with standard deviation of 10 degrees
PR	Rotation in pitch and roll with standard deviation of 10 degrees

Table 1: Dataset Descriptions

3.4 Implementation Details

The hyperparameters of the network are the same as in [10]. The network is trained for 50 epochs with a batch size of 8 using an Adam Optimiser where β_1, β_2, and ε are 0.9, 0.999 and 10^{-8} respectively. The learning rate λ is 10^{-4} for 30 epochs and is then halved every 10 epochs. Colour augmentations and flipping of the input image pairs is also performed randomly for data augmentation, similar to [10]. A detailed description of the computational performance on different platforms is given in Appendix B.

4 Results

To show our improvements we begin by analysing the effect of the Filled Disparity Loss on the performance on the overall dataset. We also evaluate the effect of image texturedness on performance. This is followed by an analysis of the effect of training with datasets that contain representative rotations on overall performance, as well as over the range of rotations.

Metrics We use the metrics described in [10]. Given ground truth depth D^* and estimated depth D, the following metrics are defined. (1) Absolute Relative Error: $\frac{1}{|N|}\sum_{i\in N}\frac{\left|D_i-D_i^*\right|}{D_i^*}$, (2) Squared Relative Error: $\frac{1}{|N|}\sum_{i\in N}\frac{\left\|D_i-D_i^*\right\|^2}{D_i^*}$, (3) RMSE: $\sqrt{\frac{1}{|N|}\sum_{i\in N}\left\|D_i-D_i^*\right\|^2}$, (4) RMSE log: $\sqrt{\frac{1}{|N|}\sum_{i\in N}\left\|log\left(D_i\right)-log\left(D_i^*\right)\right\|^2}$, (5) δ_t: $\%$ of D_i $s.t.$ $max(\frac{D_i}{D_i^*},\frac{D_i^*}{D_i})<t$, where $t\in\{1.25, 1.25^2, 1.25^3\}$.

Performance Metrics on Untextured Scenes The effect of the new loss term can be seen by altering its weight α_{fd}. The results displayed in Table 2 are from training and testing on the pitch and roll (PR) dataset. We can see that increasing α_{fd} leads to an improvement in almost all error metrics, except $\delta < 1.25$ at $\alpha_{fd} = 0.6$. Without the new loss function, the network performs considerably worse when trained and tested on this domain. The decrease in the errors stagnates at larger values of the loss weight as it reaches the optimum.

α_{fd}	Lower is better				Higher is better		
	Abs. Rel.	Sq. Rel.	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
0.0 (Monodepth)	1.7676	89.3917	13.692	0.707	0.667	0.782	0.840
0.1	1.1126	49.5702	10.575	0.572	0.685	0.803	0.865
0.2	0.3696	2.8818	6.301	0.417	**0.687**	0.816	0.882
0.3	0.3262	2.3222	6.175	0.396	0.683	**0.819**	0.892
0.4	**0.3172**	2.1787	6.201	**0.392**	0.676	**0.819**	**0.895**
0.5	0.3200	2.1690	**6.109**	**0.392**	0.674	**0.819**	**0.895**
0.6	0.3181	**2.1549**	6.246	0.394	0.664	0.815	0.894

Table 2: Performance of networks with changing α_{fd} on the PR dataset.

Influence of Texturedness on Performance We are interested in seeing the effect of texturedness on the network's performance. Texturedness is measured by the percentage of pixels with a Sobel gradient above our threshold of 0.1. In Figure 3 it is apparent that for lower levels of texture the error is much higher for networks trained with a lower loss weight α_{fd}. As the texturedness of an image increases, the errors converge to the same value. This confirms that the new loss function significantly improves estimation in textureless regions and preserves performance in textured regions.

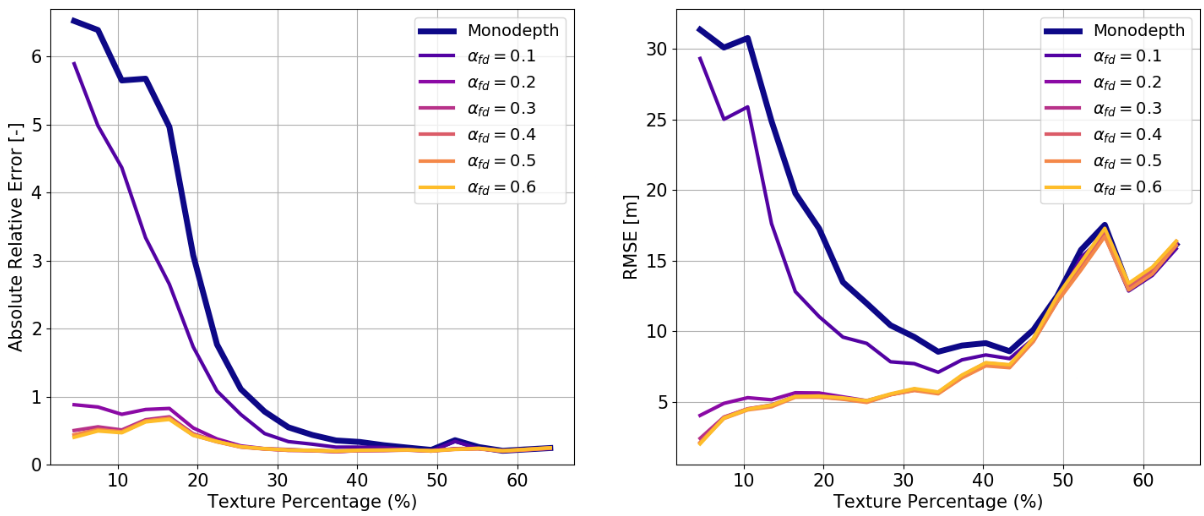

Figure 3: Absolute Relative error (left) and RMSE (right) of the PR test set as a function of the percentage of textured pixels in an image for varying α_{fd}. Lower is better.

Qualitative Performance on Untextured Scenes The qualitative results of the network with the Filled Disparity Loss can be seen in Figure 4. The errors from Monodepth are quite severe as disparity estimation in close-by, untextured, regions are extremely low (faraway regions). It is apparent that the additional loss term has helped to guide the network in textureless areas to correct for the ill-posed image reconstruction. This is especially apparent in dark environments where texture is lost (row 4), which can be common indoors. Additionally, in textured regions where Monodepth performs well, it is apparent that the performance is not degraded. Even though there is large improvement, our network still does not completely capture the scale of disparities in near, untextured, regions (rows 1, 3, 4). However, when the disparity is inverted to find depth, the differences at close range are small.

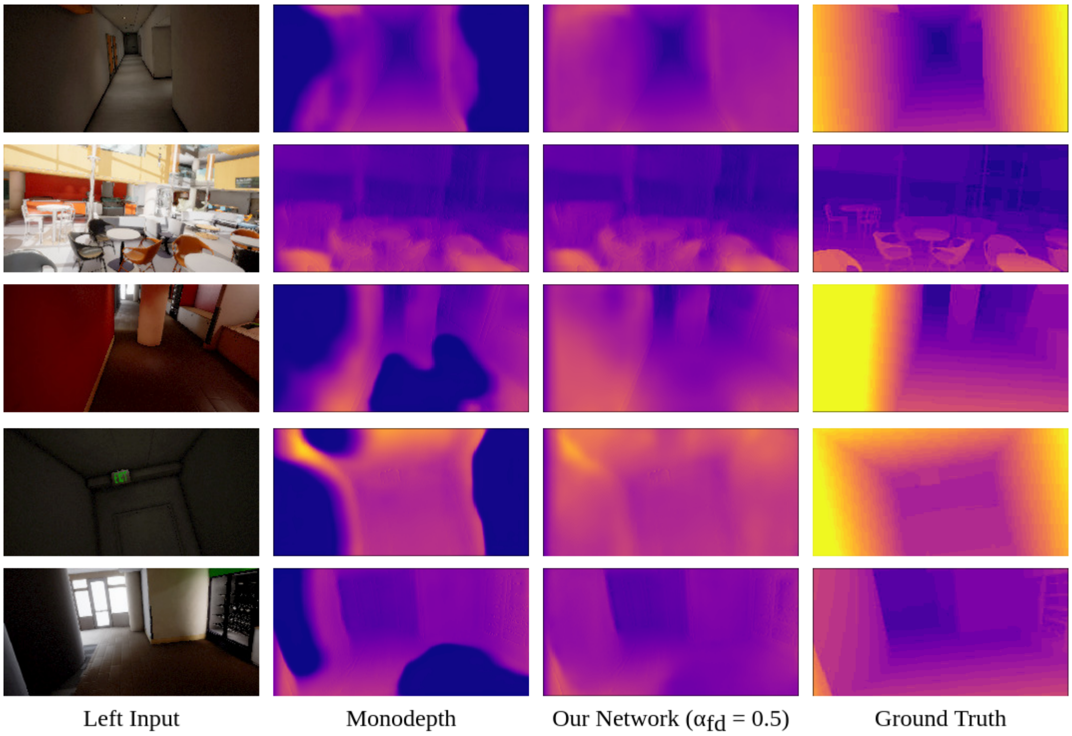

Figure 4: Disparity maps for different scenes on the PR test set

Performance Metrics on Rotated Datasets Our results in Table 3 show the effect of training on the different datasets when tested on the nominal (N) and pitch and roll (PR) datasets. The networks are all trained with $\alpha_{fd} = 0.5$. The first result of interest is that training the networks on datasets that are rotated does not seem to significantly affect the performance on the nominal dataset.

Training Dataset	Testing Dataset	Lower is better				Higher is better		
		Abs. Rel.	Sq. Rel.	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
N	N	0.3156	2.3348	6.366	0.394	0.673	0.815	0.893
R	N	0.3089	2.1977	6.424	0.393	0.670	0.815	0.894
P	N	0.3134	2.5294	6.370	0.391	0.675	0.819	**0.897**
PR	N	**0.3052**	**2.1101**	**6.287**	**0.387**	**0.679**	**0.821**	**0.897**
N	PR	0.7245	7.4258	10.232	0.673	0.317	0.542	0.701
R	PR	0.4737	3.9536	7.992	0.507	0.483	0.701	0.821
P	PR	0.3461	2.3993	6.697	0.421	0.611	0.789	0.880
PR	PR	**0.3200**	**2.1690**	**6.109**	**0.392**	**0.674**	**0.819**	**0.895**

Table 3: Performance of networks on the test sets of nominal (N) and Pitch Roll (PR)

When tested on the pitch and roll test set, the discrepancy between the networks is noticeable, as seen in Table 3. The network trained on the nominal dataset performs extremely poorly on the PR dataset when compared to the others and its performance on the nominal dataset. As expected, with more axes of rotation in the training dataset the network performs better. Training on the PR dataset has similar results when testing on the nominal and PR test sets. The network trained solely on pitch (P) performs significantly better than that trained only on roll (R). This is probably because training on pitch allows the learning of correct vertical height depth cues that arise from pitch rotation [23].

Qualitative Performance on Rotated Datasets Looking at a comparison of disparity maps in Figure 5 we can see that the performance of the network trained on the nominal dataset is severely degraded and struggles to retain any resemblance of structure when testing on rotated datasets. With more representative rotations in the training set the preservation of structure in the depth maps greatly improves.

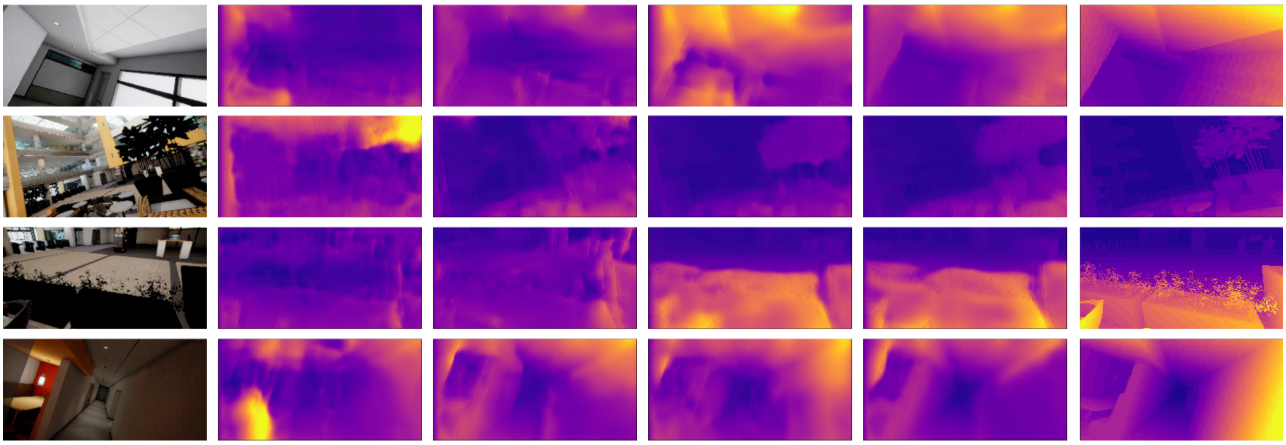

Figure 5: Comparison of disparity maps on PR test set. Networks trained with $\alpha_{fd} = 0.5$.

Performance over Range of Rotation From Figure 6 it is evident that training on the pitch and roll (PR) dataset does not reduce performance on the pitch dataset (P) when compared to the network trained only on pitch (P). As expected, the error of the network trained on the nominal dataset (N) is much larger when the absolute value of the pitch angle increases. Also promising is that for the two networks trained with rotations, their performance remains quite constant over the range of pitch. The spread of error on the rotated dataset results are also much smaller than the network on nominal (N), indicating more consistent performance. Supplementary analysis for the roll (R) and the pitch and roll (PR) test sets are given in Appendix C.

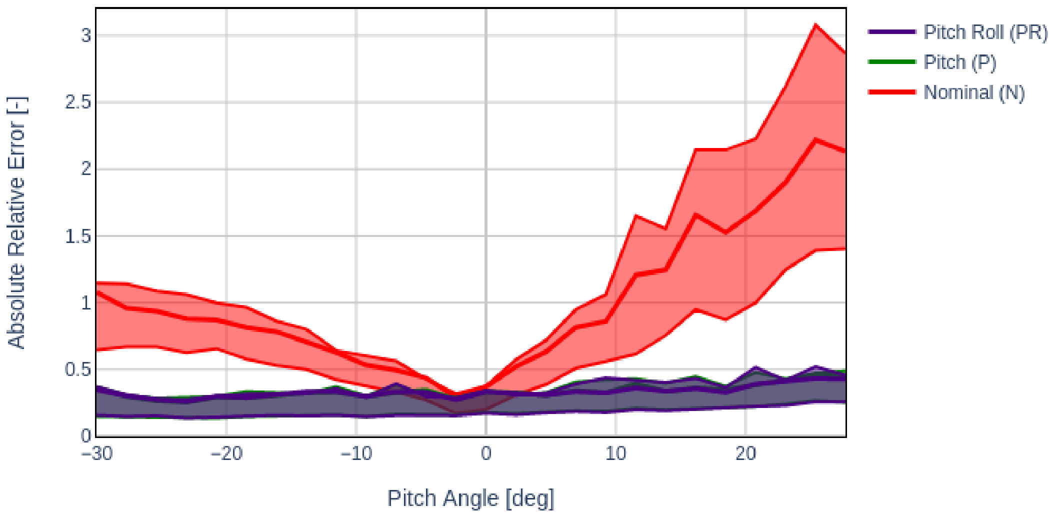

Figure 6: Comparison of networks trained on nominal (N), pitch (P) and pitch and roll (PR) datasets on the pitch (P) test set as a function of pitch angle. The 25th and 75th percentiles of the data are shown by the shaded areas. Pitch (P) and pitch and roll (PR) are almost identical.

5 Conclusion and Recommendations

In this work, we have presented a new Filled Disparity Loss term to improve depth estimation in textureless regions of images. Our method successfully estimates depth in untextured regions of indoor environments whilst preserving performance on textured regions. Additionally, we have demonstrated the ability of self-supervised networks for monocular depth estimation to generalise over rotations of scenes given a representative dataset.

For further work, there are a few insights into possible areas of improvement. First, gathering and testing on a real-world dataset would be important to assess how this method would perform on a physical platform. Secondly, it would be interesting to investigate the effect of height as it was kept constant for these experiments. A particularly exciting line of research would be to take advantage of this method for online learning of monocular depth in indoor applications. Several computational and hardware improvements would have to be considered, but it would be very interesting for indoor navigation.

Overall, this work allows for more mobile applications of self-supervised monocular depth estimation in complex, indoor, environments.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. URL https://www.sciencedirect.com/science/article/pii/S1077314207001555. Similarity Matching in Computer Vision and Multimedia.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 377–384 vol.1, 1999.

[4] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:2650–2658, 2015.

[5] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, 3(January):2366–2374, 2014. ISSN 10495258.

[6] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[7] Ravi Garg, B. G. Vijay Kumar, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9912 LNCS:740–756, 2016.

[8] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Computer Vision – ACCV 2010*, pages 25–38, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[10] C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611, 2017.

[11] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October(1):3827–3837, 2019.

[12] Heiko Hirschmüller. Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2005.

[13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

[14] H. Jung, Y. Kim, D. Min, C. Oh, and K. Sohn. Depth prediction from a single image with conditional adversarial networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1717–1721, 2017.

[15] Aran C.S. Kumar, Suchendra M. Bhandarkar, and Mukta Prasad. Monocular depth prediction using generative adversarial networks. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2018-June:413–421, 2018.

[16] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[17] A. Pilzer, S. Lathuilière, N. Sebe, and E. Ricci. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9760–9769, 2019.

[18] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards Real-Time Unsupervised Monocular Depth Estimation on CPU. *IEEE International Conference on Intelligent Robots and Systems*, pages 5848–5854, 2018.

[19] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[20] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005.

[21] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.

[22] Peter Sturm and Bill Triggs. A factorization based algorithm for multi-image projective structure and motion. In Bernard Buxton and Roberto Cipolla, editors, *Computer Vision — ECCV '96*, pages 709–720, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[23] T. Van Dijk and G. De Croon. How do neural networks see depth in single images? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2183–2191, 2019.

[24] Yunhan Zhao, Shu Kong, and Charless Fowlkes. When perspective comes for free: Improving depth prediction with camera pose encoding, 2020.

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

A Data Collection

Data collection for this work was done in the Airsim *Building_99* simulation environment using binaries[1] from Microsoft [21]. The drone platform with stereovision and disparity ground truth for the left view was used to collect data in the environment. Movement of the drone was done by setting the exact pose of the drone as it allowed for control of the distribution of rotations. The translation of the drone followed a set path that allowed it to view the different parts of the building environment. The path taken over the building floor plan can be seen in Figure 7. Examples of the different scenes in *Building_99* can be seen in Figure 8

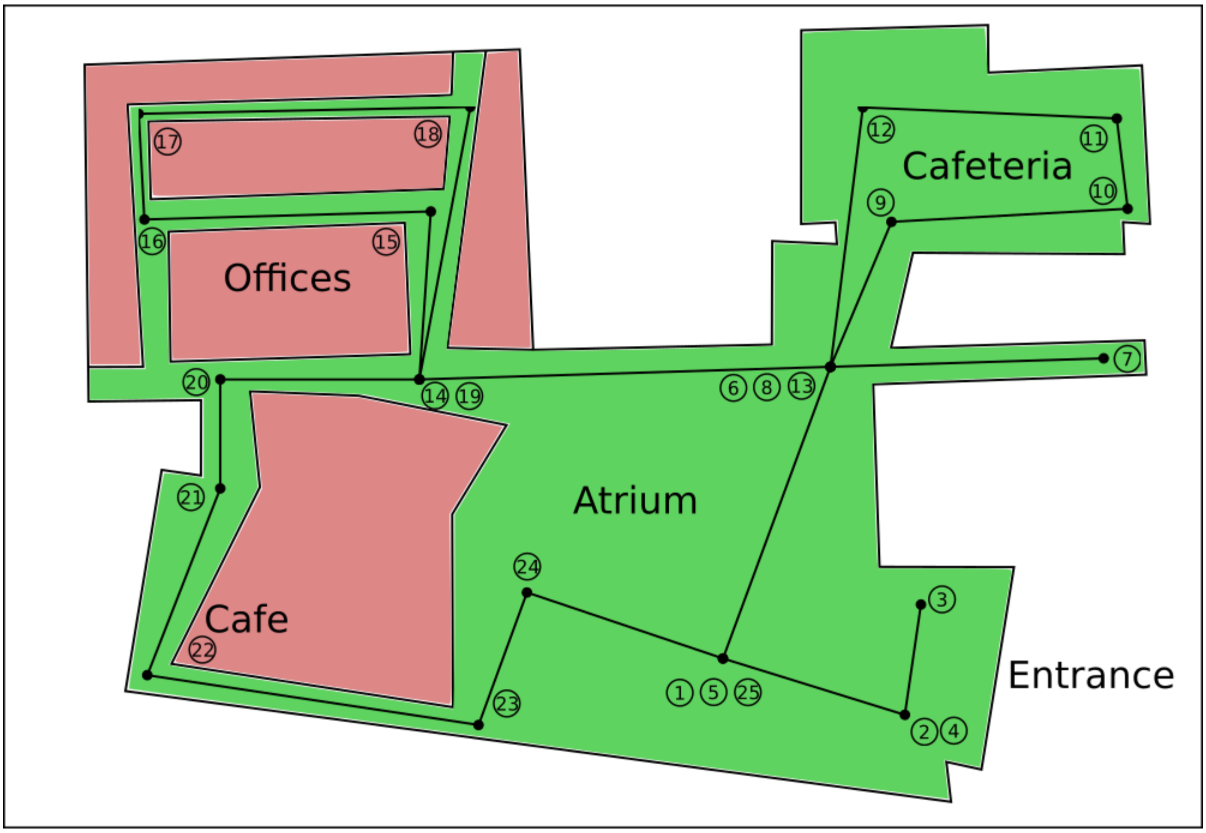

Figure 7: Floor plan of Building_99 Simulation environment with labelled route

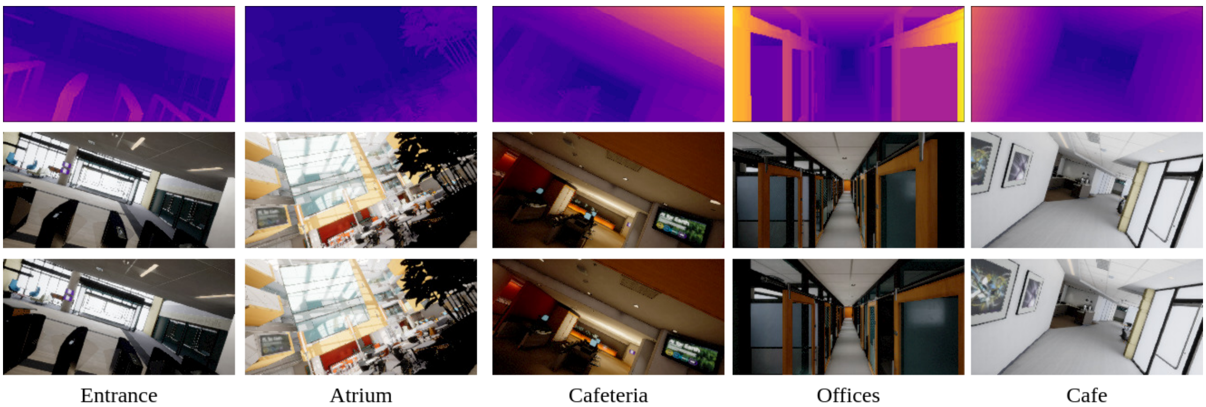

Figure 8: Example scenes of locations in *Building_99* under Pitch and Roll Rotation. Shown are the left image (bottom), right image (middle) and the ground truth disparity (top)

[1]The specific binary version can be found at https://github.com/microsoft/AirSim/releases/tag/v1.3.1-linux

B Computational Performance

The addition of the Filled Disparity Loss increases the computational load required for training when compared to Godard *et al.* [10]. On a single GTX 1080 Ti Monodepth [10] trains at 48.3 [examples/s] whilst our network achieves 29.5 [examples/s]. As the Filled Disparity function is only imposed in training, the speed in inference, 82.6 [examples/s], is the same for both.

In the interest of improving computational performance, a simple trimming process is performed where the number of layers in both the encoder and decoder is reduced to a percentage of the original size. The results of the training speed and accuracy on the Pitch and Roll (PR) dataset on three different platforms can be seen in Table 4. The three Nvidia platforms considered are the GTX 1080 Ti, Jetson TX2 and Jetson Nano. The smaller Jetson platforms are interesting in their ability to be placed on light mobile robots, such as drones, to enable online learning.

Network Size [%]	Training Speed [examples/s]			Error Metrics				
	GTX 1080 Ti	Jetson TX2	Jetson Nano	Abs. Rel.	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
20	58.80	6.67	1.81	0.348	6.689	0.632	0.795	0.879
40	55.13	5.88	1.68	0.331	6.382	0.656	0.808	0.887
60	43.98	5.08	N/A	0.320	6.195	0.667	0.815	0.893
80	35.60	4.51	N/A	**0.313**	6.119	**0.677**	**0.821**	**0.896**
100	29.50	3.86	N/A	0.320	**6.109**	0.674	0.819	0.895

Table 4: Training speed and error metrics for differing network size on three Nvidia platforms. N/A is used to indicate that it was not possible to train the network for that size. Some error metrics are removed in the interest of space.

As expected, the training speed improves for a decrease in the size of the network on all platforms. On the Jetson Nano, only smaller sized networks can be trained due to memory constraints. Reducing the batch size did not enable training of the network. An interesting result in Table 4 is that the performance in accuracy is not degraded by too large an amount whilst retaining only 20% of the layers.

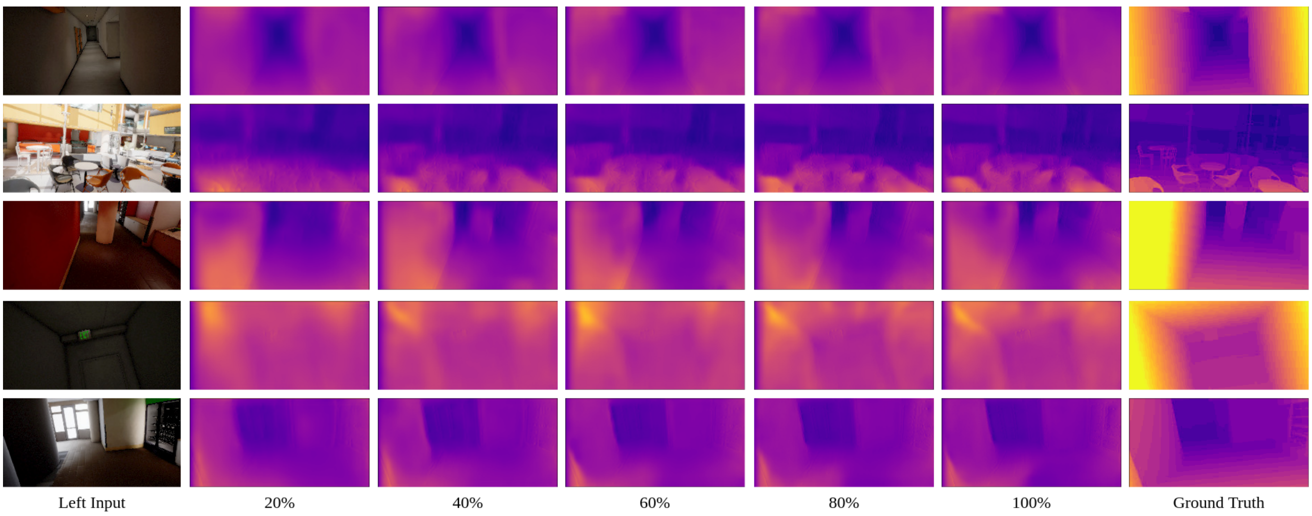

Figure 9: Comparison of disparity map outputs for networks with differing sizes. Sizes are given as percentage of nominal network size.

Figure 9 shows a qualitative comparison of the different network sizes. The scale of the disparity estimation is quite constant and the performance of the Filled Disparity Loss is retained in untextured areas. Though it does seem that detail is lost in more textured areas. For example, in the second row, the chairs which are delineated quite well in the original 100% sized network become a single blur in the network 20% the original's size. This seems promising for producing coarse estimations on low-power and light platforms.

C Supplementary Rotational Analysis

In addition to the rotational analysis given in section 4 a few additional results are presented here. First, an analysis of the effect of the roll angle on performance on the roll (R) test set is given followed by a discussion of the effect on performance over both the pitch and roll angle on the pitch and roll (PR) test set.

In Figure 10 a comparison of the performance on the roll (R) test set is shown for 3 different networks (trained on nominal (N), roll (R) and pitch and roll (PR)). As expected the results are quite similar to that of Figure 6. The performance of the network trained on the nominal (N) dataset is similar to the others for low roll angles but steadily gets worse for larger roll angles. Contrary to pitch, the decrease in performance for the roll is symmetric. This is probably because roll angles change vertical cues symmetrically whereas pitch does not. Similar to pitch, the performance for the networks trained on roll (R) and pitch and roll (PR) is maintained over the range of roll angles. As for the pitch, the mean result is near the 75th percentile as the average is skewed by larger outliers in error.

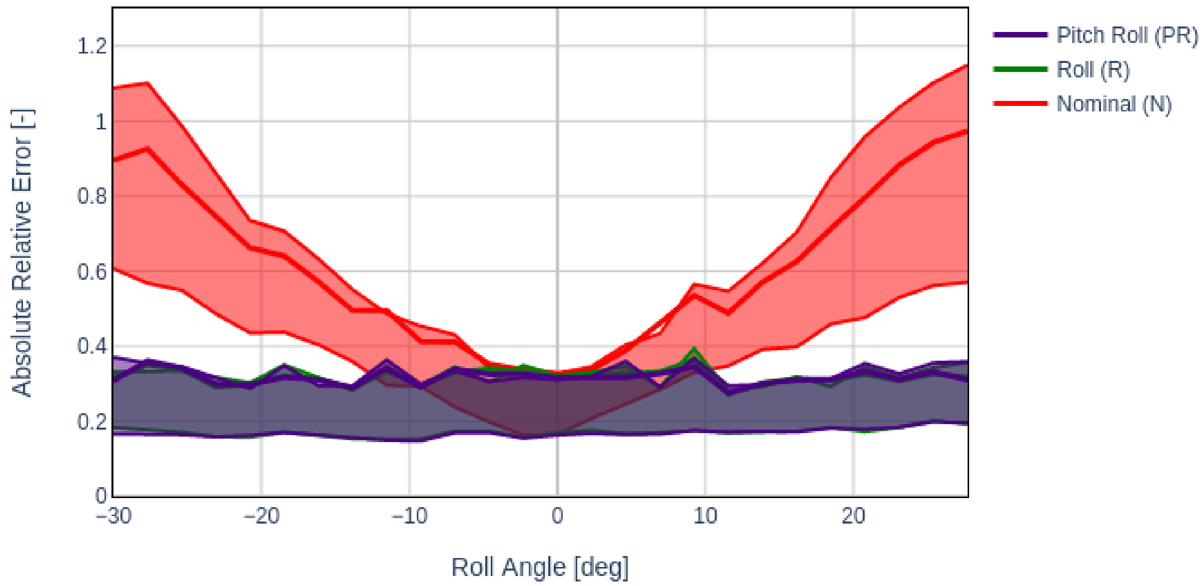

Figure 10: Comparison of networks trained on nominal (N), roll (R) and pitch and roll (PR) datasets on the roll (R) test set as a function of roll angle. The 25th and 75th percentiles of the data are shown by the shaded areas. Roll (R) and pitch and roll (PR) are almost identical.

For a 3D analysis, Figure 11 shows the performance on the pitch and roll (PR) dataset over the range of pitch and roll angle, for 4 networks trained on each of the datasets. The first significant result is that the network trained on the nominal (N) dataset performs far worse, overall, than the other networks. The performance near smaller angles of roll and pitch is good for the network trained on the nominal (N) dataset. However, for larger angles the performance quickly degrades to large errors. This is especially the case for positive pitch angles. The network trained on the roll (R) dataset has a similar shape to that of the nominal, that larger pitch angles degrade performance substatially. However, for each pitch angle, along the range of roll angles, the performance is fairly consistent.

Both the networks trained on the pitch (P) and pitch and roll (PR) seem to have more consistent, and better, performance over the range of rotations. Although, performance on the pitch (P) trained network does seem to slightly worsen when going to extreme roll angles. The performance of the pitch and roll (PR) trained network is constant over the entire range of 2D rotations and is overall the best. From these results, it is apparent that pitch is the more important rotation for improving performance on dynamic platforms. Also, it can be seen that training on both pitch and roll is the most effective for improving performance over the entire range of expected rotations.

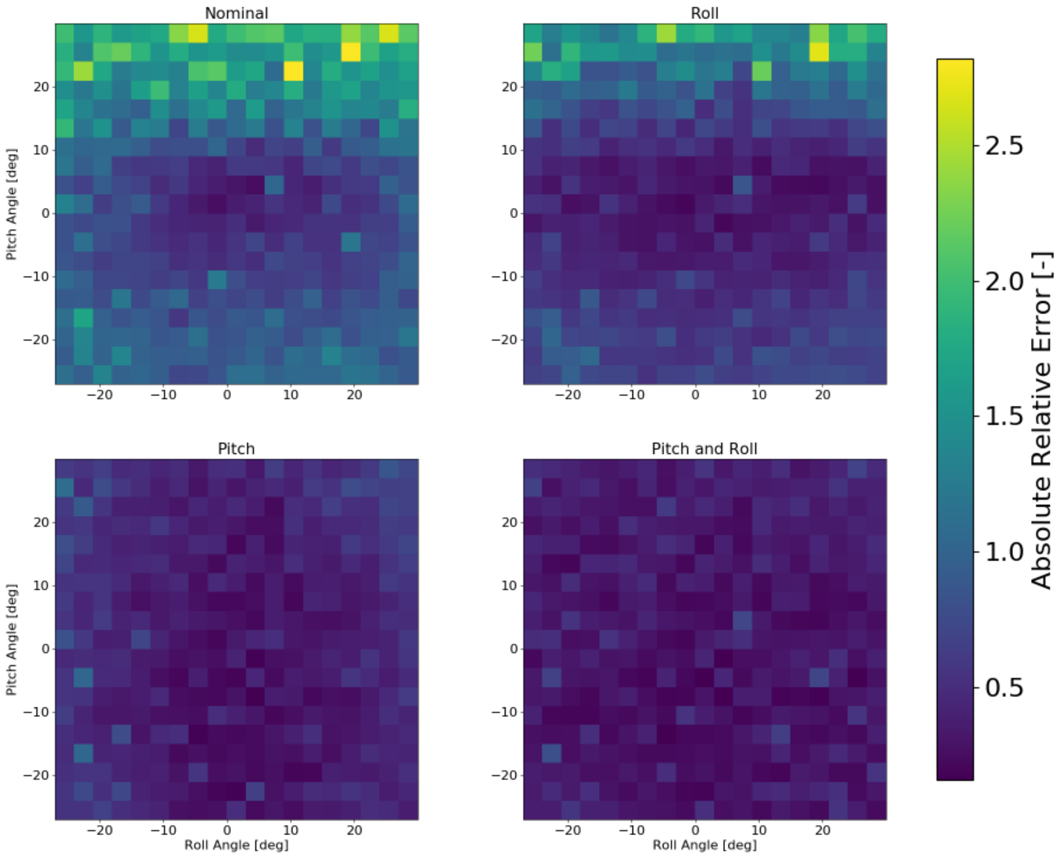

Figure 11: Comparison of performance of networks trained on all four datasets on the pitch and roll (PR) test set as a function of both pitch and roll angle.

II

Preliminary Research[1]

[1]The Preliminary Research was completed for the AE4020 Literature Study course.

3

Literature Review

In this chapter the review of the relevant literature is presented. First, various depth estimation methods are discussed in section 3.3. Next different domain adaption techniques are detailed in section 3.4. This is followed by an examination of the different methods to improve computational speed in section 3.5. Some details for the platforms for implementation are given in section 3.6 followed by the conclusions of the report in section 3.7. During the course of the thesis the focus shifted from online learning on a drone indoors to solving the issues of indoor rotated depth estimation.

3.1. Executive Summary

Depth estimation is an important capability for modern systems and robots. This extends from different fields such as 3D scene reconstruction [64] to robot navigation [32]. In the field of autonomous robotics, depth estimation is important for tasks such as obstacle avoidance, object detection, navigation and visual odometry.

The field of depth estimation has existed for a while and continuously sees improvement in attempts to make better predictions over time [64]. However, depth estimation is often tested for specific environments and users, not covering the entire spectrum of domains. This is especially true for methods that attempt to leverage contextual cues in the domain [52]. Therefore, when deploying an autonomous robot, that can operate fully alone, it would be ideal that our method for estimating depth could adapt to its surroundings and learn on its own. This desire can be made possible through self-supervised learning; a paradigm in which, instead of being fed supervised labels by a human, an algorithm can learn by comparing its prediction of the world with its observation of the world.

This literature study delves into assessing what the current body of knowledge consists of in the field of depth estimation and online adaption. This is done in order to determine what is required to make self-supervised learning of depth possible on a mobile low-power robot, such as a Micro Air Vehicle (MAV).

Estimating depth ranges over several different methods. Classically, stereo image pairs have been leveraged to estimate the location of a projected point in 3D space with methods such as SGM [20]. These methods, by themselves, are optimisation based and use no learning methods. Adaptions with CNNs are possible to allow them to learn, however, these are computationally expensive and complex in their structure [67][62][55][28].

A newer method is to leverage contextual cues in single images to estimate depth. This began with Saxena [45] in 2005 with handcrafted filters to extract and relate cues in an image. However, more recently, these methods have extended to using CNNs, starting with Eigen [9] in 2014. This inspired a large number of other papers that also learn with ground truth depth, but change the structure and loss functions to pursue higher accuracy [64][24][8][29]. One large issued identified was that it was very difficult to compile a large training set of ground truth depth data, especially when training on specific new domains. This lead to an important development by Garg *et al.* [13] to develop a self-supervised method. Essentially, Garg saw that instead of estimating depth, estimating disparity from a stereo pair would allow for simpler collection of data as only a second camera was need. This loss method, with image warping, can be seen in Figure 3.1.

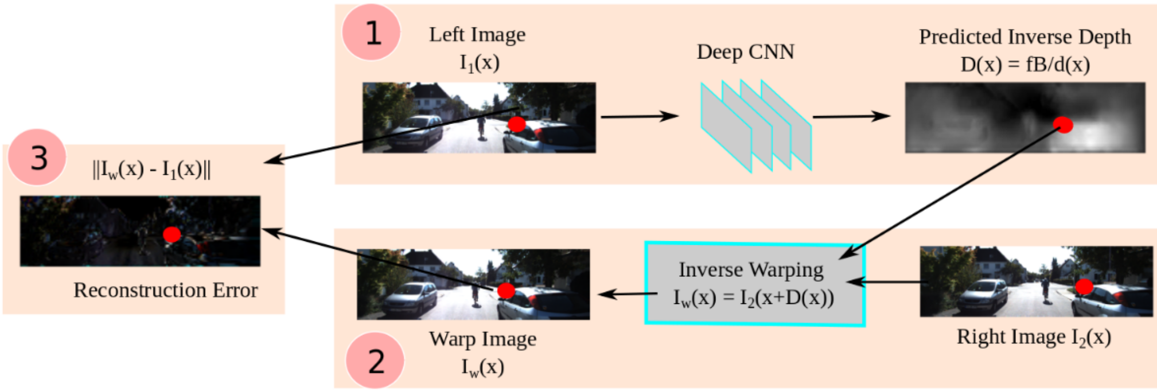

Figure 3.1: Self-supervised training architecture [13]. Predicted inverse depth is the same as image disparity

Like Eigen's paper there have been a large amount of additions to Garg's method [15][16][41][8][37]. One of the first, large, improvements was by Godard *et al.* [15] with a change to the loss term that produced a disparity map for both directions of warping to check consistency between them. Another noticeable improvement, by Poggi *et al.* [41], was to use a pyramidal structure to analyse the image with decoders at different resolutions, and then up-sampling them to higher resolution decoders. This method retained performance, but decreased runtime considerably, especially as it allowed to choose which layer of resolution to terminate inference at. Although these self-supervised methods do not produce as accurate results as the ground truth methods do on similar domains [64], they allow for easier adaption to different domains by the means of simpler data collection methods. One of the most recent innovations, that made data collection even simpler, was to use sequences of monocular images. Zhou *et al.* [68] discovered that, by leveraging the change in pose between subsequent monocular frames depth could be learned in a self-supervised way with only one camera. This class of method allows for data collection with the least amount hardware, however, also leads the lowest accuracy compared to the other methods. Also, it requires multiple networks to run in parallel and more data to be stored in memory. There have also been methods that fuse these learning estimate methods with the more classical methods [31][10][11].

These methods are compared to find the most apt for self-supervision on a low-power computational hardware equipped drone. This means the most important aspects are possibility for self-supervised learning and inference and training speed. From a comparison of the methods in Figure 3.2 it is seen that Garg and Godard have the most recommended methods for these purposes. This is because they boast the fastest training speeds whilst maintaining acceptable accuracy.

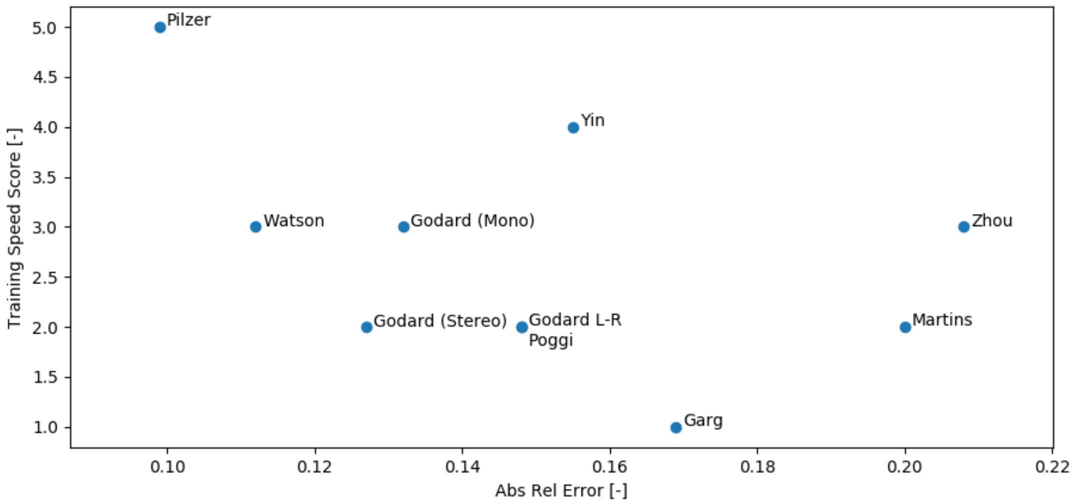

Figure 3.2: Scatter plot of depth estimation methods accuracy and training speed

There have been a few papers that examine the application of these learning methods on drones. Pinard *et al.* [40] use two frames in an image sequence, one corrected for pose transition to the other, as inputs for their networks. This serves as a good adaption for applications on drones as pose is accounted for, however, this method is only demonstrated with processing on a computer receiving a feed from the drone. Zhao *et al.* [66] encode the pose of the drone, in height and pitch, into a 4th channel input. This methods works very well for inference on exotic poses at a low additional cost of computation. Yang *et al.* [58] demonstrate one of the very few complete applications on a drone using SLAM and auto-encoders together. However, this runs on quite a bulky, powerful, computational setup as well as only being tested in a very constrained obstacle course. There is still work to do to develop learning based depth estimation methods for drones on lower power computational hardware, applied in more diverse environments.

The aforementioned depth methods offer a way to interpret an environment that they have been trained for, however, they are quite constrained to the environment they train on and do not generalise well to other domains [64]. In order to be applicable to a drone, it would be preferable to work irrespective of the domain.

One of the obvious methods is to try train on varying domains. Many do this by training on different synthetic domains, however, this is susceptible to lighting and in the end isn't fully able to generalise [2]. Another method is to use estimated confidence measures to selectively train with 'good' estimates of depth online. However, this requires more computation and still inherits the bias and errors of SGM, or other classical depth methods, while only getting a sparse loss calculation [50].

Architectural changes also provide a promising direction for online domain adaption. For example, employing LSTM networks [30][27], allows to leverage temporal information, from past inputs, in an inexpensive manner. Alternatively, Zhang *et al.* [63] use adapter units that learn to change the statistical features of normalisation layers outputs in order to fit the new domain better. This seems to be quite powerful, however, also employs quite complicated, non-standard, network shapes which are moved around a lot in the different phases of training and deployment. Apart from architecture, it is also possible to change the gradient descent method. Finn *et al.* [12] use MAML that essentially adapts weights on the training data, due to loss on test data. This allows training of the network to not be independent of future inputs. However, this method is complicated to employ on simple frameworks such as Darknet because of its outer-loop optimisation method.

Due to the constraints of the platform for this research it is also important to ensure that the computation is not too slow in both inference and training. One the first obvious methods is to lower resolution of input images. Peluso *et al.* [35] show the operation of the network at 32x32 resolution by using SGM supervisory signal which can be seen in Figure 3.3.

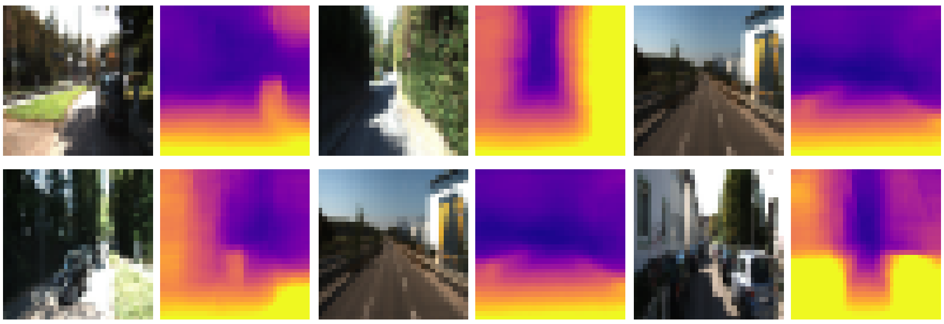

Figure 3.3: Examples of depth estimation on 32x32 image inputs

Changes to architecture are also powerful. Mobile Net [21] employ the use of Depthwise Separable Convolutions to speed up convolutions. The effect is quite powerful and is employed by Wofk *et al.* [54] for the purpose of depth estimation. However, the functionality to carry this out on a simple framework, such as Darknet, is not yet possible. A promising direction is to prune the network before it is deployed for online training as seen by Wofk *et al.*.

Quantisation is also a promising feature that changes the representation of weights to lower bit representations to decrease computation time and memory space. This method is powerful for inference, but remains very difficult to use for back-propagation. Wofk *et al.* achieve considerable performance increase using this method. For training time, it is recommended to use batch normalisation for faster convergence to a stable network. It is also possible to not train the entire network each iteration. Brock *et al.* [6] suggest progressively freezing layers of the network to stop to training them. These methods are useful, but may prove difficult to implement on a framework such as Darknet.

There must also be considerations for implementation. The hardware platform is self-contained on a StereoJevois camera. This has two cameras, CPU, GPU, USB and serial output. The stereo input can be in colour, or black and white, with the latter having a higher input rate. However, the input rate is not too important as it likely cannot be processed as fast. The coding for the learning method is possible in Darknet and Tensorflow, however, Darknet is preferred due to its simplicity in changing the source code to be compiled to binaries for the Jevois. For simulation the most promising candidate is Airsim as it has high visual fidelity, stereo camera and a simple API that makes prototyping and testing easier.

From the different areas analysed there were two clear gaps in the literature. First, it is clear that although there are a large number of depth estimation methods available, there is not too much work with regards to application of drones. There are some efforts in this area, however, either the methods do processing off-board [39] or they employ powerful hardware in constrained, obstacle-course, environments [58]. There has not been work to process fully on-board a lightweight drone for natural, indoor, environments. Secondly, as seen in the discussion of adaption, there has clearly been no work on online self-supervised learning of depth for a physical drone, in a real-world environment. Therefore, this thesis will address both the use, and online learning, of self-supervised depth estimation on a drone, in real environments.

This leads to the following research question:

- How can we carry out online self-supervised learning of depth estimation on a drone?
    1. Which depth prediction method is best suited to deployment on a drone for online domain adaption on low-power hardware?
        (a) Which method offers the best trade-off between inference speed, training speed and accuracy?
        (b) Which method is best suited to adapt online to a new domain the fastest and most robustly?
        (c) To what degree is the proposed method feasible for implementation on hardware with low computational power?
    2. How should the drone obtain new data such that it can adapt to the new domain as quickly and robustly as possible?
        (a) What information about the environment is most important to capture, to robustly adapt to the new domain?
        (b) At what rate should the drone obtain data for the learning process, in order for the drone to learn quickly and robustly?
        (c) How the should the drone be controlled dynamically in order to facilitate robust domain adaption?
        (d) How can the drone learn in a manner that prevents unlearning of previous domains?
    3. What is the overall performance of the proposed depth prediction method?
        (a) How well does the drone adapt in terms of accuracy, compared between different domains?
        (b) How fast does the drone reach a threshold of accuracy between different domains?
        (c) What is the inference time, training time and memory allocation of the proposed method on the drone?

To answer these questions, first some exploratory work of the methods will be conducted. The depth estimation methods of interest will be Godard's [15] with pose encoding [66]. This will be tested for different dynamic situations in simulation and then moved to the StereoJevois platform. This is due to its sequential structure, simpler for implementation, and the auto-encoders previous application on drones for estimation [58][40]. For adaption, of the methods discussed, the most promising for implementation are LSTM, as it provides temporal information for better scale estimation, and batch normalisation, for increased speed convergence of network. Finally, for increase of speed, resolution, offline pruning and quantisation will be explored. An outline of time planning is given in the Gantt chart in Figure A.2.

3.2. Background

In order to contextualise the discussed literature a brief overview is given of the basic concepts. This literature study focuses on both depth and learning methods. Therefore, in subsection 3.2.1 an overview of depth geometry is given followed by a description of the important common features of a CNN in subsection 3.2.2.

3.2.1. Stereo Depth Geometry

Depth estimation of a scene from a stereo pair is a well posed problem [64]. The essence of the problem is to find the corresponding projection of a point on both of the pair images. These are found on the epipolar plane that connects the point in the scene with two cameras. This can be seen in Figure 3.4.

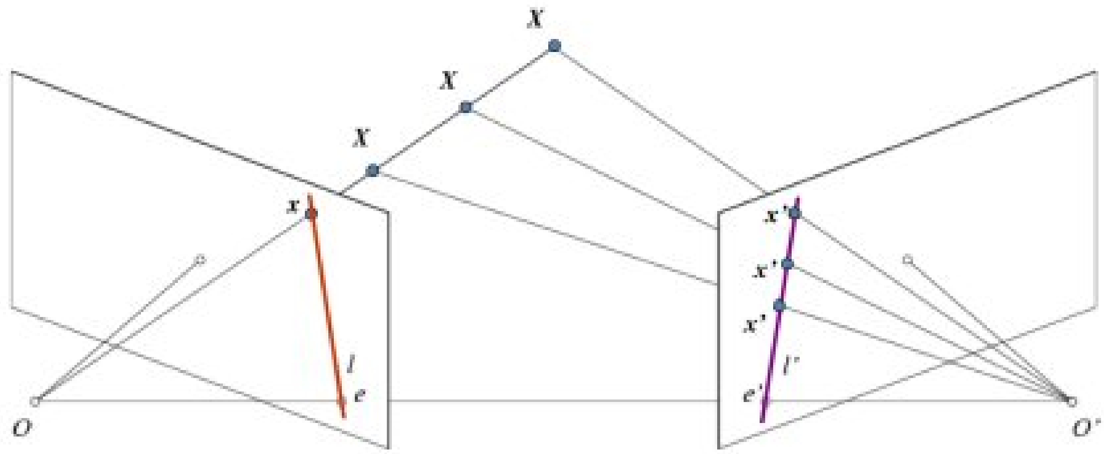

Figure 3.4: Epipolar Geometry [5]

Point **X** is projected onto point x on the left plane. From just the left plane image, the length of the vector **OX** is indeterminate. It can be solved by using information of the right plane, namely the projection of the point in space on to the right plane. The differing projection of **X** on to the right plane, **x'**, varies along the epiline **l'**. With the plane **OO'X** determined, the 3D position of point **X** can be given with the known translation between the two cameras.

In the case of rectified stereo vision, one image plane is only horizontally translated from the other without any rotation; the two images are coplanar. This makes every point's epiline horizontal, corresponding points need no vertical disparity. For two rectified images, the only problem is to find the horizontal pixel displacement between the points on the two images, also known as the disparity. This disparity can be translated to depth by use of Equation 3.1 where f is the focal length of the cameras and B is the horizontal distance between the cameras [64].

$$d = \frac{f * B}{D} \tag{3.1}$$

3.2.2. Convolutional Neural Networks

As this review centres on learning for depth estimation methods a brief overview of one of the most popular learning methods for image analysis, CNNs, is given. CNNs are essential adaptions of Artificial Neural Networks in order to handle machine learning concerning images. The architecture of a neural network consists of several layers; in general the most essential layer is the convolutional layer that contains the convolution, activation and pooling. There are also additional important features such as skip-architecture and upsampling. These features can be seen in Figure 3.5.

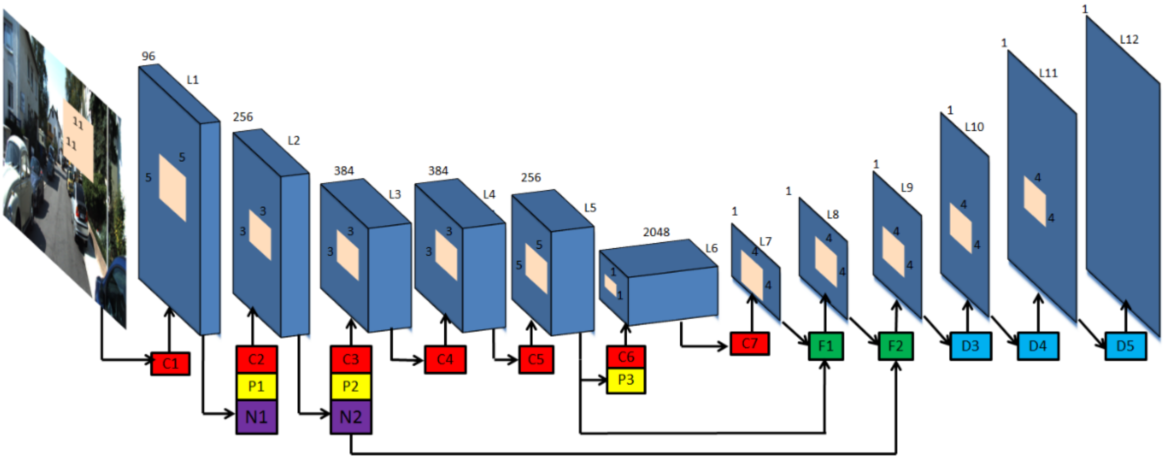

Figure 3.5: Autoencoder network architecture. The blocks C (red), P (yellow), L (purple), F (green), D (blue) correspond to convolution, pooling, batch normalisation, deconvolutions and upsampling layers respectively [13]

Convolutional Layer The convolutional layer is the essence of the CNN architecture. It can be seen by layers L1-L6 in Figure 3.5. The layer consists of three steps: convolution, activation and pooling. The first step is to perform a convolution operation over the input channels by matrix multiplication of kernels. Each convolutional layer has a specified number of kernels which determines the number of output channels of the layer.

The kernels are defined by a few parameters which include their dimension, padding and stride [17]. The dimension simply defines the height and width of the kernel as they are a square matrix. The padding defines how large the border of zero pixels is added onto the image in order to be able to pass the kernel over the perimeter values of the image. This allows for capturing more information from the edges of images. Finally, the stride defines the number of pixels traversed to perform the next convolution. Therefore, the learning of the network is done in the values of the kernels, demonstrating the ability to keep learned parameters low. The output dimensions, width and height, of the convolutions are given according to Equation 3.2. Where N is the image dimension, P is the padding, K is the kernel dimension and S is the stride.

$$N_{out} = \frac{N_{in} + 2P - K}{S} + 1 \tag{3.2}$$

After this, the output 'image' is passed through an activation function to transform the output values. These functions are similar to the those of a neural network such as sigmoid, tanh and ReLU. The functions are normally used to add non-linearity to the system, as well as, to clip values that are negative, to zero. Finally, each of the output channels of the convolutions can be optionally passed through a pooling layer. The pooling layer is a method of downsampling where the image is separated into rectangular sections that do not overlap and a non-linear function is performed to output a single value. The most common form of down-sampling uses max-pooling which just outputs the maximum value of the section. There are also linear functions such as averaging in pooling.

These convolutions in the end are used to reduce the information in the image to minimal representations that retain important features from the original input. The output of the convolutional layers can be then passed onto different layers such as a fully connected neural network or upsampled back to the original resolution.

Upsampling The convolutions are used to reduce information, however, that compressed information can also be upsampled to output higher resolutions images than the convolutional final output. This can be done by means of unpooling or deconvolution, also known as transposed convolution. Unpooling uses non-learning methods such as max-unpooling or nearest neighbour to fill in or interpolate unknown space. However, deconvolution uses the mechanism of convolution with larger padding and stride to pass kernel filters that increase the resolution of the image. This allows for learnable upsampling to recover an image from an encoded output of the convolutional layers.

Skip Architecture Skip architecture allows for the connection of convolutional layer outputs to the upsampling layers which has two main benefits. The first, is that it allows for losses to be better back-propagated through the network. Secondly, it allows for the upsampling layer to retain details from the earlier convolutional layers for constructing an output image [13].

3.3. Depth Estimation Methods

There are a plethora of existing depth estimation techniques that use a variety of techniques and a variety of input types. In this exploration of existing depth techniques, the methods are first delineated by their type of input for estimating depth, and then further divided by the technique to train the method. First, a review of methods that have an input of stereo-pairs is given in subsection 3.3.1. This is followed by methods that have an input of a single image (monocular) in subsection 3.3.2. A fusion of the two input types is discussed in subsection 3.3.3. After an exploration of the methods, an overall comparison of the different methods is given in subsection 3.3.4. Finally, there is a discussion of the depth networks that have been used for drones in subsection 3.3.5 and a brief conclusion to the chapter is given in subsection 3.3.6. A general diagramatic overview of the architecture of the most relevant methods discussed, and how they are grouped, can be found in Figure A.1.

3.3.1. Stereo Depth Estimation

The first depth prediction methods consisted of having dual stereo pair images. As explained with the basics in subsection 3.2.1, points in 3D space are triangulated by using the position of the point on both right and left projection planes. There has been extensive research and development into classical optimisation based methods and newer, learning based methods. The general structure of stereo depth estimation comprises of 4 steps: matching cost computations, cost aggregation, disparity computation/optimisation and disparity refinement [46].

Classical Classical stereo algorithms can be split into three main categories: local, global and semi-global methods. These define the level at which comparisons of matches of pixels are made for estimating disparities. In general, local methods depend on information of neighbouring pixels which is fast in computation but can be misled by discontinuities, whilst global methods attempt to optimise over all disparities which is more accurate but more computationally expensive [4]. Semi-global attempts to leverage the advantages of both methods. This was shown by Bebeşelea-Sterp *et al.* [4] in a comparative study where graph-cut optimisation, a global method, outperformed other methods in accuracy but required a much higher computational cost. However, Sum of Squared Differences (SSD) was shown to be less accurate and more sparse, but led to much faster computational time.

An extremely popular semi-global method is Semi-Global Matching (SGM), which uses mutual-information based matching cost method [19]. This method, and other classic methods, boast the ability to generalise on different settings, however, do suffer from large amounts of memory use [20]. It has a simplified implementation in OpenCV, known as Semi-Global Block Matching (SGBM), that reduces memory usage and computation time using block matching [5]. An example of disparities is shown in Figure 3.6. It can be seen that even SGBM still results in a slightly sparse disparity map. It also leaves the edge of image without a depth estimation as the images are offset. One thing that stereo vision algorithms don't leverage are contextual cues of images which can be better trained over machine learning processes [45].

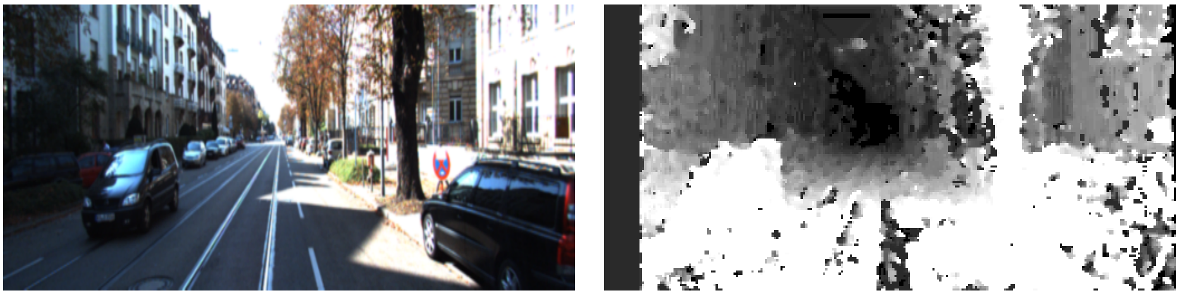

Figure 3.6: Demonstration of the output of SGBM. Input to the algorithm from KITTI [14] (left) Disparity output from SGBM algorithm (right)

Learning Based As opposed to more classical methods there is also the possibility for learning based methods on stereo inputs. Normally, these methods subsitute some, or all, of the earlier described 4 steps of stereo matching. For example, Zhong *et al.* [67], substitutes only the feature extraction and disparity computation with 4 neural networks, 2 for each image. This method in particular allows for self-supervised learning as the disparities are calculated for both the left and right images, and allows for warping each opposite image to compare to the actual image pair for loss calculations. However, it is clear that the complicated architecture, which can be seen in Figure A.1, is quite heavy with 4 networks and non-linear functions and weight sharing that make backpropogation in an online context quite computationally demanding and requiring a complicated framework. A similar architecture is used for Activestereonet [62], however in this approach a camera with an active IR depth sensor is used providing ground truth depth labels. Another difference with this network is that because a second disparity map is not required, a confidence map is generated instead from the right image. This is quite useful in attempting to describe the quality of the estimation in a mobile robotic application.

However, there are methods that simply use a completely end-to-end network to estimate depth like Yang *et al.* [55] and Liang *et al.* [28]. In this paper two auto-encoders are used to produce features from the two images and they are passed through a 3D decoder network to produce the final disparity maps. The final decoder has several stages, allowing for early stopping in consideration of computational time. It is shown that the network can stop earlier to retain faster processing than SGM, whilst maintaining higher accuracy. Although, this network is running on a high-end GPU as opposed to SGM which runs on the CPU. However, for a network architecture it is quite heavy employing two auto-encoders and one 3D decoder. It boasts its performance specifically for high-resolution images, which is perhaps out of the scope of the low resolution inputs of small cameras on a dynamic vehicle. A further trade-off of the performance of these networks is given in subsection 3.3.4.

3.3.2. Monocular Depth Estimation

Although inferring depth from a single image is an ill-posed problem, it allows for dense inference of depth [64] by leveraging contextual information in a learning manner. A powerful way to do this is to learn contextual clues in the image through means of machine learning. Since the publication of one of the first methods in 2005 by Saxena *et al.* [45], there are now a large amount of existing methods. To simplify the collection of papers they are separated by their training methods; namely, they are separated into training with ground truth depth labels, stereo image pairs and sequences of monocular images.

Ground Truth Training At the beginning of the development of monocular depth estimation methods it was standard to use ground truth depth as a supervisory signal. These were collected by means of secondary sensors such as LiDAR. The first paper in this field was written by Saxena *et al.* [45] using Markov Random Fields. Different image features are found at local and global scales and also sampled at different resolutions. With these features a probabilistic model is trained on parameters per row of the image. Although this functions relatively well for the images given, it requires images to be of similar shape, such as rotational angle [45] [9]. Therefore, any rotated images, which are extremely common for dynamic vehicles, will probably perform quite poorly using this technique. It also requires a quadratic optimisation routine to be run on forward inference. A quite impressive fact is that the method only needs to train on 319 images.

The next big step came quite a bit later, in 2014, from Eigen *et al.* [9] using two CNNs: one for producing a coarse, but dense, depth map and the second for refining the coarse estimation. This paper was made possible, in part, to the KITTI dataset [14] released in 2013 that comprised of a large collection of images and depth maps from a LiDAR sensor. Of course at the time this performed very well, but in light of new methods this paper does not perform as well anymore in terms of accuracy [64]. Since 2014 there have been many papers in the same vein making changes to the structure of the network such as adversarial networks [24] and multi-task networks [8]. The use of multi-task networks are quite powerful for example in Chen *et al.*'s paper [8] where semantic segmentation is done in a parallel network. This is useful as the depth over an object is quite smooth, whilst the transition in depth to another object nearby in a scene is harsher. However, the drawback of these more complicated networks is that they have higher inference time, take up more memory, have longer training times and in the case of multi-task networks require more collected data.

There is also an interesting adaption of stereo matching to monocular inputs for a network developed by Luo *et al.* [29]. In this architecture a view synthesis network first produces a synthesis right image from a left image input. Then the input left image and the synthesis image go through a stereo matching network to produce the depth. This method is powerful in terms of accuracy, but is not computationally efficient.

When training with ground truth labels it is possible to use data augmentation [9] and synthetic images [65] to train for better domain generalisation. However, overall the use of ground truth label is often expensive and restricted to certain domains.

Stereo Input Training Due to the restrictiveness of requiring ground truth of depth labels with supervised learning, Garg *et al.* [13] proposed a novel loss method to allow for self-supervised learning. The crux of this method is to use the monocular input's predicted disparity map to warp the stereo pair to a copy of the original monocular input and calculate the loss from that. The general method can be seen in Figure 3.7 for more clarity. This network employs an auto-encoder with skip architecture in order to propagate forward finer details in the convolutional layers to the layers in upsampling layer.

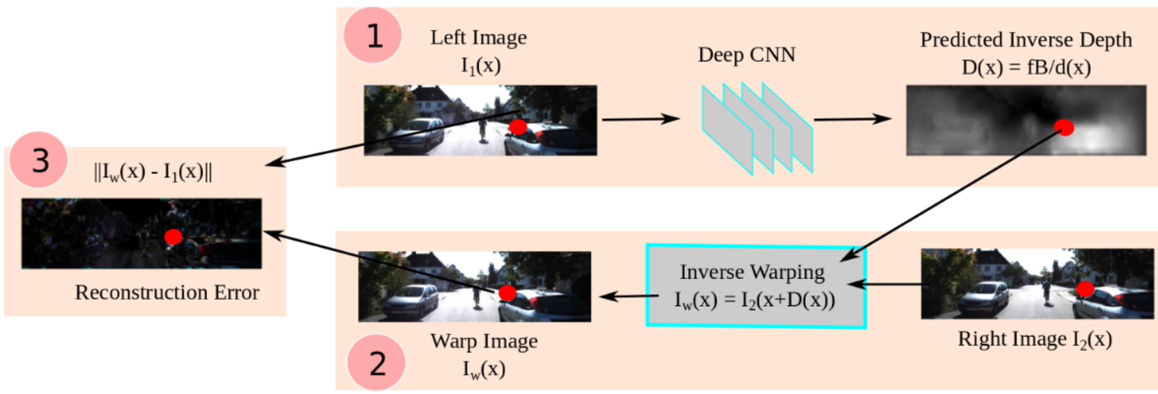

Figure 3.7: Self-supervised training architecture [13]. Predicted inverse depth is the same as image disparity

However, a pitfall of this paper is that the loss term for the inverse warping is not fully differentiable and a linearisation with Taylor Series must be used [15]. The use of a Taylor Series also requires that the previous disparity estimation must be held in memory. Godard *et al.* [15] also claim that Garg *et al.*'s method to only minimise photometric loss results in good image reconstruction but poor depth estimation [15]. However, this method still employs one of the most simple architectures available.

The next large advance in this field came from Godard *et al.* [15] who introduced a novel loss term with left-right consistency between the predicted disparity maps. Instead of just warping one image with a single disparity map, two disparity maps are generated and then the two images in the stereo pair are warped to be compared for the loss. This method significantly increases accuracy, as seen in subsection 3.3.4, and does not significantly increase the time for inference. However, it does increase the size of the output and the loss calculation as the loss is calculated for both disparity maps, 2 warped images and at 4 different resolutions. Chen *et al.* propose a slight modification to the network setup to only output one disparity and instead flip the right image and then flip the output disparity to produce the second disparity map [8]. In this case both disparity maps originate from their actual respective scene and the second view does not have to be indirectly synthesised. The network architecture and training method for Godard *et al.*'s work can be seen in Figure A.1.

Following Godard *et al.*'s innovation on the loss function, more networks were developed that innovated more on network structure. For example, many methods attempted to chase higher accuracy using more intricate networks such as cycled GAN networks [37] or knowledge distillation [38]. These architectures use multiple networks to train a single network to deploy. Whilst it is beneficial in the trade-off between inference time and accuracy, it is severely impaired in computations for online training requiring many additional networks.

A very promising paper developed by Poggi *et al.* [42] shows a novel structure, PyD-Net, which draws inspiration from the classical computer vision technique of using a pyramid of features. Essentially, the network is broken into tiers of resolution, until the lowest tier, 1/64 of the original resolution, to calculate the disparity at each level. This disparity is then send to upper layers in the pyramid to resolve disparities at higher detail. The structure of the network can be seen in Figure A.1. The purpose of sampling at these lower resolutions is that often certain, important, contextual depth clues are retained at lower resolutions [45]. This architecture was trained in a similar manner to Godard *et al.* left-right consistency loss form. However, in this case the loss is back propagated from each individual layer of the pyramid rather than once sequentially through the whole network. The novel aspect of this network is the increase of the speed in inference compared to other methods such as Goddard *et al.* [15] with similar accuracy. At similar resolutions PyD-Net runs up to 3 times faster on the GPU and 10 times faster on the CPU than Goddard's.

Generally, using stereo pairs for training allows for a elegant self-supervised learning of depth that maintains a high accuracy. This is especially attractive in the case for online self-supervised learning. However, a gap in these proposed papers is that they are not implemented in live conditions and only train on manicured datasets such as KITTI [14].

Monocular Sequence Training Sometimes it may be the case that even less information is available to the system. For example, the only data that is collected could be a stream of monocular images, with no stereo pairs captured. In this case novel methods are required. The first of these came from Zhou *et al.* [68] in 2017, where a network with monocular input depth prediction was trained on a monocular image stream. Essentially, a depth estimation network runs in parallel to a network that estimates the change in camera pose between the neighbouring frames. The output of both networks are used to estimate the target view that can be used to calculate the loss of both networks. The structure of this network can be seen in Figure 3.8.

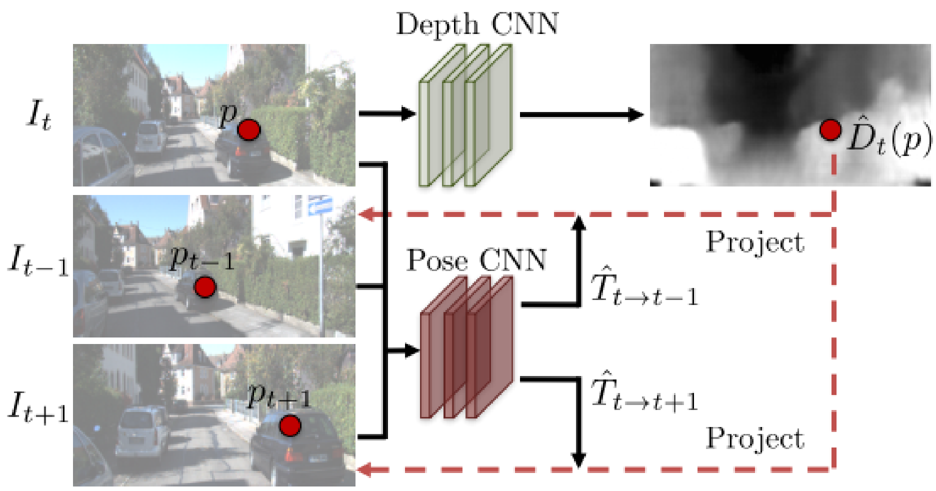

Figure 3.8: Depth and EgoMotion Network architecture [68]

This method allows for self-supervised training with less input data while retaining accuracy. However, this method, and others that perform on monocular streams, perform worse than those trained on stereo pairs and ground truth labels as seen in subsection 3.3.4. Another aspect to consider is that this network takes in quite a manicured data-set that always maintains horizontal controlled translations between frames. This work has not been tested on its application to more dynamic monocular sequences like that of a drone where there may be severe vertical translation or rotation.

Following this method there have been several papers proposing similar architectures to leverage monocular sequences. The following year, in 2018, Yin *et al.* [60] released a paper that attempted to leverage Optic Flow as an additional quantity estimated by a separate network. This resulted in a definite increase in accuracy [64] but at the loss of performance speed in back-propagation. However, for use on a dynamic vehicle the addition of an optical flow network may also prove beneficial for its tasks.

Godard *et al.* released another paper in 2019 [16] focused on training with monocular sequences. They proposed changes to the loss function in order to better handle occlusions, visual artefacts and handle pixels that violate the camera assumptions. Occlusions are a large issue in monocular sequences because as objects move in time previously unseen objects in a scene may become more clear. With these changes not too large of a change in computational speed is needed, whilst a large improvement in accuracy is gained within the category [64].

Because of the mix of this field with application in Visual Odometry there is quite a bit of development from that angle as well. For example, Yang *et al.* [56], employs a novel uncertainty estimation network in their D3VO pipeline alongside estimating pose and depth. The estimated uncertainty is useful as it is used on the back-end to optimise the estimated pose transformation, alongside the depth. Although, this architecture is developed for use by drones in mapping unknown environments the entire system is trained on the KITTI architecture as well and performance hasn't been shown on a deployed drone.

Overall, networks that allow for training monocular sequences are great for more constrained camera hardware. However, in terms of computational load, the back-propagation is much more demanding than the stereo training networks. Also, more information must be held in memory as images must be stored over time. Though, providing temporal information allows for other useful parameters such as pose estimation and optical flow to be leveraged.

3.3.3. Fusion Based Methods

Few papers have also found the benefit of fusing different types of methods together, both in the inference and in training.

Facil *et al.* [10] estimates a dense depth map using a CNN and a sparse depth map using multi-view geometric methods. The multi-view methods excels on highly textured and high parallax cases, whilst the single view CNN performs better on texture-less objects and object structures. The fusion of both is done by means of an interpolation based on pixel weights that leads to a better performance than either of the methods separately. Weights are calculated based on the likelihood for each pixel from the multi-view output to relate to the pixel from the single-view output. It must be noted that the CNN is tested on the same scenes that it is trained which means good performance is expected [31].

Martins *et al.* [31] take a different route in that SGM is used as a supervisory signal to train a monocular depth estimator network in a self-supervised way. This is done offline before deployment on a drone. When deployed, the SGM estimation is fused together with the dense monocular estimation network. Fusion works mainly through a confidence map generated by a vertical Sobel filter to detect edges that indicate where SGM works better as well as a weight dependent on the ratio of the depths of the two estimates. Although the network does not train online, it operates in new environments and proves to be effective in producing estimates with better results than the individual methods. As discussed in subsection 3.3.1, SGM can be made faster and more memory friendly by implementation of the OpenCV SGBM module which prevents this method from being too slowed down. One qualm with the method is that the monocular input network learns the deficiencies of the chosen stereo geometry method [64]. An overview of the network can be seen in Figure A.1 but can also be seen from the author in Figure 3.9. In Figure 3.9 it is clear that the estimation of depth through the CNN enhances the thin detail of the thin table support whilst the SGM network can estimate the larger distances more accurately to ground truth.

A different approach to these is to fuse depth estimates through a network. This is done indirectly in some networks such as Eigen *et al.* [9] where the networks are multi-scale and more coarse depth estimate are refined with a secondary network. However, a more direct example, where different methods of depth estimation are fused in a learnable way, was devised by Ferrera *et al.* [11]. In this approach the

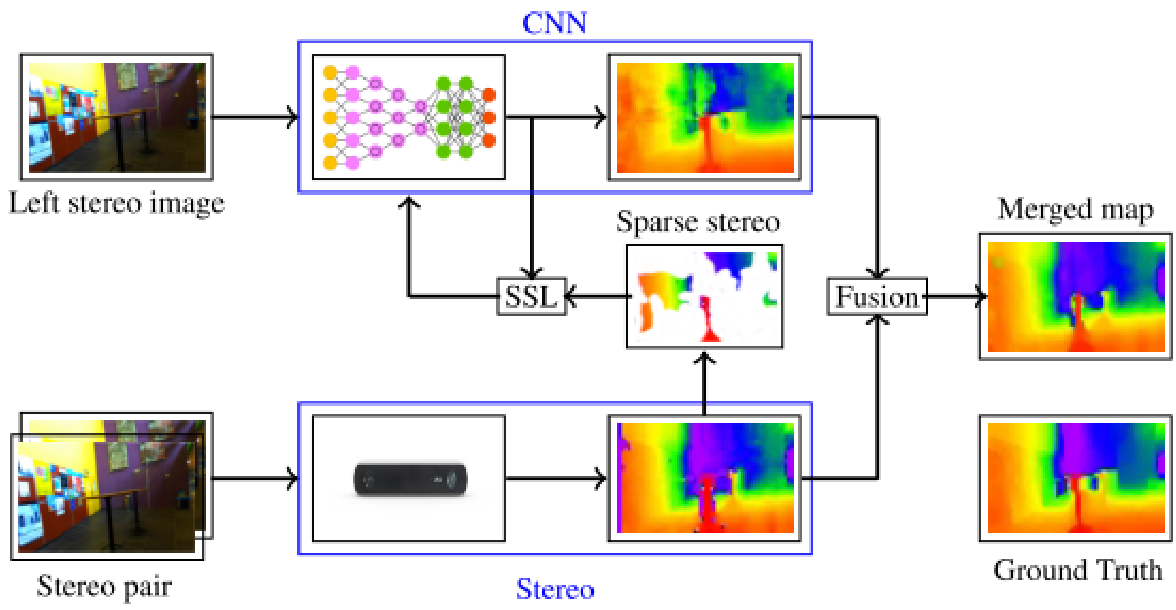

Figure 3.9: Overview of SGM/monocular depth estimation fusion architecture [31]

refinement network has multiple inputs: it takes in the stereo image pair, the disparity from a geometric method and the disparity from a CNN based branch and outputs a final disparity map using a trainable network. This method does not use full back-propagation through all networks, only the final fusion network. Adding a network, this method is adds significant computation time in through-put, however this method allows for learning on the fusion. A diagram of the network can be seen in Figure A.1.

A more indirect approach of fusion is to use other depth predictions in the learning processes. For example, Watson *et al.* [53] use a supervisory SGM signal in the loss function to improve depth estimations. When the loss of the estimation is larger than the loss for the SGM method, then an additional loss term is added based on the loss of the SGM prediction. This way offers a method to involve better depth predictions without enforcing the network to learn the secondary depth prediction method. The improvement to Godard [15] can be seen in the accuracy metric in Table 3.1. The change this method incorporates in the learning process can be seen in Figure A.1.

Overall, the fusion of different depth prediction methods allows for an improvement of depth predictions. Although it may be slightly more computationally intensive, if done in a lightweight manner, can improve the prediction substantially, at an acceptable increase in computational effort.

3.3.4. Comparison of Depth Estimation Methods

It is essential to understand which class of methods are better suited for online learning of depth estimation therefore they must be compared on both accuracy and computational speed. Conducting a trade-off of these various methods can be quite complicated as at times they are not directly comparable, especially because they do not use the exact same hardware and image format constraints for analysis. The papers traded-off, as seen in Table 3.1, are chosen for their ability to perform self-supervised learning.

Accuracy For analysing accuracy of the networks there are a common few metrics employed as listed below [64], where d_i is predicted depth, d_i^* is depth ground truth and N is the total number of pixels. In this comparison the absolute relative error is used as it provides the most geometrically understandable metric. Although this is chosen, there is not too large a difference in performance between the different metrics. It should be noted that not all the values found for the different papers are based on the same dataset, therefore the comparison is not completely accurate.

$$RMSE = \sqrt{\frac{1}{|N|}\sum_{i\in N}\left\|d_i - d_i^*\right\|^2}$$

$$RMSElog = \sqrt{\frac{1}{|N|}\sum_{i\in N}\left\|log\left(d_i\right) - log\left(d_i^*\right)\right\|^2}$$

$$AbsRel = \frac{1}{|N|}\sum_{i\in N}\frac{\left|d_i - d_i^*\right|}{d_i^*}$$

$$SqRel = \frac{1}{|N|}\sum_{i\in N}\frac{\left\|d_i - d_i^*\right\|^2}{d_i^*}$$

Inference Speed For comparing the computation speed for inference and back-propagation the method is even more convoluted, unfortunately. Most methods do not give the actual inference time, as in this field a large focus is placed on accuracy instead. Therefore, only qualitative scores can be given, on a range from 1 to 5 (1 being the best), based on the network architecture and loss method. From an observation of the networks almost all the papers receive the same score on inference speed. This is because regardless of the training method, and the networks involved, when deployed they all take the form of an encoder-decoder network. However, Poggi *et al.* [42] are the exception that manage to deploy their network much faster than the others due to network architecture and the method of compilation on hardware. It should be noted that there is the possibility to prune the network to improve the computational speed as the cost of accuracy.

Training Speed However, for training speed the results are more varied. The baseline is set at Garg's [13] paper for the fastest training as the loss is the simplest and there is only a single network to be trained. At the next level, 2, are the papers that employ more complicated loss functions. For example, Godard *et al.* [15] added the consistency term and require two outputs of the network, which is adopted by Poggi *et al.* [42] as well. Watson *et al.* are rated as three because they employ the more complicated loss term as well as having the supervisory SGM signal. Martins *et al.* fall into level 2 as they calculate the supervisory signal by another algorithm, namely SGBM.

All the networks that train using mono sequences all perform worse on training speed as they require additional networks to estimate pose, must train both networks and require storing more images in memory, placing them at level 3. Yin *et al.* [60] perform worse than their counterparts in this ordeal as they also train an optical flow network. Finally, the worst performing in training speed are those designed by Pilzer *et al.* [37] [38]. This is because they use GAN networks and knowledge distillation to compress all the learning into a single network. This may be efficient to achieve higher accuracy with low speed on inference, but takes a lot of computational effort to train with up to 4 networks to be trained in the case of knowledge distillation.

Paper	Keywords	Abs Rel	Inference Speed	Training Speed
Garg [13]	Stereo Training	0.169	2	1
Godard [15]	Stereo Training	0.148	2	2
Godard [16] (Stereo)	Stereo Training	0.127	2	2
Watson [53]	Stereo Training	0.112	2	3
Poggi [42]	Stereo Training	0.148	1	2
Pilzer [37]	Stereo Training	0.152	2	5
Pilzer [38]	Stereo Training	0.098	2	5
Zhou [68]	Mono Sequence	0.208	2	3
Yin [60]	Mono Sequence	0.155	2	4
Godard (M) [16]	Mono Sequence	0.132	2	3
Martins [31]	Fusion	0.200	2	2

Table 3.1: Comparison of relevant depth prediction methods for accuracy (abs relative difference) and inference and training speed (rated from 1 to 5)

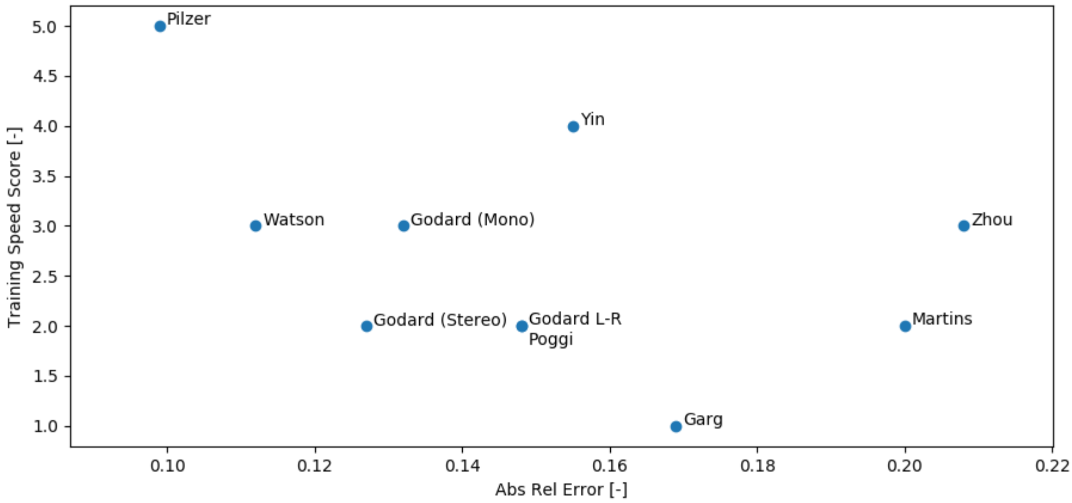

Figure 3.10: Scatter plot of depth estimation methods accuracy and training speed

From Table 3.1 and Figure 3.10 it is clear that on average, networks using stereo pairs for training can maintain better accuracy whilst maintaining low computational effort on training. It seems that although the knowledge distillation method developed by Pilzer is the most accurate its terrible training speed makes it infeasible for online learning. If accuracy is weighted less than the computational effort then the best methods are between Garg's and Poggi's. In terms of complexity Garg's architecture is much simpler to employ due to its sequential structure whilst Poggi has a convoluted method for back-propagation. Godard's stereo method [16] does also fair quite well, especially with consideration of possiblity to prune the network. A fusion like the one devised the paper by Martin *et al.* also seems quite feasible to employ online.

3.3.5. Implementations for Drones

There have also been efforts to implement these methods on drones. This research shifts away from simple estimation of depth and considers the artefacts and effects of estimating depth on a vehicle with more exotic motion. This consideration is shown to be important in a paper by van Dijk *et al.* [52]. It is shown that for different rotations of an image, the various algorithms explored previously perform worse.

One of the largest issues of implementation on a dynamic vehicle like a drone is the issue of rotation and constantly changing pose. This is addressed by Pinard *et al.* [40] by encoding the change of pose in a video stream into the network training, which can be seen in Figure 3.11.

A stream of videos is obtained and two images are fed into the DepthNet network: the target frame and the warped reference frame, corrected for the rotation from target frame. The pose is estimated

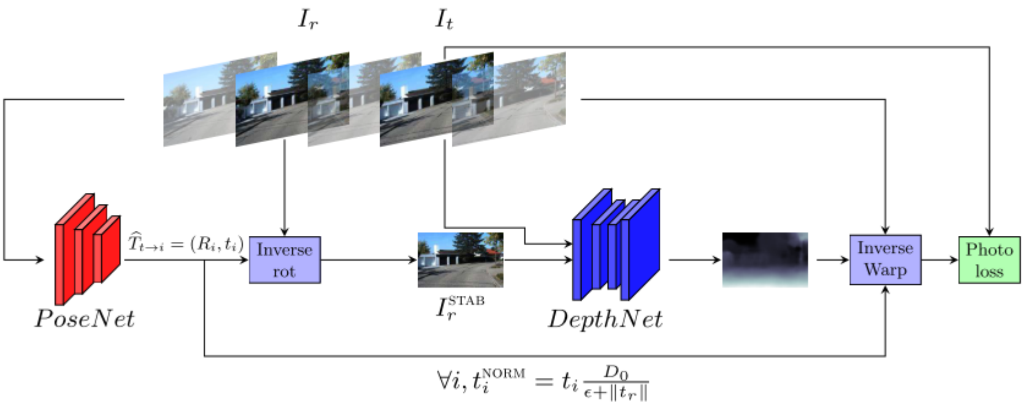

Figure 3.11: Training structure for stablised DepthNet [40]

from the PoseNet network [68]. In the inference stage PoseNet is not required as the stabilisation can be done by the drone, or with inertial sensors. The issue with using the motion the encode information on the pose of the drone is that it requires that the drone is moving to estimate depth. However, despite this flaw, the network performs quite well in estimating depth maps for rotation, even managing to compute depth for a flipped image, which is impossible for naive networks [68]. Though, this comes with a flaw that for the non-flipped images, the estimation is slightly worse than the naive, non-robust, prior solutions. Another issue with this method is that the training method is quite heavy as it requires calculations of loss on multiple frames and the additional PoseNet. Also, for inference on a live drone, this method streamed video to a laptop with a GPU equipped, instead of running on board [39].

A more lightweight alternative to this is implemented by Zhao *et al.* [66]. They propose a much simpler solution that encodes the drones vertical position and pitch into a 4th channel input, concatenated with 3 channel RGB monocular input. This requires additional sensors which are normally available in the form of IMU and Gyro for drone applications. The encoding is done by estimating the depth of each image row in relation to the fixed height of the room (that has an infinitely long floor and no walls). This can be seen more clearly in Figure 3.12.

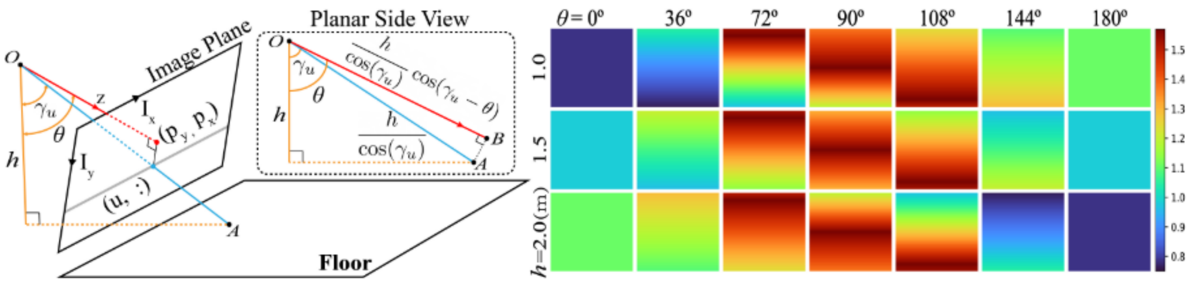

Figure 3.12: Encoding to depth for camera pose (height and pitch). u is row in image, (p_x, p_y) are principle point of image, h is camera height, θ is pitch, z is camera view axis. [66]

The depth of pixels in each image row is the projection of the distance to the floor $_{OA}$, or ceiling, projected onto the camera view axis, z. Because of issues of infinity at 90 degrees, row depth is passed through inverse tan function (monotonically increasing) that maps all positive values of depth to the interval $\left\{0, \frac{\pi}{2}\right\}$. One could also naively add two channels to the input, one with the height value and one with pitch. However, that leads to lower performance than this geometrically motivated method [66].

The authors report that using this camera pose encoding leads to much better performance in more exotic poses and similar performance in more common poses (angled on the horizon and at mid-height) when compared to methods that don't consider pose. This is even the case with noise on the pose estimation. It also outperforms a method that estimates surface normals in training [59]. This method uses ground truth to train the network, however, it seems that it should not be too difficult to employ the same strategy with an aforementioned self-supervised method as only an input channel is added. Two apparent geometric drawbacks are that this method assumes a ceiling height, which could be misleading for outdoor scenes, and it does not account for roll motion. However, these issues can be addressed to make the solution more viable for the motion of drones in indoor and outdoor settings. Unfortunately, this method is not employed yet on an actual flying drone 'in the wild'. Yet it does propose a simple, yet powerful, method to be more robust to drone motion.

Niu *et al.* [34] compare the implementation of autoencoders [13] and multi-scale deep networks [9] for use on robotic swarms in a forest, specifically with a focus on motion blur (simulated). It is shown that for both methods depth accuracy drops with increased simulated gaussian and horizontal blur. The effect of blur is more apparent on the autoencoders compared to multi-scale deep networks, however, by not too large an amount. Also, it is shown that the autoencoders perform much better in run-time with a minimal drop in accuracy.

Finally, Yang *et al.* [58] demonstrates an obstacle avoidance strategy that uses on board estimation of depth and the confidence of that estimation. The network for the estimation has a similar form to that of Garg [13], however, it also takes in a sparse depth map as input. It is possible for the sparse depth map to come from secondary sensors but the authors use the ORB-SLAM algorithm [33] in parallel to derive this sparse map, which only requires monocular images and pose information. The network is trained offline on ground-truth depth maps.

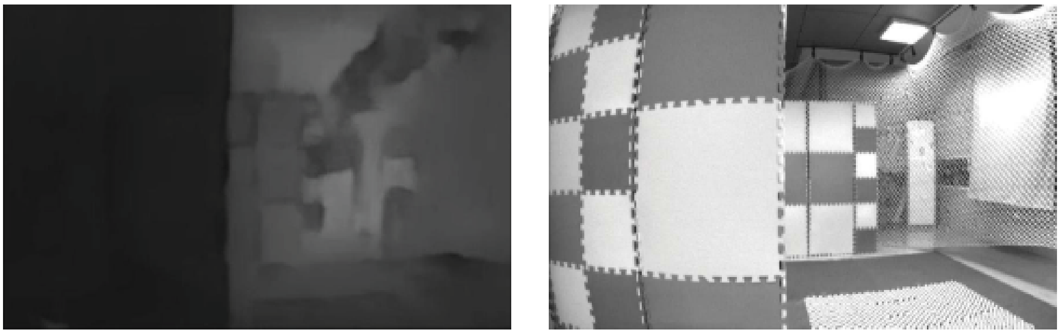

Figure 3.13: Drone footage of obstacle avoidance with depth maps and confidence in indoor real environment. Depth map (left) raw images (right) [58]

A great aspect of this paper is that the authors also tested the performance on the drone in a real environment as seen in Figure 3.13. They use a NVIDIA Jetson TX2 running in the ROS (Robotic Operating System) framework, on a 0.6 m diameter drone and managed to achieve inference at 12 FPS. They manage to fly through a short obstacle course with a 100% success rate, but no data is given on depth prediction accuracy and it seems a low velocity is used causing very little rotation. However, it is important to note that the network was not trained on images from this real environment, yet it could manage to navigate. Although the hardware is quite powerful for this application and the environment quite constrained, the paper does indicate that the depth estimation can be used on a live drone in a controlled setting.

3.3.6. Conclusion

It is clear that there are a large range of different methods available for predicting depth, especially as many had to be left out in this review. With the given comparison a simple trade-off on speed and accuracy was detailed between the different frameworks. As well as this, an examination of current drone methods some examples of application on drones was also given. However, what is still lacking, is proof of application on-board a MAV with low computational hardware in more exotic, natural, environments. Although, there have been efforts to compensate for the effects of drone motion on camera input. Generally, it seems that there is room to explore the performance of depth estimation on a larger range of motion and environments with a live, on-board, platform. From this collection of methods it seems that most promising direction is to pursue simpler architectures, as identified in subsection 3.3.4, such as Garg *et al.* [13] and Godard *et al.* [16]. Although Poggi performs well on inference time, its more complicated network structure, will provide a large barrier. As well as this, the input of pose encoding, discussed in subsection 3.3.5, will also be explored in the application of this project.

3.4. Domain Adaption

All of the papers discussed in section 3.3 are trained offline on well manicured data sets. Although they boost good performance on the same dataset they are trained on, they are not as robust to changes in domain, such as real world applications on drones [51]. One method that is normally employed, is to simply train the pre-trained model on a small subset of training images of the new domain. However, this is impractical and may take more resources to collect images for supervisory learning models. This chapter addresses the methods by which networks can adapt to new domains to maintain performance. Specifically, subsection 3.4.1 delves into techniques for adaption in the field of depth estimation, followed by subsection 3.4.2 addressing general adaption techniques outside of the field, and finally the chapter ends with subsection 3.4.3 for a brief conclusion.

3.4.1. Depth Adaption

Although there exists a large body of work for estimating depth using learning based methods, there are considerably fewer works on the adaption of the networks to different domains. The few methods that do exist rely on different methods that are organised into three here. The most common is using synthetic data sets, followed by training with confidence measures in the loss function and finally by designing novel architectures for effective training and preventing network forgetfulness.

Synthetic It is quite common in the literature for networks to use synthetic data sets to attempt better generalisation on different domains [64] [63] [30]. Atapour-Abarghouei *et al.* note that this still does not practically generalise a network well to many different domains [2]. Their paper proposes to minimise the difference between the feature distributions of the source and target domains. This is achieved with two steps: training a depth model over synthetic data and training another model to perform style transfer of images from synthetic to the real-world. The style transfer network is trained using a GAN configuration, leveraging adversarial training and cycle-consistency. However, the authors identified it is very difficult for style transfer to accommodate large changes in lighting and saturation. Also, more importantly it is evident that style transfer is for the same class of domain, for example traffic in the synthetic and real domain. This means that a completely new setting, that does not exist in either the synthetic or real-world data set, will still remain difficult.

Confidence Measures An interesting approach is to learn online using confidence measures. This was proposed by Tonioni *et al.* in 2017 [49] for deep stereo networks and expanded to include monocular networks in 2020 [50]. These papers essentially impose an additional term in the loss function so that they consider smoothing, image reconstruction and confidence guided loss. The confidence guided loss is based on the difference between the disparity estimated by the network and the sparse disparity estimated by a stereo algorithm, such as AD-CENSUS [61] or SGM [20]. This difference is weighted by a confidence mask. Confidence measures are used to evaluate how good the stereo estimated depth guiding the loss is. This reduces the effect of noisy stereo estimates leading to incorrect loss. Evidently, this method requires the use of a stereo camera to provide a secondary sparse estimate to guide online learning. The confidence measure used is found from Poggi *et al.* [41] paper on trading-off confidence measures for efficiency on embedded systems. This estimation of a confidence measure is guided by a hyperparameter, τ. In the 2017 paper [49] this parameter is handtuned to optimal performance, however, in the 2020 iteration of the study [50] the hyperparameter is learned through gradient descent during the learning procedure. The effects of the adaption process can be seen on Figure 3.14.

From the image, and the absolute relative error in disparity, it is clear that the adaption scheme is extremely useful for improving estimates of the network. The paper in question uses this method offline with collected stereo pairs to improve the estimates of different networks. However, if there is access to a stereo camera this should be feasible online. The only obstacle is the increased time to calculate loss by means of calculating the stereo algorithm output and estimating the confidence measure. A notable point is that the authors of the paper do not collect stereo images from the 'wild' but rather

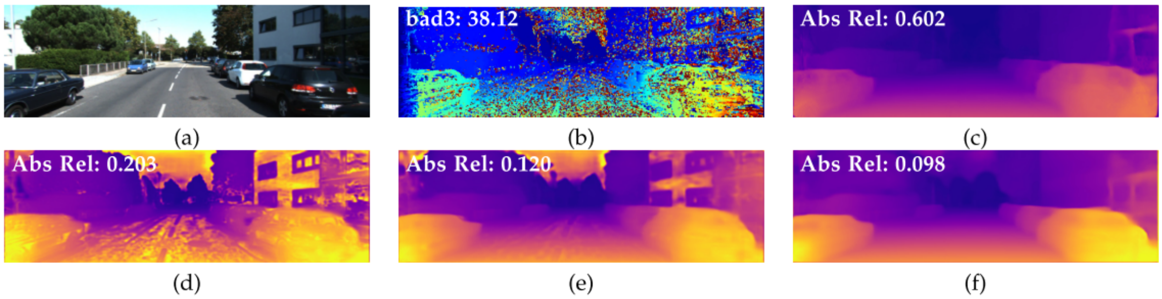

Figure 3.14: Adaption of monocular depth estimation network (Network originally trained on Cityscapes). a) input image from KITTI b) Disparity estimation from AD-CENSUS c) disparity output of monocular network d) adaption with stereo estimate only e) adaption with confidence mask f) full adaption with hyperparameter optimisation. Image taken from [50]

use the stereo images from the manicured KITTI dataset [14]. In the context of more dynamic inputs, such as rotations of images or motion blur, this adaption scheme has not been evaluated. This raises concerns on its applicability to low resolution and dynamic inputs on real-world applications.

Architecture Changes The last class of adaption scheme discussed for depth networks is employing changes to the architecture of the networks. An interesting method for managing adaption on different domains is to employ Long Short Term Memory (LSTM) layers to avoid forgetfulness of a network. LSTM layers are functions of both the current input as well as previous outputs. For example, Mancini *et al.* [30] employ two LSTM layers between the encoder and decoder as can be seen in Figure 3.15. In this case the authors attempt to leverage the LSTM network in order to refine the features extracted by the encoders. The network is trained on synthetic datasets that produce exact depth values.

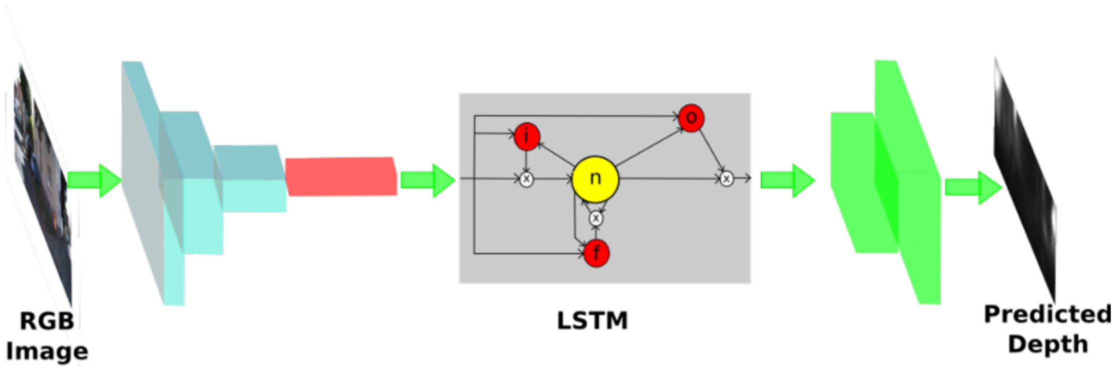

Figure 3.15: LSTM Network with two LSTM layers with 180 neurons between encoder and decoder [30]

Looking at the performance of the network the authors claim that the when both this novel network and Eigen *et al.*'s network [9] are trained on the same dataset, when transferred to a dataset on a new domain this network performs better. This is the case for many different new domains such as traffic and forests. However, the issue with this is that it is unclear if the network performs better because it can adapt better of if the LSTM addition simply adds more complexity. It does not show the performance of the network over time in a new domain, therefore not indicating whether LSTM causes better performance over time with continuously improved estimates being stored in the memory neurons.

In contrast, Li *et al.* [27] suggest an LSTM framework that employs online adaptive learning in the context of depth estimation for visual odometry. Convolutional LSTM is interleaved between convolutional layers as seen in Figure 3.16. Convolutional LSTM layers are used here in order to leverage past experience.

This paper also employs a novel parameter update method. The most common method to update parameters in a network functions according to Equation 3.3, where θ are the parameters of the network, α is the learning rate of the network, $\nabla_{\theta_i}\mathcal{L}(\theta_i, \mathcal{D}_i)$ is the gradient of update dependent on the

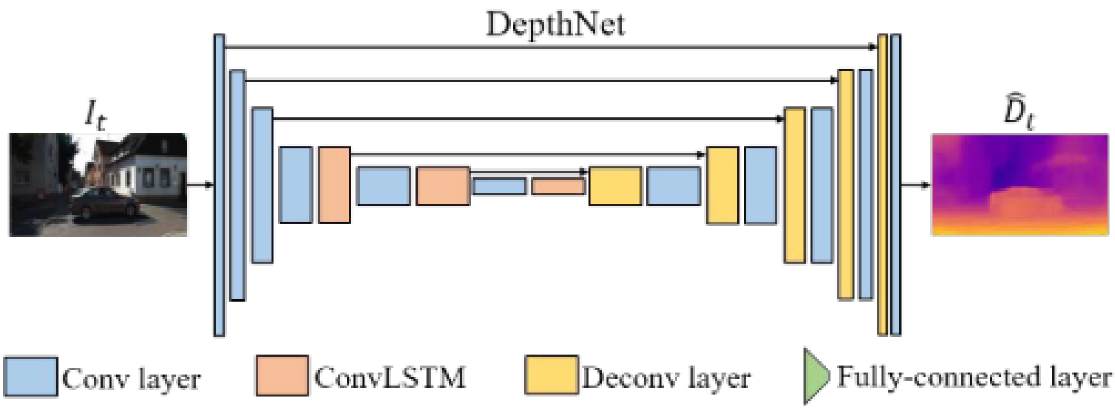

Figure 3.16: Depth Network with Convolutional LSTM layers [27]

current loss, $\mathcal{L}$, and $\mathcal{D}_i$ is the current data input.

$$\theta_{i+1} = \theta_i - \alpha \nabla_{\theta_i} \mathcal{L}(\theta_i, \mathcal{D}_i) \tag{3.3}$$

Li claims that this approach is naive and that the gradients on different iterations are stochastic in relation to each other. They imply that this leads to slow convergence and may introduce bias [27]. They offer a different meta-learning update scheme optimising over a series of inputs as seen in Equation 3.4.

$$\min_{\theta_i} \mathcal{L}(\theta_i - \alpha \mathcal{L}(\theta_i, \mathcal{D}_i), \mathcal{D}_{i+1}) \tag{3.4}$$

Essentially, instead of optimising on the current input, the update scheme attempts to optimise the adapted network on the next input. This allows for training not to be independent between subsequent data inputs. This is similar to work done by Finn *et al.* [12] on Model Agnostic Meta Learning (MAML) discussed in subsection 3.4.2. Changing the size of the data considered per iteration affects both the adaption accuracy and computational speed. It was found that accuracy only increases until a data window size of 15 and thereafter decreases.

The final step this paper takes is to align the features of the training domain with that of the new domain. This is done by using collecting the feature statistics, from the Normalisation Layer [3], at the end of training and adapting the mean and variance to better fit the new domain features. Although, the paper does not show the performance of the depth network directly, the performance in Visual Odometry, for which the depth estimation is essential, vastly outperforms other networks like GeoNet on new domains [60]. Overall, even though this paper might be more relevant to the temporal relation of visual odometry, it proposes novel and effective changes for online adaption for depth estimation. Although there are quite a few additions to the network, the training scheme manages to run at 30 FPS on the NVIDIA GTX 1080Ti GPU.

In 2020 Zhang *et al.* [63] suggest a new framework for online adaption, specifically for depth estimation, to prevent forgetting in network learning. They employ a novel statistic feature alignment with adapters and then adapt these adapters instead of the entire network. This quite a complicated scheme, and can be seen better in Figure 3.17.

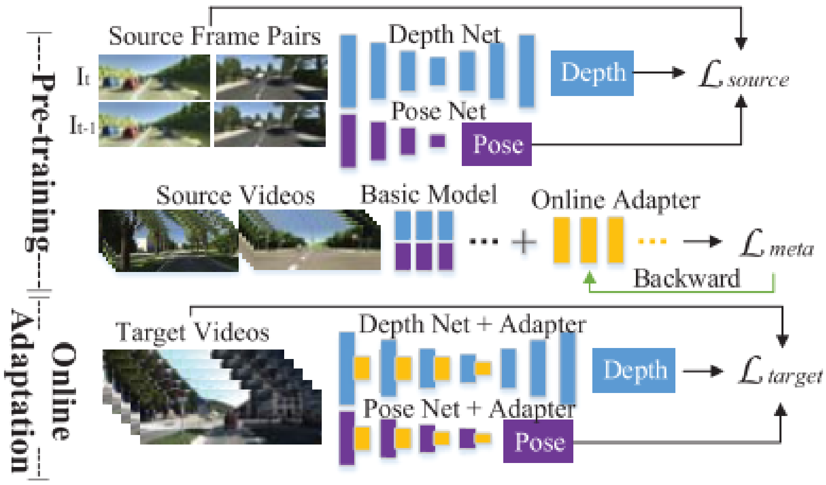

Figure 3.17: Feature Adaption based learning framework with pretraining and online training phases [63]

The figure shows that there are a few distinct steps: first the model is trained on a synthetic data set, followed by the training of the online adaptors of feature statistics, and then finally online, the adapters and the decoder layers are updated. The online adaption of the adapters is also done in a meta-learning manner, similar to Li *et al.* [27], in that the parameters are optimised over a series of data inputs. Using this method Zhang *et al.* boast quite impressive results as can be seen in Figure 3.18

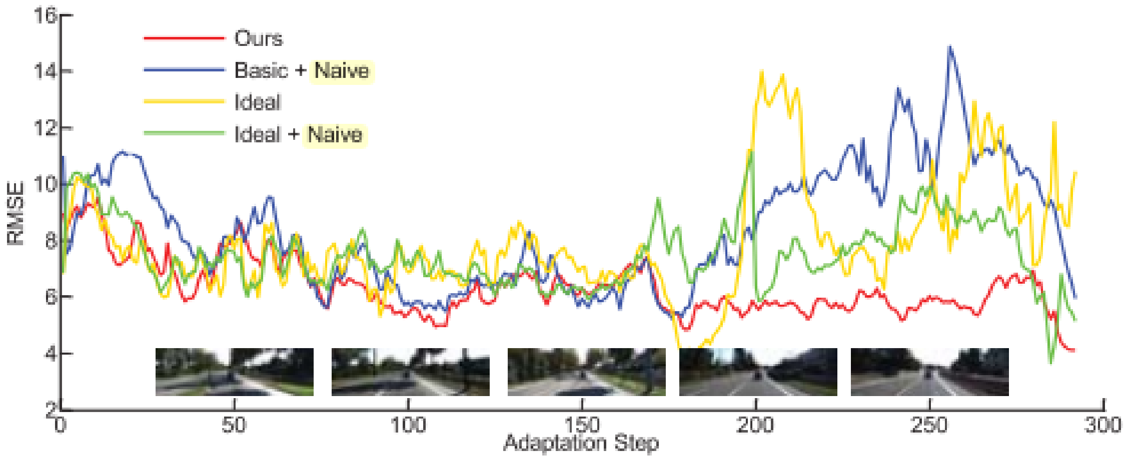

Figure 3.18: RMSE over time for different adaption methods on the same KITTI video [14] [63]

From the figure it is clear that the adaption of the authors framework works the best compared to naively updating the system like in Equation 3.3. This is evident from the spike in RMSE at roughly adaption step 200. Although, the other methods do eventually find better estimates they are much less robust to the change in the short term. This method seems to perform exceptionally well in adapting and doesn't seem to add too much computational effort as the adapter layers are only 1/9 the size of the encoder. The authors claim to operate in real-time, however, there is no indication of the training speed or inference speed. This method employs an interesting concept to only adapt the shift of feature statistics after normalisation layers in order to preserve previously learned knowledge whilst adapting to new domains.

For the final framework considered here, Tonioni *et al.* [51] released a paper in 2019 focusing on a novel architecture and update model, named MADNet (Modularly ADaptive Network) to make online adaption fast. They employ a similar pyramidal structure to Poggi *et al.* [42], however, they adapt it for stereo input rather than monocular input. This structure can be seen in Figure 3.19.

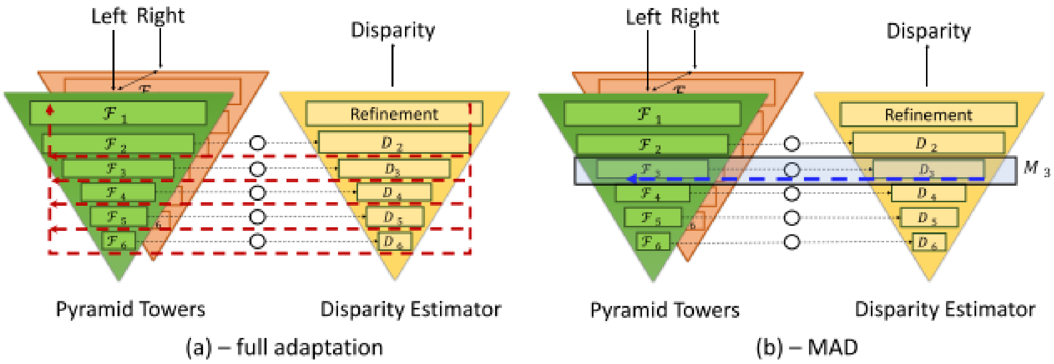

Figure 3.19: MADNet Pyramidal structure a) fully adaptive network update b) modular adaption of network [51]

The innovation of this method is that each layer of the pyramid comprises of a single module that can be updated independently from the others. This significantly decreases the computational effort per iteration with claims that backpropagation on KITTI full resolution can be done at 40 FPS. The method on selecting which layer to update is done through estimating which one currently has the worst performance in terms of loss.

The performance of this method is noticable when examining Figure 3.20. As can be seen, MADNet performs better in the long run than no adaption at all. It slowly converges to the network performance tuned on the ground truth for the new domain. It is also clear that the full adaption converges faster, however, it also takes more computational power.

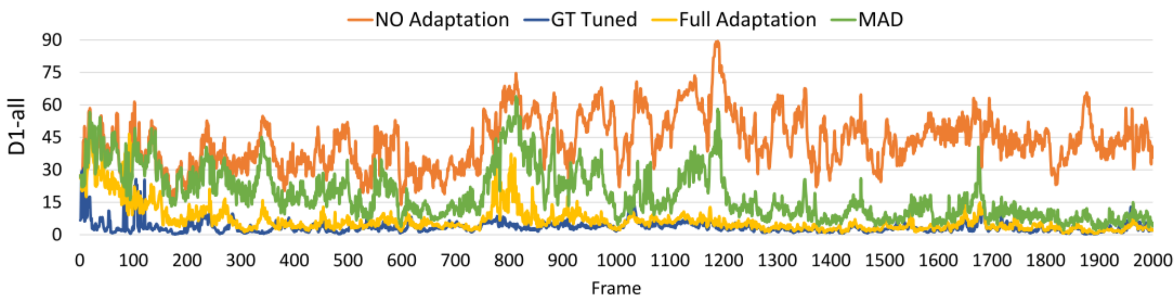

Figure 3.20: D1-all (percentage of pixels with disparity error > 3) over time on new domains comparing MADNet to other methods [51]

Tonioni *et al.* claim that only training the last few layers of a normal, sequential, network, which has a comparable computational effort to MADNet, has a much worse performance in adapting to new domains. However, the results were not compared to a sequential network being fully updated every iteration. Overall, this work seems like a promising direction for updating networks in a fast and effective manner which can be easily be moved over to a monocular framework such as from Poggi *et al.* [42].

3.4.2. General Adaption

There are also interesting papers that address adaption to new domain to generic machine learning problems such as Elastic Weight Consolidation (EWC) and MAML.

Elastic Weight Consolidation For instance, Kirkpatrick *et al.* [25] address catastrophic forgetting in neural networks. They introduce a concept known as EWC which prevents the more important weights to task A from being changed too much when adapting the network to task B. Although this process is very effective, it is quite particular to the task at hand. This can be seen by a largely differently approach for identifying written numbers compared to reinforcement learning for an Atari game. Also, the technique only seems to be implemented when the task is switched in a very discrete way. In the context of depth estimation it may be the case that the domain is constantly shifting, therefore, potentially making this method redundant.

Additionally, Chaudhry *et al.* [7] state that EWC requires calculating a Fisher matrix for each task and regularising over all of them is practically infeasible with multiple tasks and millions of parameters. Instead, Chaudhry *et al.* suggest only maintaining one diagonal Fisher matrix that is updated as it trains over new tasks. This is further improved upon by Schwarz *et al.* [47] with online EWC. This method allows for 'graceful' forgetting in which if an older task cannot be further optimised upon, it is slowly forgotten (which was not possible with EWC). Online EWC performs faster and reduces the number of required parameters by many orders of magnitude. Although this results in lost overall accuracy, in the context of online adaption on low memory and power hardware, it is more effective.

Model-Agnostic Meta-Learning Finn *et al.* [12] devise a MAML method that attempts to solve new learning tasks for a network with a small number of training samples. Essentially, this method is a more general form of Li's [27] update rule, where the parameters of the network, θ, are updated by optimising performance of the updated network over the range of tasks required of the network. Within the inner loop, for each task the parameters are updated through normal gradient descent, and then in the outer-loop the most optimal parameters are found for loss over all the tasks with the newly updated parameters from the inner loop. This method is quite useful, however, if the new parameters want to account for loss in the trained domain then the network would have to constantly store input data from previous domains to ensure they are still considered in subsequent optimisations. On low memory hard-ware this may be difficult to achieve. It also significantly increases computation time per iteration to find the gradients of the parameter over all the tasks in the outer optimisation step.

3.4.3. Conclusion

From the papers analysed it is clear in the last 3 years the research area of adaption, and specifically online adaption, is rife with innovation. Many papers have emerged attempting to better adapt networks to new domains and tasks in an effective and quick way. An issue that still remains, however, is that adaption techniques for depth estimation do not extended their domain to dynamic vehicles. The datasets used for the differing domains may include long-term shifts such as the overall environment (traffic to forest), however, they do not approach the issue of adaption in the context of a dynamic, low hardware, vehicle. Without an analysis at the edge of domain adaption it seems unclear whether these methods could make depth estimation in this dynamic environment feasible, as they so claim. For the purpose of this work it is necessary to pursue a lightweight solution that is not too complex to implement. Of the methods that were explored the most plausible would be to implement a LSTM layer in order to leverage temporal information within a domain. It may also be interesting to explore confidence estimates, however, this would add to training time where as the addition of LSTM may be more lightweight.

3.5. Network Speed

When running an online network there are two types of speed that are relevant. Both the inference speed, as well as the training speed, are essential to running a fast online adaptive network. In this chapter papers in both the field of depth estimation and the broader machine learning field are examined for strategies to improve speed. This chapter is separated into techniques to improve inference speed in subsection 3.5.1, training speed in subsection 3.5.2 and a brief conclusion in subsection 3.5.3.

3.5.1. Inference

Increasing the inference speed is an obvious goal for research in this domain as there is an effort to move neural networks to many different domains. With such an interest in the subject, many different strategies have been identified to reach this objective.

Resolution One of the most obvious changes to improve speed, on inference, is to reduce the resolution of the input. Of course, the input size must stay consistent with the chosen stride and padding of the beginning layer in order to be feasible. The effects of changing the input resolution can be seen in Figure 3.21. It is clear that the marginal effect on inference time is reduced once the image is reduced to 90x30 which already holds very little information. Perhaps, with this size of input image the size of the network (number of filters) could also be reduced as the amount of information to be extracted by passing filters is greatly reduced.

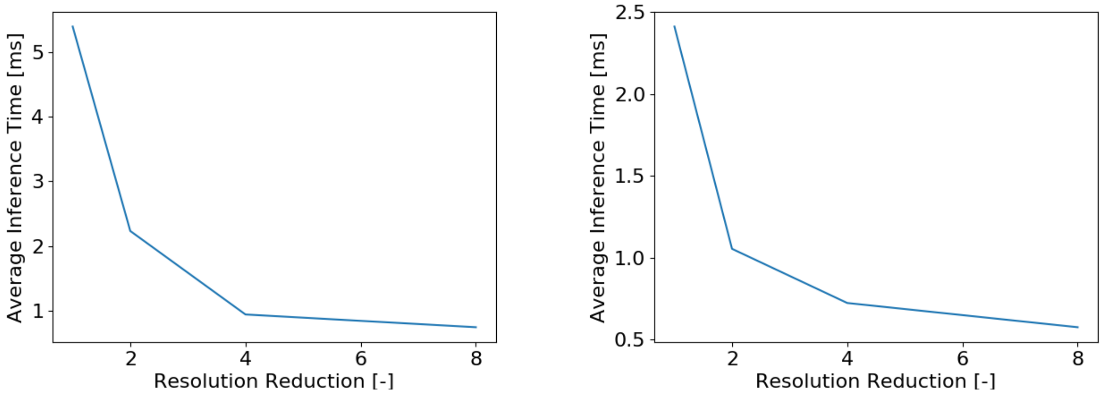

Figure 3.21: Inference time with scale reduction of original input resolution of 360x120. 32 filter 3x3 convolution with ReLU activation function and max pooling (left). Deconvolution with 1 3x3 kernel filter (right)

In the context of running on low-power computational units, reducing the input size requires a trade-off of required accuracy. For example, Peluso *et al.* [35] implement the pyramidal structure developed by Poggi *et al.* at the very edge of resolution. They make the network much shallower and are able to reduce the parameters from 1.9 million to 100 thousand. The input resolutions are extremely reduced to levels of 48x48 and 32x32 which can be seen in Figure 3.22. Depending on the application a choice must be made whether this level of detail is acceptable.

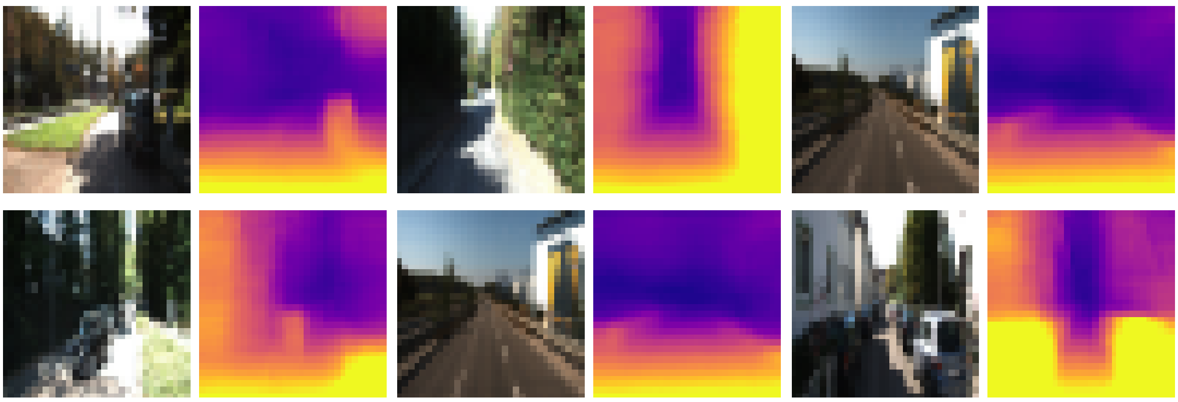

Figure 3.22: Examples of depth estimation on 32x32 image inputs

This network employs supervisory training by means of SGM. Although the reason for this is not given, it is suspected that this is due to the fact that using a self-supervised method with stereo pairs is ineffective at these resolutions as many details may be lost between the stereo pairs. Instead, finding the depth based on a classical stereo algorithm at original resolution and downsampling it to be used as a supervisory signal seems more accurate.

Network Structure There are also interesting changes that can be made to the network architecture that result in faster inference. One of the novel ways is to employ Depthwise Seperable Convolutions developed by Howard *et al.* [21] for use on their MobileNet. In essence this changes the convolution structure from a single convolution layer to two convolutions, first with a separable convolution followed by a pointwise convolution of dimension 1x1. The difference can be seen in Figure 3.23.

This change reduces the computational effort by the factor given in Equation 3.5 where D_K is kernel size and N is the number of kernels. The authors boast much faster performance with a minor drop in accuracy. They find a log linear dependance between accuracy and computation. They also claim that

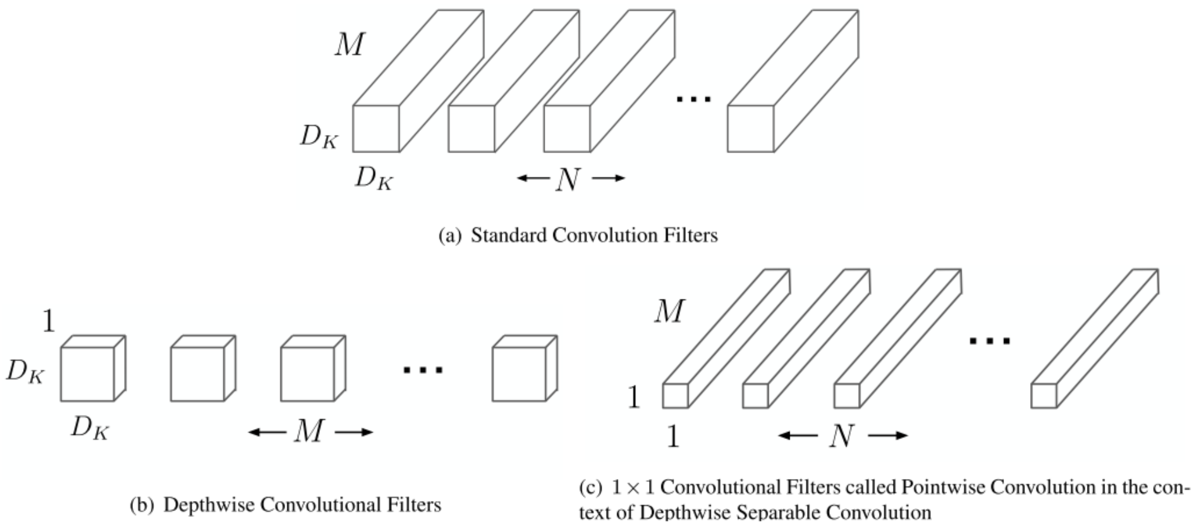

Figure 3.23: Comparison of standard convolutional filters in a) that are replaced with combination of depthwise b) and pointwise convolutions c). D_K is kernel size, M is the number of input channels and N is the number of kernels [21]

there is less need for data augmentation techniques as smaller models (less parameters) have less issues of over-fitting.

$$\frac{1}{N} + \frac{1}{D_K^2} \tag{3.5}$$

This method has also been used once by Wofk *et al.* [54] for their FastDepth network. In this network Wofk employs the use of MobileNet as an encoder as well as converting the decoder deconvolution operations to a depthwise separable shape. This results in much faster computational speeds as can be seen in Figure 3.24. Wofk also noticed that the decoder can also be designed with simple convolution followed by bearest neighbour interpolation to upsample, which results in the same accuracy with higher computational speed. At times the deconvolution operation introduces checkerboard artefacts in the output. The move from the ResNet architecture with Up Projection to MobileNet with deconvolution decreases the computation speed by a factor of up to 10 times. This is further enhanced by implementing depthwise operations in the decoder. The accuracy of each of these networks remain roughly the same.

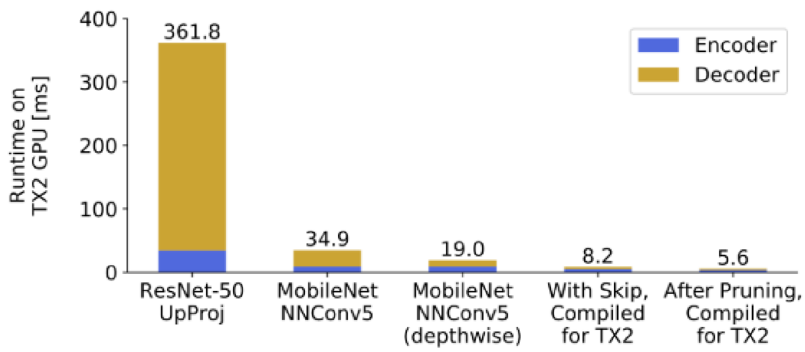

Figure 3.24: Effect of network architecture changes by FastDepth on computational speed [54]

Wofk *et al.* use skip connections in additive manner, rather than concatenative, as it reduces the number of channels to analyse in the decoder whilst maintaining similar accuracy. These proposed changes in architecture are evidently quite effective and do not pose many restrictions on the network.

Pruning Something else employed by Wofk *et al.* is the use of post training pruning. Pruning is the process of removing model parameters in effort to reduce computational time, but maintain accuracy.

The pruning of the decoder is made easier as the author removed deconvolutions and only has convolutions remaining. Wofk *et al.* use NetAdapt [57] to prune the network, which iterates over different network proposals and selects the best accuracy/complexity trade-off per round. The effectiveness of this approach can be seen by the reducing of channel size in Figure 3.25 and the decrease in runtime in Figure 3.24.

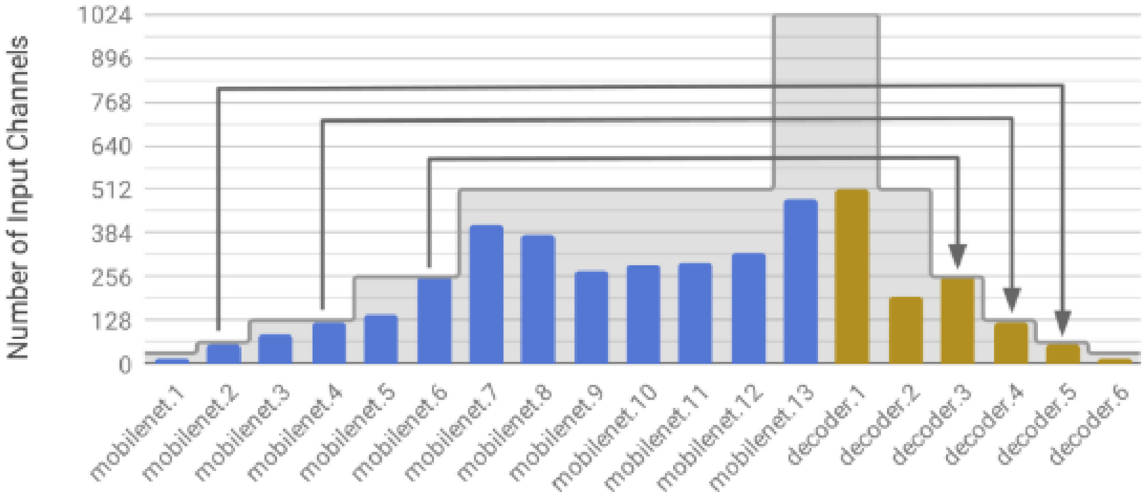

Figure 3.25: Reduction in input channels to each layer in FastDepth due to pruning. Grey area indicates the values for the network prior to pruning. [54]

One drawback of this, is that it runs in an offline manner. However, this could also be used to estimate how far the network could be reduced before employing it for online learning.

There are some methods for online pruning, for example, Haider *et al.* [18] suggest using gates to activate layers during forward pass via optimisation of an objective function. However, this work has not seen any similar implementation or peer review.

Quantisation Another promising direction is the quantisation and compilation of networks for deployment. Wofk *et al.* [54] use a hardware specific (TVM) compilation to increase the network speed by a factor 2 as seen in Figure 3.24. However, this is for a network ready for deployment, rather than one that undergoes online training.

Quantisation is a process by which floating point values for parameters in the network are reduced to 16 or 8 bit representations. This is done in a way where the accuracy is reduced as little as possible. Approaches range from performing the same transformation over the entire network to methods that change the representation per layer instead. Peluso *et al.* [36] make the remark that there is no quantisation solution that can be applied to a general network, it must be determinedb by the specific network and platform. Peluso performs a quantisation of the PyD-Net model from Poggi [41] for application on a ARM based platform. Peluso performs the quantisation dyanmically per layer aiming to minimise the distance of the original values from the quantised result. Peluso *et al.* claim that the improved performance does not stem from decrease arithmetic complexity, but rather from more efficient use of memory bandwidth due to lower bit representations. They boost hardly any drop in performance from PyD-Net with heavily increased energy efficiency (up to x6), less memory occupied to store weights and lower usage of RAM by a factor of x4.5. This paper shows excellent results, but yet again, it seems that this method requires a large effort offline and is not trainable.

Jacob *et al.* [23] propose a method allows for integer-only arithmetic to apply on integer-only hardware. Interestingly, they design a training procedure that simulates quantised integer-only computations but performs back-propagation in floating point arithmetic. The authors claim that this restores model accuracy to near-identical levels to the floating point method. They show this method to be effective on MobileNet [21] architecture

3.5.2. Training

Although there is quite an effort in decreasing the inference time of a network, there is considerably less effort on attempting to decrease the backprogation pass. This is due to there being powerful hardware available on offline training and less powerful technology when deploying the network; most frameworks do not train online.

However, some of the efforts for increasing speed on inference do also increase speed on back-propagation. For example, if the resolution is lowered, backpropagation speed for CNNs will increase. This is because the derivative of layer weights, with respect to its output, is equal to the input to the layer. Therefore, in the backward pass, when gradients are convolved, these with lower resolutions will be much less computationally intensive. This is also the case with pruning a network and training it afterwards.

Apart from changes in the raw amount of computations to train a network, there are also ways in which a network can converge faster in training. For example, Krizhevsky *et al.* [26] claim that in testing their network, ImageNet, on the task of classification, it converged the fastest when using ReLU activation functions. Apparently, using other activation functions, such as tanh, result in converging to 25% error rate in more than x6 the time.

Another very popular way is to use batch normalisation layers in the network, which are present in many of the depth estimation networks discussed in section 3.3. Batch Normalisation makes the training of a network faster and more stable [44] by stabilising the distribution of layer inputs. This is done by controlling the mean and variance of inputs through an additional layer. Santurkar *et al.* claim that this is due to the normalisation layer smoothing the landscape of the gradients, leading to faster convergence. However, there are claims that it causes gradient explosion initialisation [44]. Batch Normalisation is designed for training that has mini-batches to train on, which may not be the case for online learning, where the training batch may only be of size one. Lei Ba *et al.* [3] offer an adaption to this with Layer Normalisation, which performs a similar effect but with only on single training cases.

Finally, there are also training strategies in which the entire network does not have to be trained every iteration, like the approach of Tonioni *et al.* [51] discussed in subsection 3.4.1. For example, Brock *et al.* [6] suggest progressively freezing layers of the network to stop to training them. A similar method is proposed by Huang *et al.* [22] in which only a random set of layers are trained each iteration. Both these methods achieve gains in training time and also claim their framework prevents over-fitting as they acts as regularisers. However, their method may not be so simple to implement on low level machine learning frameworks.

3.5.3. Conclusion

Overall, there are quite a few ways to speed up the performance of the network. The most efficient methods, that could be used well in an online manner seem to be those that change the architecture of network. This includes changing resolution, pruning before deployment and quantisation. Employing these methods would be beneficial to deploying a depth estimation network on low-power hardware.

3.6. Implementation

Although there are many interesting and useful suggestions from papers in the previous chapters, there must also be a consideration of the the actual platform for this implementation. First, the SteroJevois camera is discussed in subsection 3.6.1 followed by a discussion of the framework for implementing CNNs on the StereoJevois in subsection 3.6.2. Finally, a discussion of the simulation platform Airsim is given in subsection 3.6.3.

3.6.1. StereoJevois

The StereoJevois is a hardware adaption of the Jevois Smart Machine Vision Camera[1] to have stereo vision. Jevois is a small 1.3 MP camera equipped with a video sensor, quad-core CPU, dual GPU, USB video and serial port. This allows all image processing and computer vision to be on-board the camera system. Unfortunately, there is no IMU system on the StereoJevois, however values can be input to the system through the serial bus.

[1]Jevois: `www.jevois.org`

The computer vision algorithm and resolution output is determined by the selected output resolution and frame rate of the camera. The computer vision algorithms can be implemented in different languages including Python, C and C++.

The Jevois was changed to its stereo form purely by means of hardware; the same Jevois software receives the input in the same as it would if there was only a single camera. The images arrive interleaving in the vertical direction and must be separated by a pre-processing function before they can be used. With this hardware implementation there are two modes: one with a full resolution dual image in black and white and the second of half vertical resolution in colour. They are toggled by making a short circuit between a debug pin (D1) and ground on the circuit board. These two outputs can be seen in Figure 3.26. The black and white image was taken in the same environment as the colour and was left dark to demonstrate the differences. Although it is difficult to see they are of the same scene.

Figure 3.26: Raw outputs of the StereoJevois. Colour output of StereoJevois (left) Black and White output (right)

Modules can be written for the Jevois in JevoisInventor and set at the desired resolution and frame rate output. When implementing modules, compilation is not necessary as Jevois generates shared objects itself.

3.6.2. CNN Framework

There are two open-source CNN frameworks available on the Jevois platform: Darknet [43] and TensorFlow [1]. Darknet is an implementation of CNN architecture written in C and CUDA. Darknet is primarily implemented for use in classification tasks with CNNs that have encoders followed by fully connected layers rather than decoders. Although there are capabilities for decoder layers, the use cases with Darknet are extremely sparse and informally made with very little documentation. Additionally, Darknet is restricted to a sequential structure and does not have any implementations of loss functions outside of supervised classification tasks.

Tensorflow is an extremely popular framework for designing and training neural networks of various shapes in the Python programming language. It does have an API written in C++ that is accessed by Jevois. Tensorflow is more mature in terms of development, however, their implementation in C++ is also quite new.

Jevois uses both just to call the forward pass of networks that have already been trained offline. As it seems, the implementation to learn online for CNNs is non-existent on the Jevois and must be developed. To choose between the two frameworks will require more time of analysing the capabilities.

3.6.3. Airsim

Airsim is an open source simulator for simulating autonomous vehicles such as drones and cars [48]. Airsim is quite useful in that it allows for simulated depth maps and stereo vision. It can be used to retrieve training data for training offline or it can be used to simulate online training with live control. This is possible through Python or C++ script, meaning that things can be tested on an easier machine learning framework like PyTorch first.

This simulator can be run on Unreal Engine, however, it is simpler to run the binaries for the specific

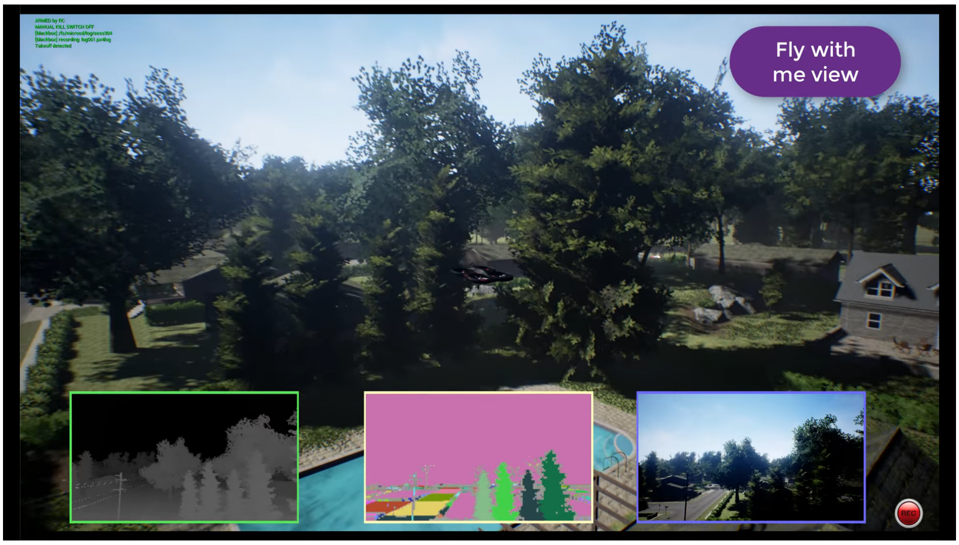

Figure 3.27: Screenshot of airsim drone simulation with depth map, segmentation map and first person view [48]

environment. A benefit of this simulator is that there are a variety of different environments with both indoor and outdoor scenes. This framework can serve to experiment with methods for depth estimation and control before being deployed a real drone. Noise artefacts and other effects can be used to attempt to better approximate the real domain.

3.7. Research Questions and Conclusion

The conclusion to the literature review is split into three sections. First, the development of the research questions is given by identifying the research gap in subsection 3.7.1. This is followed by a brief project outline of the research direction and organisation in subsection 3.7.2. Finally, a conclusion to the paper is given in subsection 3.7.3

3.7.1. Research Questions

Research Gap From the different research areas addressed in this review, it is possible to extract two major gaps in the research. First, it is clear that although there are a large number of depth estimation methods available, there is not too much work with regards to application of drones. Subsection 3.3.5 does identify some efforts in this area, however, either the methods do processing off-board [39] or they employ powerful hardware in constrained, obstacle-course, environments [58]. There has not been work to process fully on-board a lightweight drone for natural, indoor, environments. Secondly, as seen in section 3.4, there has clearly been no work on online self-supervised learning of depth for a physical drone, in a real-world environment. Therefore, this thesis will address both the use, and online learning, of self-supervised depth estimation on a drone, in real environments.

Research Questions With this research gap identified, the research question and subquestions can be stated below.

- How can we carry out online self-supervised learning of depth estimation on a drone?
    1. Which depth prediction method is best suited to deployment on a drone for online domain adaption on low-power hardware?
        (a) Which method offers the best trade-off between inference speed, training speed and accuracy?
        (b) Which method is best suited to adapt online to a new domain the fastest and most robustly?
        (c) To what degree is the proposed method feasible for implementation on hardware with low computational power?

2. How should the drone obtain new data such that it can adapt to the new domain as quickly and robustly as possible?
    (a) What information about the environment is most important to capture, to robustly adapt to the new domain?
    (b) At what rate should the drone obtain data for the learning process, in order for the drone to learn quickly and robustly?
    (c) How the should the drone be controlled dynamically in order to facilitate robust domain adaption?
    (d) How can the drone learn in a manner that prevents unlearning of previous domains?
3. What is the overall performance of the proposed depth prediction method?
    (a) How well does the drone adapt in terms of accuracy, compared between different domains?
    (b) How fast does the drone reach a threshold of accuracy between different domains?
    (c) What is the inference time, training time and memory allocation of the proposed method on the drone?

3.7.2. Research Planning

This section develops a plan for apporaching this thesis. First, a preliminary answer to research sub-question **1** is given in subsection 3.7.2 in order to guide the beginning of thesis. This is followed by a break down of the organisation of time for the thesis in subsection 3.7.2.

Research Direction This review has gathered knowledge of different depth estimation techniques, domain adaption methods, computational speed increase techniques and specifics of the hardware platform.

It is clear that computational effort in inference, and especially training, is a larger constraint than accuracy of the estimation. Also, it is clear that not only computational effort is constraining, but also solution complexity, as the proposed platforms for learning, such as Darknet, are quite rigid in their possible structures. Therefore, the solution to pursue, must attempt to target the most simple changes that have the largest effects.

From the comparison of methods in subsection 3.3.4, it is clear that the depth estimation method that employs the least computational effort and complexity, allowing for self-supervised learning, is using the autoencoder with left-right consistency loss by Godard *et al.* [15]. Although, it is slightly more expensive for training than Garg *et al.* [13], it is marginal for the considerable increase in accuracy. This structure of a sequential auto-encoder is suggested also due to its popularity on the few solutions that have been implemented for mobile robots [66] [34] [58], particularly drones, as seen in subsection 3.3.5.

Additionally, there a few features from different papers that will be explored, that add vital components in a lightweight manner, to attempt to address the issues of domain adaption. First, to make the method more robust to the effect of drone motion, pose encoding will be explored in a similar manner to Zhao *et al.* [66] as seen in subsection 3.3.5. A suggested simple change, is to include the roll pose as well. Also, LSTM layers and batch normalisation will be explored as they are some of the few possible methods discussed in section 3.4 that can be applied on a constrained platform like Darknet. Finally, for increasing computational speed, resolution changes, offline pruning and quantisation will be explored as described in section 3.5.

Research Organisation To approach the planning of the research the project was broken down into a few work-packages. These consists of: Literature Study, Implementation, Verification, Indoor Testing, Performance Analysis and Report Writing. Implementation is done both on the simulation and hardware platform. The planning can be seen in the Gantt Chart in Figure A.2.

In the chart there are quite a few concurrent work-packages. The first concurrent work-packages are implementation and verification as they are done together while iterating the developed code. The second concurrent work-packages are indoor testing and performance analysis as the data must be analysed while testing to see if things are behaving accordingly. Finally, performance analysis and report writing happen concurrently. The time for each is estimated on personal experience with working with implemented code on hardware, however, there is the possibility of delays in implementation of hardware that are very possible. This is especially true because of the relative novelty of the computational hardware that was developed for the purpose of this project. If corona disallows on campus testing, then the Gantt chart must re-evaluated for new testing practices. Overall, this outline is quite ambitious for implementing this software/hardware solution; however, this is the required tempo to fully address the problem.

3.7.3. Conclusions

In conclusion, this paper has surveyed the body of research relevant to the online adaption of drones for depth estimation to arrive at a set of research questions to use. In section 3.3 the various methods to estimate depth on a drone were examined. This extended from classical to machine learning methods, and a fusion of the two, ending with a comparison of the different approaches. This was followed by examination of online adaption methods of neural networks both within the field of depth estimation and generally. Finally, different methods for increasing the speed of networks, for both inference and back-propagation, were identified and discussed. Additionally, a brief overview of the platform for implementation was given.

With these identified the main research gaps were identified and the research questions could be formed as well as a project plan. This literature review forms the motivation for a thesis to develop, implement and analyse an online adaptive self-supervised depth estimation network on a drone.

A

Additional Figures

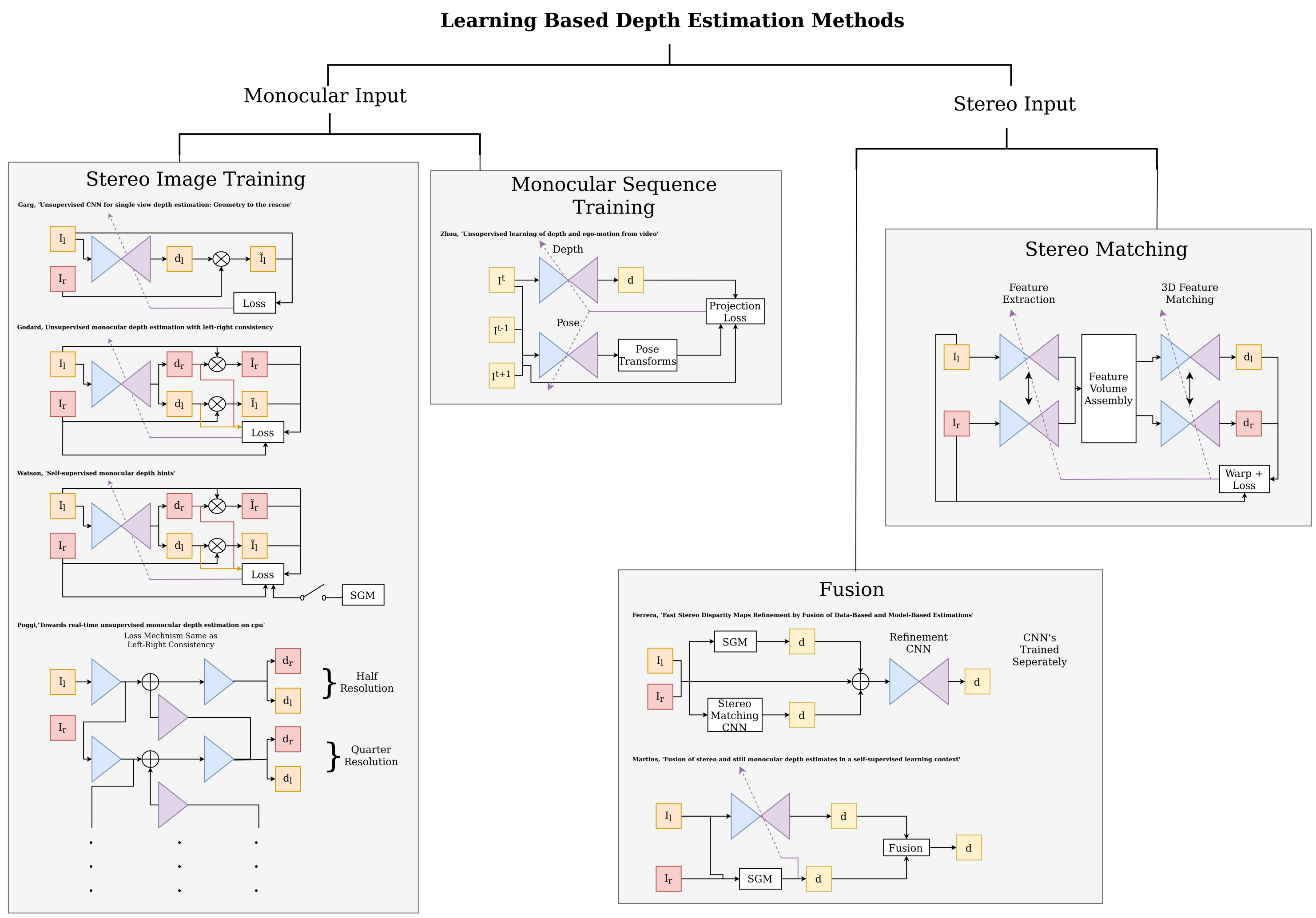

Figure A.1: Overview of relevant learning based depth estimation methods. I are the images and the subscripts indicated whether they are left or right. Superscript indicates the image place in time. $\tilde{I}$ indicates that the image of the scene is synthesized. d is the disparity with the same subscripts as the images I

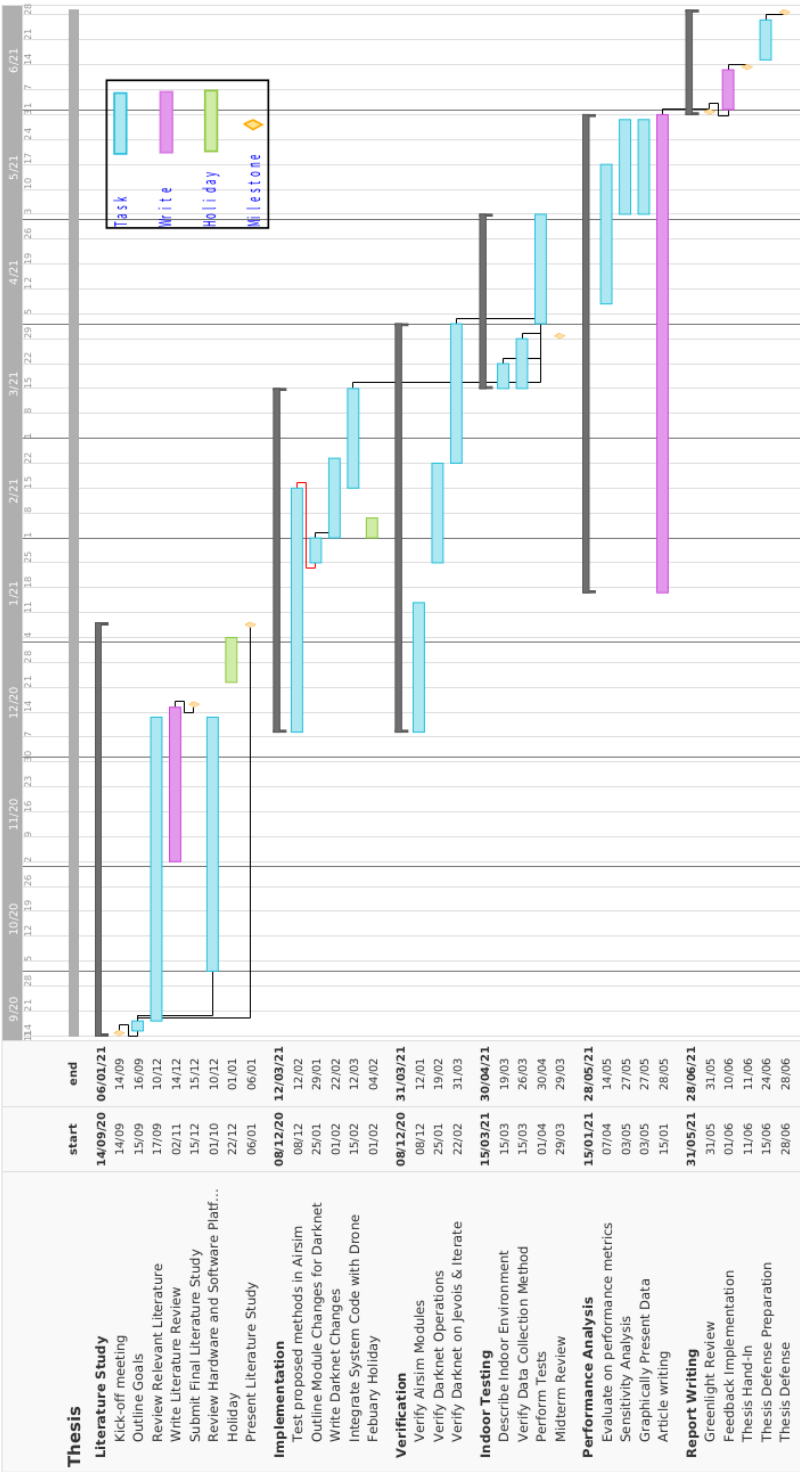

Figure A.2: Gantt chart for project planning of the thesis

III

Conclusions

4
Conclusion

With the advances proposed in this paper, self-supervised monocular depth estimation has been considerably improved in untextured indoor rotated environments. This contribution is useful for making numerous applications, such as autonomous navigation, augmented reality and scene reconstruction, more accurate and safer in increasingly complex environments.

The main conclusions of the paper are given as answers to the initially posed research questions. The first question concerns the improvement in the performance of self-supervised monocular depth estimation methods in untextured regions of an image.

Q1 How can self-supervised monocular depth estimation be improved for untextured regions in images?

Q1.1 To what extent is accuracy in untextured regions improved compared to previous methods?

Q1.2 Is there a trade-off in performance in textured regions with previous methods?

To improve performance in untextured regions the Filled Disparity Loss was presented as an addition to conventional self-supervised depth estimation methods. Disparity estimation at edges, where image reconstruction loss is better posed, was used to interpolate disparity within textureless regions. From the results in section 2.4 there was a demonstrable increase in the accuracy on indoor scenes when compared to Monodepth [15]. The effect of the new loss term was quite clear as the performance in the chosen metrics improved as the weight of the Filled Disparity Loss increased. However, it was clear that the improved disparities were still smaller when compared to the ground truth disparity in near regions. This is due to the Filled Disparity Function only being able to interpolate between values at edges. When hallways, which dominate a lot of indoor spaces, were present, the Filled Disparity Function could not increase estimates of the disparity at the edges of images. However, it is noted that when inverting to find the depth the differences in these regions are minimised.

To answer the second part of this question an analysis of the performance as a function of the 'texturedness' of an image was given. It was shown that when the weight of the new loss term was increased, performance in images with low 'texturedness' was significantly improved and performance in images with high 'texturedness' was maintained. This implies that there little to no trade-off in performance in textured regions of an image when compared to previous methods [15].

The second research question addressed pertained to improving performance in rotated scenes.

Q2 How can self-supervised monocular depth estimation be improved for rotations in pitch and roll?

Q2.1 Is the improvement in accuracy different for rotations in pitch and roll?

Q2.2 How well does the improvement in accuracy generalise over the entire range of rotation?

To improve performance over pitch and roll rotations, a passive approach was used where instead of changing the network, only the range of rotations present in the dataset was changed. It was demonstrated that training with a dataset that contains rotations in both pitch and roll is extremely effective in improving the performance on rotated images. From the comparison of overall error metrics, it seemed that pitch had more of an effect on the performance of the network when tested on the pitch and roll test set. This was due to the pitch altering vertical depth cues more drastically, and in a less symmetrical manner, than rotations in roll [52].

When looking at comparisons of accuracy over the range of rotations it was clear that training for the representative rotations improved performance over the entire range of rotations. When training on images rotated in pitch, the performance for non-rotated images was the same as for training on the nominal dataset. Performance for the pitch trained network did not change over the entire range of pitch angles in the test set. Also, the network trained on both pitch and roll did not trade-off performance on the pitch test set when compared to the network trained on the pitch dataset.

Overall, the results obtained with this research not only show an improvement in the performance for untextured indoor rotated scenes, but it also demonstrates that there has not been a trade-off for textured non-rotated scenes. Hopefully, this work will expand the use of self-supervised monocular depth estimation to more complex domains.

5

Recommendations

From the work done in this paper there are a few recommendations for further research identified.

- The largest improvement for this research would be to collect indoor data under different rotations for a real-world domain. Although the results are significant, showing them closer to a domain for application would be useful. This would incorporate more real-world effects of cameras, and stereo cameras, that may pose problems to this network.
- Another large area for improvement is the generalisation of this method to any domain. Although performance is improved indoors, there is the issue of outdoor performance. With the sky as a generally untextured region of an image, the depth of whatever is in front of the sky would be interpolated into the sky as a consequence of the Disparity Filler Function. There are ways to remedy this, for example, using a sky classification network in parallel or using assumptions to detect the sky. However, these may require labelled data or have more assumptions that break in different domains.
- It is attractive that the Filled Disparity Loss function is differentiable. However, there is an undesired effect in the propagation step where disparity extends outwards, perpendicular to the detected edges. Although this effect was not too influential in the end product, it would be better for the generated disparity map to be more physically correct: to interpolate linearly between edges.
- In this research, changes in the pose were limited to pitch and roll whilst height was held constant. It would be significant to understand the effect of varying height of images in varying domains.
- An exciting area that this could be applied to is for online learning on drones. This way depth could be learned for different domains with exploration, rather than trained offline each time the domain is changed. Several computational and hardware improvements would have to be considered, but it may be interesting for indoor navigation.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] A. Atapour-Abarghouei and T. P. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2800–2810, 2018. doi: 10.1109/CVPR.2018.00296.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[4] Elena Bebeselea-Sterp, Raluca Brad, and Remus Brad. A Comparative Study of Stereovision Algorithms. *International Journal of Advanced Computer Science and Applications*, 8(11):359–375, 2017. ISSN 2158107X. doi: 10.14569/ijacsa.2017.081144.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Freezeout: Accelerate training by progressively freezing layers, 2017.

[7] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 556–572, Cham, 2018. Springer International Publishing.

[8] Po Yi Chen, Alexander H. Liu, Yen Cheng Liu, and Yu Chiang Frank Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:2619–2627, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00273.

[9] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, 3(January): 2366–2374, 2014. ISSN 10495258.

[10] José M. Fácil, Alejo Concha, Luis Montesano, and Javier Civera. Deep single and direct multi-view depth fusion. *CoRR*, abs/1611.07245, 2016. URL http://arxiv.org/abs/1611.07245.

[11] M. Ferrera, A. Boulch, and J. Moras. Fast stereo disparity maps refinement by fusion of data-based and model-based estimations. In *2019 International Conference on 3D Vision (3DV)*, pages 9–17, 2019. doi: 10.1109/3DV.2019.00011.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1126–1135. JMLR.org, 2017.

[13] Ravi Garg, B. G. Vijay Kumar, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9912 LNCS: 740–756, 2016. ISSN 16113349. doi: 10.1007/978-3-319-46484-8_45.

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

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

[16] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob(1):3827–3837, 2019. ISSN 15505499. doi: 10.1109/ICCV.2019.00393.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] Muhammad Umair Haider and Murtaza Taj. Comprehensive online network pruning via learnable scaling factors, 2020.

[19] Heiko Hirschmüller. Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2005. ISSN 01628828. doi: 10.1109/TPAMI.2007.1166.

[20] Heiko Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008. ISSN 01628828. doi: 10.1109/TPAMI.2007.1166.

[21] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv*, 2017. URL http://arxiv.org/abs/1704.04861.

[22] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 646–661, Cham, 2016. Springer International Publishing.

[23] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. doi: 10.1109/CVPR.2018.00286.

[24] H. Jung, Y. Kim, D. Min, C. Oh, and K. Sohn. Depth prediction from a single image with conditional adversarial networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1717–1721, 2017. doi: 10.1109/ICIP.2017.8296575.

[25] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/content/114/13/3521.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL https://doi.org/10.1145/3065386.

[27] Shunkai Li, Xin Wang, Yingdian Cao, Fei Xue, Zike Yan, and Hongbin Zha. Self-Supervised Deep Visual Odometry With Online Adaptation. *CVPR*, pages 6338–6347, 2020. doi: 10.1109/cvpr42600.2020.00637.

[28] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang. Learning for disparity estimation through feature constancy. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2811–2820, 2018. doi: 10.1109/CVPR.2018.00297.

[29] Y. Luo, J. Ren, M. Lin, J. Pang, W. Sun, H. Li, and L. Lin. Single view stereo matching. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 155–163, 2018. doi: 10.1109/CVPR.2018.00024.

[30] Michele Mancini, Gabriele Costante, Paolo Valigi, Thomas A. Ciarfuglia, Jeffrey Delmerico, and Davide Scaramuzza. Toward Domain Independence for Learning-Based Monocular Depth Estimation. *IEEE Robotics and Automation Letters*, 2(3):1778–1785, 2017. ISSN 23773766.

[31] Diogo Martins, Kevin van Hecke, and Guido de Croon. Fusion of stereo and still monocular depth estimates in a self-supervised learning context, 2018.

[32] Jeff Michels, Ashutosh Saxena, and Andrew Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 593–600, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102426. URL https://doi.org/10.1145/1102351.1102426.

[33] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. doi: 10.1109/TRO.2015.2463671.

[34] Chaoyue Niu, Danesh Tarapore, and Klaus-Peter Zauner. Depth estimation on embedded computers for robot swarms in forest, 2020.

[35] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia. Enabling monocular depth perception at the very edge. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1581–1583, 2020. doi: 10.1109/CVPRW50498.2020.00204.

[36] Valentino Peluso, Antonio Cipolletta, Andrea Calimera, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Enabling Energy-Efficient Unsupervised Monocular Depth Estimation on ARMv7-Based Platforms. *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019*, pages 1703–1708, 2019. doi: 10.23919/DATE.2019.8714893.

[37] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe. Unsupervised adversarial depth estimation using cycled generative networks. In *2018 International Conference on 3D Vision (3DV)*, pages 587–595, 2018. doi: 10.1109/3DV.2018.00073.

[38] Andrea Pilzer, Stéphane Lathuilière, Nicu Sebe, and Elisa Ricci. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. *CoRR*, abs/1903.04202, 2019. URL http://arxiv.org/abs/1903.04202.

[39] Clement Pinard, Laure Chevalley, Antoine Manzanera, and David Filliat. Multi range real-time depth inference from a monocular stabilized footage using a fully convolutional neural network. *CoRR*, abs/1809.04467, 2018. URL http://arxiv.org/abs/1809.04467.

[40] Clément Pinard, Laure Chevalley, Antoine Manzanera, and David Filliat. Learning structure-from-motion from motion. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 363–376, Cham, 2019. Springer International Publishing. ISBN 978-3-030-11015-4.

[41] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Efficient confidence measures for embedded stereo. In Sebastiano Battiato, Giovanni Gallo, Raimondo Schettini, and Filippo Stanco, editors, *Image Analysis and Processing - ICIAP 2017*, pages 483–494, Cham, 2017. Springer International Publishing. ISBN 978-3-319-68560-1.

[42] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards Real-Time Unsupervised Monocular Depth Estimation on CPU. *IEEE International Conference on Intelligent Robots and Systems*, pages 5848–5854, 2018. ISSN 21530866. doi: 10.1109/IROS.2018.8593814.

[43] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[44] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 2483–2493. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf`.

[45] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005. ISSN 10495258.

[46] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, 2001. doi: 10.1109/SMBV.2001.988771.

[47] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress compress: A scalable framework for continual learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4528–4537, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/schwarz18a.html`.

[48] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL `https://arxiv.org/abs/1705.05065`.

[49] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano. Unsupervised adaptation for deep stereo. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1614–1622, 2017. doi: 10.1109/ICCV.2017.178.

[50] A. Tonioni, M. Poggi, S. Mattoccia, and L. D. Stefano. Unsupervised domain adaptation for depth prediction from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10): 2396–2409, 2020. doi: 10.1109/TPAMI.2019.2940948.

[51] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi DI Stefano. Real-time self-adaptive deep stereo. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:195–204, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00028.

[52] T. Van Dijk and G. De Croon. How do neural networks see depth in single images? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2183–2191, 2019. doi: 10.1109/ICCV.2019.00227.

[53] Jamie Watson, Michael Firman, Gabriel Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:2162–2171, 2019. ISSN 15505499. doi: 10.1109/ICCV.2019.00225.

[54] D. Wofk, F. Ma, T. Yang, S. Karaman, and V. Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108, 2019. doi: 10.1109/ICRA.2019.8794182.

[55] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5510–5519, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00566.

[56] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry, 2020.

[57] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 289–304, Cham, 2018. Springer International Publishing.

[58] X. Yang, J. Chen, Y. Dang, H. Luo, Y. Tang, C. Liao, P. Chen, and K. Cheng. Fast depth prediction and obstacle avoidance on a monocular drone using probabilistic convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2019. doi: 10.1109/TITS.2019.2955598.

[59] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. *CoRR*, abs/1907.12209, 2019. URL http://arxiv.org/abs/1907.12209.

[60] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018. doi: 10.1109/CVPR.2018.00212.

[61] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In Jan-Olof Eklundh, editor, *Computer Vision — ECCV '94*, pages 151–158, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. ISBN 978-3-540-48400-4.

[62] Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas Funkhouser, and Sean Fanello. Activestereonet: End-to-end self-supervised learning for active stereo systems. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 802–819, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01237-3.

[63] Z. Zhang, S. Lathuilière, E. Ricci, N. Sebe, Y. Yan, and J. Yang. Online depth learning against forgetting in monocular videos. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4493–4502, 2020. doi: 10.1109/CVPR42600.2020.00455.

[64] Chao Qiang Zhao, Qi Yu Sun, Chong Zhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9): 1612–1627, 2020. ISSN 1862281X. doi: 10.1007/s11431-020-1582-8.

[65] Shanshan Zhao, Huan Fu, Mingming Gong, and Dacheng Tao. Geometry-aware symmetric domain adaptation for monocular depth estimation. *CoRR*, abs/1904.01870, 2019. URL http://arxiv.org/abs/1904.01870.

[66] Yunhan Zhao, Shu Kong, and Charless Fowlkes. When perspective comes for free: Improving depth prediction with camera pose encoding, 2020.

[67] Yiran Zhong, Yuchao Dai, and Hongdong Li. Self-supervised learning for stereo matching with self-improving ability. *CoRR*, abs/1709.00930, 2017. URL http://arxiv.org/abs/1709.00930.

[68] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619, 2017. doi: 10.1109/CVPR.2017.700.