

AN ANALYSIS OF STEREO DEPTH ESTIMATION UTILIZING ATTENTION
MECHANISMS, SELF-SUPERVISED POSE ESTIMATORS & TEMPORAL
PREDICTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

UTKU OĞUZMAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONIC ENGINEERING

MAY 2022

Approval of the thesis:

**AN ANALYSIS OF STEREO DEPTH ESTIMATION UTILIZING
ATTENTION MECHANISMS, SELF-SUPERVISED POSE ESTIMATORS
& TEMPORAL PREDICTIONS**

submitted by **UTKU OĞUZMAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronic Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Sinan Kalkan
Electrical and Electronics Engineering, METU _____

Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Osman Serdar Gedik
Computer Engineering, Ankara Yıldırım Beyazıt University _____

Date: 18.05.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Utku Oğuzman

Signature :

ABSTRACT

AN ANALYSIS OF STEREO DEPTH ESTIMATION UTILIZING ATTENTION MECHANISMS, SELF-SUPERVISED POSE ESTIMATORS & TEMPORAL PREDICTIONS

Oğuzman, Utku
Master of Science, Electrical and Electronic Engineering
Supervisor : Prof. Dr. Abdullah Aydın Alatan

May 2022, 111 pages

By the recent success of deep learning, real-world applications of stereo depth estimation algorithms attracted the interest of many researchers. Using the available datasets, synthetic or real-world, the researchers begin analyzing their ideas for practical applications. In this thesis, a thorough analysis is performed of such an aim. The state-of-the-art stereo depth estimation algorithms are tried to be improved by incorporating attention mechanisms to the current networks and better initialization strategies in time. For this purpose, different amounts of attention modules are applied to one of the most successful stereo depth estimator networks. The performance of the proposed attention-based neural networks that is trained with the synthetic stereo datasets under a supervised setting is compared against the performance of a baseline algorithm and it yielded superior results. When these neural networks are finetuned using a small annotated real-world dataset, the baseline algorithm had a better performance. Secondly, the temporal information available in the synthetic datasets is leveraged by teaching the proposed neural network how to initialize the current iteration by using the previous predictions. Finally, in order to finetune the neural network better for real-world use with the temporal information, a large unannotated real-world dataset is utilized under a self-

supervised training setting using ego-pose estimation and optical flow networks. In general, it is observed that these settings yield better results against state-of-the-art methods in the synthetic-to-real world supervised training settings, and they are comparable after the finetuning operation.

Keywords: Stereo Depth Estimation, Attention Modules, Self-supervised Learning, Finetuning

ÖZ

DİKKAT MEKANİZMALARINI, KENDİ-KENDİNİ DENETLEMEYLE ÖĞRENİLMİŞ POZ KESTİRİCİLERİ VE ÖNCEKİ TAHMİNLERİNİ KULLANAN STEREO DERİNLİK KESTİRİCİLERİN BİR ANALİZİ

Oğuzman, Utku
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. Abdullah Aydın Alatan

Mayıs 2022, 111 sayfa

Derin öğrenmenin yakın tarihteki başarısıyla birlikte, stereo derinlik kestirme algoritmalarının gerçek dünya uygulamaları birçok araştırmacının ilgisini çekmeyi başarmıştır. Kullanıma hazır olan birçok sentetik ve gerçek dünya veri seti sayesinde, araştırmacılar fikirlerini pratik uygulamalar için analiz etmeye başlamıştır. Bu tezde benzer bir yaklaşım için dikkatli bir analiz gerçekleştirmiştir. En gelişmiş stereo derinlik algoritmaları, dikkat mekanizmalarının ve daha iyi başlatma stratejilerinin dahil edilmesiyle birlikte iyileştirilmeye çalışılmıştır. Bu amaç için, en başarılı stereo derinlik algılama ağlarından birine farklı miktarlarda dikkat modülleri eklenmiştir. Tasarlanan dikkat-tabanlı nöral ağın denetimli öğrenme yöntemiyle ve sentetik veri setleri kullanılarak eğitilmesiyle birlikte ortaya koyduğu performans baz alınan algoritmalarla karşılaştırılmıştır, daha iyi sonuçlar elde edilmiştir. Bu nöral ağların ince ayarını yapmak için küçük, etiketlenmiş ve gerçek-dünyaya ait görseller bulunduran bir veri seti kullanıldığında baz alınan algoritmanın daha iyi sonuçlar ortaya koyduğu gözlemlenmiştir. İkinci olarak, veri setinde var olan zamansal bilgilerden, nöral ağa şu anki özyinelemesini önceki tahminlerini kullanarak başlatmasını öğretmek yoluyla istifade edilmeye çalışılmıştır. Son olarak, gerçek dünya uygulamaları için ve tasarlanan nöral ağın ince ayarını daha iyi yapabilmek

için, bir poz kestirme ve optik akı ağı ve büyük bir etiketlenmemiş gerçek-dünya veri seti kullanılıp bir denetimsiz öğrenme ince ayar operasyonu gerçekleştirilmiştir. Genel olarak, sentetik görsellerle eğitip gerçek dünya işlerinde denenen test yapılarında ağılarımız baz alınan ağıdan daha iyi performans göstermiştir. İnce ayarlar sonunda ise sonuçlar karşılaştırılabilir düzeylerde kalmıştır.

Anahtar Kelimeler: Stereo Derinlik Kestirimi, Dikkat Modülleri, Kendi Kendini Denetimli Öğrenme, İnce-ayar

To My Family

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Prof. Dr. A. Aydın Alatan, for all his efforts, guiding me in the computer vision field, and believing in me in my every step.

I can never express my gratitude to my mother, Zehra Oğuzman, who always created the most comfortable environment everywhere, every time so that I could learn more with least amount of time and effort.

And to quote Franz Kafka, “One learns when one has to, one learns when one needs a way out, one learns at all costs.”

TABLE OF CONTENTS

ABSTRACT.....	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS.....	xi
LIST OF TABLES	xv
LIST OF FIGURES	xvii
LIST OF ABBREVIATIONS	xx
1 INTRODUCTION	1
1.1 Motivation and Problem Definition	1
1.2 Scope of the Thesis	2
1.3 Methodological Contribution of the Thesis	3
1.4 Outline of the Thesis	3
2 FUNDAMENTALS OF STEREO GEOMETRY, CONVOLUTIONS AND ATTENTION MODULES.....	5
2.1 Notation.....	5
2.2 Pinhole Models.....	5
2.2.1 Projection Using the Homogenous Coordinates	6
2.3 Convolutions	7
2.3.1 Pooling Layers	8
2.3.2 Filter Hyperparameters	9
2.4 Attention Modules.....	11

2.4.1	Scale Operation	13
2.4.2	Masking Operation	13
2.4.3	Addition & Normalization Operation.....	13
2.4.4	Concatenation Operation	13
2.4.5	Linear Layer	14
2.4.6	Positional Encoding Module	14
3	RELATED WORKS ON LEARNING-BASED STEREO DEPTH ESTIMATION.....	15
3.1	Introduction.....	15
3.1.1	Traditional Methods	15
3.1.2	Deep Learning	18
3.1.2.1	Supervised Deep Learning	18
3.1.2.2	Self-Supervised Deep Learning	23
3.2	Self-Supervised Depth Estimation with an Ego-Pose Estimator Network	26
3.3	Stereo Depth Estimation Methods Using Iterative Refinements	28
3.4	Stereo Depth Estimation Methods Using Attention Mechanisms	32
3.5	Comparison of the Methods in the Literature	37
3.5.1	Error Metrics for Stereo Depth Estimation	37
3.5.2	Depth Estimation Comparison	37
4	EFFECTS OF ATTENTION MODULES, INITIALIZATIONS AND SELF SUPERVISED TRAINING ON STEREO DEPTH ESTIMATION	39
4.1	Experimental Settings	39
4.1.1	Selected Network Architecture.....	39
4.1.2	Optimization	40

4.1.3	Datasets	40
4.2	Analysis 1: Attention-based Improvements for RAFT-Stereo Algorithm	43
4.2.1	Hyperparameter Search.....	45
4.2.2	Experiment 1 - Training the Network with Various Number of Self & Cross Attention Modules	47
4.2.2.1	Synthetic-to-Real Generalization Performance	48
4.2.2.2	Comparing Performance on Textured and Textureless Areas...	50
4.2.2.3	Finetuning the Networks.....	51
4.2.3	Experiment 2 - Training the Network with Self-Attention Modules Only	54
4.2.3.1	Synthetic-to-Real Generalization Performance	55
4.2.3.2	Comparing Performance on the Textured and Textureless Areas	57
4.2.3.3	Finetuning the Networks.....	58
4.2.4	Experiment 3 - Training the Network with Cross-Attention Modules Only	60
4.2.4.1	Synthetic-to-Real Generalization Performance	60
4.2.4.2	Comparing Performance on the Textured and Textureless Areas	62
4.2.4.3	Finetuning the Networks.....	63
4.2.5	Discussion of the Analysis 1	65
4.3	Analysis 2: Exploitation of Temporal Information in RAFT-Stereo Algorithm.....	67
4.3.1	Experiment 1 - Using Previous Predictions and Optical Flow to Initialize RAFT-Stereo Algorithm.....	68

4.3.1.1	Hyperparameter Search	72
4.3.1.2	Synthetic-to-Real Generalization Performance.....	73
4.3.1.3	Finetuning the networks	78
4.3.2	Experiment 2 - Finetuning RAFT-Stereo Algorithm with Ego-Pose Estimation Network and Optical Flow Network.....	81
4.3.2.1	Performing finetuning operation on the RAFT-Stereo algorithm with a standard ego-pose estimator	84
4.3.2.2	Performing finetuning operation on the RAFT-Stereo algorithm with an ego-pose estimator that uses optical flow information.....	86
4.3.3	Discussion of the Analysis 2	90
5	CONCLUSIONS	91
	REFERENCES	93

LIST OF TABLES

TABLES

Table 3.1 Comparison of the performance of 4 stereo depth estimation method ...	38
Table 4.1 The tasks and the utilized datasets during those tasks are explained.....	42
Table 4.2 The hyperparameter grid search combinations for networks with attention modules	46
Table 4.3 The result of the hyperparameter search, validated on KITTI training dataset. The traditional % error used for KITTI is considered to be the 3px % error	46
Table 4.4 Comparison of the networks that are trained with synthetic images only	48
Table 4.5 The synthetic-to-real experiments. The networks are only trained on SceneFlow datasets and validated with KITTI training dataset.....	51
Table 4.6 The performance comparison of finetuned networks by KITTI training dataset	52
Table 4.7 The synthetic to real experiment with self-attention modules	55
Table 4.8 The synthetic-to-real experiments with networks that is built with self-attention modules.	57
Table 4.9 The comparison between the proposed networks in experiment 2 and competing networks.	58
Table 4.10 The synthetic to real generalization results, compared with the best competing algorithms.....	62
Table 4.11 Performance on the Textured and Textureless areas	63
Table 4.12 Performance comparison of the networks finetuned with the KITTI dataset	65
Table 4.13 The synthetic to real task with all of the proposed networks and the baseline network [4].....	66
Table 4.14 The results of the proposed finetuned networks and the finetuned baseline network [4].....	67

Table 4.15 Hyperparameter grid search values	72
Table 4.16 Results of hyperparameter search for the network with 1x1 CNN filter	72
Table 4.17 The result of the hyperparameter search for the network with U-Net...	73
Table 4.18 Quantitative comparison of the networks and the baseline network.....	74
Table 4.19 The baseline network and the proposed networks finetuned with KITTI 126 training subset.....	78
Table 4.20 Quantitative results of the network with the standard ego-pose estimator vs the finetuned baseline network	86
Table 4.21 Quantitative network results with an ego-pose estimator that uses optical flow vs the finetuned baseline network.	88

LIST OF FIGURES

FIGURES

Figure 2.1 Pinhole camera setting [7]	6
Figure 2.2 The convolution filter operating on a layer [125].....	8
Figure 2.3 (a) Max pooling, (b) average pooling operations [125].....	8
Figure 2.4 K many convolution filters that belongs to one layer [125].....	9
Figure 2.5 Stride operation shown in a 1D tensor for the sake of simplicity [125]..	9
Figure 2.6 Zero-padding of the feature map with the required number of zeroes [125].....	10
Figure 2.7 Four types of normalization operations [125]	10
Figure 2.8 (a) Transformers architecture, (b) an attention module, (c) one head [8]	12
Figure 3.1 Taxonomy of the network architectures for stereo-based disparity estimation, focusing on different component of the pipeline (Revised from [15]).	16
Figure 3.2 (a, b) Two networks, (c) their appearance loss, (d) multi-scale loss [6]	27
Figure 3.3 RAFT-Stereo architecture is the architecture mainly used in the thesis [4].....	28
Figure 3.4 The correlation lookup operation around current disparity estimate.....	30
Figure 3.5 GRU modules in the Multi-Level Update Operator stage [4]	31
Figure 3.6 The STTR Architecture [61]	32
Figure 3.7 Alternating cross & self-attention modules in STTR architecture [61].	33
Figure 3.8 (a) Geometric constraints of stereo matching, (b) the attention mask [61]	35
Figure 4.11 An example from every dataset utilized in this thesis: The first row is an SceneFlow (FlyingThings3D and Monkaa) examples, the second row is ETH3D and Middlebury examples, the last row is a KITTI example.....	41
Figure 4.2 General architecture of the Analysis 1 (Revised from [4])	45
Figure 4.3 A general architecture for the first experiment of the Analysis 1	47
Figure 4.4 Comparison of the baseline network with the proposed networks.....	49

Figure 4.5 A typical example of the effects of the Laplacian filter.....	50
Figure 4.6 Examples from finetuned networks in this experiment where the cascade of self and cross attention modules are utilized.....	53
Figure 4.7 The general architecture of the second experiment (Revised from [4])	54
Figure 4.8 The qualitative results of the synthetic to real generalization test with self-attention modules	56
Figure 4.9 The qualitative results of the networks that have self-attention only and the baseline network.....	59
Figure 4.10 The architecture of the network in Experiment 3 (Revised from [4])	. 60
Figure 4.11 The synthetic-to-real generalization performance of the proposed networks and the baseline network [4].....	61
Figure 4.12 The qualitative results of the finetuned networks and the baseline network.....	64
Figure 4.13 The architecture of the network with 1x1 CNN filter (Revised from [4])	69
Figure 4.14 The architecture of the network with U-Net (Revised from [4])	70
Figure 4.15 The U-Net blocks utilized in the network with U-Net which utilizes 3 previous predictions.....	70
Figure 4.16 The architecture of the network that utilizes optical flow (Revised from [4], [46])	71
Figure 4.17 Qualitative results of the networks with 1x1 CNN filter and the baseline network [4]	75
Figure 4.18 Qualitative results of the networks with U-Net and the baseline network [4]	76
Figure 4.19 Qualitative results of the network with the optical flow and the baseline network.....	77
Figure 4.20 Qualitative results of the finetuned baseline network and the proposed finetuned networks with 1x1 filters	79
Figure 4.21 Quantitative results of the finetuned baseline network and the proposed finetuned networks with U-Net	80

Figure 4.22 Quantitative results of the finetuned baseline network and the proposed finetuned networks with optical flow.....	81
Figure 4.23 Architecture of the network with the standard ego-pose estimator (Revised from [4]).....	84
Figure 4.24 Qualitative results of the network with the standard ego-pose estimator vs the finetuned baseline network.....	85
Figure 4.25 The proposed ego-pose estimator architecture inspired by Flowdometry algorithm [131]	87
Figure 4.26 Architecture of the network with an ego-pose estimator that uses optical flow information (Revised from [4])	87
Figure 4.27 Qualitative results of the network with an ego-pose estimator that uses optical flow vs the finetuned baseline network.....	89

LIST OF ABBREVIATIONS

ABBREVIATIONS

3D : 3 Dimensional

AR : Augmented Reality

CNN : Convolutional Neural Network

EPE : End Point Error

GAN : Generative Adversarial Network

GRU : Gated Recurrent Unit

MRF : Markov Random Field

SfM : Structure from Motion

SVM : Support Vector Machine

VR : Virtual Reality

CHAPTER 1

INTRODUCTION

One of the most impressive capabilities some animals developed for survival is depth perception. Vision is an inseparable part of tool usage, exploration, locomotion, planning and understanding many concepts. Computer vision started as a research field to reverse engineer this capability for robotic applications. From the famous 1966 Summer Vision Project [1] where initial researchers thought it could be achieved within a summer, to today, researchers frequently come up with many novel methods and build upon many different vision ideas. From SfM methods to SVMs, and even early neural network ideas, researchers were after the quality and the quantity of the vision humans possess using engineering methods.

1.1 Motivation and Problem Definition

After the prominent work of Hinton [2] in 2012, deep networks that are trained with supervised targets, optimized through backpropagation and methods such as convolution [3], became suddenly popular due to the limited computation opportunity it provides. This opportunity let many researchers start working on the problem of vision without much effort. A considerable portion of this research focuses on the stereo dense depth estimation problem, which is the problem of estimating depth of each pixel with respect to one of the cameras.

With the rise of deep learning, utilizing these calculations for practical real-life purposes became plausible. Depth estimation is a prime necessity when it comes to AR/VR applications, drones with autonomous capabilities and most importantly autonomous cars. Autonomous car applications are the most serious daily robotic application that needs to be 100% safe in order not to danger any bystander's life.

Hence, datasets that focus on road scenarios such as KITTI[4] and ETH3D[5] have been gathered together and shared as open-source for the research communities. The computer vision research community utilizes these famous datasets frequently.

Some significant attempts to estimate a dense depth map from stereo images include [4], [5]. They require a dense depth label as the ground truth signal, which is hard to obtain in the necessary amount. Therefore, self-supervised training methods are preferred in some phases, for finetuning a self-supervised network or training them from the start, such as [6].

This thesis focuses on improving stereo depth estimation, especially one of the current best stereo depth estimation system RAFT-Stereo [4]. Finetuning a pre-trained stereo network offers great flexibility to make accurate inferences for dense depth estimations without a large real-world ground truth dataset.

1.2 Scope of the Thesis

This study aims to improve the performance of a stereo depth estimation with various attentional and temporal techniques. First, the effects of adding various attention modules are investigated to the RAFT-Stereo algorithm, specifically changing the correlation volume with attention modules. Attention modules have the ability to shuffle the context around them. The hypothesis for this work is that they have the ability to in-paint the textureless patches in high dimensional space to match them more precisely, even though those areas considered alone lack features. Hence the effect they pose against the textured areas will be investigated as well. Second, effects of various temporal information are investigated in RAFT-Stereo algorithm. Previous frames have valuable information when it comes to depth estimation and the supervised training methods for stereo networks. One can extract important information by trying to predict an input from the other inputs via the intermediate concepts like depth, ego-pose estimation, and optical flow.

All of the work in this thesis is based on RAFT-Stereo [4] algorithm, since it is one of the best and up-to-date stereo depth estimation methods of stereo matching. Although RAFT-Stereo is on the top 3 of the Middlebury 2014 stereo leaderboards, it still has some room for improvement. All of the codes developed in this thesis available will also be available on Github soon.

1.3 Methodological Contribution of the Thesis

To sum up, the main contributions of this study can be expressed as follows:

- An assessment of the effect of various attention modules to improve the stereo depth estimation,
- An inspection of the utilization of various past disparity estimates to improve the stereo depth estimation performance.
- An inspection of the effect of a self-supervised finetuning procedure using the temporal information of ego-pose estimation, optical flow, and previous predictions.

1.4 Outline of the Thesis

This thesis work consists of 5 chapters. Chapter 1 introduces the motivation and contributions. Chapter 2 presents relevant stereo geometry subjects, namely pinhole cameras, stereo depth, disparity, and relevant deep learning subjects, namely convolutions and attention modules. Chapter 3 informs the reader about the current literature and approaches to supervised depth estimation, self-supervised depth estimation, and vision transformers. Chapter 4 explains the proposed models, tests, and results. Chapter 5 finalizes the thesis by stating the conclusion drawn from the experimental analysis.

CHAPTER 2

FUNDEMENTALS OF STEREO GEOMETRY, CONVOLUTIONS AND ATTENTION MODULES

The self-supervised stereo depth estimation methods that will be discussed in Chapter 3 rely on some key concepts of the fundamentals of stereo geometry and deep learning. Those concepts are covered in this chapter.

2.1 Notation

The general notation that persists in this thesis is that matrices are denoted with bold letters, and vectors are indicated with a bar on top:

$$Ex: \mathbf{A}\bar{x} = \bar{b}$$

2.2 Pinhole Models

A camera can be modeled as an optical device that maps the electromagnetic radiation bouncing from the surfaces residing in the 3D world into a 2D image plane made of a light-sensitive sensor array. This thesis section discusses the mostly linearized geometric calculations under this setting and uses rectified stereo images.

Suppose that a camera projection is performed on the plane where $z = f$. That plane at f is called *image plane*, and f is called *focal length*. The *camera center* is the origin of this coordinate system. The center of the image plane is called the *principal point*. The line that goes through the camera center and the principal point is called the *principal axis*. The plane parallel to the image plane and goes through the camera center is called the *principal plane*. The projection of a 3D point \bar{X} can be computed as the intersection of the corresponding line, which passes through the camera center and \bar{X} and the image plane (Figure 2.1).

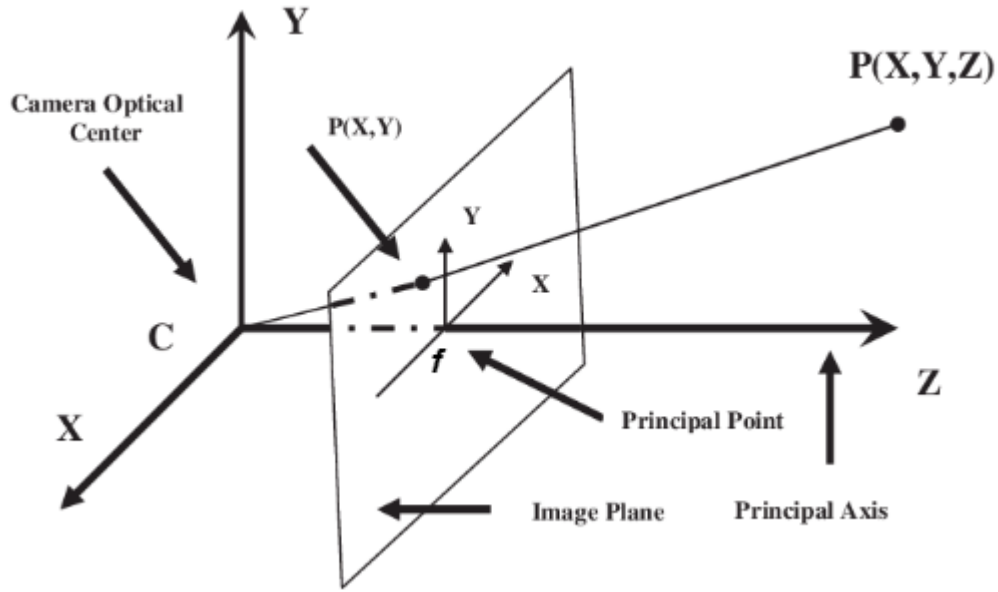


Figure 2.1 Pinhole camera setting [7]

2.2.1 Projection Using the Homogenous Coordinates

The pinhole projection can be expressed as follows:

$$(X, Y, Z)^T \rightarrow \left(f \frac{X}{Z}, f \frac{Y}{Z}\right)^T \quad (1)$$

However, this is a nonlinear operation and cannot be represented with simple 3 by 3 matrix transformations due to the division operation. Hence the homogenous coordinate system is utilized to represent this projection as a linear mapping.

Every point in this coordinate system has an additional dimension now: $(X, Y, Z, 1)^T$. The transformations within this space are done using a 4D transformation matrix. The representation of a standard 3D point after the conversion to the homogenous transform can be acquired by this linear multiplication now:

$$\begin{bmatrix} fX \\ fY \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2)$$

The 3 by 4 matrix on the right-hand side of the equation is called *camera projection matrix* and it is shortened by P:

$$\bar{x} = \mathbf{P}\bar{X} \quad (3)$$

The pinhole camera equation defined in equation (3) and Figure 2.1 assumes that the origin of the image plane is at the principal point. But this may not be true in all cases. There may be an offset between the two. Hence equation (3) must be corrected as:

$$(X, Y, Z)^T \rightarrow \left(\frac{fX}{Z} + c_x, \frac{fY}{Z} + c_y\right)^T \quad (4)$$

Hence, equation (2) can also be corrected as:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (5)$$

If one has been given a rectified & vertically aligned stereo image couple where a certain 3D point is marked in both images, the difference in the x-axis of those 2 points gives the *disparity*. *Baseline* is the given distance between 2 cameras. With all that knowledge, one could infer the depth of that 3D point with respect to the principal plane:

$$depth = \frac{focal\ length \times baseline}{|disparity| + (c_{x,l} - c_{x,r})} \quad (6)$$

2.3 Convolutions

Convolutions in deep neural networks are almost the most common module when vision tasks are considered. The convolution layers are essentially filters that perform convolution operations as it scans the input image (I), scanning through its height (H) and width (W) dimensions and finally a nonlinear function is applied to the result. The hyperparameters of a filter include filter size (F) and stride (S). The resulting output (O) is called *feature map* or *1 layer of the feature map*, to be exact.

When applied at the same layer, many of these filters create whole feature maps with the *channel size* (C). These features calculated for a specific location in the image can be used to correlate certain parts of the stereo image pairs. Some similarity metrics applied at the resultant feature maps can help us match most of the pixels in the images. This method assumes the occlusion areas are negligible, however it aims to fill those areas using the context in both images.

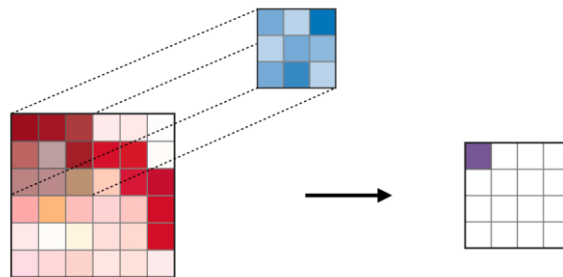


Figure 2.2 The convolution filter operating on a layer [125]

2.3.1 Pooling Layers

Pooling is a down-sampling operation, typically applied after a convolution, it is a spatially-equivariant operation, it is not a learnable filter. The most common pooling operations are *max pooling* (where the filter applies a max operation in its receptive field) and *average pooling* (where the filter averages every entry in its receptive field). Max pooling preserves most of the detected features, while average pooling is used to down-sample the feature map for computational costs reasons.

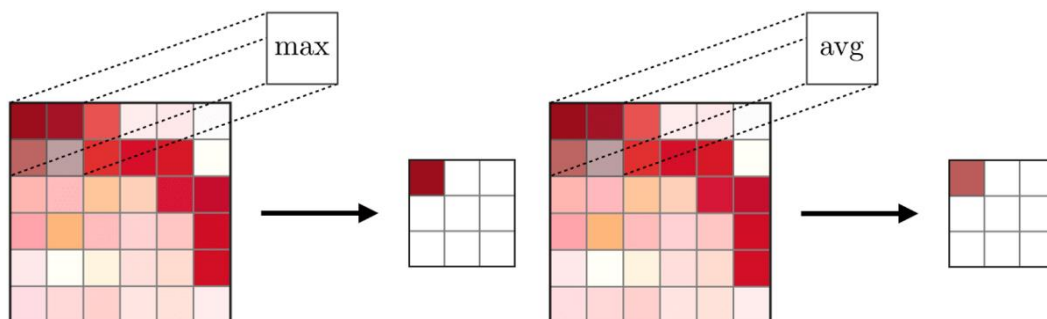


Figure 2.3 (a) Max pooling, (b) average pooling operations [125]

2.3.2 Filter Hyperparameters

These convolution filters have many hyperparameters that affect the course of the convergence process of the neural network. The important ones are these:

Channel Dimension: A standard convolutional filter is a 3D filter. Its height & width are equal to each other. Applying K many of them to the (C, H, W) dimensions of the previous layer makes them a 3D filter $(C$ by F by $F)$. The channel dimension of the layer is determined by the number of 3D filters that transformed the representation of the previous layer in every layer.

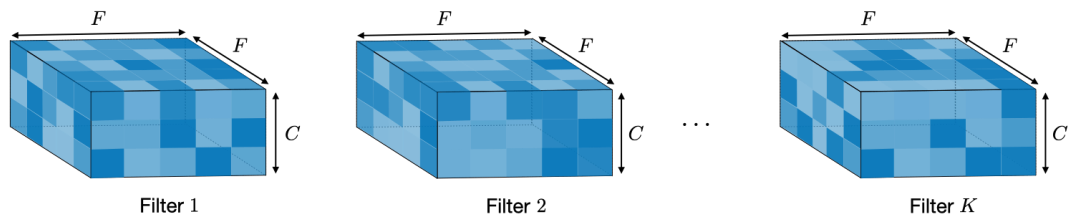


Figure 2.4 K many convolution filters that belongs to one layer [125]

Stride: Stride S is the number of pixels that the convolution filter jumps in the height and width axis after it finishes its previous convolution operation. There is usually no stride in the channel dimension.

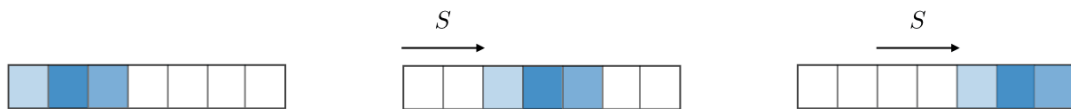


Figure 2.5 Stride operation shown in a 1D tensor for the sake of simplicity [125]

Zero-Padding: Zero-padding is the process of adding P number of zeroes to each side of the feature map. It is usually used to match the new feature maps' height and width size to the specified/required input size for a specific architectural selection.

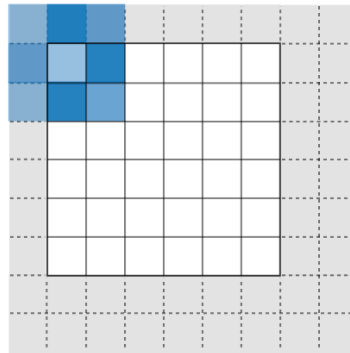


Figure 2.6 Zero-padding of the feature map with the required number of zeroes [125]

Normalization: Normalization is the act of calculating the average and the variance of some data points along a dimension and then normalizing those data points by subtracting the average from every data point and dividing every data point by the square root of the variance. The type of normalization indicates which data points are included in the normalization calculation and along which dimensions they calculated.

- **Batch normalization:** Batch normalization focuses on standardizing every input in the mini-batch direction during training. Hence for every input of shape (N,C,H,W) entering the network, the normalization process is conducted along the mini-batch direction N (Figure 2.7).

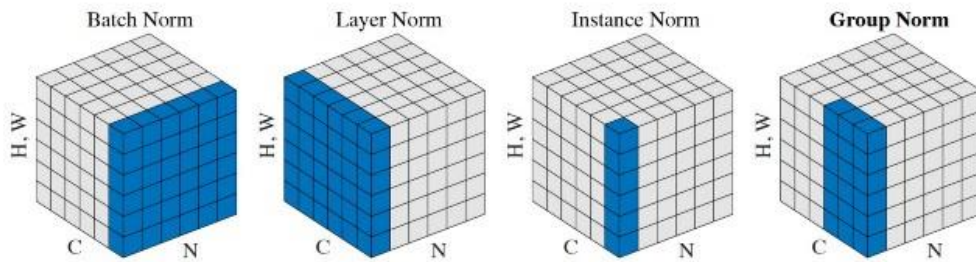


Figure 2.7 Four types of normalization operations [125]

- Layer normalization: Layer normalization is calculated over the whole (C,H,W) dimensions instead. Different samples do not affect other samples' normalization parameters (Figure 2.7).
- Instance normalization: Instance normalization is calculated per channel along the H & W axis. Every sample in the minibatch is independent of each other in terms of calculating their normalization parameters (Figure 2.7).
- Group normalization: Group normalization is similar to the instance normalization, however instead of calculating it on only one channel, it includes some nearby channel *groups*. Hence, this method is naturally between the Instance Normalization method and Layer Normalization method (Figure 2.7).

2.4 Attention Modules

The aim of the attention mechanisms in general is to make the computations focus only on the parts of the input that matter by making the network pick some representations of the data over others. In computer vision, especially in feature matching problems, the issues around textureless areas can benefit from these concepts. Attention mechanisms can force textureless areas to attend to nearby textured areas to anchor themselves. Gathering weighted V vectors in this manner can create echoes of the textured areas into the textureless areas in terms of edges and contours. These echoes may create in-paintings of the textureless areas and raise the opportunity for a more precise disparity map regarding feature correlations between stereo image pairs.

The attention modules are another mechanism to be utilized in this thesis. The ultimate purpose of attention modules in vision tasks is to blend certain types of features together [8]. Basically, an attention module takes a set of embeddings (H.W many vectors, with C embedding dimensions). It first applies 3 different learnable linear transformations W_K , W_Q , W_V on the initial vector set to get the new K , Q & V vector sets, and it treats them as continuous *key*, *query*, and *value* tuples in a database.

Just like querying a database, there are many values (V) in the database; those values have corresponding keys (K) to search the database quickly and access the necessary values. The key is the most similar to the current query (Q) becomes the result. However instead of calculating the results in a discrete/hard fashion, attention modules use a soft output, meaning the resultant answer for the query is the weighted combination of the V vector sets according to how much their K and the Q vectors matched up.

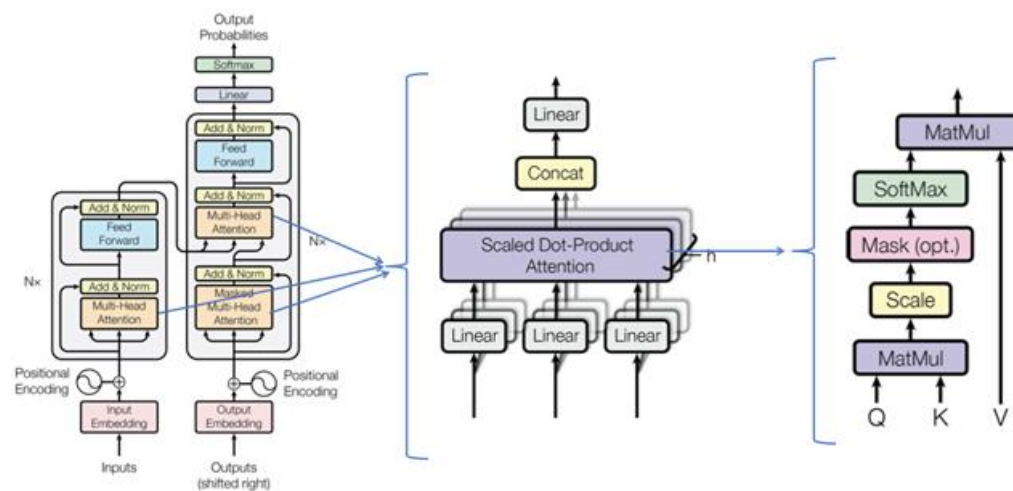


Figure 2.8 (a) Transformers architecture, (b) an attention module, (c) one head [8]

Implementation details are explained in Chapter 3.4. After the original vector set is transformed into Q , K & V vector sets, Q and K sets are multiplied. Every vector in Q is multiplied with every vector in K by the dot-product. Using the dot-product as the similarity metric, vectors in Q can *attend* to any other vector in K , no matter where they are located. Then, the module composes the output by using the vectors in V , first turning the dot-product result into probabilities (with the help of a softmax operation) and multiplying the V vectors with the softmax result. Its implementation includes the following other details.

2.4.1 Scale Operation

By the heuristic exploration, Vaswani et al. [8] observed that they need to scale the resultant attention matrix by the square root of the number of embedding dimensions (d) to prevent the exploding and vanishing of the gradients. With this operation, the resultant vector from the multiplication is always scaled to have norm of \sqrt{d} at most.

2.4.2 Masking Operation

If the architecture requires specific locations to be unattendable, such as causality constraints or stereo matching scenarios, masking the weights of unattendable locations via multiplying them by $-\infty$ makes their softmax output 0, and the final softmax still sums up to 1.

2.4.3 Addition & Normalization Operation

The Addition is a ResNet [9] style skip connection technique to prevent gradients from exploding or vanishing. Also, layer normalization after the Addition is most common.

2.4.4 Concatenation Operation

Vaswani [8] introduces the idea of heads (h). The idea is that one can divide the embedding dimension into h heads and process them in a parallel fashion to speed up the process. In theory, the system can still discover some features which are independent of other features. This idea is mainly favored due to the data parallelization opportunities in GPUs. After these separate head calculations, the results of each head are concatenated into one vector.

2.4.5 Linear Layer

A final linear transformation is observed to stabilize the training. It prepares the output for the next layer and gives the network more expressive power.

2.4.6 Positional Encoding Module

Suppose positional encoding techniques are not utilized in Transformers. In that case, no matter where a vector is located in the V, K, and Q vector sets, the attention vector it produces will always be the same. Positional encodings try to inject a sense of position into attention modules either by summing specific sine functions [8], adding them into the product of Q and K [11]–[14], or to the product of the softmax and V vectors [14]. They can be learned [12] or fixed [10], [11]. The equations regarding the *relative positional encoding* method have been explained in Equation 23 in Chapter 3.4.

CHAPTER 3

RELATED WORKS ON LEARNING-BASED STEREO DEPTH ESTIMATION

In this chapter, the related works from the literature are briefly explained.

3.1 Introduction

The task of disparity prediction between rectified stereo image pairs is an essential problem in computer vision. It is a challenging task due to the non-ideal conditions in the image pairs, such as occlusions, textureless areas, reflective areas, transparent areas, thin structures, repetitive textures, and noise. Matching cost aggregations are also unfortunately often ambiguous. In naïve implementations, erroneous matches might have a lower cost than the correct ones due to the mentioned disturbances above. Moreover, the error in the disparity term affects different depths with different ratios, the further the region of interest, the smaller the error should be. A taxonomy for stereo depth estimation can be seen in Figure 3.1.

3.1.1 Traditional Methods

Traditionally there are four steps in stereo feature matching: feature extraction, cost aggregation and regularization, disparity estimation, post-processing.

Feature Extraction: The traditional methods, like SGM [16] and cost filtering [17], are robust and efficient cost aggregation methods. They are not differentiable; therefore, they cannot be trained to increase their performance. They do not need the ground truth depth labels, however when ground truths are available, the traditional methods like [18]–[20] can learn the hyperparameters.

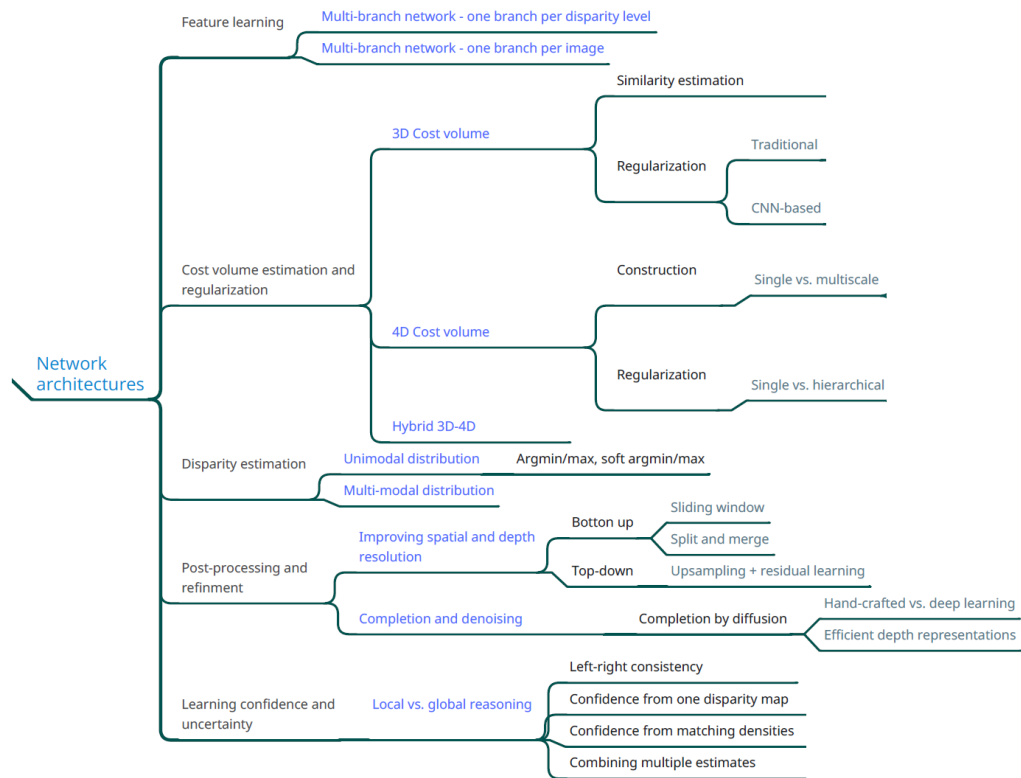


Figure 3.1 Taxonomy of the network architectures for stereo-based disparity estimation, focusing on different component of the pipeline (Revised from [15])

Naturally, the pixels in one image should not be matched to multiple pixels in the other image. This geometric constraint can be helpful in resolving some ambiguities. Ohta et al. [21] was the first attempt to use dynamic programming where intra and inter-epipolar line information are combined with a uniqueness constraint. This constraint is missing in most learning-based methods due to the shortcomings of the current view of stereo matching methods that construct a *cost volume*, which will be explained later. The approaches that consider the disparity prediction from a sequence-to-sequence matching can avoid this issue.

Cost aggregation and Regularization: Early traditional work in the field mainly focused on designing better matching costs [22], [23]. Commonly used methods included mutual information [16], normalized cross-correlation [24], Census transforms followed by Hamming distance [25].

Local methods, such as [26] and [27] try to find the matching points within a pre-defined window. There are algorithms that use handcrafted schemes to find local correspondences [16], [17], [28], [29]. Desirable disparities might have either the lowest matching costs (sum of absolute differences, SAD) or the highest correlation costs (normalized cross-correlation, NCC). The optimization strategies utilized in the local algorithms are denoted as *winner-takes-all* (WTA) algorithms [30]. Global algorithms formulate the stereo matching problem as energy minimization problems, utilizing algorithms like MRF-based optimization methods. Global methods generally achieve better performance; however, they have NP-hard complexities. Klaus et al. [31] use belief propagation to solve it, Kolmogorov et al. [32] use graph-cut to get suboptimal results. Utilization of the spatial context in the scene [33] is common practice. Semi-Global Matching (SGM) [16] approximates the MRF-based methods by performing a cost aggregation in all directions to improve the accuracy and efficiency [34]. However, traditional methods still suffer from inaccuracy (local algorithms) or high computational costs (global algorithms).

Disparity Estimation: Early work also focused on efficient inference algorithms [35]. There are parametric models, such as slanted plane [36], that aim to reduce the optimization parameters. There are other algorithms to avoid the full disparity space using PatchMatch [37] and super-pixels [38]. Some approaches use random forests and decision trees to converge to a solution quickly [39]–[42]. Hierarchical algorithms [43] use slanted support windows to amortize the matching cost computation in tiles. These methods require camera-specific learning or serious post-processing on their results.

Post-processing: Given a set of noisy matches, the cost optimization stage aims to recover a consistent depth map, subjected to geometric constraints, such as smoothness and planarity. These constraints can be naturally formulated as an optimization problem, maximizing some visual similarity measure subject to these constraints. Geometric constraints, such as occlusion and matching uniqueness, led to the success of non-learning-based methods [16], and it is mostly missing in the

learning-based techniques. For the occluded regions, pixels do not have a disparity by definition.

3.1.2 Deep Learning

The favorite method of most contemporary computer vision researchers' favorite methods are deep learning methods since Hinton et al. successfully trained a convolutional neural network with backpropagation via AlexNet in 2012. There are many computer vision tasks that deep learning helps, such as object detection [44] panoptic segmentation [45]; optical flow [46]; super-resolution [47] and generative networks [48]. After deep convolutional networks proved helpful, the computer vision field rarely used classical methods for the above tasks[49].

The most straightforward solution for a dense depth estimation task is to predict depth values for every pixel on the left image of the stereo pair. There are many setups for this task, such as semi-supervised monocular estimation [50], supervised stereo estimation [51] or supervised multi-view estimation [52]. There are also self-supervised counterparts of those setups [6], [53].

3.1.2.1 Supervised Deep Learning

The research of stereo matching with supervised deep learning can be mainly categorized into four categories: better feature matching [54], [55], better regularization [56], [57], learning all the depth estimation steps in an end-to-end fashion [57], [58] or refining the computed disparity [58]. The first category replaces the handcraft features with learned deep features. The second category learns to regularize the matching cost aggregation, such as spatially variant penalty parameters. The third category formulates the stereo matching problem as a supervised regression task that implements a combination of these different steps in a deep learning pipeline, hence one optimization signal flow optimizes the whole

architecture, rather than individual modules. Finally, the fourth category iteratively refines the solution.

Better feature matching: PSMNet [59] extended the matching accuracy of the previous work with pyramid feature extractions and stacked hourglass blocks. It increased the 3D convolutions layers to 25 for the matching cost aggregations. However, these changes increased the memory usage, and the computational cost, which is mitigated by down-samplings, and this action causes an inevitable loss of precision. The use of explicit matching cost volume results in more accurate prediction on the datasets such as KITTI [60] and FlyingThings3D [58]. Nevertheless, this increases the computational cost significantly and sometimes limits the operation resolution.

Another challenge of these cost volume systems is their limited disparity range. Disparity values, in theory, can range from zero to the image width, which can depend on the resolution of the images, the baseline distance of the cameras, and the distance of the object to the camera pair. The best works that use cost volumes are typically constrained with a maximum of 192 pixels [15]. This constraint is necessary to implement a memory-feasible method, however, is not flexible to the properties of the physical scene or camera setup. The close objects, potentially closer than the disparity of 192 pixels, may be essential in most robotic applications. STTR [61] removes this constraint to better estimate the depths of close objects.

Some readers might argue that these techniques are having difficulty generalizing outside the domain they were trained in; hence they cannot be readily used on datasets that do not have ground truth training data. There have been several efforts to improve the generalization ability of deep stereo networks, such as adding new network components [62] or generating additional data [63]. DSMNet [62] tries to improve the generalization ability of GA-Net architecture by normalizing the features used to construct the cost volume and utilizing a non-local graph-based filtering approach that reduces dependence of GA-Net on local patterns. DSMNet achieves better generalization than the prior work, however it still uses 3D

convolutions in its architecture design. This high computational cost limits the operation resolution of DSMNet. MADNet [64] tries to solve this domain adaptation problem with its MAD algorithm, which independently trains sub-portions of the network.

Some of the recent works [42], [43], [65] increased the efficiency of the disparity estimation while maintaining accuracy. They are based on three major ideas: sparse features for high resolution matching cost computation, efficient disparity optimization schemes that do not use explicit matching cost volumes, and iterative image warps using slanted planes to minimize image pair dissimilarity. HITNet [5] leveraged the planar geometry of the scene as a geometrical constraint in the network design by guiding the stereo predictions using those predicted tiles. In the forward pass, tile method of HITNet must decide, if each pixel lies on a plane. To learn this behavior, they impose several additional loss terms on the angle of the tiles and the decision weights.

STTR [66] utilizes Transformer with self and cross-attentions combined with the optimal transport theory to avoid using cost volumes. This design selection lets them match pixels explicitly and densely while imposing a uniqueness constraint.

After the success of Vaswani [8] in the field of NLP in 2017, researchers started working for more efficient architectures that can perform the same outcome to increase the adaptability of the new architectures. The researchers looked for hashing algorithms [14], hierarchical structures [107], methods to make the attention modules converge easier [11], or alter how the attention process works completely to run them faster [12]. This interest has carried over to the computer vision field, and researchers started looking for a way to implement such Transformers for computer vision tasks. Liu et al. [10] experimented with hierarchical transformers using shifting windows to ensure the Transformer architecture blends the context efficiently. Fan et al. [108] researched ways of reducing the quadratic requirements of the Transformers by processing the information in multiple scales, as in convolutions. El-Nouby et al. [109] looked for a way to faster Transformers by

transposing the Q, K & V vectors to reduce the required computational cost. Xiao et al. [110] experimented with the effect of using convolutions and transformation in the same architecture and concluded attention modules require early convolutions to converge easier.

Better Regularization: Deep learning was initially applied to improve matching costs in a stereo pipeline. Zbontar and LeCun [71] are the first to propose a network for evaluating a match score between image pairs. Their matching costs are processed using traditional methods such as semi-global matching, consistency checking, and filtering. Early works [55], [71]–[73] also trained Siamese networks to extract pairwise features or predict matching costs.

In stereo depth estimation research, the predominant approach has been the usage of 3D convolutional neural networks (ConvNet). First, a 3D cost volume is built by the enumeration of integer disparities, and then a 3D ConvNet was then utilized to filter the cost volume [59], [62], [74]–[77]. This enumeration could be a search range over the scanline on rectified stereo pairs or all possible disparity matches using plane sweep algorithms [78]. Filtering and regularization are usually applied to cost volumes next.

Some follow-up works [79], [80] focused on a down-sampled version of the cost volume to provide a trade-off between speed and accuracy. Fast approaches down-sample the cost volume in spatial and disparity space. Such algorithms attempt to recover fine details by edge-aware up-sampling layers. They can perform in real-time, however thin structures and depth edges create problems. There are also cascaded models [59], [77], [81]–[84] to search the disparity space in a coarse-to-fine fashion. [83] uses multiple residual blocks to improve the disparity prediction. [84] relies on a hierarchical cost volume, and with that, the network can be trained on high-resolution images and generate different resolution outputs on demand. However, all these methods are trained with expensive matching cost volumes or multiple refinement layers that prevent real-time performance.

GANet [77] introduced the SGA and LGA layers. SGA implements a differentiable version of SGM to improve matching cost aggregation, especially on occluded, textureless, and reflective areas. LGA aims to improve the disparity prediction on the objects' thin structures and depth edges to recover the loss of precision during down-sampling. SGM-Net [85] tries to enhance the performance of the traditional cost aggregations by predicting the penalty parameters of SGM using a neural net where [86] learned to fuse proposals by optimization in stereo matching. [87] looks at aggregating costs using a minimum span tree.

Most stereo matching networks can also be categorized as either 2D or 3D convolutions. ES-Net [88] adopts the ideas from PWC-Net [89] (an optical flow algorithm) to the stereo depth estimation problems. Given that warping images may introduce potential errors at low resolutions, ES-Net constructs a multi-scale cost volume at a higher resolution, which leads to better performance.

End-to-end Training: Mayer et al. [58] proposed the first end-to-end trainable stereo matching network based on FlowNet architecture [90]. They released a large synthetic dataset that made training convolutional networks for stereo possible. End-to-end networks are proposed to learn all steps jointly, yielding more accurate results [58], [91]–[93]. Some end-to-end methods, such as DispNet, link the matching and disparity prediction, and it directly computes the correspondence field between stereo images by minimizing a regression loss.

The efforts, as in GC-Net [75], have suggested incorporating explicit matching cost volumes that encode the cost of assigning a disparity for a pixel. It was the first algorithm that incorporates the 3D convolutions in the pipeline. In this setting, 3D convolutions are used as a differentiable approximation of the classical filtering algorithms such as SGM. GCNet incorporates feature extraction, matching cost aggregation, and disparity prediction into a single end-to-end training. Stereo images are first mapped through concatenation or correlation operator [74]. 3D convolutions layers then filter the cost volume before being mapped to a pixel-wise depth estimate through a differentiable argmin indexing operator.

Refining Disparity: In contrast, RAFT-Stereo uses only 2D convolutions and a cost volume by a single matrix multiplication and avoids 3D convolutions. Additionally, they introduced multi-level GRU units that maintain hidden states at multiple resolutions with cross-connections, however it still generates a single high-resolution disparity update.

Another line of work has looked at replacing the more costly components of the 3D networks with more lightweight modules. Liang et al. [82] first proposed a 2-stage refinement method for stereo. Bi3D [94] proposed estimating depth with a series of binary classification stages. Rather than calculating if an object is at a specific depth D , as current stereo methods do, Bi3D classifies areas as being closer than depth D or farther than depth D . When a longer time or more computational budget is available, it can compute depth queries with various levels of quantization.

To refine the disparity prediction, [83] uses a 2-stage ConvNet by first estimating the disparity prediction and then refining it. FADNet [95] also uses a 2-stage network design as well; similarly, the second one is used to refine the disparity prediction from the first network.

On the other hand, Lipson [4] suggested a multi-level correlation and refinement procedure that finds correlations between the stereo image couple and calculates the depth in a supervised way.

3.1.2.2 Self-Supervised Deep Learning

Since the ground truth data is, most of the time, hard to obtain (usually by other sensors like LIDAR or depth camera) or hard to annotate (both in terms of human perception and the tools available), to abandon the tedious task of gathering dense depth ground truths, self-supervised and weakly supervised methods are explored widely in the community. Research on self-supervised training usually explores three main elements: better architectures, loss functions, or image warping models.

Better Architectures: Zhong [96] suggested a self-supervised learning method for stereo matching using feature volumes. Many relevant works extend these ideas to valuable different directions [8], [24], [30].

Caron et al. [44] used self-supervision techniques similar to [111] to turn the attention maps into a semantic segmentation mask. Moreover Li et al. [66] used the default dense transformer modules to calculate an attention map, which they utilized as the cost matrix of an optimum transport problem to solve the stereo feature matching task.

PVStereo proposes a new ConvNet architecture, denoted as OptStereo, inspired by the traditional optimization-based methods. OptStereo first builds multi-scale cost volumes and uses an RNN unit to refine disparity predictions at high resolutions iteratively. This architecture avoids the error accumulation problem in the coarse-to-fine methods. PVM creates semi-dense disparity predictions, and the generated disparity images are used to supervise DCNN training.

Loss Functions: MonoDepth2 [6], despite its name, contains a stereo training procedure in the existence of stereo image pairs. They proposed a novel minimum reprojection loss designed to handle occlusions robustly. After the reprojection is applied, if there is an occlusion at the previous frame and no occlusion at the next frame in a specific pixel location, they ignore the greater loss created by the occlusion. They also propose an auto-masking loss to ignore training pixels that violate the assumption of the camera motions.

Moreover, Liu et al. [99] proposed Flow2Stereo, which leverages the geometric constraints behind stereoscopic videos, all relations between two consecutive frames and stereo image pairs, to predict disparity and optical flow estimation in a self-supervised manner. Tulyakov et al. [100] prepared a method to generate disparity estimations in a weakly-supervised way.

Zhong et al. [96] proposed a self-supervised stereo matching approach. It uses a novel training loss to exploit the *loop constraint* in the image warping process and improve the predictions in textureless areas.

Image Warping: Zhong et al. [96] aim to solve the stereo matching problem using warped images. They show that warping an input image to mimic its pair using depth as an intermediate variable is enough to converge the neural nets. To overcome the fact that multiple solutions satisfy this setup, they develop a 3D regularization method that pushes the trivial solutions in the high dimension feature volume. Their feature matching method chooses the disparity that minimizes both the appearance space and the high dimensional feature volume. They use the left-right consistency check loss function to handle the textureless areas better.

Deep3D [101] proposed a model with discretized depth to effectively utilize the novel view synthesis methods. [57] improved this approach by predicting continuous disparities, and [102] improved the results by including a left-right depth consistency term. Self-supervised training methods are extended with other consistency terms [103], semi-supervised data [104], [105], GANs [106], [107], and temporal formation [103]–[105].

Self-supervised training typically relies on making assumptions about the appearance of object properties (brightness consistency) and surface properties (e.g., Lambertian) between consecutive frames. The method in [111] shows a local structure-based appearance loss [54] significantly improves the depth estimation performance compared to a simple pairwise pixel difference [57], [101], [112]. The method in [113] extended this approach to include an error-fitting term. [114] combined it with an adversarial-based loss to drive the network to produce realistic-looking warped predictions.

The following research topics will be examined in the upcoming sections since therefore utilized during the proposed technique:

1. Self-supervised depth estimation methods with an ego-pose estimator network,
2. Stereo depth estimation methods using iterative refinements,
3. Stereo depth estimation methods using attention mechanisms.

3.2 Self-Supervised Depth Estimation with an Ego-Pose Estimator Network

The most famous self-supervised ego-pose and depth estimation examples are arguably MonoDepth2 [6]. The proposed technique has three different depth estimation methods: monocular, stereo, and monocular-stereo combination. For this thesis, the latter one is taken into the focus.

In this approach, Godard et al. [6] presented three novel ideas: a minimum reprojection loss for appearance matching, a full resolution multi-scale sampling method that performs all image sampling at the input resolution, and a novel and simple auto-masking loss to ignore the occluded training pixels.

Their self-supervised depth estimation method frames the estimation problem as a novel view-synthesis problem by training 2 networks to predict the appearance of a target image from the viewpoint of a source image. Godard et al. express the relative pose transformation for each source image $I_{t'}$ with respect to the target image I_t , as $T_{t,t'}$. Finally, the total loss they utilize can be written as:

$$L = \mu L_p + \lambda L_s \quad (7)$$

The authors [6] predict a dense depth map D_t that minimizes *total photometric reprojection error* L_p where:

$$L_p = \sum_{t'} pe(I_t, I_{t' \rightarrow t}) \quad (8)$$

$$I_{t' \rightarrow t} = I_{t'} \langle \text{proj}(D_t, T_{t \rightarrow t'}, K) \rangle \quad (9)$$

where $pe(\cdot)$ is *photometric reprojection error* using L_1 distance in pixel space. $proj(\cdot)$ is the resulting 2D coordinates of the projected depths D_t in $I_{t'}$ and $\langle \cdot \rangle$ is the sampling operator. K is the pre-computed intrinsic matrix, $I_{t' \rightarrow t}$ is the warped image. $pe(\cdot)$ error is defined as,

$$pe(I_a, I_b) = \frac{\alpha}{2} (1 - SSIM(I_a, I_b)) + (1 - \alpha) \|I_a - I_b\|_1 \quad (10)$$

where α is chosen to be $\alpha = 0.85$. Another loss metric they prefer is the edge-aware smoothness regularization. This metric is calculated as,

$$L_s = |\partial_x d_t^*| e^{-|\partial_x I_t|} + |\partial_y d_t^*| e^{-|\partial_y I_t|} \quad (11)$$

where d_t^* is the *mean-normalized inverse depth*, used to discourage the shrinking of the depth estimation.

In stereo training, the stereo image pair is used as the source (left) and target image (right). In the monocular setting, relative poses are not already determined, so it is possible to train a self-supervised pose estimation network to predict the relative poses $T_{t \rightarrow t'}$ which is used in the $proj(\cdot)$ formula. During training, Godard et al. solve for the depth and ego-pose unknowns simultaneously. In the mixed setting of monocular and stereo, the source image set also includes temporally adjacent frames (future and past frames) and the opposite stereo image.

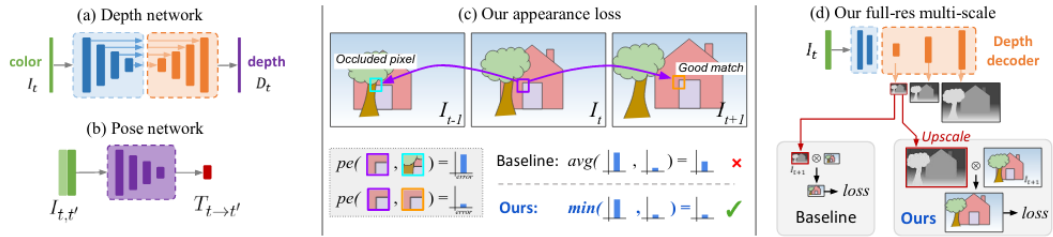


Figure 3.2 (a, b) Two networks, (c) their appearance loss, (d) multi-scale loss [6]

They also suggest that instead of the L_p equation above, a per-pixel minimum projection error would yield better results, since the existing self-supervised depth estimations average together the reprojection error into each of the available source images. If some pixels are visible in the target image, however not visible in the source image, this situation creates problems. In order to solve it, they use per-pixel

mask μ . When the network tries to predict the correct depth for such a pixel, it will probably not match the target and that induce a high penalty during training. Therefore, averaging the photometric error overall source images at each pixel is not the best method in this case. However, using a minimum operation significantly reduces artifacts at the image border:

$$\mu = [\min_{t'} pe(I_t, I_{t' \rightarrow t}) < \min_{t'} pe(I_t, I_{t'})] \quad (12)$$

3.3 Stereo Depth Estimation Methods Using Iterative Refinements

Iterative refinements are a relatively new concept for stereo depth estimation. There were previous similar cascaded models [59], [81]–[83], however they rely on expensive cost-volume filtering operations using 3D convolutions. An iterative refinement method which is arguably easier to understand and easier to calculate is RAFT-Stereo [4]. The architecture of this paper will be the main baseline for the experiments conducted in this thesis.

RAFT-Stereo has 4 main stages: Feature Encoder stage, Context Encoder stage, Correlation Pyramid stage, and Multi-Level Update Operator stage. The general architecture of RAFT-Stereo is shown below:

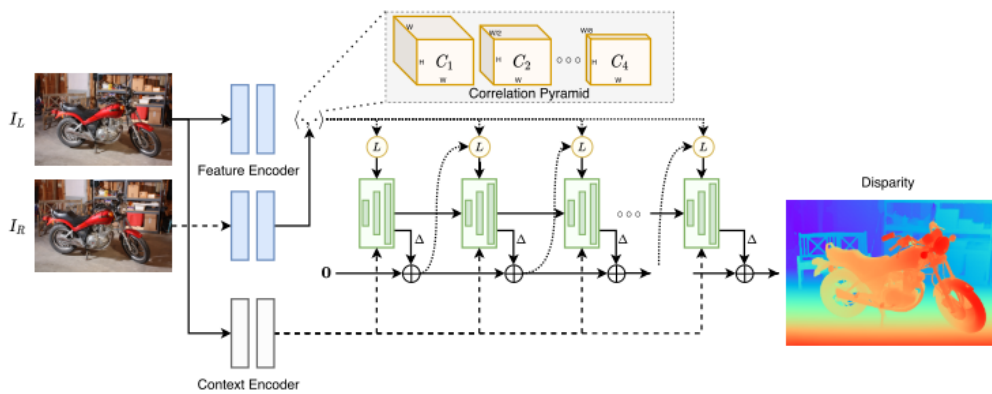


Figure 3.3 RAFT-Stereo architecture is the architecture mainly used in the thesis [4]

Given a rectified stereo image pair I_L and I_R , RAFT-Stereo extracts feature maps from the input images with convolution layers. Then the algorithm builds a 3D cost volume where the 3rd dimension is the disparity along the x-axis. Afterwards, it uses multi-level GRU [100] units to add incremental details to its disparity estimation d . Their loss function is a simple standard L_1 loss.

There are two different feature extractors in RAFT-Stereo: the feature encoder and the context encoder. Feature encoder is both applied to the left and right image. It builds a dense feature map that the correlation volume will use. The feature maps are 1/8th of the input resolution in the H- and W-axis, every element has 256 channels, and instance normalization is the preferred normalization technique. The context encoder has almost the same design, except it has batch normalization, and it is only applied to the left image. The context encoder has three outputs injected into three GRUs separately; the same context outputs are used in every refinement iteration in the Multi-Level Update Operator stage.

Lipson et al. built up a 3D correlation volume to build up the correlation pyramid. Using the dot-product as the similarity metric and $f, g \in R^{H \times W \times D}$ feature maps extracted from the left and right images, the 3D correlation volumes are calculated as,

$$C_{i,j,k} = \sum_h f_{ijh} \cdot g_{ikh} \quad (13)$$

where $C \in R^{H \times W \times W}$. The 3rd dimension of volume C is the disparity axis, the displacement along the horizontal axis W . Geometrically, from the left image to right image, all disparities must be positive. Hence, the correlation volumes are only calculated for the positive disparities.

In order to finalize building the correlation pyramid by using 4 correlation volumes, they perform 1D average pooling on the 3D correlation volume along the disparity axis. The k^{th} layer of the Correlation Pyramid is constructed via an average pooling layer with kernel size 2, stride 2. This operation results in a new lower-resolution volume C^{k+1} with dimensions of $H \times W \times W / 2^k$. This means only the disparity

dimension is shrinking between the correlation volumes, which is a valuable outcome for estimating detailed and fast disparity maps.

For indexing into this correlation pyramid, they use a correlation lookup operator L_C , similar to the one defined in RAFT[46]. Given the current estimate of disparity d in a pixel location, they construct a 1D indexing grid along the disparity dimension, with integer offsets around the current estimate in each correlation volume, as it is illustrated in Figure 3.4. The grid is used to index the correlation pyramid in each four correlation volumes. The retrieved values are then concatenated into a single feature map. In each pixel location from each correlation volume, $2r + 1$ many variables are indexed. r is the *correlation radius* and by default, it is selected as $r = 4$. Hence, the correlation feature is of size $H \times W \times (8r + 4)$.

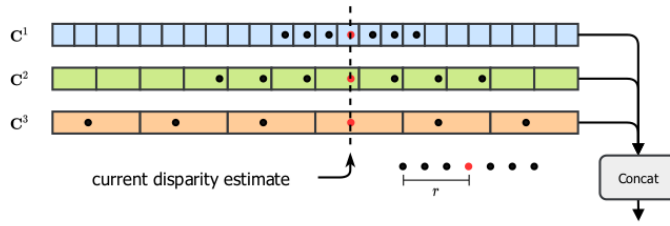


Figure 3.4 The correlation lookup operation around current disparity estimate

The Multi-Level Update Operator predicts a series of 2D disparity fields in each iteration for N times:

$$d = \{d_1, d_2, \dots, d_N\} \quad (14)$$

starting from the initial $d_0 = 0$ case. N is the iteration number, which can be any number as long as the execution time permits. In the original algorithm, this parameter is selected as $N = 32$. They use the current disparity estimate in every pixel location during each iteration update to index the Correlation Pyramid and produce the new set of correlation features.

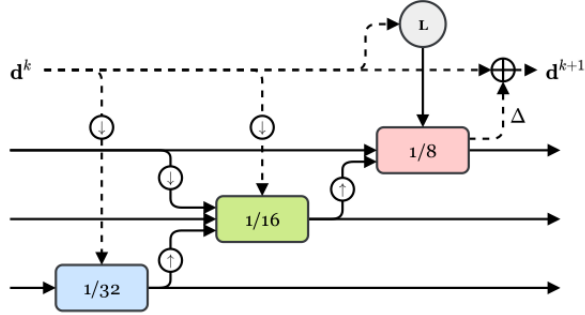


Figure 3.5 GRU modules in the Multi-Level Update Operator stage [4]

The disparity features are passed down through 2 convolution layers (shown with the descending dashed arrow in Figure 3.5).

Multi-Level Update Operator stage uses 3 GRUs that simultaneously operate on the disparity estimates at $1/8^{\text{th}}$, $1/16^{\text{th}}$, $1/32^{\text{nd}}$ of the resolution of the input. GRUs are cross-connected and use each other's hidden states as inputs. Every GRU gets the context features, and every GRU gets the disparity features created by the current disparity estimate. But only the last GRU indexes the correlation pyramid and gets the correlation features as input. The GRUs update their corresponding hidden states, and the last GRU is used to predict the disparity update Δ_k . That Δ_k is simply added to the current disparity estimate d^k to get the next estimate d^{k+1} .

Note that the $1/8^{\text{th}}$ resolution GRU has 4 times the parameter of the $1/16$ resolution GRU and 16 times for the other. To increase the quality of the hidden states, the two lower resolution GRUs are updated several times with the same input before the higher resolution GRU is ready to calculate again.

Because of this architecture, the predicted disparity field is calculated at $1/8^{\text{th}}$ of the original input resolution. To obtain a disparity map with the same dimensions as the original input, they use a convex up-sampling method, the same one used in RAFT [46].

The loss metric of RAFT-Stereo is simply a standard L_1 distance,

$$L = \sum_{i=1}^N \gamma^{N-i} \|d_{gt} - d_i\|_1 \quad (15)$$

where γ is chosen as $\gamma = 0.9$.

It should be noted that in this algorithm, there is no Fully Connected Network module. In every layer, only convolution filters are applied on the activation pattern of the previous layer. Hence this architecture can take inputs with any resolution.

3.4 Stereo Depth Estimation Methods Using Attention Mechanisms

Another state-of-the-art method that focuses on stereo matching problems is the STTR technique [66]. STTR uses Transformers [8] to relax the limitation of a fixed disparity range which is common in many convolutional correlation techniques. Instead of pixel-wise intensity correlation popular in classic stereo depth estimation methods, the authors use a customized Transformers attention mechanism with alternating between self and cross attention modules. Their method also identifies the occluded areas, provides a confidence map, and imposes uniqueness constraints for the stereo matching with the help of optimal transport calculations. STTR also provides a relative positional encoding for their customized attention mechanisms to define discriminative features during the matching process. Their architecture is shown below:

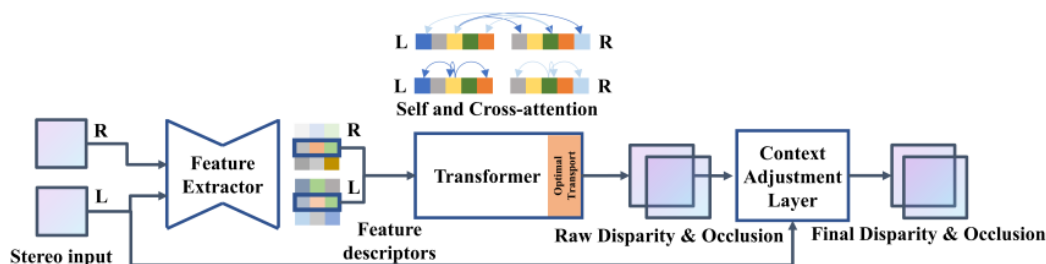


Figure 3.6 The STTR Architecture [61]

The feature extractor of STTR is a standard hourglass network similar to [101]. The features on each pixel location are denoted as e_l of size C_e . The resultant feature map is at the same resolution as the input image.

The self-attention module computes the attention of every pixel only along the epipolar line in the same image. The cross-attention module performs a similar

search; however, it searches on the other image. For N-1 layers, the authors compute self and cross attentions one after another. The Nth layer uses the most attended pixel

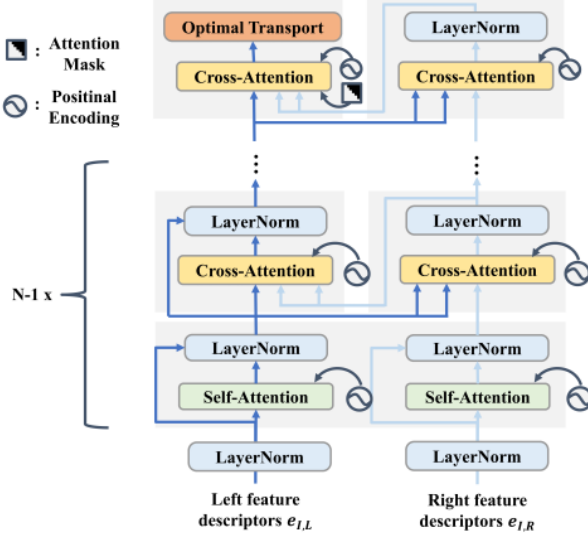


Figure 3.7 Alternating cross & self-attention modules in STTR architecture [61]

in the attention map to estimate the raw disparity. There are exclusive operations for the last layer: optimal transport calculation for uniqueness constraint and attention mask for search space reduction.

STTR utilizes multi-head attention modules. The *original feature map* input e has a *feature descriptor* e_I for every activation I . The multi-head module split the *channel dimension* C_e of those feature descriptors e_I into N_h *heads*, creating feature descriptors $e_{h,I}$ of channel dimension $C_h = C_e/N_h$ per head.

For each attention head h , three sets of linear projections give the $Q_{h,I}$, $K_{h,I}$ and $V_{h,I}$ vectors, this projection is repeated for every activation I in the original feature map:

$$Q_{h,I} = W_{Q_h} e_{h,I} + b_{Q_h} \quad (16)$$

$$K_{h,I} = W_{K_h} e_{h,I} + b_{K_h} \quad (17)$$

$$V_{h,I} = W_{V_h} e_{h,I} + b_{V_h} \quad (18)$$

Then $Q_{h,I}$, $K_{h,I}$ and $V_{h,I}$ vectors for each row r of the original feature map e are simply concatenated to acquire $\overline{Q_{h,r}}$, $\overline{K_{h,r}}$ and $\overline{V_{h,r}}$ tensors.

An attention map is calculated by first an outer-product and then a softmax operation. This calculation effectively multiplies every $Q_{h,I}$ with every other $K_{h,I}$ to create attention matrix $\alpha_{h,r}$ per head h for every row r . It should be noted that this softmax is the only nonlinearity step in this module:

$$\alpha_{h,r} = \text{softmax}\left(\frac{\overline{Q_{h,r}^T \cdot K_{h,r}}}{\sqrt{C_h}}\right) \quad (19)$$

The weighted $\overline{V_{0,r}}$ tensor is calculated by multiplying the attention matrix $\alpha_{h,r}$ with the $\overline{V_{h,r}}$ tensors and concatenating of the N_h followed by a linear projection. This way we gathered a vector for every activation in the row r :

$$\overline{V_{0,r}} = W_0 \text{concat}(\alpha_{1,r} \overline{V_{1,r}}, \dots, \alpha_{N_h,r} \overline{V_{N_h,r}}) + b_0 \quad (20)$$

Then, the residual addition step is calculated by extracting the vector $V_{0,I,r}$ from the $\overline{V_{0,r}}$ tensor and adding it to the original feature descriptor e_I :

$$e_I \leftarrow e_I + V_{0,I,r} \quad (21)$$

For self-attention modules, K, Q, and V vector sets are calculated from the same image, however the cross-attention module V is computed by the target image. Cross attention modules are connected bidirectionally.

It should be noted that every feature e_I belongs to one activation in the original feature map, the attention module processes information for every row in the original feature map separately from other rows and the I^{th} slice of the processed information for row r is then used to update the activation e_I . This is why the resolution of the original feature map can increase independently from the Transformer architecture.

The authors' relative positional encoding scheme try to resolve the ambiguities created by the pixel similarities in textureless places. Relative encoding forces the features in the image to anchor themselves to the nearest prominent features using the relative positional information. They chose to encode the relative pixel distance instead of an absolute position encoding scheme due to its shift-invariance abilities.

In the original transformer, the position embedding tensor was an absolute encoding tensor, and it was directly added to the input feature descriptor:

$$e = e_l + e_p \quad (22)$$

This selection makes the final values in the attention map:

$$\alpha_{i,j} = e_{l,i}^T W_Q^T W_K e_{l,j} + e_{l,i}^T W_Q^T W_K e_{p,j} + e_{p,i}^T W_Q^T W_K e_{l,j} + e_{p,i}^T W_Q^T W_K e_{p,j} \quad (23)$$

The last term is a *position-position* term due to having 2 e_p terms. Such strategy makes it an absolute value in a particular pixel location. In order to make the position encoding relative, Li et al. excluded that term:

$$\alpha_{i,j} = e_{l,i}^T W_Q^T W_K e_{l,j} + e_{l,i}^T W_Q^T W_K e_{p,j} + e_{p,i}^T W_Q^T W_K e_{l,j} \quad (24)$$

Their optimal transport calculations use the negative of the attention map as the cost matrix M , computed by the N^{th} cross attention module without a final softmax, since optimal transport will normalize the attention values. Then, it aims to assign each pixel in the right image at most one pixel in the left image. For that purpose, they use an entropy regularized optimal transport setup [102].

Given a cost matrix M and 2 marginal distributions a and b of length I_w (a and b are the results of the cross attentions in both images at the same height), the optimal transport needs to find the optimal coupling matrix T :

$$T = \underset{T \in R^{I_w \times I_w}}{\operatorname{argmin}} \sum_{i,j=1}^{I_w} T_{ij} M_{ij} - \gamma E(T) \quad (25)$$

$$s. t. : T \mathbf{1}_{I_w} = a, T^T \mathbf{1}_{I_w} = b$$



Figure 3.8 (a) Geometric constraints of stereo matching, (b) the attention mask [61]

$E(T)$ is the entropy regularization term. This equation is solved for every line in the image. Li et al. [61] also masks the attention map such that any point in the left image cannot be matched to a pixel more right one in the other image. They use a lower triangular binary mask on the attention map to impose such constraints.

The occlusion module uses disparity candidates in a modified *winner-take-all* (WTA) approach. The raw disparity is the most probable match k , acquired from the optimal transport matrix T . Then, the most probable match's 3-pixel neighbors $N_3(k)$ and their probabilities are normalized such that the sum of their probabilities t_l sums to 1:

$$\tilde{t}_l = \frac{t_l}{\sum_{l \in N_3(k)} t_l} \quad (26)$$

The authors weigh the candidate disparities d_l 's by the probabilities \tilde{t}_l to get the raw disparities $\tilde{d}_{raw}(k)$:

$$\tilde{d}_{raw}(k) = \sum_{l \in N_3(k)} d_l \tilde{t}_l \quad (27)$$

They use the residual probabilities of the pixels that are outside of the 3-pixel boundary as the occlusion probability:

$$p_{occ}(k) = 1 - \sum_{l \in N_3(k)} t_l \quad (28)$$

The raw disparities $\tilde{d}_{raw}(k)$ and the occlusion probabilities $p_{occ}(k)$ so far are calculated within a line, independent of other lines. Therefore, the disparities lack contextual cues. Content Adjustment Layer in Figure 3.6 uses convolutions to adjust the estimated values, conditioned on the input image globally.

They use a Relative Responsive loss L_{rr} which is proposed in [101] on the assignment matrix T , for both sets of matched pixels M and sets of unmatched pixels U . The goal of the network is to maximize the attention on the actual target location. Since the disparity is subpixel, they use linear interpolation between the nearest integer pixels to find the matching probability t^* . For the i^{th} pixel in the left image with ground truth disparity $d_{gt,i}$:

$$t_i^* = \text{interp}(T_i, p_i - d_{gt,i}) \quad (29)$$

$$L_{rr} = -\frac{1}{N_M} \sum_{i \in M} \log(t_i^*) + \frac{1}{N_U} \sum_{i \in U} \log(t_{i,\phi}) \quad (30)$$

The $\text{interp}(\cdot)$ is the linear interpolation and $t_{i,\phi}$ is the unmatched probability. The first term on the right-hand side is the matched terms' loss; the second is the unmatched pixel loss. They also use 3 other losses: smooth L_1 loss on raw disparities, the smooth L_1 loss on final disparities and the binary entropy loss and then sum them all up:

$$L = w_1 L_{rr} + w_2 L_{d1,r} + w_3 L_{d1,f} + w_4 L_{be,f} \quad (31)$$

3.5 Comparison of the Methods in the Literature

The comparison of the stereo depth estimation performance of the methods explained in this chapter is provided in this section. The definitions of the metrics are explained below.

3.5.1 Error Metrics for Stereo Depth Estimation

The metrics below are the most commonly used metrics for stereo depth estimation (the lower number is better):

- EPE (in pixels): The mean absolute disparity error.
- 3 px error (%): It gives the percentage of pixels with a matching error of more than 3 px. This metric can also be used for 1 or 2 px.

3.5.2 Depth Estimation Comparison

The results of the depth estimation algorithms reported in Table 3.1. The reported results are based on their reports in their papers, other papers that replicated them [103], online benchmark sites of the datasets, and the tests in this thesis. When

conflicting results are found, the online benchmarks are assumed to be true. As one can see in Table 3.1, in all metrics, RAFT-Stereo and STTR achieves the best results, which is no surprise because they are the most recent papers in this table.

Table 3.1 Comparison of the performance of 4 stereo depth estimation method

Metric	Middlebury Q Dataset		ETH3D Dataset	
	2px (%)	EPE (px)	1px (%)	EPE (px)
Monodepth2	30.76	7.94	12.56	0.60
HITNET	12.80	3.29	3.11	0.22
RAFT-Stereo	<u>9.36</u>	<u>2.71</u>	<u>3.28</u>	0.19
STTR	8.51	2.33	4.98	0.37

CHAPTER 4

EFFECTS OF ATTENTION MODULES, INITIALIZATIONS AND SELF SUPERVISED TRAINING ON STEREO DEPTH ESTIMATION

Based on the literature survey, RAFT-Stereo is selected as a promising technique to examine. This algorithm has two main drawbacks, one of which is its performance in homogenous regions, whereas the other one is the lack of an initialization stage for better estimates. In this chapter, two primary analyses have been performed for the improvement of the state-of-the-art stereo depth estimation method, RAFT-Stereo, by utilizing some design choices:

- Analysis 1: The effects of adding various self & cross attention modules of Transformers into RAFT-Stereo depth estimation network.
- Analysis 2: Initialize the RAFT-Stereo depth estimation network with various previous disparity predictions or use ego-pose estimation to finetune the pre-trained supervised RAFT-Stereo depth estimation network with a self-supervised method.

4.1 Experimental Settings

4.1.1 Selected Network Architecture

RAFT-Stereo [4] is used to conduct all the experiments and to propose improvements for its estimation capabilities. The experiments are conducted by using the PyTorch [119] library (version 1.10). As it is explained in the related work section, RAFT-Stereo has 4 stages: Feature Extractor, Context Extractor, Correlation Pyramids, and Multi-Level Update Operator. Some modifications have been made throughout this

thesis on Correlation Pyramids, Multi-Level Update Operator, or overall training method.

4.1.2 Optimization

The default optimization method of RAFT-Stereo is AdamW [104] optimizer. In most of the tests, one-cycle learning rate schedule [105] with a maximum learning rate of $2 \cdot 10^{-4}$ is used. Batch size is selected as 2 or 4, according to the memory requirements of avail GPU. Models are trained on synthetic data for 200.000 steps. The epoch size has been calculated accordingly using the parameters above. In the L_1 loss scheme of RAFT-Stereo, the gamma value is left untouched $\gamma = 0.9$.

4.1.3 Datasets

The SceneFlow synthetic dataset, a combination of FlyingThings3D, Driving & Monkaa, is utilized during the supervised synthetic training. These are stereo image datasets from different domains, and the purpose of using images from different domains is to extract and expect some level of generalization. Since SceneFlow datasets have temporal continuation between many of its frames, these datasets are also used in the Analysis 2 where the proposed method aims to utilize previous frames' disparity predictions.

The raw KITTI dataset has 12919 unannotated real-world stereo images. In the self-supervision experiments, this *raw KITTI dataset* is used as the primary training dataset since the self-supervised experiments converge relatively slowly and requires more example to extract useful information through the consistency.



Figure 4.11 An example from every dataset utilized in this thesis: The first row is an SceneFlow (FlyingThings3D and Monkaa) examples, the second row is ETH3D and Middlebury examples, the last row is a KITTI example

The *KITTI training dataset* comprises only 200 stereo images with ground truth information. The number of images is enough for the validation purposes during the synthetic training phase. However, this number of images is not enough to separate this dataset into a training set and a validation set during the finetuning phase afterwards. Therefore, a 10-fold cross-validation scheme over this dataset is utilized during the finetuning phase.

Middlebury 2015, ETH3D and KITTI training datasets are utilized as the validation datasets during all attention experiments, however due to their lack of temporal nature, the temporal information experiments use only the KITTI training dataset.

Middlebury dataset has three resolutions: Middlebury-F is full resolution, Middlebury-H is half resolution, and Middlebury-Q is quarter resolution. The Middlebury-H and Middlebury-Q are utilized for all attention experiments.

Middlebury-F is not utilized because none of the networks in any experiment could fit into the VRAM of available GPU (Nvidia RTX 3090).

In all supervised synthetic training experiments, 320x736 pixel crops have been used, same as the RAFT-Stereo algorithm. In the finetuning phase, the input images are cropped to 320x1024 pixels. This setting is different from the RAFT-Stereo algorithm as they use 320x1000 pixel crops during the finetuning phase. The reason is that RAFT-Stereo algorithm progressively decreases the size of the activations until it reaches to 1/8th resolution of the original input image, while the proposed networks in this work reach 1/16th resolution of the original input image. This is why the image width must be divisible by 16, and an image width of 1024 is selected for this purpose.

Table 4.1 The tasks and the utilized datasets during those tasks are explained

Tasks \ Datasets	FlyingThings3D	Driving	Monkaa	ETH3D	Middlebury	KITTI training dataset
Training	✓	✓	✓			
Validation				✓	✓	✓
Finetuning						✓ (10-fold cross validation)

In all supervised training experiments including the finetuning phase, the datasets have been augmented with various augmentation techniques. To be specific, the image saturation was adjusted between 0 and 1.4; the right image was perturbed to simulate imperfect rectification, which is common in datasets such as ETH3D and Middlebury. Moreover, the image and disparity data have been stretched by random factors in the range $[2^{-0.2}, 2^{0.4}]$ to simulate a range of disparity distributions. All these scenarios are explained in Table 4.1.

4.2 Analysis 1: Attention-based Improvements for RAFT-Stereo Algorithm

Transformer architectures are generally composed of the self-attention and cross-attention modules, as mentioned in Chapter 2. According to Li et al. [36], these networks can be quite beneficial for stereo disparity estimation. In STTR, these modules are believed to be creating echoes in the intermediate high-dimensional representation, especially in textureless areas. They gather *echo-like* in-paintings, and in theory, they can make the rest of the networks' task somehow easier for the dense feature matching mechanism on the textureless areas. The reason for obtaining echoes in high dimensional space is the relative position encoder in the attention mechanism. The encoder makes it possible for an individual pixel attend to relative positions in a row, whenever a nearby edge pixel is attended in a layer, some of the feature of that edge pixel is included into the feature output for the original pixel due to the attention mechanism. When this process is repeated, the features of the edge pixel expand into the nearby homogenous regions in a periodic manner. This analysis begins with the hypothesis that this self and cross attention cascade can be a suitable replacement for the correlation volumes used in the more mature feature-matching methods. However, this idea might follow that, it can also be detrimental for the textured areas. Those echoes may decrease performance in the textured areas and most importantly, around the edges. Hence this hypothesis requires further investigation.

Another interesting point is that according to the STTR algorithm, the *in-painting* effect is relatively less noticeable when the self-attention module processes information and relatively more pronounced when the cross-attention module processes information. There are other design choices for this task, such as using cross-attention modules only, or using the self-attention module except the last one. The reason to use only cross-attention modules is to make the *in-painting* effect more pronounced using the depth edges and the reason for using self-attention modules except the last one is to make the *in-painting* effect more pronounced using the edges in the same image. This analysis will investigate the results of these choices as well.

However, there is an important difference between this analysis and the original STTR method. STTR approach always uses the self and cross-attention modules one after another. Attention map of the last cross-attention module is the input of an optimal transport algorithm that tries to use this information such that the last attention map can be trained into a correlation volume module. In contrast, different combinations, and number of self and cross-attentions have been tested in this thesis, and no optimal transport algorithm has been used to sort the attention calculations. Instead, in this work the highest quality feature maps of the correlation-volume of RAFT-Stereo are changed with the transformers' attention maps. This modification can be thought as bringing STTR algorithm into the end-to-end training category closer or making the correlation volume of RAFT-Stereo more learnable.

In this stage, various combinations of the attention modules can be utilized: cross-attention modules only, self-attention modules only, or both of them one after another. It should be reminded that the last attention module needs to be a cross-attention module, since cross attention is the module where an attention calculation is produced by utilizing the information coming from the other image and not coming from the same image.

For that purpose, in the Analysis 1, after the Feature Encoder of RAFT-Stereo, different amounts of cross & self-attention modules are placed, and the improvement opportunities that these attention modules pose are investigated. To keep the disparity index the GRUs use the same; the final attention map of the last cross-attention is subsampled four times & turned into four *Attention Volumes* in an *Attention Pyramid*. It should be noted that, as explained in Chapter 3.3 and 3.4, the convolution and attention modules utilized in this thesis are flexible in terms of the resolution of the input.

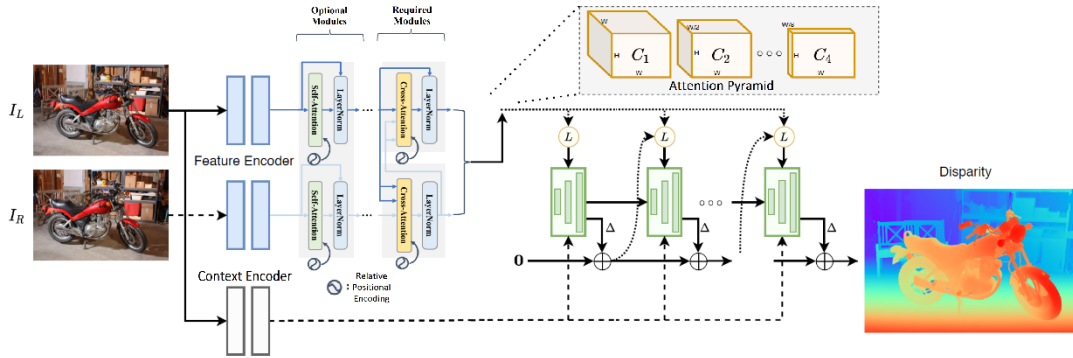


Figure 4.2 General architecture of the Analysis 1 (Revised from [4])

The experiments performed in this part are:

- Training a network with various numbers of self & cross-attention modules,
- Training a network with various numbers of self-attention modules and one last cross-attention module,
- Training a network with various numbers of cross-attention modules only.

Every experiment has been conducted on a Nvidia RTX 3090 GPU. Every training lasted 3.5 days with a total of 200.000 steps in the end. In all of the experiments, the parameters of the RAFT-Stereo architecture have been initialized with the Kaiming initialization. The parameters of the attention modules have been initialized with the values of a pre-trained STTR network, deleting the unnecessary modules starting from the last. The N=12 networks have 12,401M parameters, N=8 networks have 12,135M parameters and N=4 networks have 11,869M parameters.

4.2.1 Hyperparameter Search

Before starting the experiments, a quick hyperparameter search for the networks with attention modules is conducted. The initial trial and error phase shows that $4 \cdot 10^{-3}$ is a quite aggressive learning rate for these tasks since the results were always inferior to other alternatives. On the other extreme, $1 \cdot 10^{-5}$ was a small learning rate to finish training within 200.000 steps, therefore it is later used as the finetuning learning rate

parameter. The potential improvement by dropout was considered as well. Dropout rate of 0.15 is tested and at that dropout rate, the learning process was always affected negatively. The reason for considering the hyperparameter combination in Table 4.2 is based on this observation.

Table 4.2 The hyperparameter grid search combinations for networks with attention modules

Dropout \ learning rate	$2 \cdot 10^{-4}$	$11 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
0.0	(0.0, $2 \cdot 10^{-4}$)	(0.0, $11 \cdot 10^{-5}$)	(0.0, $2 \cdot 10^{-5}$)
0.1	(0.1, $2 \cdot 10^{-4}$)	(0.1, $11 \cdot 10^{-5}$)	(0.1, $2 \cdot 10^{-5}$)

The hyperparameter combinations in Table 4.2 are used to train the networks explained in Figure 4.2. During the hyperparameter search N is always used $N = 8$ and use them for all the trainings because the architecture of the networks in this analysis are relatively similar. The networks are trained with the SceneFlow dataset and validated with the KITTI training dataset. Since these results are not finetuned, the respective baseline network for this case is the model denoted as *raftstereo-sceneflow* in Table 4.3 from the RAFT-Stereo project page.

Table 4.3 The result of the hyperparameter search, validated on KITTI training dataset. The traditional % error used for KITTI is considered to be the 3px % error

Networks	EPE	3px % error
Network with (0.0, $2 \cdot 10^{-4}$)	1.0431	4.8565
Network with (0.0, $11 \cdot 10^{-5}$)	1.0953	5.4873
Network with (0.0, $2 \cdot 10^{-5}$)	1.0641	5.1375
Network with (0.1, $2 \cdot 10^{-4}$)	1.0449	4.9220
Network with (0.1, $11 \cdot 10^{-5}$)	1.0402	4.7782
Network with (0.1, $2 \cdot 10^{-5}$)	1.0892	5.1496
Baseline: raftstereo-sceneflow	1.1779	5.6985

As one can observe in Table 4.3, in the synthetic-to-real generalization phase, the proposed networks performed better in terms of both 3px % error and EPE errors. Examining the results, the rest of the experiments in this section is conducted with dropout rate of 0.1 and a learning rate of $11 \cdot 10^{-5}$, which are shown by bold fonts in Table 4.3.

4.2.2 Experiment 1 - Training the Network with Various Number of Self & Cross Attention Modules

In this first experiment, different amounts of both self & cross attention module pairs are added to RAFT-Stereo (Figure 4.2). These attention modules (and the feature encoder that feeds them) are the modules defined in STTR algorithm [61]. The parameters on Github page of the authors are used to initialize the proposed networks. For this experiment, the total number of attention modules (N) is selected as N=4, N=8, N=12, as these different numbers will work as the ablation tests in this experiment.

The attention stage starts with the self-attention module. The purpose of these choices of N is to understand the effects of these blocks of self and cross-attention cascades better. By stacking more (N=12) or less (N=4) of them, we effectively perform ablations tests on these blocks.

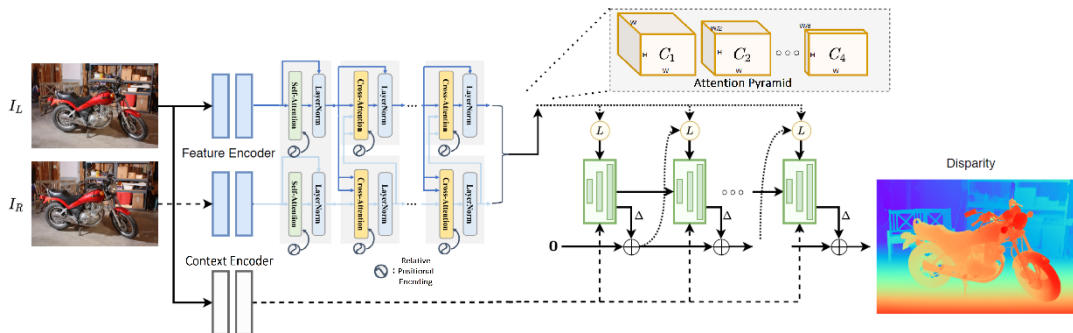


Figure 4.3 A general architecture for the first experiment of the Analysis 1

4.2.2.1 Synthetic-to-Real Generalization Performance

The initial phase of the experiment is the synthetic-to-real generalization phase. A qualitative comparison regarding the three tests that is conducted is shown in Figure 4.4. There are quite minor differences, however the proposed networks at this stage perform slightly better than the baseline. In the first row, please notice the trees; in the second row, please notice the indentations; and in the third row, please notice hand breaks.

Table 4.4 Comparison of the networks that are trained with synthetic images only

Datasets	KITTI	Middlebury		ETH3D
Networks trained with only synthetic images	3px % error	2px % error of half set	2px % error of quarter set	1px % error
Cross & Self, N=4	5.3174	13.6253	10.6802	3.1843
Cross & Self, N=8	4.7782	8.5712	7.4183	2.2652
Cross & Self, N=12	<u>4.9097</u>	<u>10.8557</u>	8.1948	<u>3.0630</u>
Baseline (raftstereo-sceneflow)	5.6985	12.59	9.36	3.28
HD ³ [122]	26.5	37.9	20.3	54.2
Gwcnet [123]	22.7	34.2	18.1	30.1
PSMNet [59]	16.3	25.1	14.2	23.8
GANet [124]	11.7	20.3	11.2	14.1
DSMNet [125]	6.5	13.8	<u>8.1</u>	6.2
STTR [61]	6.74	13.9	8.51	4.98

The quantitative results of the synthetic-to-real generalization phase and results of other competing algorithms are shown in Table 4.4. The proposed networks in this phase performed better than both the baseline network and other competing

algorithms in all datasets, when it comes to the percentage of the stereo disparity outliers.

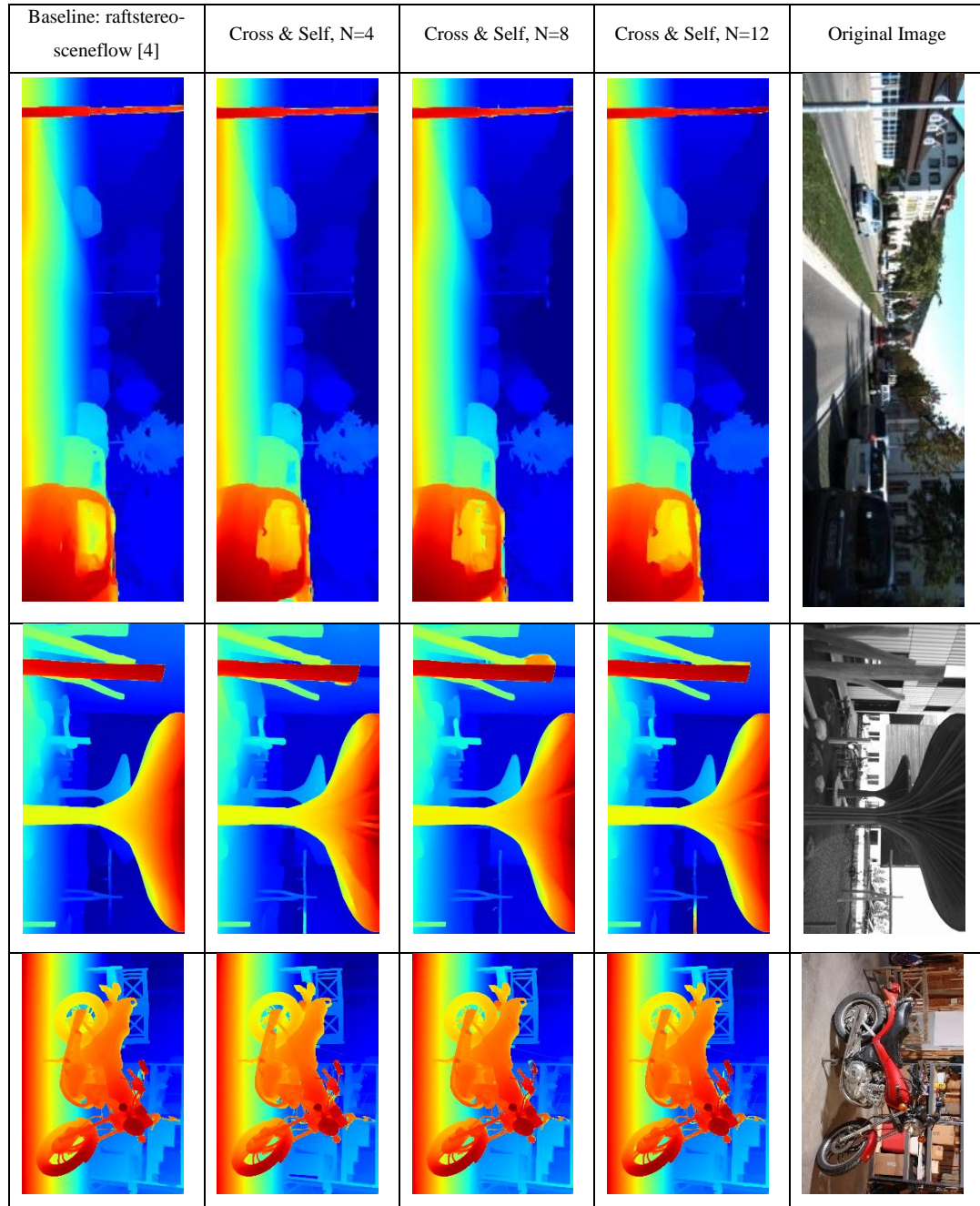


Figure 4.4 Comparison of the baseline network with the proposed networks

4.2.2.2 Comparing Performance on Textured and Textureless Areas

In this section, the performances of the proposed networks in the textured areas and textureless areas are investigated, the results are compared to the baseline network, for obtaining a thorough analysis. For determining the areas that are textured and textureless, a normalizing Laplacian filter is applied on the left images. The filter kernel size is heuristically selected as 33 and similarly the separation threshold that determines whether a place is textured or not is selected as 0.235. An example to understand the effect of such a filter is shown in Figure 4.5:

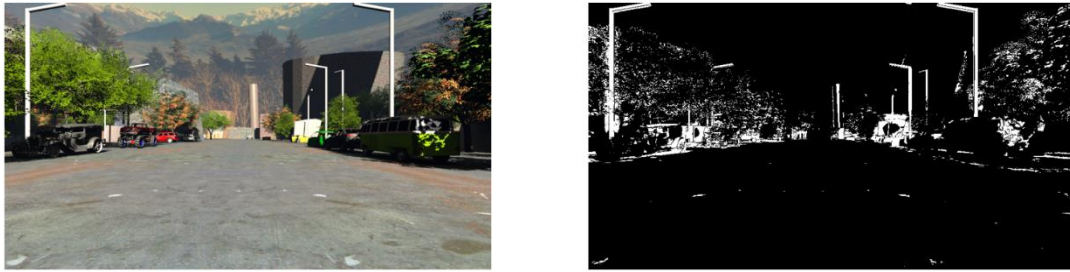


Figure 4.5 A typical example of the effects of the Laplacian filter

Then, EPE and 3px % error terms are calculated separately for each types of areas. Since original RAFT-Stereo paper did not share their final finetuned network for the KITTI dataset, which is understandable as it includes multiple folds of cross validation parameters, *their raftstereo-sceneflow* network, one only synthetically trained with SceneFlow dataset, is used for comparison. The quantitative results are shown in Table 4.5.

The proposed synthetically trained networks performed better in all three conditions: in general, textured areas and textureless areas. However, one examines at how the network depth N changes the *relative performance* of these areas (the grey area in Table 4.5). It can be observed that, relatively speaking, as adding more attention modules affected the textured areas for the worse, however the textureless areas are affected for the better (maximizing at $N=8$). This is probably due to the in-painting effect that is hypothesized in STTR paper. In the high-dimensional space of the feature output of the transformers, an *echo* that is created by the relative positioning

encoding of the STTR algorithm can affect textureless areas relatively in a positive way, however the textured areas may be affected relatively in a negative way.

Table 4.5 The synthetic-to-real experiments. The networks are only trained on SceneFlow datasets and validated with KITTI training dataset

Networks trained with only synthetic images	EPE in general	3px % error in general	EPE in textured areas	3px % error in textured areas	EPE in textureless areas	3px % error in textureless areas	Ratio of EPE in textured areas and EPE in general	Ratio of 3px % error in textured areas and 3px % error in general	Ratio of EPE in textureless areas and EPE in general	Ratio of 3px % error in textureless areas and 3px % error in general
Baseline (raftstereo-sceneflow)	1.1779	5.6985	1.3867	7.6713	1.1345	5.421	1.1773	1.3462	0.9632	0.9513
Cross & Self, N=4	1.1931	5.3174	1.3112	7.3482	1.1528	5.03	1.0990	1.3819	0.9662	0.9460
Cross & Self, N=8	1.0402	4.7782	1.3001	7.2698	0.9967	4.4277	1.2499	1.5215	0.9582	0.9266
Cross & Self, N=12	1.0561	4.9097	1.2897	7.0589	1.0149	4.6074	1.2212	1.4377	0.9610	0.9384

Even though the absolute performance is increased as one adds more attention modules, one can see there is a trade-off in the attention modules. In the original RAFT-Stereo algorithm, the relative ratios for 3px % errors are $1.3462:0.9513=1.4151$, for textured areas and textureless areas respectively. But in the N=8 network, the same ratio is $1.5215:0.9266=1.6420$. This indicates that an algorithm that discriminates between the textured and textureless areas may improve the performance even more by using a suitable attention-based network for each types of areas.

4.2.2.3 Finetuning the Networks

In order to improve the results, the networks should be trained further by a similar dataset. Hence the next task is to finetune the networks using the KITTI training dataset. During this phase of the experiment, a 10-fold cross-validation method has to be applied, since this is a small dataset with only 200 annotated images. The learning rate is selected as 10^{-5} , and data augmentations such as saturation and

stretches were applied. The *raftstereo-sceneflow* network is finetuned to acquire the unprovided finetuned RAFT-Stereo model’s results.

The qualitative results of the finetuned networks are shown in Figure 4.6 and the quantitative results is shown in Table 4.6. After finetuning, the proposed networks have come close however failed to pass the performance of the baseline network RAFT-Stereo by only 0.01 difference. This result is somehow expected due to the fact that attention modules always require more training dataset than any other network architecture. However, it should be noted that the big poles’ shape in the first row and thin lines in the second row where the proposed network performed a little better.

Table 4.6 The performance comparison of finetuned networks by KITTI training dataset

Networks finetuned with KITTI training dataset	3px % error
Cross & Self, N=4	2.7601
Cross & Self, N=8	1.9997
Cross & Self, N=12	1.9777
Baseline (RAFT-Stereo) [4]	1.96
AcfNet [126]	1.89
AMNet[127]	1.84
GANet-deep [124]	1.81
SUW-Stereo [128]	1.80
GANet + DSMNet [125]	1.77
CSPN [129]	1.74
LeaStereo [130]	1.65
STTR [61]	2.01

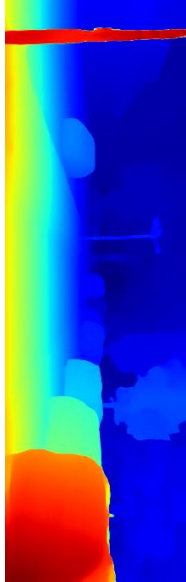
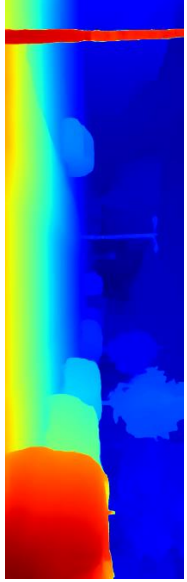
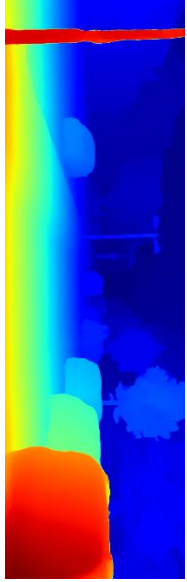



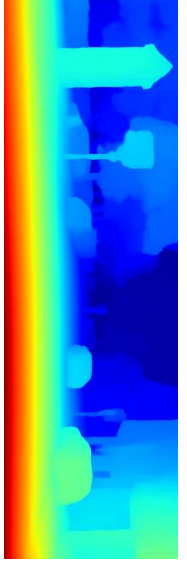
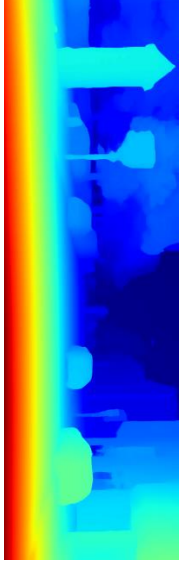
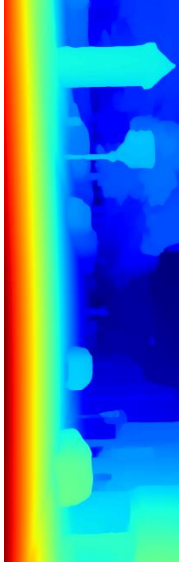

Baseline: raftstereo-sceneflow [4]	Cross & Self, N=4	Cross & Self, N=8	Cross & Self, N=12	Original Image
				
				

Figure 4.6 Examples from finetuned networks in this experiment where the cascade of self and cross attention modules are utilized

To conclude, after an initial hyperparameter search, the proposed networks have been tested, which are composed self and cross-attention modules instead of correlation volumes. It is observed that the proposed methods performed much better synthetic-to-real world generalization than both RAFT-Stereo and other competing algorithms compared in Table 4.4. However, it is failed to surpass any of them when

the networks are finetuned with the KITTI training dataset. With a deal of uncertainty, one can argue that this might be related to the existence of the attention modules in the proposed networks and the smaller number of training examples, however this question is left as a future work. As more attention modules are added, the additive modules generally increase the performance of the network up to a point, however relatively speaking, during those performance increases, a performance trade-off between the textureless areas and textured areas is observed clearly.

4.2.3 Experiment 2 - Training the Network with Self-Attention Modules Only

In the second experiment, all the cross-attention modules are removed, except the last module, since the attention calculation of the last cross attention module must be used for disparity search by the proceeding modules, instead of the original correlation-volume. This architecture is more similar to that of the original Transformer [8].

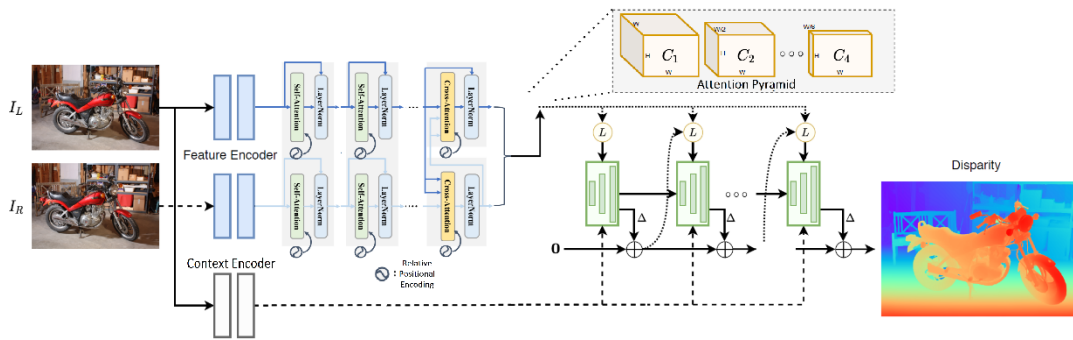


Figure 4.7 The general architecture of the second experiment (Revised from [4])

This experiment is required to understand the in-painting effects of the self-attention modules better. On the previous experiment regarding this topic, one could only conclude about the effect as an outcome of both of the attention modules. On the other hand, with this experiment and the next one, one can better understand the individual effects of the self and cross attention modules. During all tests, the

previous hyperparameter grid search results are used. Moreover, N has been chosen as $N=4$, $N=8$, and $N=12$, as these will work as the ablation tests during all tests.

4.2.3.1 Synthetic-to-Real Generalization Performance

The initial phase of the training is the synthetic-to-real generalization phase. The quantitative results of the synthetic-to-real experiments and results of other competing algorithms are shown in Table 4.7. In order to not populate this thesis with the same graph, only the best two closest algorithms are kept in the figures.

The proposed networks only could surpass other networks in synthetic-to-real generalization cases for the KITTI dataset. In all other cases, self-attention modules were not more successful than the baseline.

Table 4.7 The synthetic to real experiment with self-attention modules

Datasets	KITTI	Middlebury		ETH3D
Networks trained with only synthetic images	3px % error	2px % error of half set	2px % error of quarter set	1px % error
Self-attention only, $N=4$	5.5657	14.6747	11.3530	3.8603
Self-attention only, $N=8$	5.5306	12.8250	10.9948	3.3309
Self-attention only, $N=12$	5.7419	14.1663	11.8981	3.4133
Baseline: raftstereo-sceneflow [4]	5.6985	12.59	9.36	3.28
DSMNet [125]	6.5	13.8	8.1	6.2

The qualitative results are shown in Figure 4.8. The proposed networks in this case, relatively speaking, struggle with holes and poles.

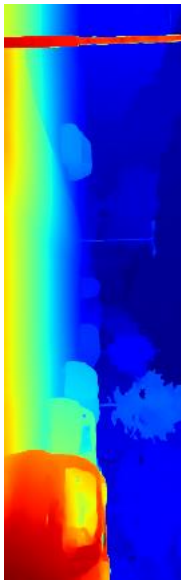
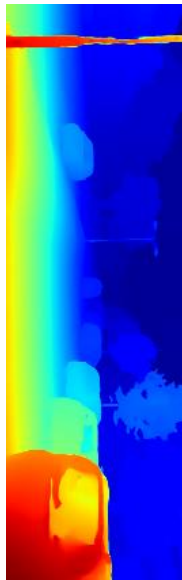
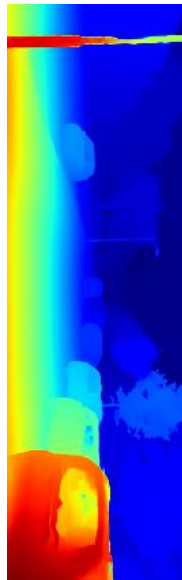


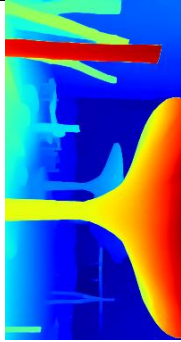
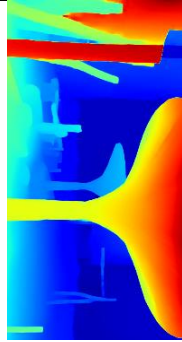
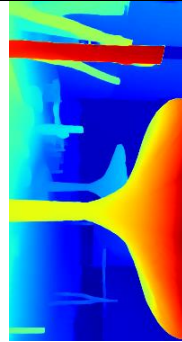
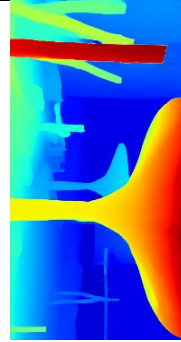

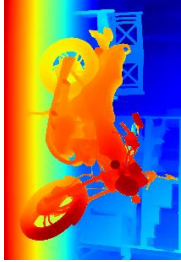
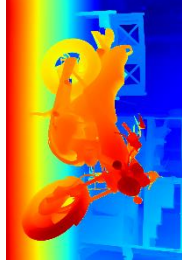
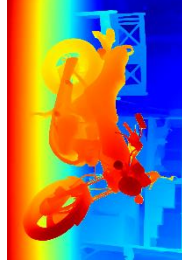
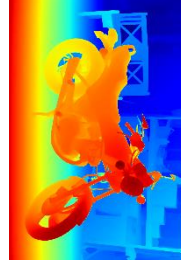

Baseline: raftstereo-sceneflow [4]	Self-attention only, N=4	Self-attention only, N=8	Self-attention only, N=12	Original Image
				
				
				

Figure 4.8 The qualitative results of the synthetic to real generalization test with self-attention modules

4.2.3.2 Comparing Performance on the Textured and Textureless Areas

A similar analysis for the textured and textureless areas is also performed in this experiment. As shown in Table 4.8, *3px % error in general* is minimum at the proposed network with $N=8$, *EPE in general* is minimum at the one with $N=4$.

With the baseline network, the ratios of the textured and textureless areas start from the ratio $1.3462:0.9513=1.4151$, and predictably, as more self-attention modules are added, at $N=12$, the ratio becomes $1.4836:0.9320=1.5918$. This result is similar to the previous texture analysis. However, *3px % errors and EPE in textured areas* decreased with N , while *3px % errors and EPE of textureless areas* are roughly aligned with their *in general* counterparts. This observation suggests that either the increase of number of self-attention modules caused this decrease on the textured area performance or the lack of enough cross-attention modules caused this phenomena. This is why the further investigation, which focuses on building networks with cross-attention modules only, will improve our understanding.

Table 4.8 The synthetic-to-real experiments with networks that is built with self-attention modules.

Networks trained with only synthetic images	EPE in general	3px % error in general	EPE in textured areas	3px % error in textured areas	EPE in textureless areas	3px % error in textureless areas	Ratio of EPE in textured areas and EPE in general	Ratio of 3px % error in textured areas and 3px % error in general	Ratio of EPE in textureless areas and EPE in general	Ratio of 3px % error in textureless areas and 3px % error in general
Baseline (raftstereo-sceneflow) [4]	1.1779	5.6985	1.3867	7.6713	1.1345	5.421	1.1773	1.3462	0.9632	0.9513
Self-attention only, $N=4$	1.1130	5.5657	1.3973	8.1588	1.0677	5.2010	1.2554	1.4659	0.9593	0.9345
Self-attention only, $N=8$	1.1807	5.5306	1.4158	7.9317	1.1351	5.1929	1.1991	1.4341	0.9614	0.9389
Self-attention only, $N=12$	1.1301	5.7419	1.4276	8.5189	1.0816	5.3513	1.2633	1.4836	0.9571	0.9320

4.2.3.3 Finetuning the Networks

The next task is finetuning the networks by using the KITTI training dataset. Again, during this phase, a 10-fold cross-validation method had to be applied because this is a small dataset with only 200 images. The learning rate is selected as 10^{-5} , data augmentations such as saturation and stretches were applied.

The qualitative results of the proposed finetuned networks are shown in Figure 4.9 and the quantitative results are shown in Table 4.9. The finetuned raftstereo-sceneflow network is used to acquire the unprovided finetuned RAFT-Stereo model and its output images.

Table 4.9 The comparison between the proposed networks in experiment 2 and competing networks.

Networks finetuned with KITTI training dataset	3px % error
Self-attention only, N=4	2.3907
Self-attention only, N=8	2.4856
Self-attention only, N=12	2.4722
Baseline (RAFT-Stereo) [4]	1.96
AcfNet [126]	1.89
AMNet[127]	1.84
GANet-deep [124]	1.81
SUW-Stereo [128]	1.80
GANet + DSMNet [125]	1.77
CSPN [129]	1.74
LeaStereo [130]	1.65
STTR [61]	2.01

After finetuning, one can observe in Table 4.9 that the proposed networks have undoubtedly improved their results on KITTI training dataset, however they are not

better than the competing algorithms shown in Table 4.9. Qualitative results show similar indications as well.

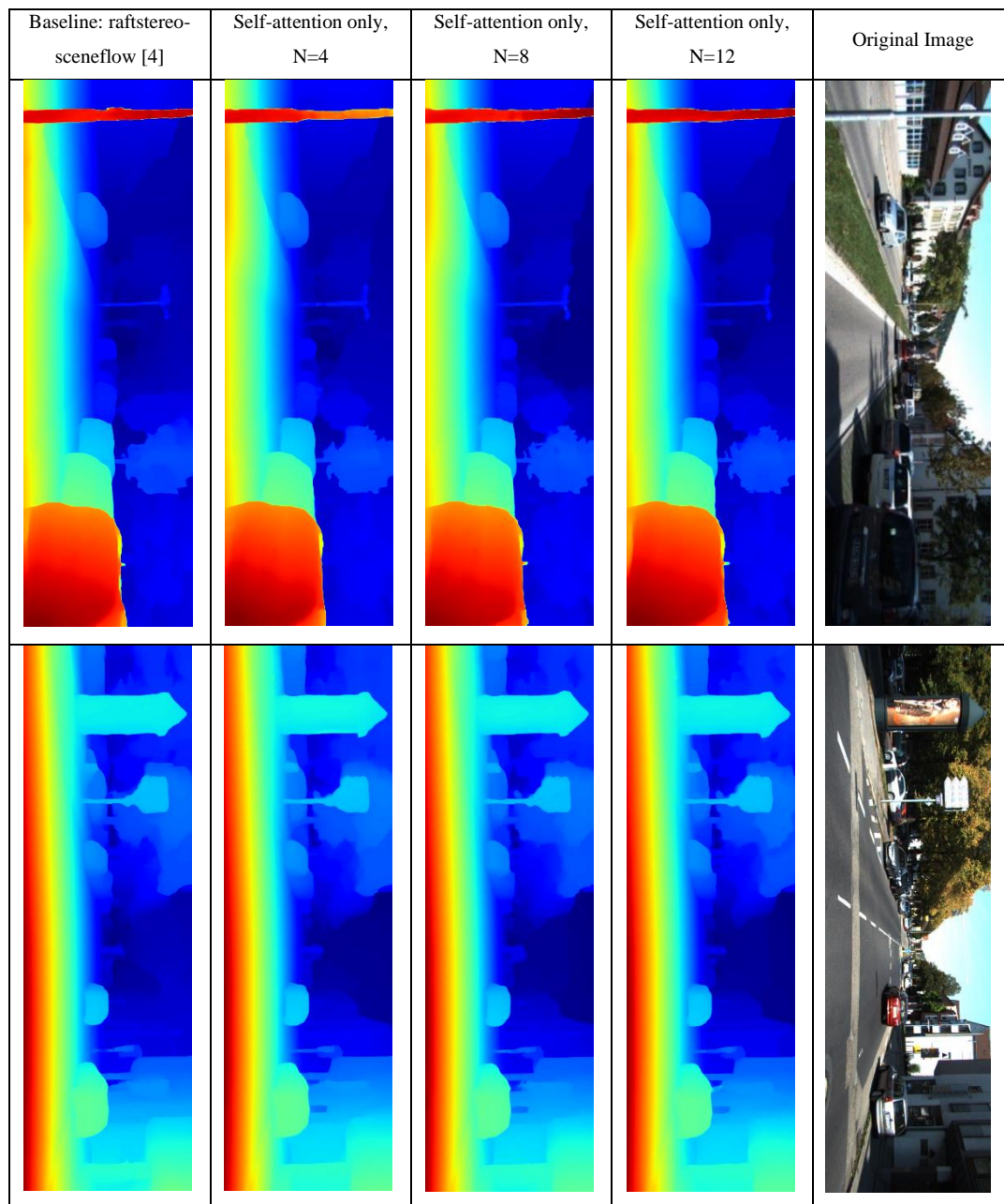


Figure 4.9 The qualitative results of the networks that have self-attention only and the baseline network

To conclude, the proposed networks with self-attention modules are tested in terms of synthetic-to-real world generalization, and it is observed that they performed better synthetic-to-real world generalization on the KITTI training dataset than both

RAFT-Stereo and other competing algorithms, the proposed networks were only competitive on the other two datasets, unfortunately. During the texture analysis, a decrease of the absolute performance on textured areas (both EPE and 3px % error) is observed as the number of self-attention module increases. And after the finetuning operation, the proposed networks stayed behind other networks.

4.2.4 Experiment 3 - Training the Network with Cross-Attention Modules Only

All the self-attention modules are removed in this final experiment, and computations are done with the cross-attention modules only. This experiment is conducted to understand the in-painting effects of the cross-attention modules in a complete manner. During all tests, the previous hyperparameter search results are used. During all tests, N is chosen as $N=4$, $N=8$, and $N=12$, as these will function as the ablation tests.

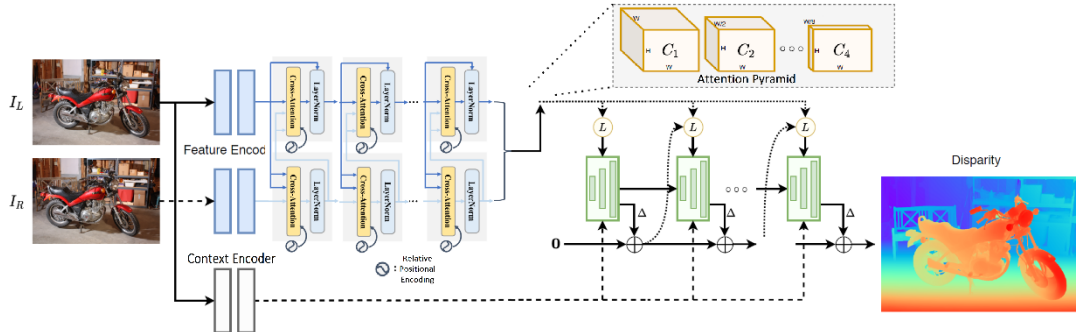


Figure 4.10 The architecture of the network in Experiment 3 (Revised from [4])

4.2.4.1 Synthetic-to-Real Generalization Performance

The initial phase of the training is the synthetic-to-real generalization training phase. The quantitative results of the synthetic-to-real experiments and results of other competing algorithms are shown in Table 4.10 and the qualitative results are shown in Figure 4.11.

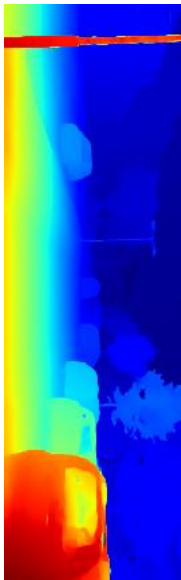

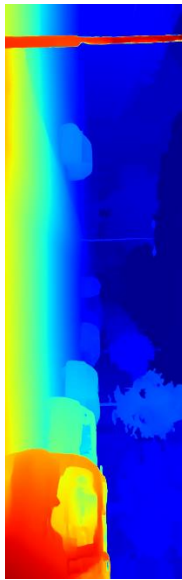
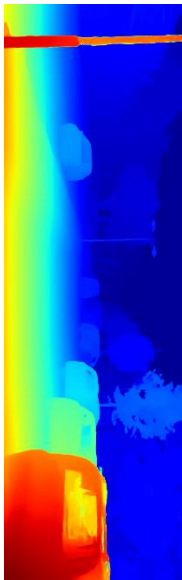

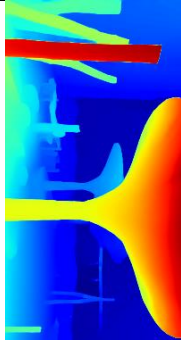
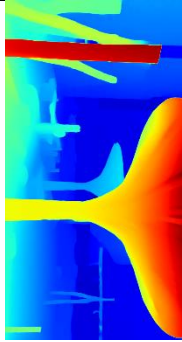
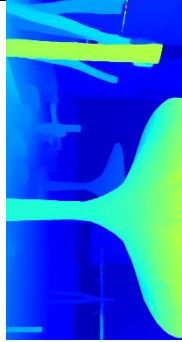
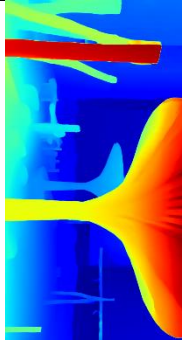

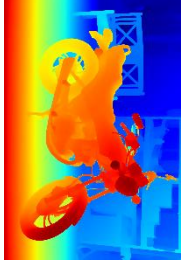
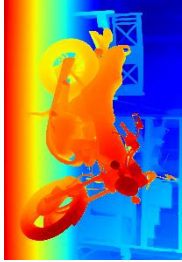
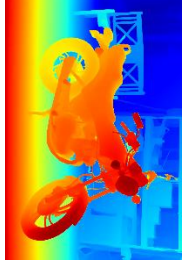
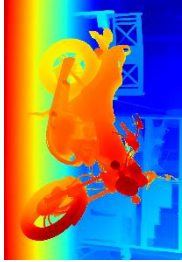

Baseline: raftstereo-sceneflow [4]	Cross-attention only, N=4	Cross-attention only, N=8	Cross-attention only, N=12	Original Image
				
				
				

Figure 4.11 The synthetic-to-real generalization performance of the proposed networks and the baseline network [4]

In the first row of Figure 4.11, straightness of the backplate of the closest car should be noted; in the second row, visibility of the indentations should be examined; and in the last row, similarly the thin rods on the upper shelf are critical.

In order to not populate this thesis with the same tables, only the closest two competing algorithms are kept in Table 4.10 and in the following tables. The proposed networks with cross-attention modules performed better on synthetic-to-real generation task than the baseline and other competitor networks.

Table 4.10 The synthetic to real generalization results, compared with the best competing algorithms

Datasets	KITTI	Middlebury		ETH3D
Networks trained with only synthetic images	3px % error	2px % error of half set	2px % error of quarter set	1px % error
cross-attention only, N=4	5.4873	9.6963	9.2349	2.8317
cross-attention only, N=8	5.1884	11.564	8.7276	3.2366
cross-attention only, N=12	5.1434	10.724	8.3792	2.6902
Baseline (raftstereo-sceneflow) [4]	5.6985	12.59	9.36	3.28
DSMNet [125]	6.5	13.8	8.1	6.2

4.2.4.2 Comparing Performance on the Textured and Textureless Areas

A similar experiment is performed for the textured and textureless areas. As shown in Table 4.11, in synthetic-to-real generalization tasks, the proposed networks, especially N=8 and N=12 perform the best. The network for N=12 performed better in both textured and textureless areas, which makes its general result the best among all.

Examining at the ratios to speak relatively, one can see that the performance in textured areas is compromised percentage-wise for the improvement of the performance in the textureless areas percentage-wise. While the baseline has 1.3462:0.9513=1.4151 ratio for the textured and textureless areas respectively, N=8 networks performance percentage has shifted to 1.4838:0.932=1.5920.

Considering the overall picture, one can argue that cross-attention modules increase the performance as their numbers increase, roughly speaking. However, *EPE in textured areas* stayed behind; this scenario does not happen in the case when both of

the attention modules are utilized before; however, a similar situation occurred in self-attention only network—indicating that for better *EPE in textured areas* performance, the interaction between the self and cross-attention modules is more useful.

Table 4.11 Performance on the Textured and Textureless areas

Networks trained with only synthetic images	EPE in general	3px % error in general	EPE in textured areas	3px % error in textured areas	EPE in textureless areas	3px % error in textureless areas	Ratio of EPE in textured areas and EPE in general	Ratio of 3px % error in textured areas and 3px % error in general	Ratio of EPE in textureless areas and EPE in general	Ratio of 3px % error in textureless areas and 3px % error in general
Baseline (raftstereo-sceneflow)	1.1779	5.6985	1.3867	7.6713	1.1345	5.421	1.1773	1.3462	0.9632	0.9513
Cross-attention only, N=4	1.0953	5.4873	1.3255	7.5061	1.0532	5.2033	1.2102	1.3679	0.9616	0.9482
Cross-attention only, N=8	1.0716	5.1884	1.3444	7.6988	1.0252	4.8354	1.2546	1.4838	0.9567	0.932
Cross-attention only, N=12	1.0731	5.1434	1.3313	7.4694	1.0299	4.8162	1.2406	1.4522	0.9597	0.9364

4.2.4.3 Finetuning the Networks

In this section, the networks are finetuned by using the KITTI training dataset. As before, during this phase 10-fold cross-validation method is applied, since this is a small dataset with only 200 images. The learning rate is selected as 10^{-5} , data augmentations such as saturation and stretches were applied. The qualitative results of the proposed finetuned networks are shown in Figure 4.12, and the quantitative results are shown in Table 4.12.

The proposed networks as before failed to surpass the baseline after finetuning, one can hypothesize this might have to do with attention-modules being data-hungry than other types of networks.

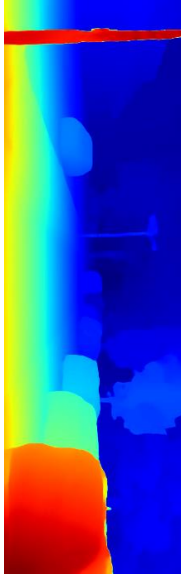
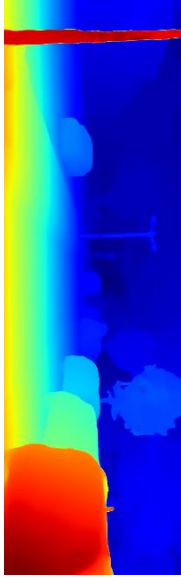
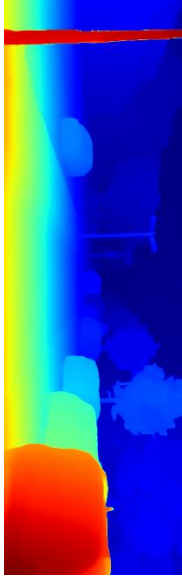
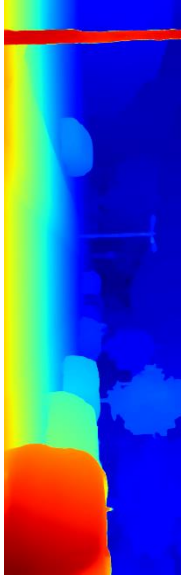

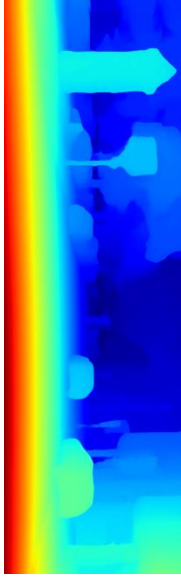
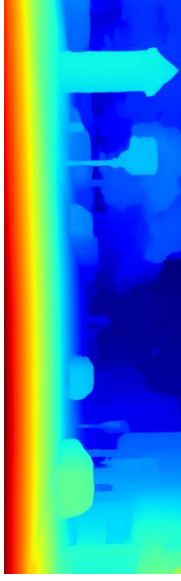
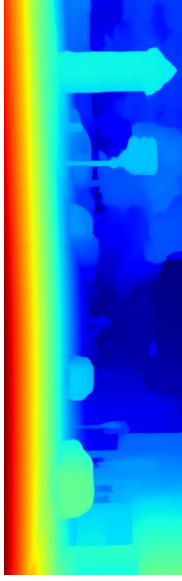
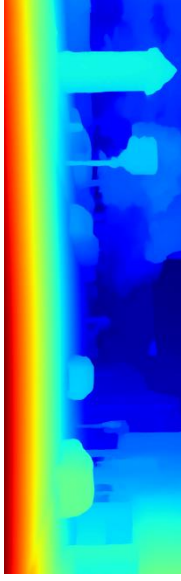

Baseline: raftstereo-sceneflow	Cross-attention only, N=4	Cross-attention only, N=8	Cross-attention only, N=12	Original Image
				
				

Figure 4.12 The qualitative results of the finetuned networks and the baseline network

To conclude, the proposed networks were tested in synthetic-to-real world generalization and observed that they all performed better synthetic-to-real world generalization in all datasets, better than RAFT-Stereo and other state-of-the-art algorithms. During the texture analysis, an increase in the performance of textured areas and textureless areas are observed. Still, the performance increase in the

textured areas was stagnant with the number of cross-attention modules. Our synthetic-to-real results improved naturally in the finetuning tests; however, they are not good enough to pass the performance of the baseline network.

Table 4.12 Performance comparison of the networks finetuned with the KITTI dataset

Networks finetuned with KITTI training dataset	3px % error
Cross-attention only, N=4	2.3816
Cross-attention only, N=8	2.2678
Cross-attention only, N=12	2.2324
Baseline (RAFT-Stereo)	1.96
AcfNet	1.89
AMNet	1.84
OptStereo	1.82
GANet-deep	1.81
SUW-Stereo	1.80
GANet + DSMNet	1.77
CSPN	1.74
LeaStereo	1.65
STTR [61]	2.01

4.2.5 Discussion of the Analysis 1

As a summary, all the results from the synthetic-to-real task are compiled in Table 4.13. Additionally, all the results from the final finetuned tasks are also compiled in Table 4.14.

The following conclusions can be stated based on the depth estimation performances observed in this analysis:

- In the synthetic-to-real world task, the best network was the version of the proposed networks that used both self and cross attention with N=8. It surpassed the baseline and the other competing networks.

Table 4.13 The synthetic to real task with all of the proposed networks and the baseline network [4]

Which modules are utilized	How many is used	EPE in general	3px % error in general	EPE in textured areas	3px % error in textured areas	EPE in textureless areas	3px % error in textureless areas
both	N=4	1.1931	5.3174	1.3112	7.3482	1.1528	5.0317
both	N=8	1.0402	4.7782	<u>1.3001</u>	<u>7.2698</u>	0.9967	4.4277
both	N=12	<u>1.0561</u>	<u>4.9097</u>	1.2897	7.0589	<u>1.0149</u>	<u>4.6074</u>
self	N=4	1.1130	5.5657	1.3973	8.1588	1.0677	5.2010
self	N=8	1.1807	5.5306	1.4158	7.9317	1.1351	5.1929
self	N=12	1.1301	5.7419	1.4276	8.5189	1.0816	5.3513
cross	N=4	1.0953	5.4873	1.3255	7.5061	1.0532	5.2033
cross	N=8	1.0716	5.1884	1.3444	7.6988	1.0252	4.8354
cross	N=12	1.0731	5.1434	1.3313	7.4694	1.0299	4.8162
Baseline network[4]		1.1779	5.6985	1.3867	7.6713	1.1345	5.4210

- If the performance of the textured and textureless areas is examined separately, the attention modules increase the absolute performance in both textured and textureless areas. This result confirms the previous hypothesis that attention modules can be a good choice for textureless areas. In other words, they did not reduce the performance by the *in-paintings* they create, while the attention modules gather context around them (either context available from the same image or the other image). The best performance result occurs by using the self and cross-attention together.
- None of the proposed finetuned networks could surpass the baseline network, even though one of them got close. This small failure might be dedicated to the fact that attention modules are generally more data-hungry or that since more parameters are added to the original baseline network, the training simply might need a larger annotated dataset.

Table 4.14 The results of the proposed finetuned networks and the finetuned baseline network [4]

Validation Dataset		KITTI training dataset	
Which modules are utilized	How many is used	EPE in general	3px % error in general
both	N=4	0.7052	2.7656
both	N=8	0.6074	1.9997
both	N=12	<u>0.5887</u>	<u>1.9777</u>
self	N=4	0.6758	2.3907
self	N=8	0.6763	2.4856
self	N=12	0.6652	2.4722
cross	N=4	0.6454	2.3816
cross	N=8	0.6269	2.2678
cross	N=12	0.6243	2.2324
finetuned baseline network[4]		0.5700	1.9600

4.3 Analysis 2: Exploitation of Temporal Information in RAFT-Stereo Algorithm

In this Analysis 2, the aim is to improve the depth estimation performance of RAFT-Stereo algorithm by initializing its iterations with previously available information. RAFT-Stereo algorithm basically calculates the depth by refining its iterations up to a defined number ($i=32$), however it disregards the previous calculation at the start of each depth calculation. One can hypothesize in this analysis that in a dataset that have temporal continuation or in real world, that disregarded information can be in fact quite useful, and initialization of the iteration from a *good enough* solution might help the algorithm converge better and be faster. In this analysis, this hypothesis is going to be tested.

Apart from initialization, one may also try to utilize the ego-pose estimation property, as it is a handy idea. 3D ego-pose estimation is usually calculated by taking adjacent images in time and calculating 6 degrees of freedom that express that 3D translation and 3D rotation between the previous frame and the next frame. When

combined with the depth calculations, the ego-pose estimation concept may even free the training algorithm from the supervised learning scheme.

Additionally, the ego-pose estimation can be calculated by the intermediate concept of optical flow [131]. Optical flow algorithms are used for tracking motion, it can be the motion of other objects or the ego-motion of the vehicle the algorithm is running on. Hence, one can use optical flow algorithms like the original RAFT [46] in this task.

Hence, the experiments performed in this analysis are:

- Initialize the iterations of the RAFT-Stereo algorithm with the previous predictions in three different ways,
- Finetune the RAFT-Stereo algorithm using an ego-pose estimation network and a self-supervision scheme.

Every attention module experiment has been conducted with an Nvidia RTX 3090 GPU. Every training lasted 3.5 days with 200.000 steps in the end. In all of the experiments, parameters of the RAFT-Stereo architecture have been started with the Kaiming initialization. The parameters of the original RAFT algorithm have been kept frozen. In the experiments, 320x720 pixel crops have been utilized for training.

4.3.1 Experiment 1 - Using Previous Predictions and Optical Flow to Initialize RAFT-Stereo Algorithm

In the first proposed experiment, different architectures are utilized to initialize the network.

First, using previous predictions by simply one layer of CNN layer with 1x1 filter size is examined, as shown in Figure 4.13. In this analysis these networks are denoted as *networks with 1x1 CNN filter*. Utility of using different amounts of previous prediction is investigated as well. The version that utilizes three previous predictions has 11,241M parameters, the version that uses two previous predictions has 11,173M

parameters, the version that utilizes one previous prediction has 11,105M parameters.

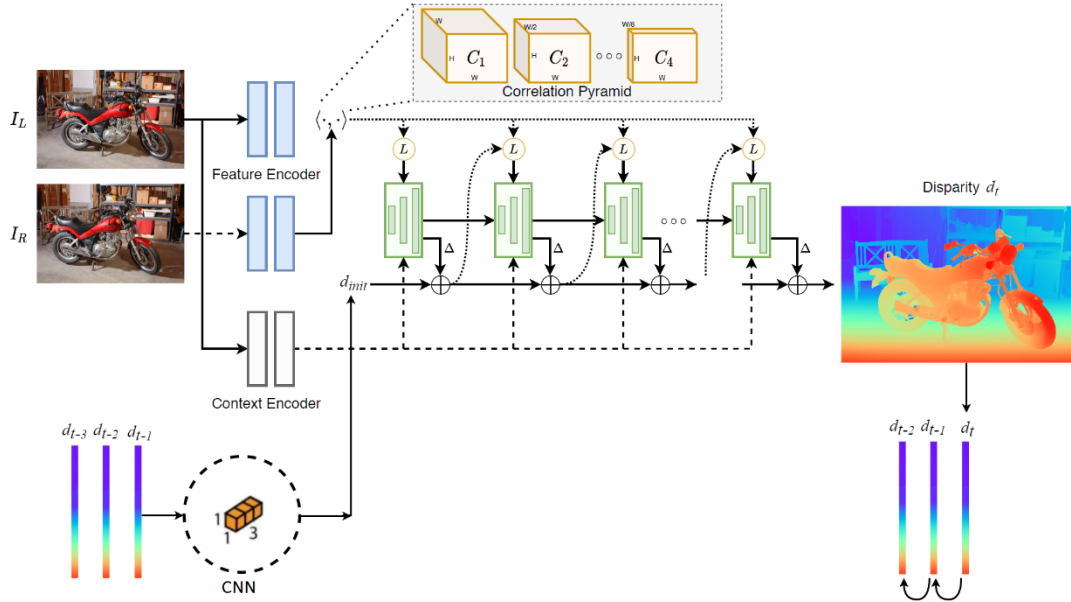


Figure 4.13 The architecture of the network with 1x1 CNN filter (Revised from [4])

Second, one may also try to utilize multiple previous predictions with deep U-Net blocks, as shown in Figure 4.14. Using different amounts of previous predictions in this network is tested as well. In this thesis, these networks are denoted as *networks with U-Net*. The version that utilizes three previous predictions has 18,879M parameters, the version that utilizes two previous predictions has 18,878M parameters and the version that utilizes one previous prediction has 18,878M parameters.

The layer definition of the utilized U-Net is illustrated in Figure 4.15. This illustration is for the network that utilizes three previous predictions. Hence its input is denoted as 320x720x3, others naturally will have inputs of size 320x720x2 and 320x720x1. In these proposed networks one, two and three previous predictions are utilized, to hopefully give the network a sense of velocity and acceleration, which requires at least two previous predictions and three previous predictions, respectively.

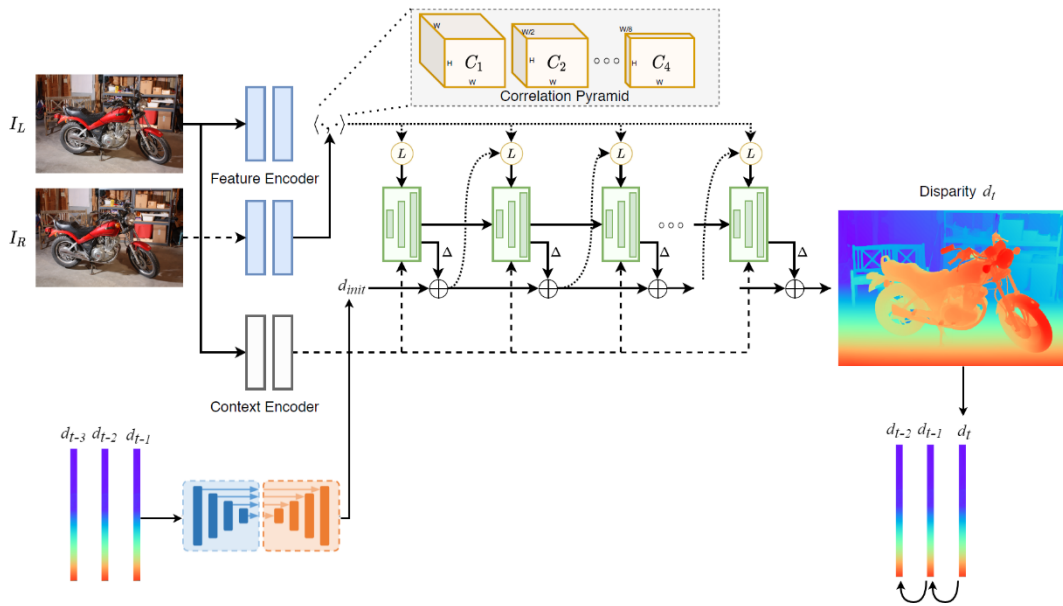


Figure 4.14 The architecture of the network with U-Net (Revised from [4])

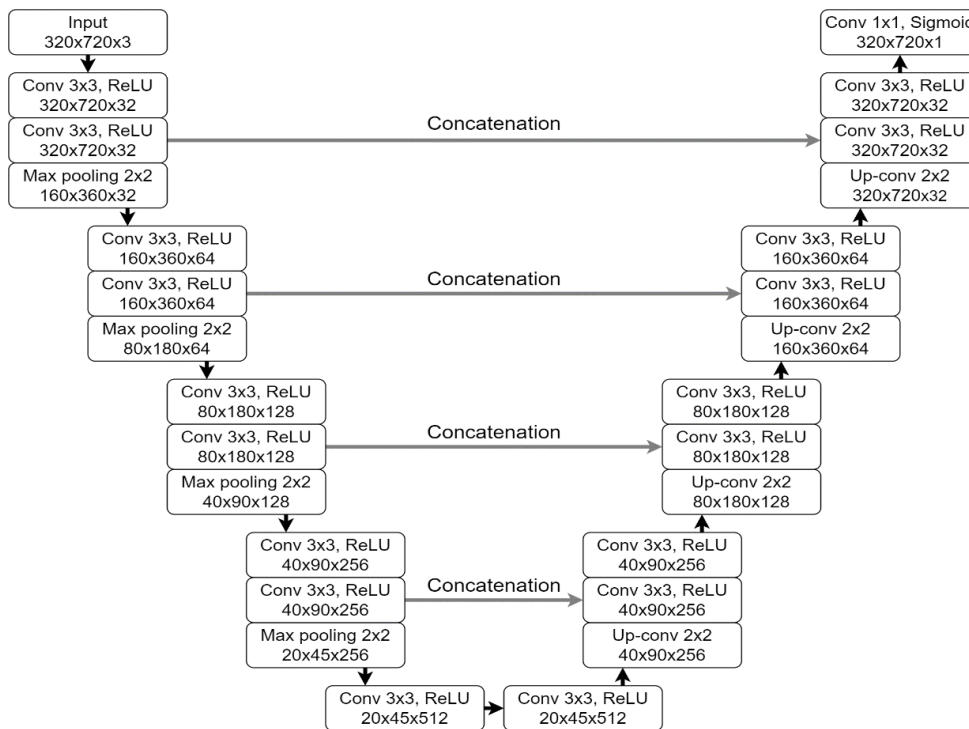


Figure 4.15 The U-Net blocks utilized in the network with U-Net which utilizes 3 previous predictions

Finally, the previous depth estimation on the image plane is shifted by using the optical flow output of the original RAFT algorithm, as depicted in Figure 4.16. Here the previous prediction and the previous optical flow data are utilized only. These networks are denoted as *networks with optical flow* in this thesis.

Once the analysis of this training scheme is set, we realize some minor points in the dataset preparing step to be solved. First of all, ETH3D and Middlebury has no temporal continuation; hence one, unfortunately, cannot use them to validate the model. Thankfully, SceneFlow has many temporally-related image sequences. This opportunity dramatically helps with the training scheme, and to use this to its full advantage, the shuffle feature in the data loader is turned off. Moreover, if a new sequence begins, the previous prediction is simply reset to all zeros.

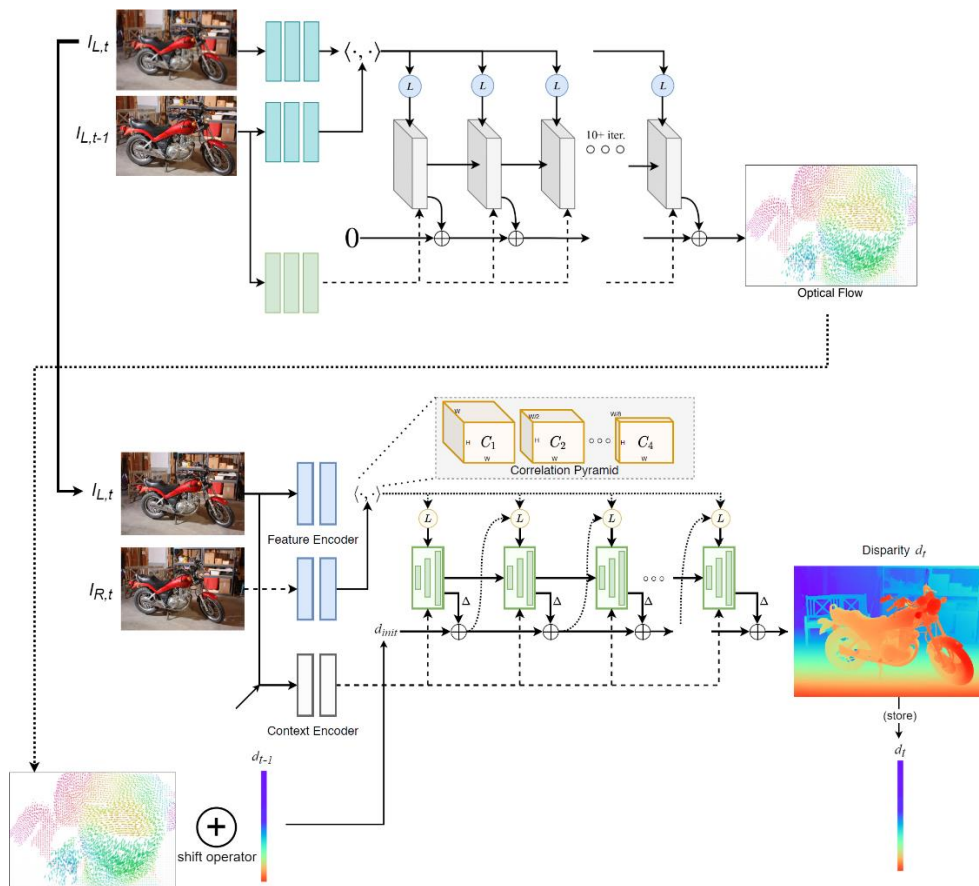


Figure 4.16 The architecture of the network that utilizes optical flow (Revised from [4], [46])

4.3.1.1 Hyperparameter Search

Before starting the following experiments, a quick hyperparameter search for the network with 1x1 CNN filter and for the network with U-Net is conducted. This hyperparameter search is performed for the learning rate and dropout—the hyperparameter grid search combination in Table 4.15.

Table 4.15 Hyperparameter grid search values

Dropout \ learning rate	$2 \cdot 10^{-4}$	$11 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
0.0	(0.0, $2 \cdot 10^{-4}$)	(0.0, $11 \cdot 10^{-5}$)	(0.0, $2 \cdot 10^{-5}$)
0.1	(0.1, $2 \cdot 10^{-4}$)	(0.1, $11 \cdot 10^{-5}$)	(0.1, $2 \cdot 10^{-5}$)

The results for the network with 1x1 CNN filter utilizing three previous frames are shown in Table 4.16. Our primary error metric is 3px % error. Hence, for the network with 1x1 CNN filter, the (0.0, $11 \cdot 10^{-5}$) option is selected to be the best. The versions that utilize two previous predictions and one previous prediction are trained using these hyperparameters.

Table 4.16 Results of hyperparameter search for the network with 1x1 CNN filter

Validation Dataset	KITTI	
	EPE	3px % error
Networks with 1x1 CNN filter		
Network with (0.0, $2 \cdot 10^{-4}$)	1.0496	5.8256
Network with (0.0, $11 \cdot 10^{-5}$)	1.0471	5.7026
Network with (0.0, $2 \cdot 10^{-5}$)	1.1461	5.9029
Network with (0.1, $2 \cdot 10^{-4}$)	1.2040	6.6989
Network with (0.1, $11 \cdot 10^{-5}$)	1.1372	5.9883
Network with (0.1, $2 \cdot 10^{-5}$)	1.2370	5.9977
Baseline: raftstereo-sceneflow [4]	1.1267	6.0991

The results for the network with U-Net are shown in Table 4.17. Looking at the results, the $(0.1, 2 \cdot 10^{-5})$ option is chosen for the network with U-Net. This means the versions that utilizes two previous predictions, and one previous prediction are trained using these hyperparameters.

Table 4.17 The result of the hyperparameter search for the network with U-Net

Validation Dataset	KITTI	
	EPE	3px % error
Networks with U-Net		
Network with $(0.0, 2 \cdot 10^{-4})$	1.1914	6.8479
Network with $(0.0, 11 \cdot 10^{-5})$	1.0695	5.8622
Network with $(0.0, 2 \cdot 10^{-5})$	1.1237	6.2052
Network with $(0.1, 2 \cdot 10^{-4})$	1.1379	6.3408
Network with $(0.1, 11 \cdot 10^{-5})$	1.0637	5.9385
Network with $(0.1, 2 \cdot 10^{-5})$	1.0472	5.4362
Baseline: raftstereo-sceneflow [4]	1.1267	6.0991

4.3.1.2 Synthetic-to-Real Generalization Performance

During the validation process of the training phase with SceneFlow dataset, one cannot utilize every image pair of the KITTI training dataset, due to lack of temporal continuation between some of the image pairs in the dataset. Out of 200 KITTI training image pairs, only 126 image pairs have visible temporal continuation. Hence, in this experiment, the validation dataset is even smaller, and the validated performance of network (denoted as *raftstereo-sceneflow*) in this smaller dataset is different. To be more specific its *3px % error* changes from 5.6985 to 6.0991 in this subset of the KITTI dataset. In this thesis, this subset of the dataset will be referred as *KITTI 126 training subset*.

The quantitative results of the synthetic-to-real experiments and results of the baseline network are shown in Table 4.18. The proposed network performed better

than the baseline network. Networks with 1x1 CNN filter and the optical flow unfortunately performed worse than the baseline.

In this table, one can observe that during this synthetic-to-real generalization task, the proposed 2 networks with U-Net performed better than the baseline network. Networks with 1x1 CNN filter and the optical flow are unfortunately worse than the baseline.

Table 4.18 Quantitative comparison of the networks and the baseline network

Validation Dataset	KITTI	
	EPE	3px % error
Networks trained with only synthetic images		
Network with 1x1 filter, utilizing 3 previous predictions	1.0471	5.7026
Network with 1x1 filter, utilizing 2 previous predictions	1.1215	6.2082
Network with 1x1 filter, utilizing 1 previous prediction	1.1784	6.5437
Network with U-Net, utilizing 3 previous predictions	1.0472	5.4362
Network with U-Net, utilizing 2 previous predictions	1.0350	5.6088
Network with U-Net, utilizing 1 previous predictions	1.0564	5.6059
Network with optical flow	1.1586	6.2715
Baseline: raftstereo-sceneflow [4]	1.1267	6.0991

Qualitative results are shown in Figure 4.17, Figure 4.18 and Figure 4.19. The most successful proposed network in this phase is the network with U-Net utilizing 3 previous predictions. In the first rows the T-shaped streetlamp, in the second rows the blue cars shape in general, and in the last rows the smoothness of the back of the red car should be noted.

The proposed network with optical flow showed some noticeable problems where there appear to be holes inside moving objects. Between 1x1 filter group, U-Net group and the optical flow versions of the network, the most successful group is U-Net, and the least successful one is the optical flow.

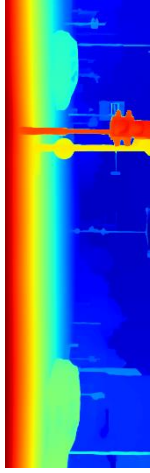
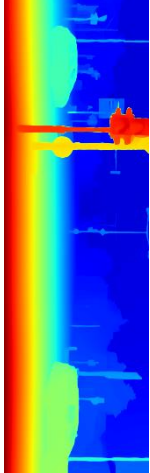
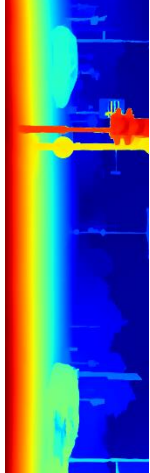
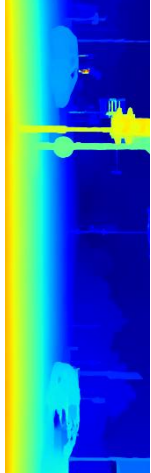

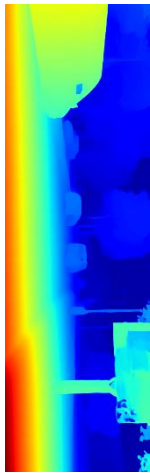
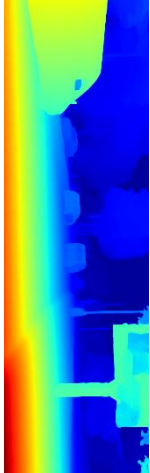
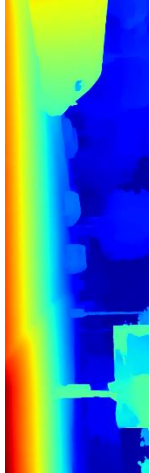


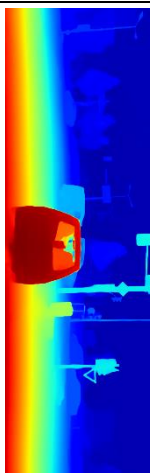
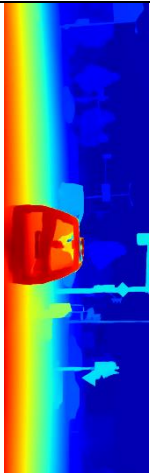
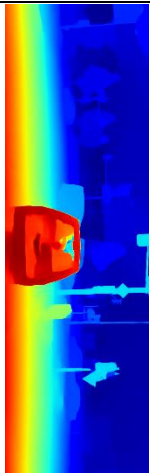
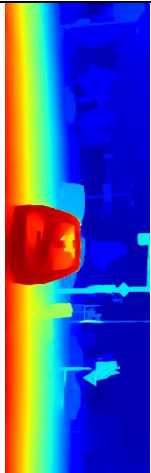

Baseline: raftstereo-sceneflow [4]	Network with 1x1 filter, utilizing 3 previous predictions	Network with 1x1 filter, utilizing 2 previous predictions	Network with 1x1 filter, utilizing 1 previous prediction	Original Image
				
				
				

Figure 4.17 Qualitative results of the networks with 1x1 CNN filter and the baseline network [4]

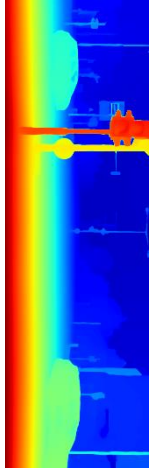
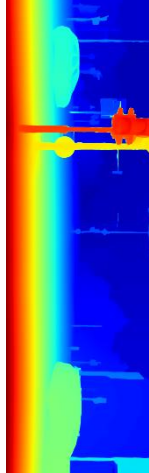
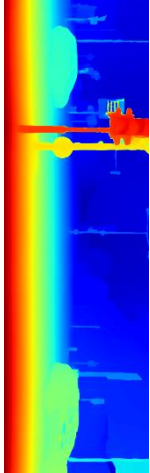
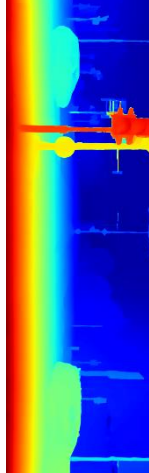

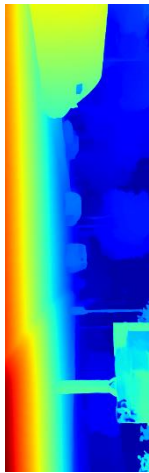
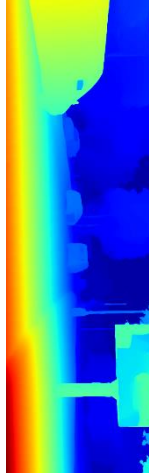
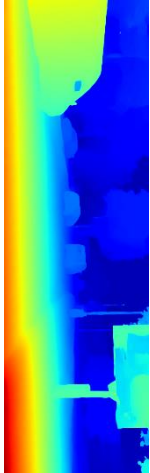
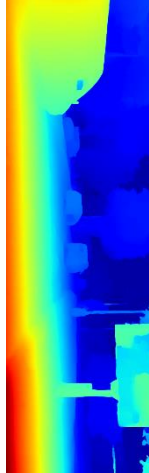

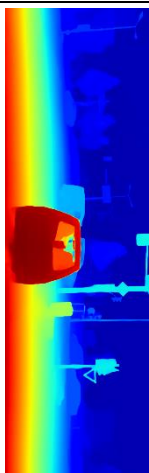
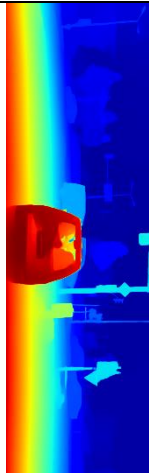
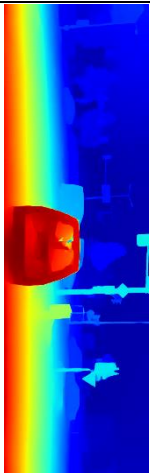
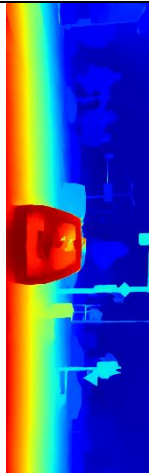

Baseline: raftstereo-sceneflow [4]	Network with U-Net, utilizing 3 previous predictions	Network with U-Net, utilizing 2 previous predictions	Network with U-Net, utilizing 1 previous prediction	Original Image
				
				
				

Figure 4.18 Qualitative results of the networks with U-Net and the baseline network [4]

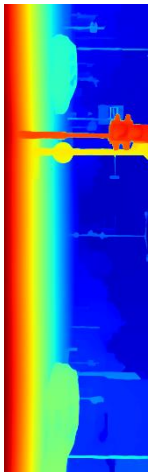
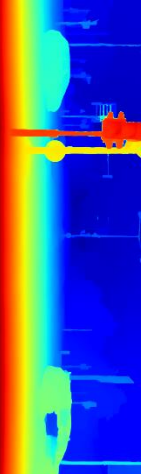

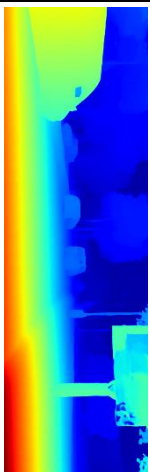
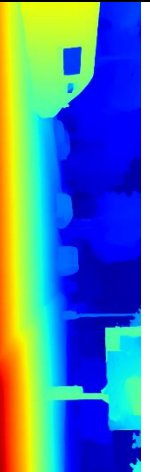

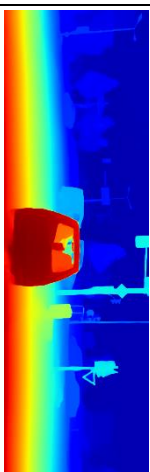
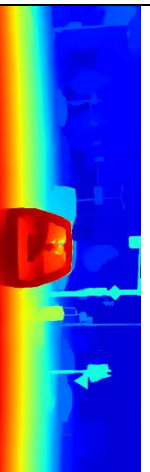

Baseline: raftstereo-sceneflow [4]	Network with optical flow	Original Image
		
		
		

Figure 4.19 Qualitative results of the network with the optical flow and the baseline network

4.3.1.3 Finetuning the networks

In this phase, the networks are finetuned using the *KITTI 126 training* subset. During this phase 9-fold cross-validation method is applied due to the fact this is a small dataset with only 126 images and 126 is divisible by 9. The learning rate is selected as 10^{-5} , data augmentations such as saturation and stretches were applied.

Quantitative results are shown in Table 4.19. The proposed networks have failed to surpass the baseline network in the finetuning phase again, even though some were more successful than the baseline network in the previous phase. Surprisingly, the network with optical flow show performance gain greater than others however it is not enough to surpass the baseline network.

Table 4.19 The baseline network and the proposed networks finetuned with KITTI 126 training subset

Dataset	KITTI	
	EPE	3px % error
Networks finetuned with KITTI 126 training subset		
Network with 1x1 filter, utilizing 3 previous predictions	0.9146	4.3818
Network with 1x1 filter, utilizing 2 previous predictions	0.8107	4.6253
Network with 1x1 filter, utilizing 1 previous prediction	0.8943	5.2571
Network with U-Net, utilizing 3 previous predictions	0.7678	3.5201
Network with U-Net, utilizing 2 previous predictions	0.7900	4.1905
Network with U-Net, utilizing 1 previous prediction	0.9288	5.2251
Network with optical flow	<u>0.7387</u>	<u>2.8018</u>
Baseline: raftstereo-sceneflow [4]	0.6266	2.7084

The qualitative results of the proposed finetuned networks are shown in Figure 4.20, Figure 4.21 and Figure 4.22. All of the proposed networks exhibit small certain unwanted elements, enough to stay behind the baseline network. Even though during the finetuning phase the results of the proposed networks improved naturally, the performance of the baseline network has improved more. According to these

observations, one can argue that original design is indeed an exemplary network architecture selection when it comes to finetuning phase.

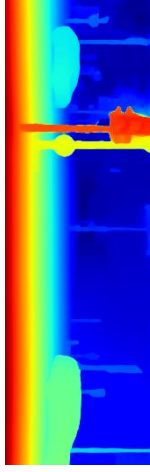
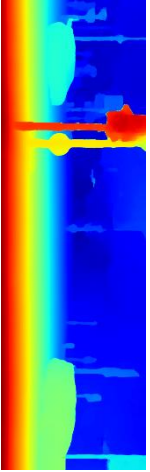
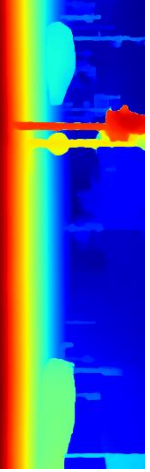
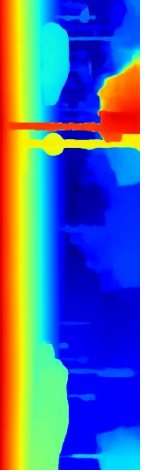

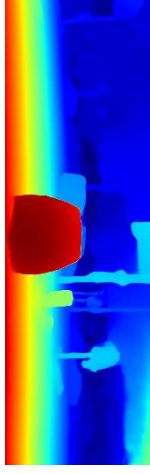
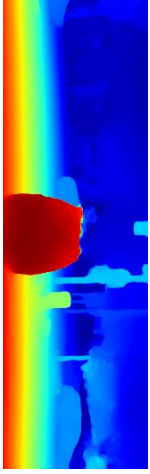
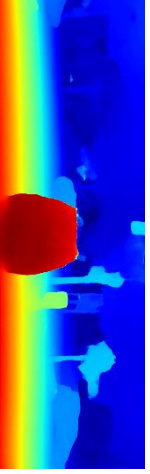
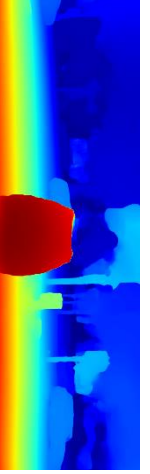

Baseline: raftstereo-sceneflow [4]	Network with 1x1 filter, utilizing 3 previous predictions	Network with 1x1 filter, utilizing 2 previous predictions	Network with 1x1 filter, utilizing 1 previous prediction	Original Image
				
				

Figure 4.20 Qualitative results of the finetuned baseline network and the proposed finetuned networks with 1x1 filters

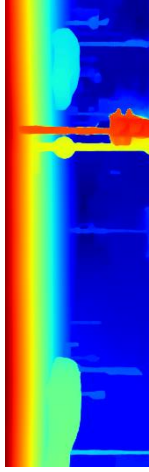
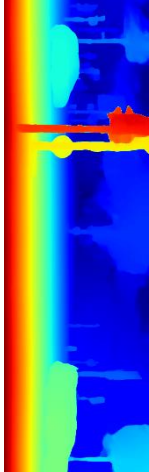
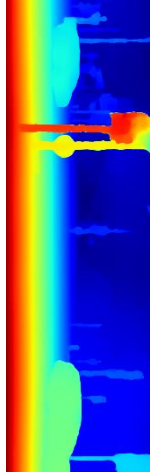
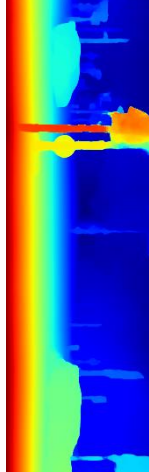

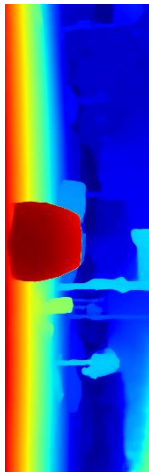
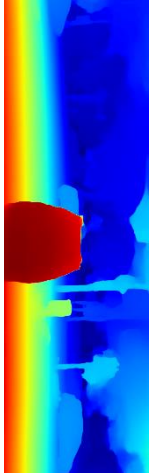
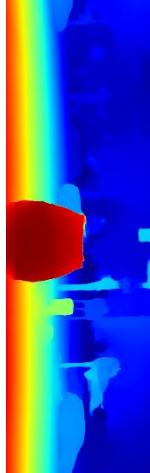
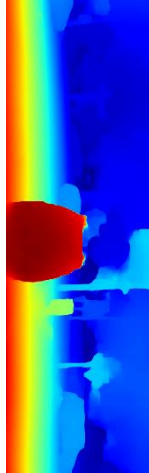

Baseline: raftstereo-sceneflow [4]	Network with U-Net, utilizing 3 previous predictions	Network with U-Net, utilizing 2 previous predictions	Network with U-Net, utilizing 1 previous prediction	Original Image
				
				

Figure 4.21 Quantitative results of the finetuned baseline network and the proposed finetuned networks with U-Net

To summarize, again the proposed networks performed better at synthetic-to-real world tasks, however after the finetuning process, they stayed behind the improvement of the original network.

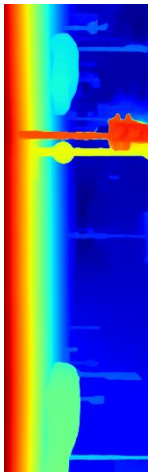
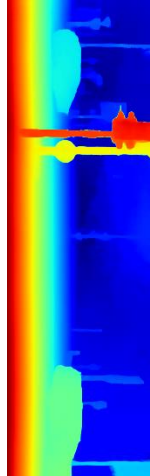

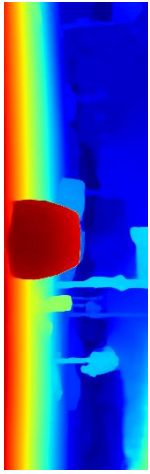
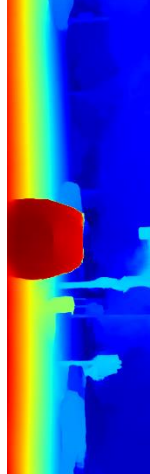

Baseline: raftstereo-sceneflow [4]	Network with optical flow	Original Image
		
		

Figure 4.22 Quantitative results of the finetuned baseline network and the proposed finetuned networks with optical flow

4.3.2 Experiment 2 - Finetuning RAFT-Stereo Algorithm with Ego-Pose Estimation Network and Optical Flow Network

In this experiment, other techniques to improve the performance of RAFT-Stereo algorithm with temporal information are tested. This time the temporal information in the form of an ego-pose estimation network and self-supervision method is utilized.

Pose estimation can be instrumental when it comes to learning depth, since in Euclidian geometry, depth of one scene and depth of the next scene can be related to each other by the concept of pose, a vector with 6 degrees of freedom, 3 for translation and 3 for rotation. For that relation, we require a constant called the *intrinsic calibration matrix*, K , of the camera. MonoDepth2 algorithm utilizes this fact in their network architecture, and they try to learn to depth in a self-supervised way by setting up an expectancy.

In this self-supervised training setting, it is useful to consider depth and ego-pose as intermediate concepts. The network only takes stereo images through time, and this selection means it can only predict what its input will be upon a specific action. The network is *primed* to develop the depth concept by our pre-determined intermediate geometrical constraint. This intermediate geometrical constraint creates expectations for the next input, and it tells what depth will be experienced if the previous depth and predictions of the ego-pose network are correct. Next the original image is warped with this information to prepare what input is expected to get in the next frame. If the depth and ego-pose is calculated correctly, one can estimate what the inputs will be in the next frame. The inputs of the ego-pose estimator in this experiment are simply 2 images adjacent in time.

As it can be observed, there is no need for annotation. For this experiment, this thesis hypothesizes that one can utilize the real-world data, skip the finetuning process, and perform similar or better than the annotated dataset. For this purpose, the raw KITTI dataset is preferred, similar to MonoDepth2 algorithm.

This simple idea of MonoDepth2 is a powerful concept, however it is not as powerful as the RAFT-Stereo algorithm for depth estimation. However, this simple idea may be utilized to improve the finetuning process of the RAFT-Stereo algorithm. As mentioned before, RAFT-Stereo is finetuned for KITTI challenge by the help of the *KITTI training dataset*, this dataset has limited amount of annotated image pairs, only 200 of them. Moreover, among some of them there is no temporal continuation. This is where this self-supervision scheme can come in. With self-supervision one

could use the whole *raw KITTI dataset* composed of 12,919 stereo image pairs and improve the network performance with a larger dataset, without a 10-fold cross-validation method in finetuning phase.

The concept of optical flow also proved itself quite useful for calculating poses, whether it is ego-poses or poses of other objects. There are some recent papers that publish certain works on this idea [131]. The activity of tracking individual pixels along the pixel trajectory in image plane (not in a 3D geometry with depth), might give relevant information about the ego-motion. Hence in this experiment, a network that utilizes an ego-pose estimator network that uses the output of the original RAFT optical flow algorithm provides is tested as well, instead of simply taking the two images adjacent in time.

The experiments performed in this section are:

- Performing finetuning operation on the RAFT-Stereo algorithm with a standard 3D ego-pose estimator network in a self-supervised scheme,
- Performing finetuning operation on the RAFT-Stereo algorithm with an ego-pose estimator network that uses 2D optical flow information in a self-supervised scheme,

Every attention module experiment has been conducted with a Nvidia RTX 3090 GPU. Every training lasted 25.000 steps in the end. Batch size has been selected as 2. In both of the experiments, parameters of the RAFT-Stereo architecture have been started with the synthetically trained *raftstereo-sceneflow* network. The parameters of the original RAFT algorithm have been kept frozen. The parameters of the ego-pose estimation network are initialized with the parameters provided by MonoDepth2 paper. Hyperparameters are not searched in this experiment because this last analysis is a finetuning process. Hence, the finetuning is performed with the same 10^{-5} learning rate previously used, without any dropouts.

4.3.2.1 Performing finetuning operation on the RAFT-Stereo algorithm with a standard ego-pose estimator

The network architecture in this test is shown in Figure 4.23. In the proposed architecture the default RAFT-Stereo architecture and the default MonoDepth2 ego-pose networks are used. When the next pair of images come, an ego-pose estimation is calculated using the current left frame and the next left frame; this calculation is a relative calculation. By the current depth and ego-pose estimates, an expected image is created by warping the original left image. The difference between this warped image and the next left image becomes the photometric loss L_p . L_s is edge-aware smoothness loss, μ is per-pixel mask explained in Section 3.2 and λ is simply the loss hyperparameter (set to be 0.001).

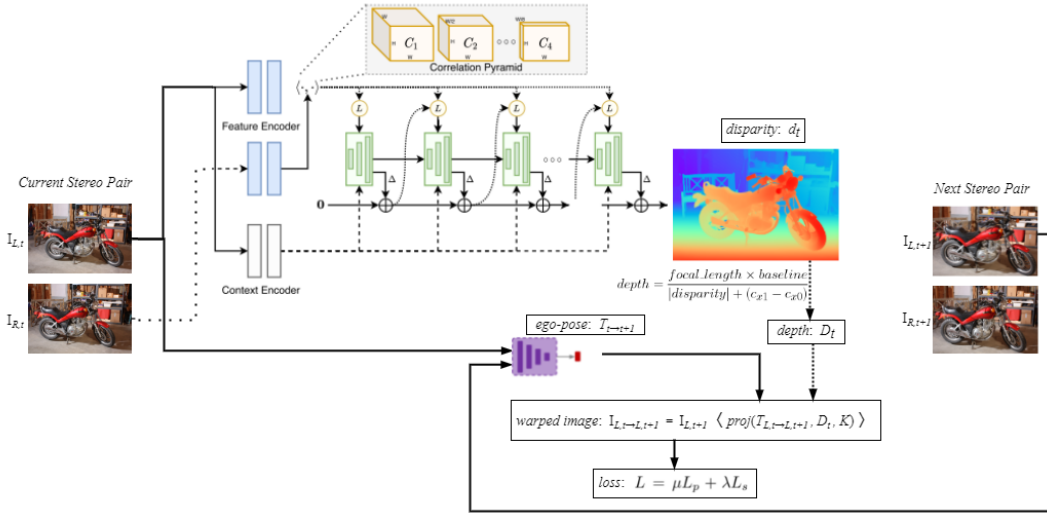


Figure 4.23 Architecture of the network with the standard ego-pose estimator (Revised from [4])

In the experiment, the trained *mono+stereo 320x1024* network from MonoDepth2 is utilized as the ego-pose estimator, the parameters of this ego-pose estimator are not frozen, it is included in the finetune phase as well.

The qualitative results are shown in Figure 4.24. One can observe some distortions on the qualitative results, probably due to the ambiguities of self-supervised learning.

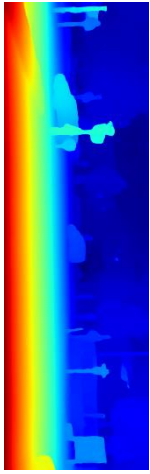
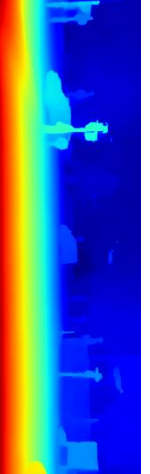

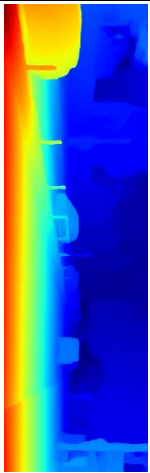
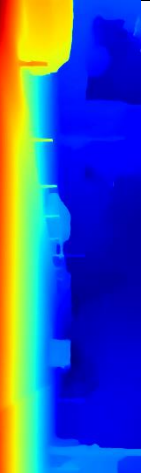

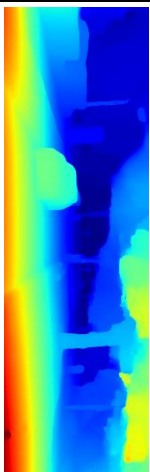
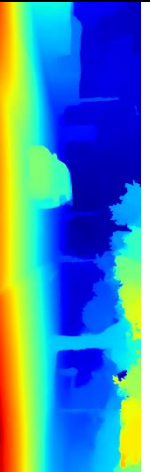

Baseline: raftstereo-sceneflow	Network with standard ego-pose estimator	Original Image
		
		
		

Figure 4.24 Qualitative results of the network with the standard ego-pose estimator vs the finetuned baseline network

Looking at these results, one can conclude that self-supervised learning, even though helped the general error reduction in a small amount, created some problems that did not exist before.

The quantitative results are shown in Table 4.20. Even though this finetuning process with 25000 steps increased the performance of the network some slight amount, it did not improve the results compared to the baseline network. This training method failed to perform better than the training method of the baseline network that uses the KITTI training dataset with 5000 steps and 10-fold cross validation.

Table 4.20 Quantitative results of the network with the standard ego-pose estimator vs the finetuned baseline network

Validation Dataset	KITTI training	
Networks	EPE	3px % error
raftstereo-sceneflow (not finetuned)	1.1779	5.6985
Network with standard ego-pose estimator	0.9377	5.4678
Baseline: finetuned raftstereo-sceneflow [4]	0.5700	1.9600

4.3.2.2 Performing finetuning operation on the RAFT-Stereo algorithm with an ego-pose estimator that uses optical flow information

The network architecture in this test can be visualized, as shown in Figure 4.26. The default RAFT-Stereo architecture is used as the depth estimator network and an architecture close to Flowdometry algorithm [131] is used to turn the optical flow information into ego-pose estimation. For this purpose, the original RAFT optical flow algorithm is utilized as the optical flow network, whose optical flow output is half the original input images in H and W dimensions. The proposed ego-pose estimator network is illustrated in Figure 4.25.

However, in order to use this untrained ego-pose network module for finetuning, first step is to need to make sure it is trained equivalently. For that purpose, the default MonoDepth2 architecture is utilized. Their *pose encoder* network is changed with an

original RAFT network (whose parameters have been kept frozen) and their *pose decoder* network is changed with the ego-pose estimation architecture depicted in Figure 4.25. Then the parameters of other modules in MonoDepth2 are loaded (same as the previous test) and the same MonoDepth2 training procedure is followed for 20 epochs with batch size 12 to acquire the trained ego-pose estimator.

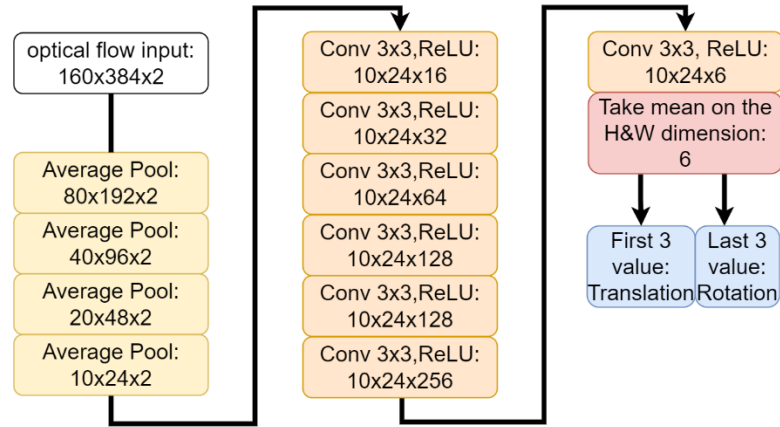


Figure 4.25 The proposed ego-pose estimator architecture inspired by Flowdometry algorithm [131]

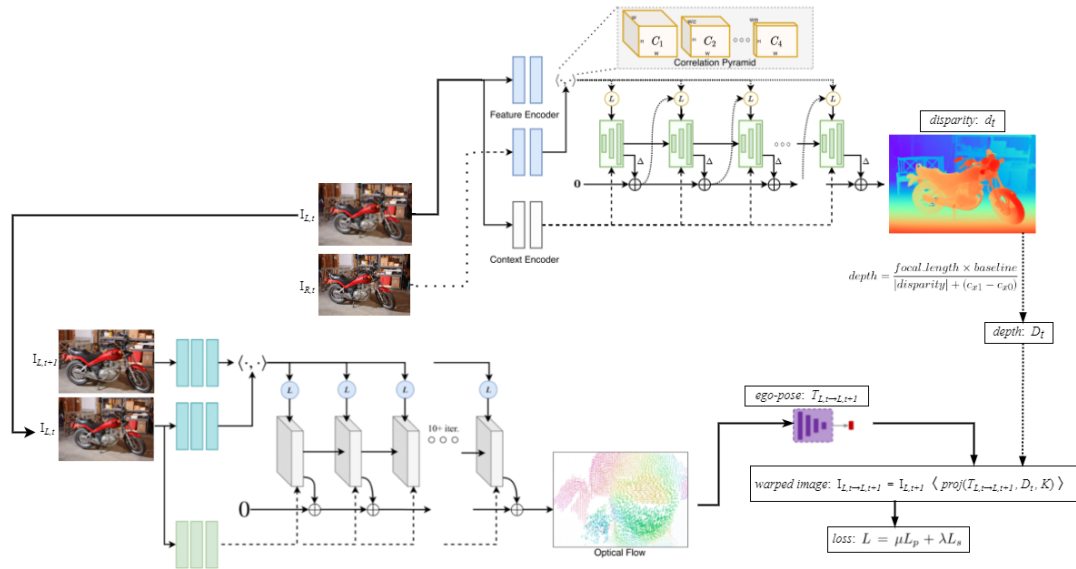


Figure 4.26 Architecture of the network with an ego-pose estimator that uses optical flow information (Revised from [4])

In this architecture when the next pair of images come, an ego-pose estimation is calculated by using the optical flow between current left frame and the next left frame. With the current depth estimation and the ego-pose estimation, a rendered image is created by warping the original left image. The difference between this warped image and the next left image becomes the photometric loss L_p . L_s is edge-aware smoothness loss, μ is per-pixel mask explained in Section 3.2 and λ is simply the loss hyperparameter.

The quantitative results are shown in Table 4.21. It showed some improvement with respect to the previous test, probably due to ego-pose estimator jobs getting easy due to the addition of optical flow. However, it is not better than training with the annotated KITTI training dataset.

Table 4.21 Quantitative network results with an ego-pose estimator that uses optical flow vs the finetuned baseline network.

Validation Dataset	KITTI training	
	EPE	3px % error
raftstereo-sceneflow (not finetuned)	1.1779	5.6985
Network with an ego-pose estimator using optical flow	0.9117	5.3669
Baseline: finetuned raftstereo-sceneflow [4]	0.5700	1.9600

The qualitative results are shown in Figure 4.27. One can realize there is some pixelization introduced, for example, around the signs, cars, and thin objects. One can see some depth edges getting blurred. Also, some holes have been falsely registered as closed and closer surfaces. However, the performance of this finetuning on synthetically trained RAFT-Stereo is slightly better than the previous self-supervision test.

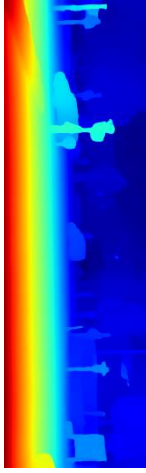
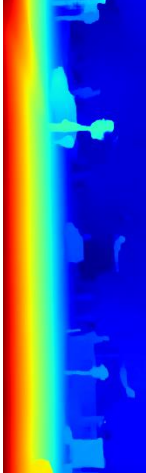

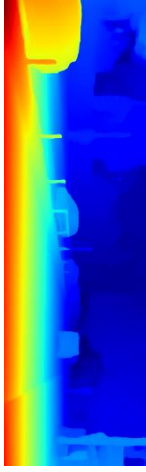
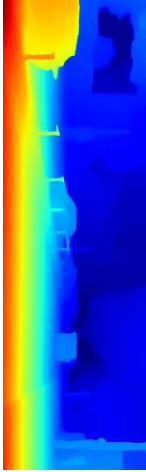

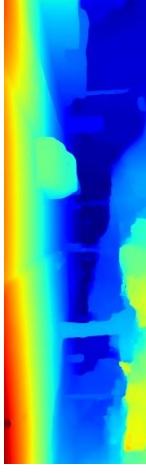
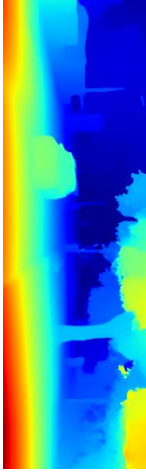

Baseline: raftstereo-sceneflow [4]	Network with an ego-pose estimator using optical flow	Original Image
		
		
		

Figure 4.27 Qualitative results of the network with an ego-pose estimator that uses optical flow vs the finetuned baseline network

4.3.3 Discussion of the Analysis 2

The following conclusions can be stated based on the depth estimation performances observed in the Analysis 2:

- In the first experiment, the aim was to improve the performance of the RAFT-Stereo algorithm by using temporal information available from the previous calculations, such as previous predictions and the optical flow estimations during the synthetic training phase with SceneFlow dataset. In this experiment, it is hypothesized that this information could improve the performance. It is observed that three networks in the U-Net family and one network in the 1x1 CNN filter family worked better than the baseline network in the synthetic training phase. The network with optical flow could not achieve this feat. Nevertheless, this confirmed the hypothesis for this experiment, which is previous predictions can improve the future calculations of an iterative network.
- After the first experiment, all the proposed networks are then finetuned with the KITTI training dataset, and it is observed that they could not pass the performance of the finetuned baseline network [4].
- In the second experiment, the baseline network is finetuned with the unannotated *raw KITTI dataset* in a self-supervised scheme by using a standard ego-pose estimator. The hypothesis was with self-supervision, a large unannotated dataset could be good replacement for a small, annotated dataset. Even though this actually improved the synthetically trained networks, the annotated 200 images KITTI training dataset could achieve this much better, and this result disproved the hypothesis for this experiment.
- The result is the same for the self-supervised scheme with an ego-pose estimator that uses optical flow. It improved the performance of the synthetically trained network, better than the standard ego-pose estimator however the annotated small KITTI dataset could do this much better, and this result disproved the hypothesis for this experiment again.

CHAPTER 5

CONCLUSIONS

This thesis investigated different aspects of improving the RAFT-Stereo algorithm, an end-to-end trainable algorithm in the top 3 of the Middlebury stereo leaderboards. Various types of experiments with attention mechanisms and temporal information utilization mechanisms are proposed to improve the performance of RAFT-Stereo algorithm.

In the Analysis 1, the focus was on the attention mechanisms to improve the performance of RAFT-Stereo algorithm. It is observed that attention mechanisms, as opposed to correlation calculations, performed better synthetic-to-real generalization, better than the baseline network and other competing networks. We had hypothesized attention mechanisms could perform better than the alternative and this result confirmed the hypothesis for this experiment. With the results of these experiments, one can argue the attention mechanisms are valuable tools that can further assist researchers in the near future.

The next focus was on the effects of the cross and self-attention modules on textured and textureless areas. The cascade of self and cross attention modules one after the other performed the best result and improved the absolute 3px % error in both textured areas and textureless areas. The regarding hypothesis was to see improvement in textureless areas. These results confirmed the hypothesis for this experiment. In almost every instance, the attention mechanism, as opposed to just correlations, performs with a little *relative* performance compromise among these areas, meaning its relative performance on textureless areas increases while its relative performance on textured areas decreases. With the results of these experiments, one can argue that these compromise phenomena may be helpful if one tries to apply different networks onto areas with different texture distribution.

Even though some of the proposed finetuned networks come very close, they could not pass the performance of finetuned baseline network. One explanation might be that the attention modules might be more data-hungry, which would indicate a larger annotated KITTI dataset is required for the networks with attention modules. Another explanation might be that small training differences such as crops size can affect the training process. Further investigation of this phenomenon may prove helpful.

In the first experiment of Analysis 2, the aim was to improve the performance of the RAFT-Stereo algorithm by using temporal information available from the previous calculations, such as previous predictions and the optical flow estimations during the synthetic training phase with SceneFlow dataset. 3 networks in the U-Net group and 1 network in the 1x1 filter group performed better than the baseline network in the synthetic training phase. Our hypothesis was the previous calculations could be helpful in an iterative network, and the results confirmed the hypothesis for this experiment. After that, the proposed networks are finetuned with the KITTI training dataset and it is observed that they could not pass the performance of the baseline network. Further investigation of this phenomenon may shed light on this issue.

In the second experiment of Analysis 2, the aim was to finetune the RAFT-Stereo algorithm in a self-supervised way. An ego-pose estimator network enabled us to use the unannotated large raw KITTI dataset. Two types of ego-pose estimators are utilized: a standard network that takes two images adjacent in time, and a network that takes optical flow information instead. It is observed that even though this self-supervised training scheme worked to improve the baseline network slightly, its effects were not as strong enough as the annotated small KITTI training dataset. We hypothesized that we could replace the annotated datasets, and this result disproved this hypothesis for this experiment. Even though this last experiment could not be as successful as hoped, this may be a worthy research area to look into further.

REFERENCES

- [1] S. Papert, “The summer vision project.” pp. 1–6, 1966. [Online]. Available: <http://dspace.mit.edu/handle/1721.1/6125>
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, pp. 145–151, 2012, doi: 10.1145/3383972.3383975.
- [3] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, “Object recognition with gradient-based learning,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1681, no. 0, pp. 319–345, 1999, doi: 10.1007/3-540-46805-6_19.
- [4] L. Lipson, Z. Teed, and J. Deng, “RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching,” *Proceedings - 2021 International Conference on 3D Vision, 3DV 2021*, pp. 218–227, Sep. 2021, doi: 10.1109/3DV53792.2021.00032.
- [5] V. Tankovich, C. Häne, Y. Zhang, A. Kowdle, S. Fanello, and S. Bouaziz, “HitNet: Hierarchical Iterative Tile Refinement Network for Real-time Stereo Matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 14357–14367, 2021, doi: 10.1109/CVPR46437.2021.01413.
- [6] C. Godard, O. mac Aodha, M. Firman, and G. Brostow, “Digging into self-supervised monocular depth estimation,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 3827–3837, Jun. 2019, doi: 10.1109/ICCV.2019.00393.

- [7] M. Dehghani, M. Ahmadi, A. Khayatian, M. Eghtesad, and M. Yazdi, "Vision-based calibration of a Hexa parallel robot," *Industrial Robot*, vol. 41, no. 3, pp. 296–310, 2014, doi: 10.1108/IR-07-2013-376.
- [8] A. Vaswani *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. NIPS, pp. 5999–6009, 2017.
- [9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
- [10] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9992–10002, 2021, doi: 10.1109/ICCV48922.2021.00986.
- [11] O. Press, N. A. Smith, and M. Lewis, "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation," pp. 1–23, 2021, [Online]. Available: <http://arxiv.org/abs/2108.12409>
- [12] S. Zhai *et al.*, "An Attention Free Transformer," 2021, [Online]. Available: <http://arxiv.org/abs/2105.14103>
- [13] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The Efficient Transformer," pp. 1–12, 2020, [Online]. Available: <http://arxiv.org/abs/2001.04451>
- [14] C. Grimm, D. Arumugam, S. Karamcheti, D. Abel, L. L. S. Wong, and M. L. Littman, "Modeling Latent Attention Within Neural Networks," no. 2016, pp. 1–15, 2017, [Online]. Available: <http://arxiv.org/abs/1706.00536>
- [15] H. Laga, L. V. Jospin, F. Boussaid, and M. Bennamoun, "A Survey on Deep Learning Techniques for Stereo-based Depth Estimation," *IEEE*

- Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020, doi: 10.1109/tpami.2020.3032602.
- [16] H. Hirschmüller, “Stereo processing by semi-global matching and mutual information,” *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. II, no. 2, pp. 807–814, 2005, doi: 10.1109/CVPR.2005.56.
- [17] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, “Fast cost-volume filtering for visual correspondence and beyond,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 504–511, 2013, doi: 10.1109/TPAMI.2012.156.
- [18] L. Zhang and S. M. Seitz, “Estimating optimal parameters for MRF stereo from a single image pair,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 331–342, 2007, doi: 10.1109/TPAMI.2007.36.
- [19] D. Scharstein and C. Pal, “Learning conditional random fields for stereo,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, doi: 10.1109/CVPR.2007.383191.
- [20] Y. Li and D. P. Huttenlocher, “Learning for stereo vision using the structured support vector machine,” *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008, doi: 10.1109/CVPR.2008.4587699.
- [21] Y. Ohta and T. Kanade, “Stereo By Intra- and Inter-Scanline Search Using Dynamic Programming,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 2, pp. 139–154, 1985, doi: 10.1109/TPAMI.1985.4767639.
- [22] M. J. Hannah, “Computer Matching of Areas in Stereo Images,” no. May 1968, p. 134, 1974, [Online]. Available:

<http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0786720%5Cnhttp://dl.acm.org/citation.cfm?id=907372>

- [23] R. Zabih and J. Woodfill, “Non-parametric local transforms for computing visual correspondence,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 801 LNCS, pp. 151–158, 1994, doi: 10.1007/bfb0028345.
- [24] Y. S. Heo, K. M. Lee, and S. U. Lee, “Robust Stereo matching using adaptive normalized cross-correlation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 807–822, 2011, doi: 10.1109/TPAMI.2010.136.
- [25] W. S. Fife and J. K. Archibald, “Improved census transforms for resource-optimized stereo vision,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 1, pp. 60–73, 2013, doi: 10.1109/TCSVT.2012.2203197.
- [26] D. Scharstein, R. Szeliski, and R. Zabih, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001*, no. 1, pp. 131–140, 2001, doi: 10.1109/SMBV.2001.988771.
- [27] K. Zhang, J. Lu, and G. Lafruit, “Cross-based local stereo matching using orthogonal integral images,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 1073–1079, 2009, doi: 10.1109/TCSVT.2009.2020478.
- [28] K. J. Yoon and I. S. Kweon, “Locally adaptive support-weight approach for visual correspondence search,” *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. II, pp. 924–931, 2005, doi: 10.1109/CVPR.2005.218.

- [29] M. Bleyer and M. Gelautz, “Simple but effective tree structures for dynamic programming-based stereo matching,” *VISAPP 2008 - 3rd International Conference on Computer Vision Theory and Applications, Proceedings*, vol. 2, pp. 415–422, 2008, doi: 10.5220/0001072904150422.
- [30] R. Fan, “Real-Time Computer Stereo Vision for Automotive Applications,” 2018. [Online]. Available: [https://research-information.bris.ac.uk/en/theses/realtime-computer-stereo-vision-for-automotive-applications\(aaca1358-f103-48a3-bad2-4a07db873ba0\).html](https://research-information.bris.ac.uk/en/theses/realtime-computer-stereo-vision-for-automotive-applications(aaca1358-f103-48a3-bad2-4a07db873ba0).html)
- [31] A. Klaus, M. Sormann, and K. Karner, “Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure,” *Proceedings - International Conference on Pattern Recognition*, vol. 3, pp. 15–18, 2006, doi: 10.1109/ICPR.2006.1033.
- [32] V. Kolmogorov and R. Zabih, “Computing visual correspondence with occlusions using graph cuts,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 508–515, 2001, doi: 10.1109/iccv.2001.937668.
- [33] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient belief propagation for early vision,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, doi: 10.1109/cvpr.2004.1315041.
- [34] R. Fan, U. Ozgunalp, Y. Wang, M. Liu, and I. Pitas, “Rethinking Road Surface 3-D Reconstruction and Pothole Detection: From Perspective Transformation to Disparity Map Segmentation,” *IEEE Transactions on Cybernetics*, pp. 1–10, 2021, doi: 10.1109/TCYB.2021.3060461.
- [35] V. Kolmogorov, “Convergent tree-reweighted message passing for energy minimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568–1583, 2006, doi: 10.1109/TPAMI.2006.200.

- [36] K. Yamaguchi, D. McAllester, and R. Urtasun, “Efficient joint segmentation, occlusion labeling, stereo and flow estimation,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 756–771, 2014, doi: 10.1007/978-3-319-10602-1_49.
- [37] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “PatchMatch: A randomized correspondence algorithm for structural image editing,” *ACM Transactions on Graphics*, vol. 28, no. 3, 2009, doi: 10.1145/1531326.1531330.
- [38] Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu, “SPM-BP: Sped-up PatchMatch Belief Propagation for Continuous MRFs,” no. 1, pp. 4006–4014, 2015, [Online]. Available: <http://www.epa.gov/iris/subst/0080.htm>
- [39] S. R. Fanello *et al.*, “UltraStereo: Efficient learning-based matching for active stereo systems,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, no. July, pp. 6535–6544, 2017, doi: 10.1109/CVPR.2017.692.
- [40] S. R. Fanello *et al.*, “Learning to be a depth camera for close-range human capture and interaction,” *ACM Transactions on Graphics*, vol. 33, no. 4, 2014, doi: 10.1145/2601097.2601223.
- [41] S. R. Fanello *et al.*, “HyperDepth: Learning depth from structured light without matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 5441–5450, 2016, doi: 10.1109/CVPR.2016.587.
- [42] S. R. Fanello *et al.*, “Low Compute and Fully Parallel Computer Vision with HashMatch,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 3894–3903, 2017, doi: 10.1109/ICCV.2017.418.

- [43] V. Tankovich *et al.*, “SOS: Stereo Matching in $O(1)$ with Slanted Support Windows,” *IEEE International Conference on Intelligent Robots and Systems*, no. 1, pp. 6782–6789, 2018, doi: 10.1109/IROS.2018.8593800.
- [44] M. Caron *et al.*, “Emerging Properties in Self-Supervised Vision Transformers,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9630–9640, 2021, doi: 10.1109/ICCV48922.2021.00951.
- [45] H. Yuan *et al.*, “PolyphonicFormer: Unified Query Learning for Depth-aware Video Panoptic Segmentation,” 2021, [Online]. Available: <http://arxiv.org/abs/2112.02582>
- [46] Z. Teed and J. Deng, “RAFT: Recurrent All-Pairs Field Transforms for Optical Flow (Extended Abstract),” *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4839–4843, 2021, doi: 10.24963/ijcai.2021/662.
- [47] W. Zhang, Y. Liu, C. Dong, and Y. Qiao, “RankSRGAN: Super Resolution Generative Adversarial Networks with Learning to Rank,” pp. 1–18, 2021, [Online]. Available: <http://arxiv.org/abs/2107.09427>
- [48] T. Karras *et al.*, “Alias-Free Generative Adversarial Networks,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. NeurIPS, pp. 438–448, 2021, [Online]. Available: <https://blog.faradars.org/generative-adversarial-networks/>
- [49] B. Wronski *et al.*, “Handheld Multi-Frame Super-Resolution,” *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 1–18, 2019.
- [50] A. J. Amiri, S. Yan Loo, and H. Zhang, “Semi-supervised monocular depth estimation with left-right consistency using deep neural network,” *IEEE International Conference on Robotics and Biomimetics, ROBIO 2019*, pp. 602–607, 2019, doi: 10.1109/ROBIO49542.2019.8961504.

- [51] F. Shamsafar, S. Woerz, R. Rahim, and A. Zell, “MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching,” *Proceedings - 2022 IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022*, pp. 677–686, 2022, doi: 10.1109/WACV51458.2022.00075.
- [52] W. N. Greene and N. Roy, “Multiviewstereonet: Fast Multi-View Stereo Depth Estimation using Incremental Viewpoint-Compensated Feature Extraction,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 9242–9248, 2021, doi: 10.1109/ICRA48506.2021.9562096.
- [53] V. G. Rares, Ambrus, S. Pillai, A. Raventos, and A. Gaidon, “3D packing for self-supervised monocular depth estimation,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2482–2491, 2020, doi: 10.1109/CVPR42600.2020.00256.
- [54] Z. Wang, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *Canadian Journal of Civil Engineering*, vol. 44, no. 4, pp. 253–263, 2017, doi: 10.1139/cjce-2016-0381.
- [55] W. Luo, A. G. Schwing, and R. Urtasun, “Efficient deep learning for stereo matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 5695–5703, 2016, doi: 10.1109/CVPR.2016.614.
- [56] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial transformer networks,” *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 2017–2025, 2015.
- [57] R. Garg, B. G. Vijay Kumar, G. Carneiro, and I. 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)*, vol. 9912 LNCS, pp. 740–756, 2016, doi: 10.1007/978-3-319-46484-8_45.

- [58] N. Mayer *et al.*, “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 4040–4048, 2016, doi: 10.1109/CVPR.2016.438.
- [59] J. R. Chang and Y. S. Chen, “Pyramid Stereo Matching Network,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 5410–5418, 2018, doi: 10.1109/CVPR.2018.00567.
- [60] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012, doi: 10.1109/CVPR.2012.6248074.
- [61] Z. Li *et al.*, “Revisiting Stereo Depth Estimation From a Sequence-to-Sequence Perspective with Transformers,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6177–6186, 2021, doi: 10.1109/ICCV48922.2021.00614.
- [62] F. Zhang, X. Qi, R. Yang, V. Prisacariu, B. Wah, and P. Torr, “Domain-Invariant Stereo Matching Networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12347 LNCS, pp. 420–439, 2020, doi: 10.1007/978-3-030-58536-5_25.
- [63] J. Watson, O. mac Aodha, D. Turmukhambetov, G. J. Brostow, and M. Firman, “Learning Stereo from Single Images,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12346 LNCS, pp. 722–740, 2020, doi: 10.1007/978-3-030-58452-8_42.

- [64] S. Rozenberg, G. Elidan, and R. El-Yaniv, “MadNet: Using a MAD Optimization for Defending Against Adversarial Attacks,” 2019, [Online]. Available: <http://arxiv.org/abs/1911.00870>
- [65] A. Kowdle *et al.*, “The need 4 speed in real-time dense visual tracking,” *SIGGRAPH Asia 2018 Technical Papers, SIGGRAPH Asia 2018*, 2018, doi: 10.1145/3272127.3275062.
- [66] Z. Li *et al.*, “Revisiting Stereo Depth Estimation From a Sequence-to-Sequence Perspective with Transformers,” 2020, [Online]. Available: <http://arxiv.org/abs/2011.02910>
- [67] P. Nawrot *et al.*, “Hierarchical Transformers Are More Efficient Language Models,” 2021, [Online]. Available: <http://arxiv.org/abs/2110.13711>
- [68] H. Fan *et al.*, “Multiscale Vision Transformers,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6804–6815, 2021, doi: 10.1109/ICCV48922.2021.00675.
- [69] A. El-Nouby *et al.*, “XCiT: Cross-Covariance Image Transformers,” 2021, [Online]. Available: <http://arxiv.org/abs/2106.09681>
- [70] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, and R. Girshick, “Early Convolutions Help Transformers See Better,” no. NeurIPS, 2021, [Online]. Available: <http://arxiv.org/abs/2106.14881>
- [71] J. Zbontar and Y. le Cun, “Computing the stereo matching cost with a convolutional neural network,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, no. 1, pp. 1592–1599, 2015, doi: 10.1109/CVPR.2015.7298767.
- [72] J. Žbontar and Y. Lecun, “Stereo matching by training a convolutional neural network to compare image patches,” *Journal of Machine Learning Research*, vol. 17, pp. 1–32, 2016.

- [73] S. Zagoruyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, no. i, pp. 4353–4361, 2015, doi: 10.1109/CVPR.2015.7299064.
- [74] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li, “Group-wise correlation stereo network,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 3268–3277, 2019, doi: 10.1109/CVPR.2019.00339.
- [75] A. Kendall *et al.*, “End-to-End Learning of Geometry and Context for Deep Stereo Regression,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 66–75, 2017, doi: 10.1109/ICCV.2017.17.
- [76] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, “MVSNet: Depth inference for unstructured multi-view stereo,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11212 LNCS, pp. 785–801, 2018, doi: 10.1007/978-3-030-01237-3_47.
- [77] F. Zhang, V. Prisacariu, R. Yang, and P. H. S. Torr, “GA-net: Guided aggregation net for end-to-end stereo matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 185–194, 2019, doi: 10.1109/CVPR.2019.00027.
- [78] R. Collins, “A Space-Sweep Approach to True Multi-Image Matching,” *3D Data Processing, Visualization and ...*, pp. 358–363, 2010, [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=517097%5Cnhttp://www2.it.lut.fi/mvpr/data/PaaKam_3dpvt2010.pdf
- [79] Y. Zhang *et al.*, “Activestereonet: End-to-end self-supervised learning for active stereo systems,” *Lecture Notes in Computer Science (including*

- subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*), vol. 11212 LNCS, pp. 802–819, 2018, doi: 10.1007/978-3-030-01237-3_48.
- [80] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi, “StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11219 LNCS, pp. 596–613, 2018, doi: 10.1007/978-3-030-01267-0_35.
- [81] S. Gidaris and N. Komodakis, “Detect, replace, refine: Deep structured prediction for pixel wise labeling,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 7187–7196, 2017, doi: 10.1109/CVPR.2017.760.
- [82] Z. Liang *et al.*, “Learning for Disparity Estimation Through Feature Constancy,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2811–2820, 2018, doi: 10.1109/CVPR.2018.00297.
- [83] J. Pang, W. Sun, J. S. J. Ren, C. Yang, and Q. Yan, “Cascade Residual Learning: A Two-Stage Convolutional Neural Network for Stereo Matching,” *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, vol. 2018-Janua, pp. 878–886, 2017, doi: 10.1109/ICCVW.2017.108.
- [84] G. Yang, J. Manela, M. Happold, and D. Ramanan, “Hierarchical deep stereo matching on high-resolution images,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 5510–5519, 2019, doi: 10.1109/CVPR.2019.00566.
- [85] A. Seki, M. Pollefeys, T. Corporation, E. T. H. Zürich, and Microsoft, “SGM-Nets: Semi-global matching with neural networks,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*

2017, vol. 2017-Janua, no. 1, pp. 6640–6649, 2017, doi:
10.1109/CVPR.2017.703.

- [86] J. L. Schönberger, S. N. Sinha, and M. Pollefeys, “Learning to fuse proposals from multiple scanline optimizations in semi-global matching,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11217 LNCS, pp. 758–775, 2018, doi: 10.1007/978-3-030-01261-8_45.
- [87] Q. Yang, “A non-local cost aggregation method for stereo matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1402–1409, 2012, doi: 10.1109/CVPR.2012.6247827.
- [88] Y. Wang, Q. Zhou, J. Xiong, X. Wu, and X. Jin, “ESNet: An efficient symmetric network for real-time semantic segmentation,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11858 LNCS, pp. 41–52, 2019, doi: 10.1007/978-3-030-31723-2_4.
- [89] D. Sun, X. Yang, M. Y. Liu, and J. Kautz, “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. D, pp. 8934–8943, 2018, doi: 10.1109/CVPR.2018.00931.
- [90] A. Dosovitskiy *et al.*, “FlowNet: Learning optical flow with convolutional networks,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 2758–2766, 2015, doi: 10.1109/ICCV.2015.316.
- [91] A. Shaked and L. Wolf, “Improved stereo matching with constant highway networks and reflective confidence learning,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, no. vi, pp. 6901–6910, 2017, doi: 10.1109/CVPR.2017.730.

- [92] A. Ranjan and M. J. Black, “Optical flow estimation using a spatial pyramid network,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2720–2729, 2017, doi: 10.1109/CVPR.2017.291.
- [93] X. Song, G. Yang, X. Zhu, H. Zhou, Z. Wang, and J. Shi, “AdaStereo: A Simple and Efficient Approach for Adaptive Stereo Matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 10323–10332, 2021, doi: 10.1109/CVPR46437.2021.01019.
- [94] A. Badki, A. Troccoli, K. Kim, J. Kautz, P. Sen, and O. Gallo, “BI3D: Stereo depth estimation via binary classifications,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1597–1605, 2020, doi: 10.1109/CVPR42600.2020.00167.
- [95] Q. Wang, S. Shi, S. Zheng, K. Zhao, and X. Chu, “FADNet: A Fast and Accurate Network for Disparity Estimation,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 101–107, 2020, doi: 10.1109/ICRA40945.2020.9197031.
- [96] Y. Zhong, Y. Dai, and H. Li, “Self-Supervised Learning for Stereo Matching with Self-Improving Ability,” 2017, [Online]. Available: <http://arxiv.org/abs/1709.00930>
- [97] M. Ye, E. Johns, A. Handa, L. Zhang, P. Pratt, and G. Z. Yang, “Self-Supervised Siamese Learning on Stereo Image Pairs for Depth Estimation in Robotic Surgery,” no. 1, pp. 27–28, 2017, doi: 10.31256/hsmr2017.14.
- [98] J. B. Grill *et al.*, “Bootstrap your own latent a new approach to self-supervised learning,” *Advances in Neural Information Processing Systems*, vol. 2020-Decem, 2020.
- [99] P. Liu, I. King, M. R. Lyu, and J. Xu, “Flow2Stereo: Effective self-supervised learning of optical flow and stereo matching,” *Proceedings of the*

IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 6647–6656, 2020, doi: 10.1109/CVPR42600.2020.00668.

- [100] S. Tulyakov, A. Ivanov, and F. Fleuret, “Weakly_Supervised_Learning_ICCV_2017_paper.pdf,” *Iccv*, pp. 1339–1348, 2017.
- [101] J. Xie, R. Girshick, and A. Farhadi, “Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9908 LNCS, pp. 842–857, 2016, doi: 10.1007/978-3-319-46493-0_51.
- [102] X. Guo, H. Li, S. Yi, J. Ren, and X. Wang, “Learning Monocular Depth by Distilling Cross-Domain Stereo Networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11215 LNCS, pp. 506–523, 2018, doi: 10.1007/978-3-030-01252-6_30.
- [103] M. Poggi, F. Tosi, and S. Mattoccia, “Learning monocular depth estimation with unsupervised trinocular assumptions,” *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pp. 324–333, 2018, doi: 10.1109/3DV.2018.00045.
- [104] Y. Kuznetsov, J. Stückler, and B. Leibe, “Semi-supervised deep learning for monocular depth map prediction,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2215–2223, 2017, doi: 10.1109/CVPR.2017.238.
- [105] Y. Luo *et al.*, “Single View Stereo Matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 155–163, 2018, doi: 10.1109/CVPR.2018.00024.
- [106] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia, “Generative adversarial networks for unsupervised monocular depth prediction,” *Lecture Notes in*

Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11129 LNCS, pp. 337–354, 2019, doi: 10.1007/978-3-030-11009-3_20.

- [107] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, “Unsupervised adversarial depth estimation using cycled generative networks,” *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pp. 587–595, 2018, doi: 10.1109/3DV.2018.00073.
- [108] V. Madhu Babu, K. Das, A. Majumdar, and S. Kumar, “UnDEMoN: Unsupervised Deep Network for Depth and Ego-Motion Estimation,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 1082–1088, 2018, doi: 10.1109/IROS.2018.8593864.
- [109] R. Li, S. Wang, Z. Long, and D. Gu, “UnDeepVO: Monocular Visual Odometry Through Unsupervised Deep Learning,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 7286–7291, 2018, doi: 10.1109/ICRA.2018.8461251.
- [110] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid, “Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 340–349, 2018, doi: 10.1109/CVPR.2018.00043.
- [111] C. Godard, O. mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6602–6611, Apr. 2017, doi: 10.1109/CVPR.2017.699.
- [112] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6612–6621, 2017, doi: 10.1109/CVPR.2017.700.

- [113] M. Klodt and A. Vedaldi, “Supervising the new with the old: Learning SFM from SFM,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11214 LNCS, pp. 713–728, 2018, doi: 10.1007/978-3-030-01249-6_43.
- [114] I. Mehta, P. Sakurikar, and P. J. Narayanan, “Structured adversarial training for unsupervised monocular depth estimation,” *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pp. 314–323, 2018, doi: 10.1109/3DV.2018.00044.
- [115] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” pp. 1–9, 2014, [Online]. Available: <http://arxiv.org/abs/1412.3555>
- [116] X. Liu *et al.*, “Extremely dense point correspondences using a learned feature descriptor,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4846–4855, 2020, doi: 10.1109/CVPR42600.2020.00490.
- [117] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport,” *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.
- [118] J. Watson, O. mac Aodha, D. Turmukhambetov, G. J. Brostow, and M. Firman, “Learning Stereo from Single Images,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12346 LNCS, pp. 722–740, 2020, doi: 10.1007/978-3-030-58452-8_42.
- [119] D. Mazza and M. Pagani, “Automatic differentiation in PCF,” *Proceedings of the ACM on Programming Languages*, vol. 5, no. POPL, pp. 1–4, 2021, doi: 10.1145/3434309.
- [120] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *7th International Conference on Learning Representations, ICLR 2019*, 2019.

- [121] L. N. Smith and N. Topin, “Super-convergence: very fast training of neural networks using large learning rates,” p. 36, 2019, doi: 10.1117/12.2520589.
- [122] Z. Yin, T. Darrell, and F. Yu, “Hierarchical discrete distribution decomposition for match density estimation,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 6037–6046, 2019, doi: 10.1109/CVPR.2019.00620.
- [123] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li, “Group-wise correlation stereo network,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 3268–3277, 2019, doi: 10.1109/CVPR.2019.00339.
- [124] F. Zhang, V. Prisacariu, R. Yang, and P. H. S. Torr, “GA-net: Guided aggregation net for end-to-end stereo matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 185–194, 2019, doi: 10.1109/CVPR.2019.00027.
- [125] F. Zhang, X. Qi, R. Yang, V. Prisacariu, B. Wah, and P. Torr, “Domain-Invariant Stereo Matching Networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12347 LNCS, pp. 420–439, 2020, doi: 10.1007/978-3-030-58536-5_25.
- [126] Y. Zhang *et al.*, “Adaptive unimodal cost volume filtering for deep stereo matching,” *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, vol. i, pp. 12926–12934, 2020, doi: 10.1609/aaai.v34i07.6991.
- [127] X. Du, M. El-Khamy, and J. Lee, “AMNet: Deep Atrous Multiscale Stereo Disparity Estimation Networks,” pp. 1–25, 2019, [Online]. Available: <http://arxiv.org/abs/1904.09099>
- [128] H. Ren, A. Raj, M. El-Khamy, and J. Lee, “SUW-learn: Joint supervised, unsupervised, weakly supervised deep learning for monocular depth estimation,” *IEEE Computer Society Conference on Computer Vision and*

- Pattern Recognition Workshops*, vol. 2020-June, no. Md, pp. 3235–3243, 2020, doi: 10.1109/CVPRW50498.2020.00383.
- [129] X. Cheng, P. Wang, and R. Yang, “Learning Depth with Convolutional Spatial Propagation Network,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2361–2379, 2020, doi: 10.1109/TPAMI.2019.2947374.
- [130] X. Cheng *et al.*, “Hierarchical Neural Architecture Search for Deep Stereo Matching,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, no. NeurIPS, pp. 6037–6046, 2019, doi: 10.1109/CVPR.2019.00620.
- [131] P. Muller and A. Savakis, “Flowdometry: An optical flow and deep learning based approach to visual odometry,” *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, pp. 624–631, 2017, doi: 10.1109/WACV.2017.75.