Deep learning based stereo matching on a small dataset

Rongcheng Wu

A thesis in fulfilment of the requirements for the degree of

Master of Philosophy

School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

October 2021

Thesis Title and Abstract

Thesis Title and Abstract | Declarations | Inclusion of Publications Statement | Corrected Thesis and Responses

Thesis Title

Deep learning based stereo matching on a small dataset.

Thesis Abstract

Deep learning (DL) has been used in many computer vision tasks including stereo matching. However, DL is data hungry, and a large number of highly accurate real-world training images for stereo matching is too expensive to acquire in practice. The majority of studies rely on large simulated datasets during training, which inevitably results in domain shift problems that are commonly compensated by fine-tuning. This work proposes a recursive 3D convolutional neural network (CNN) to improve the accuracy of DL based stereo matching that is suitable for real-world scenarios with a small set of available images, without having to use a large simulated dataset and without fine-tuning. In addition, we propose a novel scale-invariant feature transform (SIFT) based adaptive window for matching cost computation that is a crucial step in the stereo matching pipeline to enhance accuracy. Extensive end-to-end comparative experiments demonstrate the superiority of the proposed recursive 3D CNN and SIFT based adaptive windows. Our work achieves effective generalization corroborated by training solely on the indoor Middlebury Stereo 2014 dataset and validating on outdoor KITTI 2012 and KITTI 2015 datasets. As a comparison, our bad-4.0-error is 24.2 that is on par with the AANet (CVPR2020) method according to the publicly evaluated report from the Middlebury Stereo Evaluation Benchmark.

Originality, Copyright and Authenticity Statements

ORIGINALITY STATEMENT

☑ I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

COPYRIGHT STATEMENT

☑ I hereby grant the University of New South Wales or its agents a non-exclusive licence to archive and to make available (including to members of the public) my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known. I acknowledge that I retain all intellectual property rights which subsist in my thesis or dissertation, such as copyright and patent rights, subject to applicable law. I also retain the right to use all or part of my thesis or dissertation in future works (such as articles or books).

For any substantial portions of copyright material used in this thesis, written permission for use has been obtained, or the copyright material is removed from the final public version of the thesis.

AUTHENTICITY STATEMENT

☑ I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.

Inclusion of Publications Statement

Thesis Title and Abstract | Declarations | **Inclusion of Publications Statement** | Corrected Thesis and Responses

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in the candidate's thesis in lieu of a Chapter provided:

- The candidate contributed **greater than 50%** of the content in the publication and are the "primary author", i.e. they were responsible primarily for the planning, execution and preparation of the work for publication.
- The candidate has obtained approval to include the publication in their thesis in lieu of a Chapter from their Supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis.

☑ The candidate has declared that **their thesis has publications - either published or submitted for publication - incorporated into it in lieu of a Chapter/s. Details of these publications are provided below.**.

Publication Details #1

Full Title:	Deep learning based stereo cost aggregation on a small dataset
Authors:	Rongcheng Wu, Changming Sun, Zhaoying Liu, Arcot Sowmya
Journal or Book Name:	The International Conference on Digital Image Computing: Techniques and Applications (DICTA 2021)
Volume/Page Numbers:	
Date Accepted/Published:	13/09/2021
Status:	accepted
The Candidate's Contribution to the Work:	All, including the ideas, coding, experiment design, and paper writing.
Location of the work in the thesis and/or how the work is incorporated in the thesis:	All chapters excepting Chapter 2 (Literature Review)

Candidate's Declaration

I confirm that where I have used a publication in lieu of a chapter, the listed publication(s) above meet(s) the requirements to be included in the thesis. I also declare that I have complied with the Thesis Examination Procedure.

Abstract

Deep learning (DL) has been used in many computer vision tasks including stereo matching. However, DL is data hungry, and a large number of highly accurate real-world training images for stereo matching is too expensive to acquire in practice. The majority of studies rely on large simulated datasets during training, which inevitably results in domain shift problems that are commonly compensated by fine-tuning. This work proposes a recursive 3D convolutional neural network (CNN) to improve the accuracy of DL based stereo matching that is suitable for real-world scenarios with a small set of available images, without having to use a large simulated dataset and without fine-tuning. In addition, we propose a novel scale-invariant feature transform (SIFT) based adaptive window for matching cost computation that is a crucial step in the stereo matching pipeline to enhance accuracy. Extensive end-to-end comparative experiments demonstrate the superiority of the proposed recursive 3D CNN and SIFT based adaptive windows. Our work achieves effective generalization corroborated by training solely on the indoor Middlebury Stereo 2014 dataset and validating on outdoor KITTI 2012 and KITTI 2015 datasets. As a comparison, our bad-4.0-error is 24.2 that is on par with the AANet (CVPR2020) method according to the publicly evaluated report from the Middlebury Stereo Evaluation Benchmark.

Acknowledgement

Throughout the research and writing of this dissertation, I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Changming Sun, whose expertise was invaluable in formulating the research questions and methodology. Also, Professor Changming Sun and Professor Arcot Sowmya help me a lot with my paper and thesis writing.

I would also like to acknowledge Commonwealth Scientific and Industrial Research Organisation (CSIRO) and UNSW, which provide me great working places with energetic researchers.

In addition, I would like to thank my parents for their wise counsel and funding. Finally, I could not have completed this dissertation without the support of my friend, Mingzhe Wang, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Publications Arising from Thesis

List of Publications

- Rongcheng Wu, Changming Sun, Zhaoying Liu, Arcot Sowmya. Deep learning based stereo cost aggregation on a small dataset, *Accepted, DICTA 2021.*

Contents

List of Figures

List of Tables

Abbreviations

BFC	Best Fit with Coalescing
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DL	Deep Learning
FCN	Fully Convolutional Networks
LGA	Local Guided Aggregation
MRF	Markov Random Field
PBCP	Patch Based Confidence Prediction
SAD	Sum of Absolute Difference
SGA	Semi Global Aggregation
SGM	Semi Global Matching
SIFT	Scale Invariant Feature Transform
SSD	Sum of Squared Difference

Chapter 1

Introduction

With the popularity of deep learning within the computer vision area, many challenging problems from the past have become solvable. However, after a few years of deep learning applied to computer vision problems, many tasks have not been properly solved. To overcome bottlenecks in 3D vision problems, 3D information providing richer information could be useful. Recovering depth information from one or a few images is an essential prerequisite of 3D vision, and this is the main focus of this thesis.

In this chapter, the background and motivation are first discussed. After that, the camera model and its issues are outlined. Finally, the thesis contributions are listed and thesis organization is described.

1.1 Background and Motivation

There are two main methods to measure depth from images: the active and the passive approach. Most active methods rely on sending energy out, such as LiDAR [4, 5] or structured light [1]. Although these sensors can measure depth accurately, some drawbacks are apparent. Firstly, the cost of those sensors is high. Secondly, the lifespan of those

sensors can be short. In the passive approach, stereo matching and monocular depth estimation are the two most popular methods. However, most monocular depth estimation models based on deep learning rely heavily on the model fitting ability, which leads to model performance dropping significantly in the presence of domain shifts. For example, a monocular depth estimation model trained on an indoor dataset often fails on an outdoor dataset. Inspired by human binocular vision, the use of two cameras to simulate the eyes is another approach. Compared with a monocular system, the binocular system provides a triangle relationship for each 3D point. Therefore, stereo matching is often used to obtain depth, which only needs two cameras.

1.2 Environment Setup

Before introducing the details of the stereo matching algorithm, the setup of stereo cameras is introduced. For the traditional stereo camera setup, two cameras with the same parameters are placed on the same line and in the same plane. The baseline of the two cameras is the distance between the two cameras. Even if the cameras placed on the same line and plane can reduce mismatch between the left and right images, the viewing angle between the two cameras also highly affects the matching process. In order to reduce the searching space, the orientation of the two cameras are often set to be the same. Given pixels on the left image in the ideal state, the matching point is only found on the same row in the right image.

In Figure 1.1 the camera setup is shown. Given the left camera and right cameras with the same parameters, the gap between the two cameras is the baseline. This work will focus on the situation when the stereo image pairs are calibrated and rectified. Given a point in the real world, the gap between the left image pixel and the right image pixel for this point is the disparity. The distance between the 3D point and the two cameras is the depth. In general, the disparity will change with depth. In equations (1.1) and (1.2), a

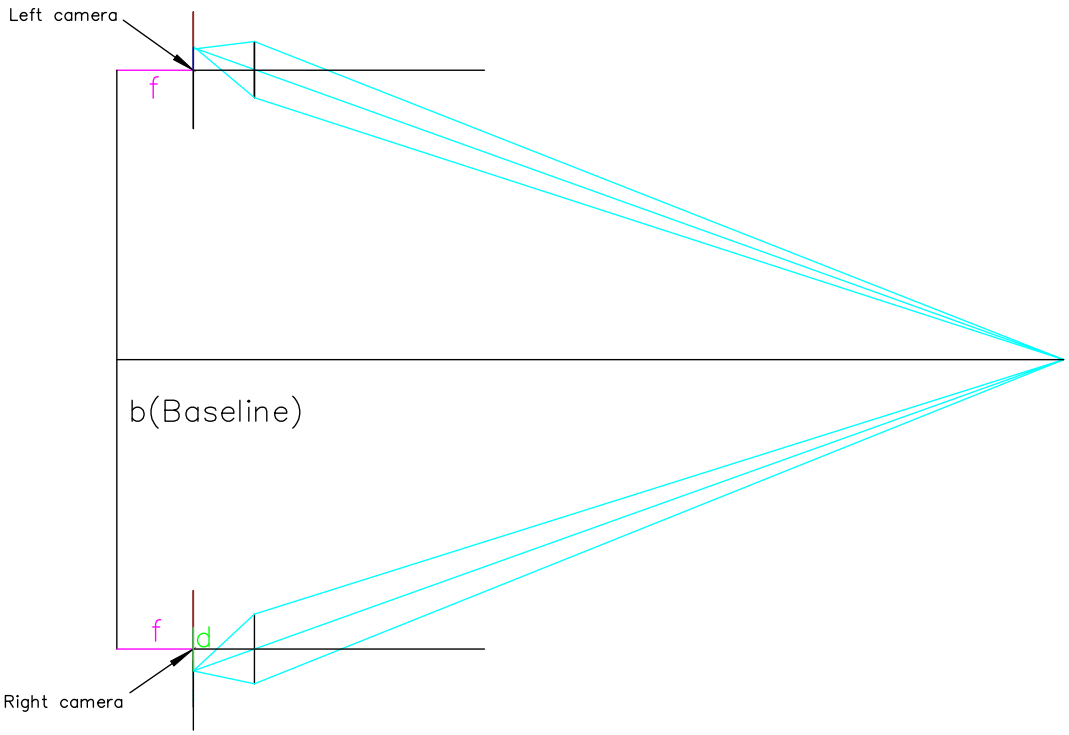


Figure 1.1: Camera model.

simple trigonometric formula that describes the relationship between depth and disparity is provided:

$$\frac{d}{b} = \frac{f}{z} \tag{1.1}$$

$$z = \frac{bf}{d} \tag{1.2}$$

where d is the disparity, f is the focal length, b is the baseline and z is depth. Clearly, the depth can be obtained from the disparity with the specific camera parameters. Therefore, recovering the depth from two images can be viewed as the problem of finding the correct disparity value. In order to find the disparity value, finding the correct matches between these two images is a necessary step.

Each pixel on the left image usually has a corresponding pixel on the right image. After the corresponding pixel is obtained, the disparity value is the position gap between the

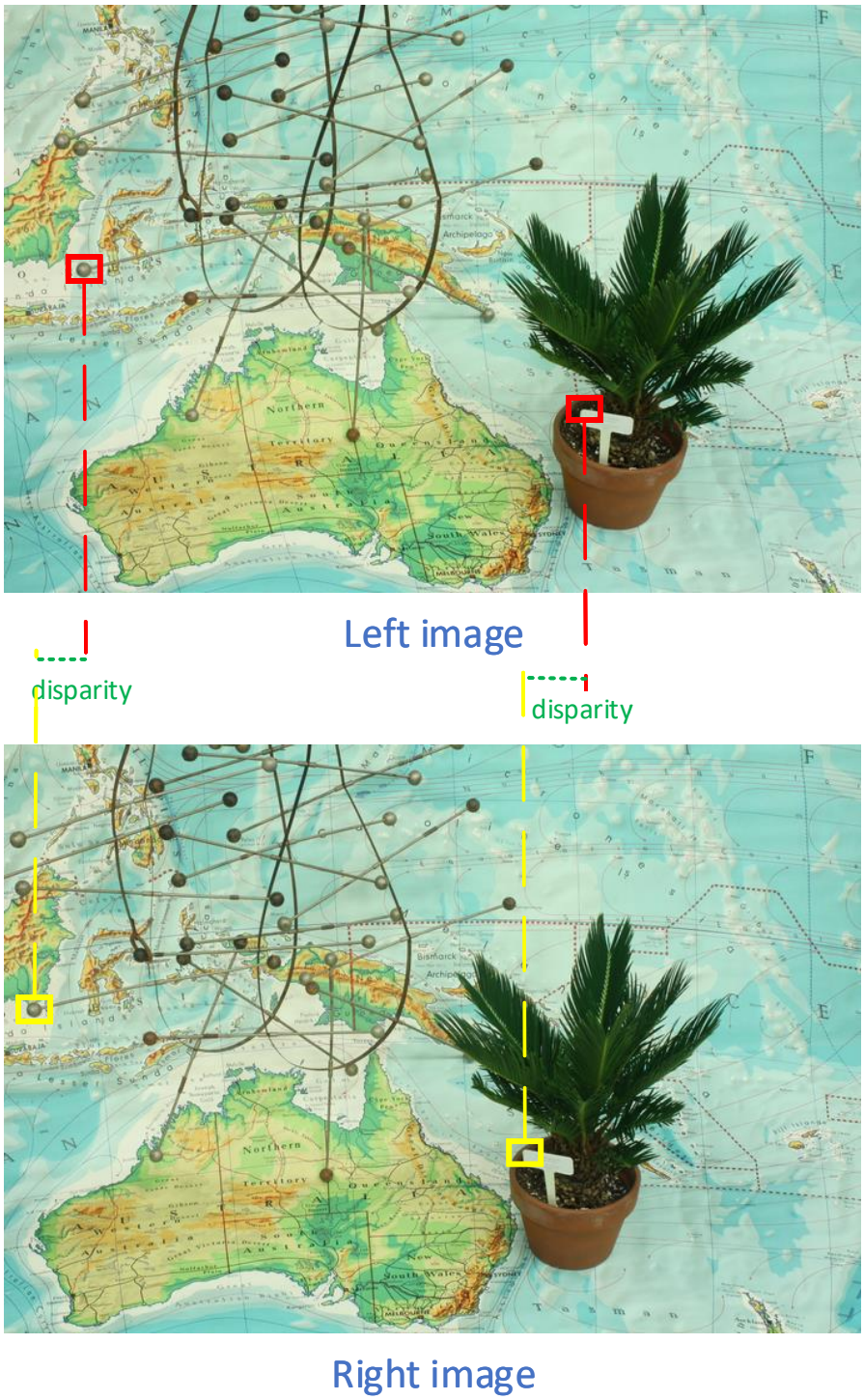


Figure 1.2: An example of disparity. The red box and red line are the pixels from the left image, and the yellow box and yellow line are the corresponding points on the right image. The disparity is shown as a green dotted line.

left and right corresponding pixels.

An example of disparity and the corresponding points is shown in Figure 1.2.

1.3 Challenges

In this part, the challenging issues of traditional stereo and deep learning based stereo matching models will be introduced.

1.3.1 Traditional Algorithm

Generally, the pipeline of stereo matching consists of four steps: matching cost computation, cost aggregation, disparity computation and refinement [6].

In traditional stereo matching, the matching cost computation step determines the quality of the final result. Even though the cost aggregation and refinement can reduce errors on the raw disparity map, these two steps still rely on the correct matching result from the cost computation step.

The main issue in the cost computation part is the mismatching. Seven main reasons cause mismatching, including textureless areas, specular reflection areas, foreshortening, occlusions, light changes, repeated patterns and window size.

1. **Textureless areas:** The reason the textureless areas leading to mismatching is that multiple positions may have the same matching cost values, because all aggregation algorithms are based on one assumption: areas of the same texture only have a very low probability of appearing. Hence, when textureless areas contain a larger number of pixels with the same matching cost values at different positions, the

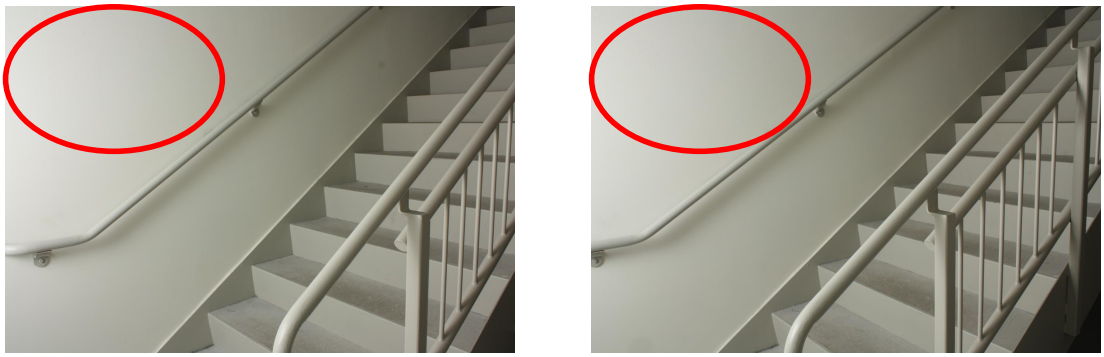


Figure 1.3: An example of textureless areas [1].

aggregation step will fail on the textureless areas. In Figure 1.3 an example of matching textureless areas is shown.

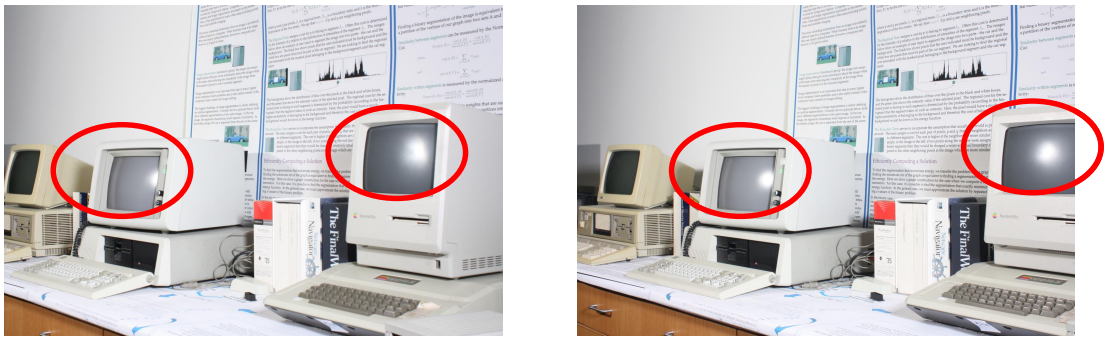


Figure 1.4: An example of specular reflection areas [1].

2. **Specular reflection areas:** Reflective surfaces violate the Lambertian assumption, which means that the brightness is the same regardless of the observer's angle of view. In other words, specular reflection areas make the observed colour of a surface point dependent on the viewpoint. Hence, different viewpoints between the left and right cameras lead to different textures on specular reflection areas. In Figure 1.4 an example of specular reflection areas is shown.

3. **Repeated patterns:** Repeated patterns leading to mismatching are similar to textureless areas: multiple similar matching cost values appear at different positions. However, repeated areas can be handled by texture segmentation. Therefore, many

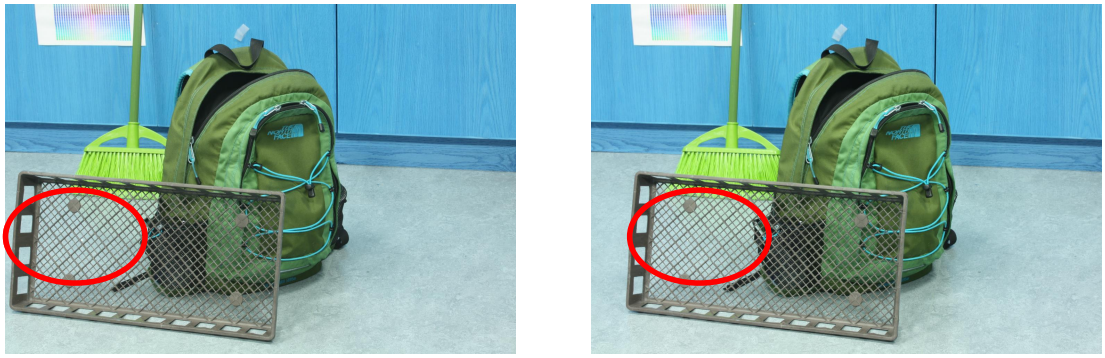


Figure 1.5: An example of repeated patterns [1].

works [7, 8] use segmentation to solve this problem. In Figure 1.5 an example of repeated patterns is shown.

Figure 1.6: An example of foreshortening [1].

item **Foreshortening:** Similar to specular reflection areas, mismatching led by foreshortening is due to change of viewpoint. Different from specular reflection areas which lead to different lighting conditions, foreshortening causes shape changes. In Figure 1.6 an example of foreshortening is shown.

4. **Occlusions:** The occlusion problem is due to objects in front blocking objects behind. Therefore, for blocked areas, the correct matching points between the left and right images cannot be found. However, occluded areas can be handled in the post-processing steps, which are named "left-right consistency check (LRC)". The LRC compares points on the left disparity map with the corresponding points on

Figure 1.7: An example of occlusions [1].

the right disparity map. Pixels are considered to be occluded points when the point difference between the left and right disparity map is larger than a threshold. In Figure 1.7 an example of occlusions is shown.

Figure 1.8: An example of different lighting conditions [1].

5. **Lighting conditions:** One of the factors leading to mismatching is different lighting conditions. Stereo matching methods rely on natural light in the environment to collect images. Due to changes in lighting angles and light intensity, the brightness difference between two images may be relatively large. In Figure 1.8 an example of different lighting conditions is shown.

6. **Window size:** For the cost computation step, the most common method is the window based method. For each pixel p in an image, a patch centred at p is used

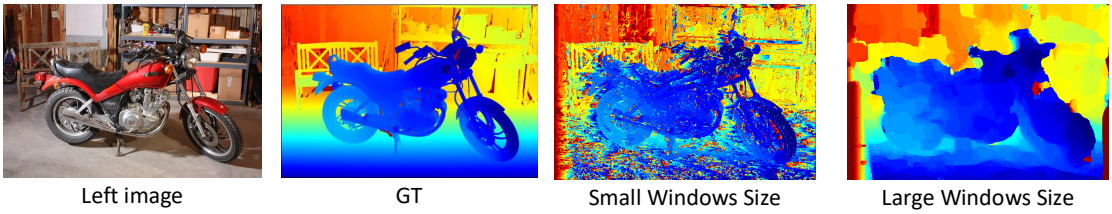


Figure 1.9: Examples of different results with different windows sizes.

to search for the correct match in the other image. Most window-based cost computation methods employ a fixed window size to search for the correct match for each pixel. However, it is non-trivial to identify an optimal window size. A small window can have less smoothing effects and better retaining of object details but incur mismatches on textureless areas.

Different window sizes in the matching cost computation step result in different raw disparity maps. In general, small windows are usually not large enough to gather sufficient information to distinguish between wrong matches and correct matches, while a large window can reduce noise in textureless areas but lose details on matching. Therefore, it is challenging to choose an optimal window size. Therefore, fixed windows are not suitable for various scenes. In Figure 1.9 examples of different results with different window sizes are shown.

1.3.2 Deep Learning

Deep learning (DL) has been used in many computer vision tasks and dominates many competition leaderboards. However, applying those deep learning models in a production environment is still a challenging mission. The main reasons preventing deep learning models from being widely applicable in the real world include: large training dataset requirements, generalisation issues, robustness issues and time and memory consumption. These are discussed now.

1. **Large training dataset requirement:** A recent consensus is that a deep learning model requires a more extensive training set to overcome the overfitting problem. In addition, training set size is also related to the model generation process. However, the cost of labeling the data is expensive in some applications. Stereo matching is one such application which requires pixel-level labeling. Even though depth sensors have become cheaper in recent years, manually correcting the errors on the depth map from sensors is unavoidable. Therefore, the lack of high accuracy labeled data is a bottleneck to applying deep learning based stereo matching in the real world. Unlike most deep learning models that rely on large training datasets, few-shot learning, which aims to gain enough generalization on a limited number of the training sets, could solve mitigate this limitation.

2. **Generalisation:** An ideal model should perform well even if the data domain or scenes are different from the training data. Unfortunately, most deep learning models still suffer from the domain adaptation problem. This means that the model performance decreases when the test data domain is different from the training data. Nowadays, the most popular method to solve the domain adaptation problem is fine-tuning. Although fine-tuning or retraining on a new domain data can improve model performance, fine-tuning cannot actually solve the generalisation problem in the production environment where data could come from various domains, and the cost to label the data is huge.

   Although many deep learning models have very high performance on the stereo matching task, one of the main issues that prevent their application in the real world is due to domain shift. In order to obtain enough training data to train a deep neural network, most works use simulated datasets to train those models. After those models are trained on a simulated dataset, fine-tuning or retraining is performed on a specified dataset. Although fine-tuning can considerably improve performance, it also requires data with ground-truth in the new environment. In some cases, it is difficult and even impossible to obtain enough data with ground truth.

In particular, stereo matching is an essential step in 3D reconstruction. Developing a few-shot DL-based stereo matching model will pave the way for 3D reconstruction with minimal supervision. Therefore, compared with improved accuracy on some datasets, using fewer training data to gain a more robust performance in the new domain is more important.

3. **Robustness:** In recent years, many works have shown that some of the popular deep learning models are fragile. One of the examples is the adversarial model [9–12], where even a small amount of noise can lead to the model collapsing. Hence, low robustness is a major factor preventing deep learning models from being widely used in some areas.

   In particular, the cost aggregation step should adapt various cost computation methods without retraining or fine-tuning. In other words, an ideal model should be robust enough to be applicable everywhere without additional training. Unfortunately, most deep learning based cost aggregation models cannot achieve stable performance on various cost computation methods. Those cost aggregation methods cannot be applied to cost computation methods different from the method used for training. For example, in Section 4.3.11, the accuracy of the aggregation model trained on the input data from Census decreases on the input data from sum of absolute differences (SAD) or sum of squared differences (SSD) cost computation methods.

4. **Time and Memory Consumption:** With improved model accuracy, deeper and larger models are used to improve their accuracy tested on some popular datasets. Most deep learning textbooks indicate that a large model will lead to overfitting and performance reduction. However, recent research as well as this research show that a larger and deeper model always has better generalisation. One example from recent research is Double Descent [13], which shows that with the model size increasing, performance initially becomes worse and then becomes better.

   Many deep learning models are successfully used on platforms with high computing

power in recent years, such as auto-driving cars with powerful GPUs. However, executing a high-performance deep learning model on mobile devices and edge devices is still a challenging mission. One of the critical factors is that the state-of-the-art (SOTA) models always have huge memory and time requirements. Hence, the Double Descent phenomenon explains why the increase of accuracy is always much slower than the increase in model size. Therefore, methods to reduce the time and memory consumption of SOTA models without sacrificing accuracy is a hot research topic.

In this work, a deeper and larger model is trained for stereo matching. The 3D convolution operation in the 3D convolution neural network is the most expensive in terms of time and memory consumption in the cost aggregation step. Because 3D convolution is widely used on the cost aggregation step, those models cannot be deployed on edge devices. Therefore, reducing memory and time consumption is a valuable research topic in the stereo matching area.

1.4 Contributions

Stereo matching takes two images and outputs their disparity map. Deep learning has been utilised [14–16] for stereo matching due to its super performance. Generally, the pipeline of stereo matching consists of four steps: matching cost computation, cost aggregation, disparity computation, and refinement [6]. Among them, cost computation and aggregation are the two most critical steps and are the main focus of this study. Based on the issues and challenges discussed in Section 1.3, our contributions are listed in this section.

1.4.1 Window Size

To address the window size issue, instead of tediously and manually selecting a suitable window size, a matching cost computation method using an adaptive window size is proposed by taking advantage of the scale-invariant feature transform (SIFT) [17] matching points as cues. The experimental results show that the propsoed adaptation of the cost computation method improves accuracy on the raw disparity map and reduces time consumption on the cost computation steps.

Compared with large fixed windows, the proposed adaptive windows method retains more details. Compared with small fixed windows, the proposed adaptive windows method reduces noise in textureless areas. The adaptive windows method is combined with Census and sum of squared difference (SSD), leading to almost 3% and 2% performance improvement respectively, compared with the regular Census and SSD.

1.4.2 Large Training Dataset Requirement

In order to overcome the drawbacks of relying on a large dataset during training, a new structure for 3D CNN is proposed, which only needs a few images for training. Comprehensive experimental results confirm the efficiency of the two proposed methods. Specifically, in end-to-end comparative experiments with no more than 11 images used for training, the recurrent 3D CNN provides a conservative 11% accuracy improvement. The overall performance was evaluated on the Middlebury Stereo Evaluation Benchmark[1], and the corresponding report shows that the proposed method achieved a bad-4.0-error of 24.2 which is on par with the AANet (CVPR2020) method.

In addition, a comparative experiment showed that the accuracy gap between the proposed model and other models increases with the decrease in size of the training set. The

[1]the proposed method is named R3DCNN in the submission to the public benchmark at `https://vision.middlebury.edu/stereo/eval3/`.

proposed R3DCNN achieves around 70% bad-4.0-error accuracy in the one-shot setting, which is 24% higher than the regular 3D U-Net [18].

1.4.3 Generalisation

To address the second limitation due to generalisation, a recurrent 3D CNN structure is constructively used to enable training with only a few training images, providing better generalisation ability and obviating the cumbersomeness of both large (simulated/real) datasets and fine-tuning.

1.4.4 Robustness

Th proposed recurrent 3D CNN not only improves generalisation but also improves the robustness. In one experiment, the propsoed cost aggregation model was trained on one particular cost computation method but directly tested on other data generated from other cost computation methods. The propsoed model maintains stable performance on the new data, while the regular 3D CNN failed in this experiment.

1.4.5 Speed and Memory Consumption

Different types of tasks always rely on different types of networks, which consist of different types of operations. The types of input data of different tasks are also significantly different.

Decomposing slow and high memory usage operations within deep networks into a set of lightweight operations could efficiently reduce memory and time consumption.

Input data is another aspect that is easily overlooked. often, input data is redundant and compressible. One example is the cost volume from stereo matching. Only some values on

the volume provide useful information. Unfortunately, most SOTA models are sensitive to changes in the distribution of input data. Therefore, one option to reduce memory consumption is to find a method to reduce input data without significant accuracy decrease during testing. Another way is to design a model that is not sensitive to changes in the distribution led by data reduction. This thesis proposes a novel data reduction method that combines these two methods and is introduced in Section 4.1.2.3.

1.4.6 Confidence Measurement

Measuring the error of the disparity map is helpful in the post-processing step. Therefore, a confidence measurement method is proposed, based on the aggregation model. The propsoed confidence measurement method in Section 4.2 does not require additional training and achieves around 75% accuracy.

1.5 Thesis Organisation

In Chapter 1 the background and the main contributions were presented. In Chapter 2 previous work on stereo matching is reviewed. In Chapter 3 the focus is on the first contribution: cost computation with adaptive windows. In Chapter 4 the main contribution on few-shot cost aggregation is described. The final contribution is confidence measurement, which is introduced in Section 4.2. Finally, the conclusion and future work are given in Chapter 5.

Chapter 2

Literature Review

This chapter presents relevant background on stereo matching. The four-step pipeline of stereo matching is introduced, including the traditional algorithms and those based on deep learning. In addition, a summary of the framework and current popular approaches to reduce memory and time consumption of deep learning models are provided.

2.1 Cost Computation

As the first step of stereo matching pipeline, cost computation affects the final result significantly. Cost computation aims to find the similarity score between pixels in the left and right images. Usually, the score used to measure similarity is the matching cost.

The cost computation for stereo matching is expressed as:

$$C(x, y) = \text{cost}(L(x, y), R(x - d, y)) \tag{2.1}$$

where L is the left image, R is the right image, C is the cost of matching the two pixels in the left and right image patches centred at (x, y) and $(x - d, y)$ respectively, d represents the disparity value, and "cost" is the cost function.

There are two types of algorithms to compute the cost value for an image pair. The first type are traditional algorithms, and most of those use low-level features of image patches to measure the dissimilarity, such as the sum of absolute difference (SAD) [19], sum of squared difference (SSD) [20], Census [21] or their combinations (AD-Census) [22]. The second type of cost computation methods are deep learning-based methods. Most of those methods replace hand-crafted features with learned features [23–25].

2.1.1 Traditional Cost Computation

The most straightforward cost computation method is to directly slide the left image over the right image and compare pixel values. However, this simple method leads to mismatches caused by noise or illumination changes. Therefore, some window-based methods use a pixel's surrounding pixels to obtain the cost value.

Some popular cost functions are listed below:

SAD:

$$\sum_{(i,j)\in W} |L(x+i, y+j) - R(x+i+d, y+j)| \tag{2.2}$$

The idea of SAD is to compute the pixel value difference between the two windows in left and right images.

SSD:

$$\sum_{(i,j)\in W} (L(x+i, y+j) - R(x+i+d, y+j))^2 \tag{2.3}$$

compared to SAD, SSD replaces the L1 loss with L2 loss.

Zero-mean SAD:

$$\sum_{(i,j)\in W} \left| L(x+i, y+j) - \bar{L}(x+i, y+j) - R(x+i+d, y+j) + \bar{R}(x+i+d, y+j) \right| \tag{2.4}$$

Locally scaled SAD:

$$\sum_{(i,j)\in W} \left| L(i,j) - \frac{\bar{L}(x+i,y+j)}{\bar{R}(x+i+d,y+j)} R(x+i+d,y+j) \right| \tag{2.5}$$

Normalized Cross Correlation (NCC):

$$\frac{\sum_{(i,j)\in W} L(x+i,y+j) \cdot R(x+i+d,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} L^2(x+i,y+j) \cdot \sum_{(i,j)\in W} R^2(x+i+d,y+j)}} \tag{2.6}$$

where L is the first image and R the second image in the image set. i is the row index of the image, and j is the column index of the image. W is the matching window. d is the disparity.

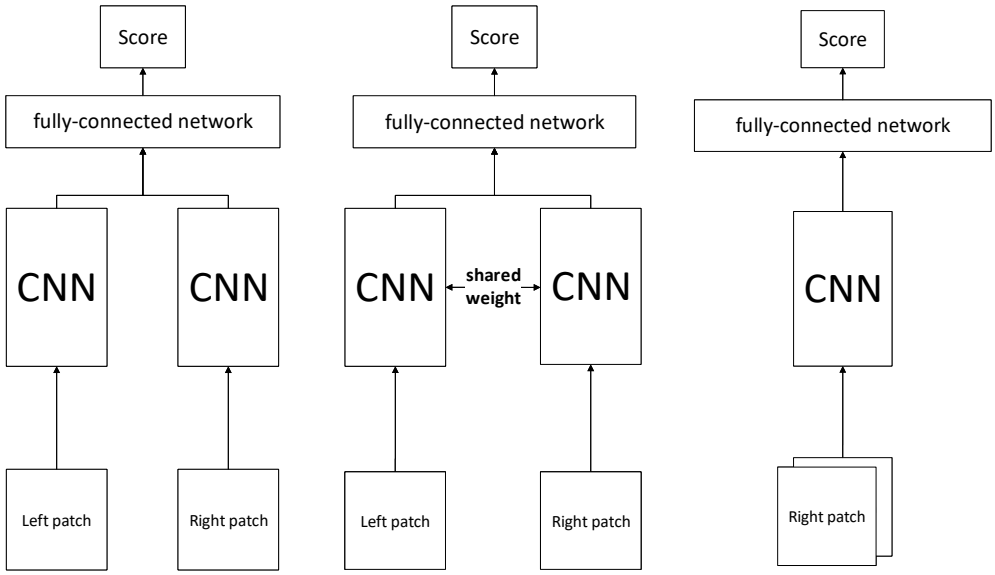
Census:

The Census [21] method uses the pixels surrounding a centre pixel of the matching windows to generate a binary string. For the values in the binary string, a pixel value less than the centre pixel is set to 1. Otherwise, it is set to 0. After the bit string is obtained, a Hamming distance is used to compute the similarity score between the left and right patches. The Census method is robust to illumination changes. Moreover, the binary string also contains location information, making the Census method more sensitive than the methods based on the sum of values of the matching windows such as SAD and SSD.

2.1.2 Deep Learning Based Cost Computation

Deep learning based cost computation takes left and right images as the input of a convolutional neural network and outputs a similarity score, as illustrated in Figure 2.1. Most deep learning based models handle the matching problem as a regression problem with L1 or L2 loss. Three main types of models are listed below.

1. **Two branch model** [23, 26, 27]: This model uses two independent branches to handle the left and right image patches. These networks concatenate the outputs of



(a) Two Branch Network (b) Siamese Network (c) Two-channel Network

Figure 2.1: Deep learning based cost computation [2].

the two branches and feed it to a fully connected network in the final layer to obtain a similarity score. The structure of this model is shown in Figure 2.1a.

(a) **The Siamese model** [14,28,29]: This type of network uses the same structure and same weights in both branches of the network. Each of the branches takes one of the image patches as input. After the output of each branch is obtained, the outputs are concatenated, flattened, and fed into a fully connected neural network. This type of model is shown in Figure 2.1b.

(b) **Two-channel model** [24]: The two-channel model uses a single branch to handle two patches by concatenating the two one-channel patches into a two-channel tensor, which is different from the two branch models that handle each patch in an independent branch. This type of model is shown in Figure 2.1c.

In order to achieve better results, three main extensions of the baseline model structure are used to replace the basic convolutional neural network with ResNet [30] in order to use skip connections between different layers.

2. **Learning multiscale features:** These methods add more branches to handle different sizes of input patches. Zagoruyko and Komodakis [24] proposed a two-branch siamese network. The outputs of all branches are combined at a fully connected neural network to estimate the similarity score. Moreover, similar to Zagoruyko and Komodakis's work, Chen et al. [29] apply two different fully connected networks on every two branches.

3. **Reducing the number of forwarding passes:** For each image, computing the similarity score for all pixels is computationally expensive. The most common approach applies the sliding windows method for each possible disparity for each pixel in the left image. Because the sliding windows are highly parallelizable and overlapped, Luo et al. [14] propose a method to reduce the time complexity by taking the row of the possible matching areas as input and computing the similarity score of all possible disparities as output. Therefore, the output of Luo et al.'s method is a disparity vector instead of a disparity value.

4. **Enlarging the receptive field:** Spatial pyramid pooling (SPP) [31] applies different max-pooling groups with different sizes of the spatial bin. Using SPP, a convolutional neural network with spatial pyramid pooling enlarges the receptive field and adjusts the different input sizes into a fixed-size output. Another approach [32] is to replace convolutions with dilated convolutions, which extends the receptive field but does not increase computational costs by filling in zero's in the convolution filters.

2.2 Cost Aggregation

Once the matching cost computation step is completed, a raw cost matrix is obtained. However, mismatches arise frequently due to textureless areas, repetitive patterns and occlusions. Cost aggregation is widely applied to reduce the mismatches and improve the accuracy of the final result. This section introduces two types of aggregation methods:

the traditional methods and deep learning methods.

2.2.1 Traditional Matching Cost Aggregation

Traditional matching cost aggregation methods are based on the assumption of smoothness: disparity should change continuously. Therefore, if the disparity of some points changes drastically, those points are highly likely to be mismatched points. Semi-global matching (SGM) [33] is the most balanced method between time consumption and accuracy.

Searching for the correct matches on the raw cost matrix is an NP-hard problem. SGM uses dynamic programming to minimize the energy function, which avoids huge time consumption.

The energy function is shown below:

$$E(d) = E_{\text{data}}(d) + E_{\text{smooth}}(d) \tag{2.7}$$

where d is the disparity value.

Generally, the energy function consists of two parts: the data and the smooth parts. The first part is the cost value on the raw cost matrix. The second part is the function that gives a higher value to discontinuous disparity points.

There are two functions in E_{smooth} of the SGM energy function: the penalty of the disparity gaps that equal 1 and the penalty of the disparity gaps larger than 1. The details of the smooth part are shown below:

$$E_{\text{smooth}} = \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 \, \mathrm{T}\left[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1\right] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 \, \mathrm{T}\left[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1\right] \tag{2.8}$$

where P_1 and P_2 are the weights of two penalty terms, where P_2 should be significantly greater than P_1, which allows the small change caused by continuous and curved surfaces.

T is the probability distribution of the corresponding intensities. N_p is the neighbourhood of p. D is the disparity map.

The simplest way to minimize the energy function is to enumerate every possible disparity value of each pixel. However, that is unacceptable due to time costs. Searching for the correct matching on the raw cost matrix is an NP-hard problem. One of the possible solutions is to apply dynamic programming on each row or line. However, using only dynamic programming on each row does not exploit the information between rows. Therefore, SGM does aggregation by using 4-16 paths dynamic programming, which combines the different path and row information to produce better accuracy. In many cases, 4-way dynamic programming is the most popular because programming of more than 4 paths increases accuracy slightly but increases the complexity significantly. The formululation of the dynamic programming problem is below:

$$L_r(\mathrm{p},d) = C(\mathrm{p},d) + \min \left\{ \begin{array}{l} L_r(\mathrm{p}-\mathrm{r},d) \\ L_r(\mathrm{p}-\mathrm{r},d-1) + P_1 \\ L_r(\mathrm{p}-\mathrm{r},d+1) + P_1 \\ \min_i L_r(\mathrm{p}-\mathrm{r},i) + P_2 \end{array} \right\} - \min_i L_r(\mathrm{p}-\mathrm{r},i) \qquad (2.9)$$

where path L_r is traversed in the direction r. The basic idea of dynamic programming is to combine the current cost value $C(p,d)$ with the lower cost of the surrounding pixels of two types: the disparity gap which equals one and that which is larger than one.

Other methods of aggregation are based on Markov random fields (MRF) and conditional random fields (CRF). Knöbelreiter et al. [34] introduced hybrid CNN-CRF, which computes the matching term by using CNN. After the matching term is obtained, it is used as a unary term of a CRF module, and another CNN computes the weight between each nodes in CRF. Knöbelreiter et al.'s method achieves a competitive result, with reduced number of parameters compared with other methods.

2.2.2 Deep Learning Based Cost Aggregation

There is no doubt that traditional algorithms can perform aggregation. But deep learning methods achieve high performance for aggregation. Using the 3D convolutional neural networks to aggregate the cost matrix is the most popular method. The 3D convolutional neural networks accept dimensions including the spatial and disparity dimensions, which significantly improves the accuracy of the final result. Khamis et al. [35] use 3D convolutions to regularise the cost matrix. They first estimate a low-resolution disparity map and then progressively improve the resolution of the disparity map using residual learning. The purpose of the low-to-high resolution method is to reduce the memory requirements. Building the cost volume is the step that consumes most of the time and memory. Chabra et al. [36] propose a method to use 3D dilated convolutions instead of the normal 3D convolutions. It reduces the computation time while capturing a wider context.

Because of the huge time and memory consumption, some works [37–40] use 2D convolutional neural networks instead of 3D CNN. The idea of these works is to use 2D convolutional networks to produce another 3D matrix. However, this approach also ignores the information from the disparity dimension and therefore suffers performance loss. Researchers have tried combining the depth direction via a gated recurrent unit (GRU) to overcome this drawback. Yao et al. [41] propose a method that uses GRU on the depth direction and 2D CNN on the spatial dimensions and turns the 3D cost matrix into a sequential 2D cost map. Yao et al.'s method reduces memory consumption significantly from 15GB to around 5GB, making it possible to handle high resolution images.

2.2.2.1 4D Cost Matrix

Compared with the 3D cost matrix, the 4D cost matrix adds the dimension of features [15, 42–46]. Therefore, the similarity measure of the 4D cost matrix is learned by the neural network instead of using hand-crafted similarity measure methods. There are two

main approaches to construct the 4D cost matrix. The first is to compute the difference between the left and right features [42]. The second is to concatenate the left and right features ([43], [15], [45], [46]). Based on the 4D cost matrix, Chen et al. [15] construct the 4D cost matrix, PSMNet, with the shape $2 \times channel \times disparityrange \times width \times high$. The first half of the channel are the left features, and the other half are the right features. Given a disparity d, the first half of the channel keeps the left feature index ranging from d to the end. The remaining half of the channel keeps the right feature index ranging from 0 to d.

The 4D cost matrix contains more information than the 3D cost matrix. Although the richer information in the 4D cost matrix can produce results with higher accuracy, a larger and deeper network is required to handle the 4D cost matrix due to a lack of information about the feature similarities.

Because the 4D cost matrix can be viewed as containing multiple channels of the 3D matrix, most works use the various structures of 3D CNN to aggregate the matching cost. U-Net [3] is a popular convolutional network in 2D image segmentation based on an encoder-decoder architecture with 2D convolutions and skip connections. Some works also extend 2D U-Net into 3D U-Net to aggregate the 4D cost matrix ([43]). One way to improve the accuracy of aggregation is to enlarge the receptive field of the U-Net. Like Kendall et al.([43]), Zhong et al. [45] also introduce residual connections in U-Net, which enlarges the receptive field without increasing the computational cost. Also, Kendall et al. [43] aggregate the cost volume step by step with four subsampling layers, making it possible to enlarge the field of view of context. Other works focus on extracting multiscale information. Wu et al. [46] aggregate 4D cost volumes into a 3D cost volume using a 3D multi-cost aggregation module, which groups the small scale volume with the larger scale volume using upsampling. After that, these volumes are fused by the 3D feature fusion module.

Inspired by the traditional cost aggregation algorithm, the semi-global matching (SGM)

technique has been combined with the deep learning methods. Not using the traditional SGM, which aggregates the cost matrix by using 4-8 paths dynamic programming to select the neighbour cost to minimize the energy function, Yu et al. [47] propose a method where two branch networks combine the SGM ideas with deep learning. One branch uses 2D CNN to handle the left image to produce the guidance (confidence) maps W_i. Another branch of the network is a 3D CNN which takes the 4D cost matrix as input and outputs the 3D cost matrix C_i. In the last step, the final output is produced as $C_i \cdot W_i$.

2.2.2.2 Reducing the 4D Cost Matrix Resource Requirement

With the huge memory and time consumption of 3D convolution network, some works focus on enhancing the speed and reducing the memory requirements. There are three main approaches: (1) reducing the size of the 3D convolutional layer; (2) refining the cost matrix and disparity map from low resolution to high resolution; (3) compressing the 4D cost matrix.

(1) Reducing the size of the 3D convolutional layer: One of the works focusing on reducing the size of the network is GANet by Zhang et al. [48]. It replaces the 3D convolutional layer with a semi-global aggregation (SGA) layer, which replaces the manually defined parameters with learnable weights. At the end of the network, the local guided aggregation (LGA) layer is applied to refine the small structures and edges. The SGA and LGA layers reduce the number of costly 3D convolutions and significantly improves the accuracy of matching textureless areas.

(2) Refining the cost matrix and disparity map from low resolution to high resolution: Instead of directly handling the full size of the input image, some works first reduce the size of the 4D cost matrix to obtain a low-resolution disparity map and then upsample and refine by using encoder-decoder networks. For example, PSM-Net by Chang and Chen [15] first estimates a low-resolution 4D cost volume and then regularises and upsamples it by

using an encoder-decoder 3D block.

Also, Wang et al. [49] introduce AnyNet, which builds the low-resolution 4D cost volume by using low-resolution 2D features. This approach learns the residuals from the lower level to produce the higher resolution cost volume. As a result, this network can return the intermediate disparities to reduce memory and time consumption.

(3) Cost matrix compression: The cost matrix consumes most of the memory in many stereo matching networks. Therefore, Tulyakov et al. [50] compress the left and right features and then concatenate them to produce a smaller cost matrix. This method reduces memory consumption without sacrificing accuracy.

2.3 Disparity Computation

The most direct way to compute the disparity from a 3D cost matrix is to use pixel-wise *argmin*, which selects the disparity or the depth level index of the minimum cost value as the disparity. However, *argmin* can only produce integer disparity values. Therefore some researchers [35, 43, 51, 52] have used a *soft-argmin* method to replace the *argmin* method to achieve subpixel results.

The *soft-argmin* operator works well on unimodal and symmetric distributions. Once this assumption is broken, it often results in an over-smoothed result. Chen et al. [53] only apply the weighted average operation in areas with maximum probability to address these issues.

2.4 Confidence Measurement

Detecting wrong values in the disparity map is meaningful. In tasks such as autonomous driving or medical imaging, even a small number of errors can lead to serious consequences.

Therefore, many researchers have focussed on confidence or uncertainty maps and on removing errors. The confidence map can also be incorporated at the aggregation step [54–56]. One example is Seki et al. [55], who combine the confidence map with semi-global matching (SGM). Unlike Seki et al., Gidaris et al. [56] detect and replace the error with a neighbouring pixel on the cost matrix by using a confidence map before passing it to an aggregation network.

Most recent works are based on supervised learning [57–62], which directly takes the cost matrix as an input of the network and produces the confidence map as output.

2.4.1 Left-Right Consistency Check

The left-right consistency check is the most popular way to measure confidence. The idea is to estimate disparity maps twice based on both the left image and the right image as references. The confidence map or error map can be produced by calculating the differences between left and right disparity maps. This method can detect the occlusion areas which are visible in one view but not in the other.

Some works [54, 55] try to combine left-right consistency check with deep learning. Seki et al. [55] propose a two-channel patch-based network which is a patch-based confidence prediction (PBCP) network. The PBCP network takes two-channel feature patches as input. The first channel enhances the differences within one channel by computing the difference between the centre of the left disparity patch and the surrounding pixels. The output of the PBCP is labelled per pixel to indicate whether the pixel is correct or not.

The two methods mentioned above [54,55] both treat the left-right consistency check as an independent step. Some works combine the left-right consistency check with aggregation or disparity estimation. For example, Jie et al. [54] use two parallel convolutional LSTM networks to compute the left and right confidence maps, which is concatenated with cost volumes and handled by the 3D LSTM step by step.

2.4.2 Confidence from a Single Raw Disparity Map

The normal left-right consistency check has to estimate the disparity maps twice on the left and right images, which increases time consumption. Hence, some works focus on estimating confidence maps on single images. Shaked and Wolf [63] directly use a two-layer fully connected network to predict the confidence map, which is supervised by the gap between the dynamically changing output and the ground truth. Instead of taking the whole image as input, Poggi and Mattoccia [60] decomposed the large image into different 9×9 patches and fed them into a CNN to predict confidence. Similar to Poggi and Mattoccia, Zhang et al. [52] use pixel-wise left-right consistency check instead of the label for supervision, and only need the left disparity map as input on inference.

2.4.3 Confidence Map from Matching Densities

There are two other methods to obtain the confidence map, both local and global. The local methods mainly focus on local consistency, in which the disparity value within one object surface is identified. Seki et al. [55] use the consistency of neighbouring pixels as the first channel input to CNN. Tosi et al. [64] introduce LGCNet to utilise the output of the two networks and concatenate the initial disparity map with the reference image as input.

2.4.4 Combining Multiple Estimators

Besides using the disparity map and reference image as cues to predict disparity maps, a combination of multiple outputs from the different models is able to achieve a better result. Haeusler et al. [57] use Random Forests [65] to handle 23 confidence maps and achieve a better result than any other confidence map in the pool. Similar to Haeusler et al., Batsos et al. [62] combine four different cost computation methods to generate a more

accurate cost matrix. In addition, Poggi and Mattoccia [60] train an ensemble regression trees classifier to compute the confidence map.

2.5 Domain Adaptation and Transfer Learning

Generalisation is said to be achieved when the deep learning model performs stably on a new domain in which the scenes are significantly different from the images used during training. Domain shift occurs when the training data distribution is different from the test data distribution. For example, the difference between the indoor and outdoor datasets or between synthetic and real datasets is a domain difference. In addition, the domain difference between training and test data is not the only one that leads to domain shift. The camera setting is also an important factor that may lead to domain shift. An example of camera parameters leading to domain shift is the disparity range. When the model is trained on a small disparity range and then applied to test data with a much larger disparity range, the pixels out of the disparity range of the training set will be missing in the output.

There are two main approaches to address the domain bias issue: adaptation by fine-tuning and adaptation by data transformation.

2.5.1 Adaptation by Fine-Tuning

Although active depth sensors can provide ground truth in many areas, labelling is not easy to do. Therefore, the most challenging part of fine-tuning is the acquisition of sufficient labels in the target domain. Some works [66, 67] rely on traditional stereo matching methods to obtain ground truth disparity labels. Also, to reduce errors from off-the-shelf stereo algorithms, these works combine the SOTA confidence measurement methods to detect and remove the errors produced by the existing stereo matching methods. In

addition, recent works show that using only sparse ground truth can also achieve fine-tuning. For instance, DispNet [37] uses very sparse ground truth to fine-tune their model.

Instead of labelled data, self-supervised methods and weakly supervised methods may be applied for fine-tuning, such as works by Godard et al. [68], Zhou et al. [69], Zhang et al. [70] and Poggi et al. [71].

The offline fine-tuning methods discussed so far suffer from a significant common drawback: fine-tuning is required each time the model is applied to a new domain. Hence, many researchers have developed online adaptation techniques. Tonioni et al. [72] convert the fine-tuning to a continuing learning problem, which upgrades the model weight frame by frame in the real environment. In addition, they also propose a lightweight network, which trains the different parts of the whole network independently. This method enhances model execution speed and achieves 25 fps in the fine-tuning step. Similarly, Zhong et al. [73] train a different random initialization deep network online, which leverages the temporal information by using LSTM in their model during prediction.

2.5.2 Adaptation by Data Transformation

Dta transformation methods have been used [74, 75] to adapt the model instead of re-training the model to adapt to a new domain. Atapour-Abarghoue et al. [74] propose a two-step pipeline to transform synthetic data to real-world data. The first step is to directly use the synthetic data to train a depth estimation network. The second step is to train a network to transform synthetic images to real-world style images. Similarly, Zheng et al. [75] transform synthetic images to real-world style.

2.6 Datasets

Some popular stereo matching datasets arte now discussed. These datasets can be divided into two categories based on the domain: simulated and real-world datasets. Based on the scenes, they are divided into indoor scenes and outdoor scenes. In this section, the influence of the domain gap is also discussed.

1. **Simulated datasets:** To obtain sufficient training data to feed data-hungry deep learning models, some works use game engines to acquire a large number of stereo pairs with ground truth, such as FlyingThings3D [37] and MPI datasets [76]. The benefit of simulated datasets includes the low cost of obtaining extensive data. However, synthetic data containing varied real-world appearances may lead to lower performance of the models on real-world datasets.

2. **Real-world datasets:** There are two popular real-world datasets: Middlebury Stereo 2014 [1] and KITTI 2012/2015 [4, 5]. The Middlebury Stereo 2014 dataset was collected using structured light, and the KITTI 2012/2015 dataset was collected using LiDAR. The dataset sizes are also different. The former contains only 10 images for training, while the latter contains 200 images.

3. **Indoor and outdoor scenes:** Different from the simulated datasets, real-world datasets contain two main types of scenes: indoor and outdoor scenes. One of the differences between indoor and outdoor datasets is the disparity range. The outputs of far and close objects often collapse when a model trained on an indoor dataset is tested on an outdoor dataset. Although some works have a large range of disparity, the models often perform well within the disparity range of the training dataset but collapse on image points that are outside the disparity range of the training datasets of extremely low data size.

2.7 Handling High-Resolution Images

Limited by GPU memory, most current deep learning-based works use low resolution images. In order to handle high-resolution images, some works use bottom-up techniques that operate in a sliding window style approach. Bottom-up techniques decompose large image pairs into small patches. After that, those small patches are used to train the model to obtain the refined result for each small patch. Finally, the patch-level results are merged into a larger output image of the same size as the input image using voting. Lee et al. [77] enhance the bottom-up techniques with a fusion network that operates in the Fourier domain to handle different patch sizes.

Split-and-merge methods reduce the memory requirements significantly. However, the bottom-up techniques trade speed for memory. They need to handle different areas of the input multiple times, which increases processing time.

2.8 Training Procedures

Unlike other computer vision tasks without clear mathematical constraints, stereo matching has clear geometric constraints, and can achieve acceptable results with fewer training data. Two types of training procedures will be introduced in this section: supervised learning and unsupervised learning.

1. **Supervised learning:** Supervised learning uses labels to guide model training. In order to avoid the unbalanced data problem, some patch matching networks randomly collect the same number of negative and positive samples for training [24, 29, 78]. In addition, data augmentation is widely used in computer vision areas. Rotating or flipping patches is a standard method to enhance the performance of the patch matching networks. Supervised learning is a powerful method to build a

high-performance network. It also requires a large number of data sampled. With the decreasing cost of depth sensors, many datasets also have ground truth by using different types of depth sensors. However, the depth map from the depth sensors may also contain noise and wrong areas. For example, the KITTI 2012 dataset does not have depth values for reflection areas.

2. **Unsupervised learning (zero-shot learning):** When labelled data cannot be obtained, the constraints in stereo matching [68] can be used to train a patch matching network. There are five constraints in stereo matching: the epipolar constraint, the disparity range constraint, the uniqueness constraint, the continuity (smoothness) constraint and the ordering constraint. With these five constraints, multi-instance learning is suitable to be applied to unsupervised learning. For example, epipolar constraints may be used to build the training set. With the epipolar constraint, all patches not centred at the same row on the right image can be seen as a negative example.

3. **Semi-supervised learning:** Although the unsupervised method can avoid relying on large quantities of ground truth depth data, the accuracy of those models is lower than the supervised models. To keep the balance between the number of labeled data and accuracy, some of the work tries to reduce the amount of manual labeling. The semi-supervised learning methods are generally based on confidence, which provides the information to remove the incorrect disparity for training. Tonioni et al [66]. propose a confidence-guided loss to reduce the impact of that unreliable disparity. The confidence map from Tonioni et al. is based on the traditional stereo matching algorithm. Kuznietsov et al. [79] also propose a method based on photo-consistent dense depth to gain reliable disparity. Unlike Tonioni et al. and Kuznietsov et al., Chen et al. [80] used an iterative method to collect the reliable disparity from the network's output. The confidence map from Chen et al. is based on the left-right consistent check on each iteration.

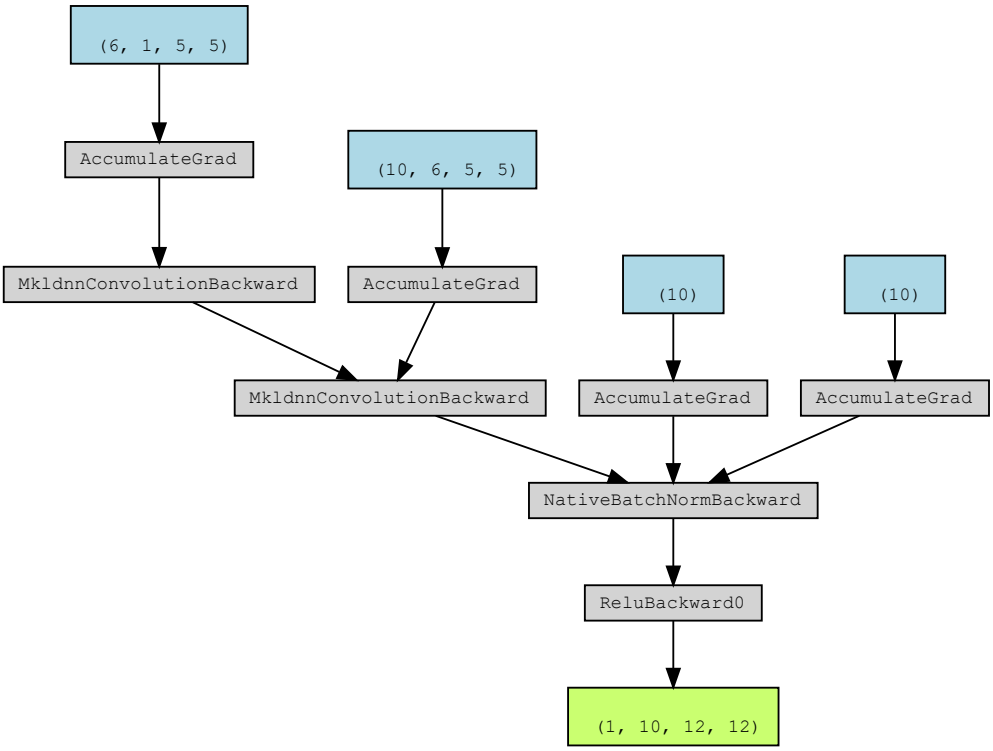


Figure 2.2: Computational graph. The blue blocks are the input tensors of different sizes, and the green block is the output tensor. The gray blocks are different types of operators.

2.9 Framework

Before considering how to optimise model quality, it is necessary to understand the pipeline of the popular deep learning framework. The two major components are tensors and operators. The tensor is a multi-dimensional array of numerical values, and the operator is a mathematical operation. The deep learning framework provides high-level programming interfaces and operators for users to build a deep learning model. After the model is compiled, the framework will create computation graphs consisting of tensors and operators. Dependency is represented by the edges between the operators and tensors. In Figure 2.2 an example of a computation graph is shown.

In order to optimize memory in more detail, the computation graphs should be analysed. Three main factors affect the maximum memory used during training: the size of different types of tensors and operators, the lifecycle of the tensors and memory block management. Gao et al. [81] and Albert et al. [82] provide two methods to estimate memory consumption.

As Gao et al. [81] indicate, there are two key points to conssider: release policy labels and dependencies. One of the release policy labels is RELEASE_ON_EXIT, which means the tensor cannot be released until the model ends. As for the dependencies, the tensors and operators cannot be released if they rely upon the other operators. Finally, memory block management is based on the best-fit with coalescing (BFC) algorithm, which consists of three steps. The first step is searching for the first free block larger than the required size. If a suitable free block is available, the required memory block is allocated from this free block, and the remaining free space will become a new free block. If a suitable free block is unavailable, the required block will be added to the tail of the memory block list. In order to enhance search speed, the free blocks are managed by a linked list. The linked list is a collection of free blocks whose block sizes are larger than a predefined and ordered bin size.

By a combination of the three steps while traversing the computational graph, memory changes can be estimated without actually running the model. In addition, this result provides an entry point for the current research through detailed analysis of the computational graph from the deep learning framework.

2.10 Current Popular Tricks

In this section, some popular tricks to reduce time and memory consumption are introduced.

2.10.1 Mixed Precision Training

Directly using half-precision to train a deep learning model will lead to significant decrease in accuracy. Therefore, a popular method is to mix the half-precision FP16 and and the full F32 tensor during training. Micikevicius et al. [83] show details of the mixed-precision training method in a modern framework. Three main approaches are used for mixed-precision training of the deep learning model: the FP32 master copy of weights, loss scaling and arithmetic precision.

The FP32 master copy of weights is maintained and updated with the weight gradient during the optimizer step. The copy of weights increases the memory requirement, however the need for FP32 master weights is not universal. Compared with full precision training, in the mixed-precision training the copy of the master weights still reduces the memory requirement. In addition, the master copy of weights relieves the problem that updates (weight gradients multiplied by the learning rate) become too small to be represented in FP16.

Loss scaling is the scaling up of gradients, which will shift values to occupy a more representable range and preserve values otherwise to zero value.

Arithmetic precision is a guideline for choosing suitable precision for operations. According to Micikevicius et al. [83], large reductions (sums across elements of a vector) should be carried out in FP32 but loaded by the FP16.

2.10.2 Pruning

Pruning is a model optimization technique that eliminates unnecessary weight tensor values to reduce memory and time consumption. In this section, four main types of pruning methods are reviewed: magnitude-based weight pruning, the impact of loss, reconstruction ability of feature output and other methods.

Magnitude-based weight pruning is based on the assumption that a larger value is on the weight tensor or the output feature. There are many classical works using this method ([84], [85]).

The impact of loss method observes the impact of the loss to determine the importance of parameters. Two recent works [86, 87] use this method.

Reconstruction ability of feature output considers the quality of the reconstruction which attempts to minimize the reconstruction error of the feature output by the network after clipping. The cropped information is not very important if the current layer is pure and does not affect the subsequent output. Two recent works [88, 89] use this method.

2.10.3 Knowledge Distillation

Knowledge distillation is the abstraction of a complex model using a learned mapping of inputs to outputs, and is achieved using a Teacher-Student model. In general, the teacher model is larger and the student model is smaller. The basic idea is to use the output of the teacher model to supervise the learning of a student model. The assumption is that the output of the teacher model contains the correct label and the similarity information between categories, which provide richer information for student training. In Hinton et al.'s [90] work, the student model learns almost the same generalisation as the teacher model, even though it is smaller. In addition, the knowledge distillation method can transfer the information from multiple teacher models into one student to improve the latter's generalisation.

2.11 Fully Convolutional Networks (FCN)

In this work, the proposed network is based on a encoder-decoder structure and fully convolutional networks (FCN) [91]. Hence, the current encoder-decoder structured networks

and FCN are introduced in this section.

The motivation of the FCN is to solve per-pixel tasks. Unlike in recognition tasks, the stereo matching task requires to output the same size as the input image instead of a one-hot vector. Long et al. [91] first introduced the FCN structure, which replaces the fully connected layer with a convolution layer in the output layer. After that, different types of FCN have been applied to other areas.

Encoder-Decoder is a model framework, and does not refer to a specific algorithm. Under this framework, different algorithms may be used to solve different tasks. First, an encoder converts the input sequence into a dense vector of fixed dimension, and the decoder stage generates the target translation from this activation state. Because this work only uses the encoder-decoder CNN, the popular encoder-decoder network used in computer vision areas is discussed.

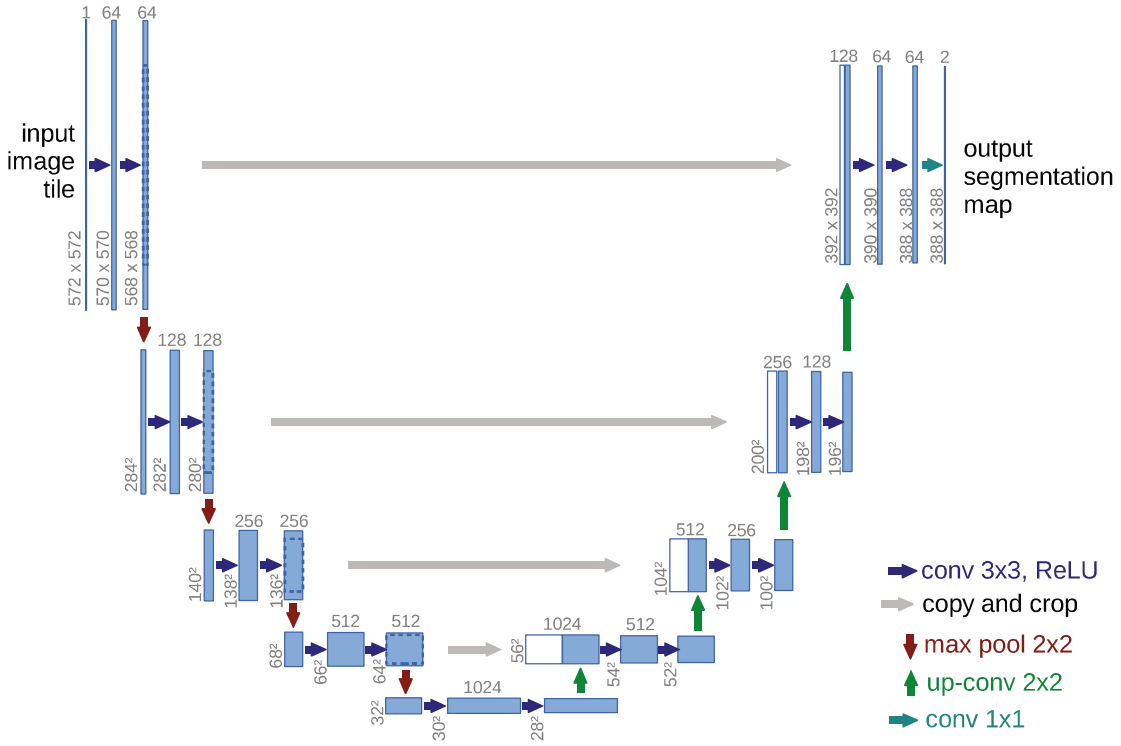


Figure 2.3: Example of U-Net [3].

The standard U-Net [3] is a fully convolutional network with the encoder-decoder structure. It consists of three major parts: encoder, decoder and skip connection. The encoder part, except for the convolution layer, is used for extracting features. Four pooling layers are used to reduce the tensor's size. In addition, four transposed convolution layers are used to recover the size of the tensor from the encoder layers. The skip connections between the encoders and decoders also prevent vanishing gradient problem. Because U-Net has achieved great success in many areas, some of the works have focussed on turning 2D U-Net into a 3D U-Net. The first work is from Cciccek et al. [18], who modifed 2D U-Net into 3D U-Net following the same structure of the 2D U-Net.

2.12 Summary

According to our contribution, some of the works related to our contribution and experiment are introduced in this chapter. The traditional and deep learning based cost computation is introduced in Section 2.1, which corresponds to our adaptive cost computation methods in Chapter 3. After that, the works focus on our major contribution to the cost aggregation step are reviewed in Section 2.2. Next, disparity computation and confidence measurement are introduced in Section 2.3 and Section 2.4. Section 2.5 reviewed domain adaption and transfer learning which are related to our Chapter 4. Finally, some datasets, framework, tricks and Fully Convolutional Network related to our experiment are reviewed in Section 2.6, Section 2.9, Section 2.10 and Section 2.11.

Chapter 3

Cost Computation

In the cost computation, it is non-trivial to identify an optimal window size. A small window can have less smoothing effects while better retaining object details but incur mismatches on textureless areas. Small windows are usually not large enough to gather sufficient information to distinguish between wrong matches and correct matches, while a large window can reduce noise in textureless areas but lose details on matching. Therefore, it is challenging to predetermine the optimized window size.

Compared to the traditional fixed windows methods, our adaptive window matching method (SIFT-Census) can reduce mismatches on textureless areas and avoid the over smoothing problem on small objects.In this chapter, a novel adaptive window matching method is introduced.

The proposed adaptive window matching is based on the idea that large matching windows can deal with textureless areas better, while smaller matching windows are more suitable for textured regions. Cues that indicate the location of richly textured areas and textureless areas are used. One of the assumptions is that richer textures are on areas surrounded by more interesting points. In contrast, an area far away from interesting points has lesser texture in that area. In addition, to improve the accuracy of the raw

disparity map, other cues are combined with the proposed method and are discussed in this chapter.

Experiments show that the propsoed adaptive window matching method outperforms fixed window methods. Particularly, when two challenging areas with textureless and small object areas appear in an image simultaneously, the proposed method can achieve higher accuracy.

3.1 Matching Cost Computation

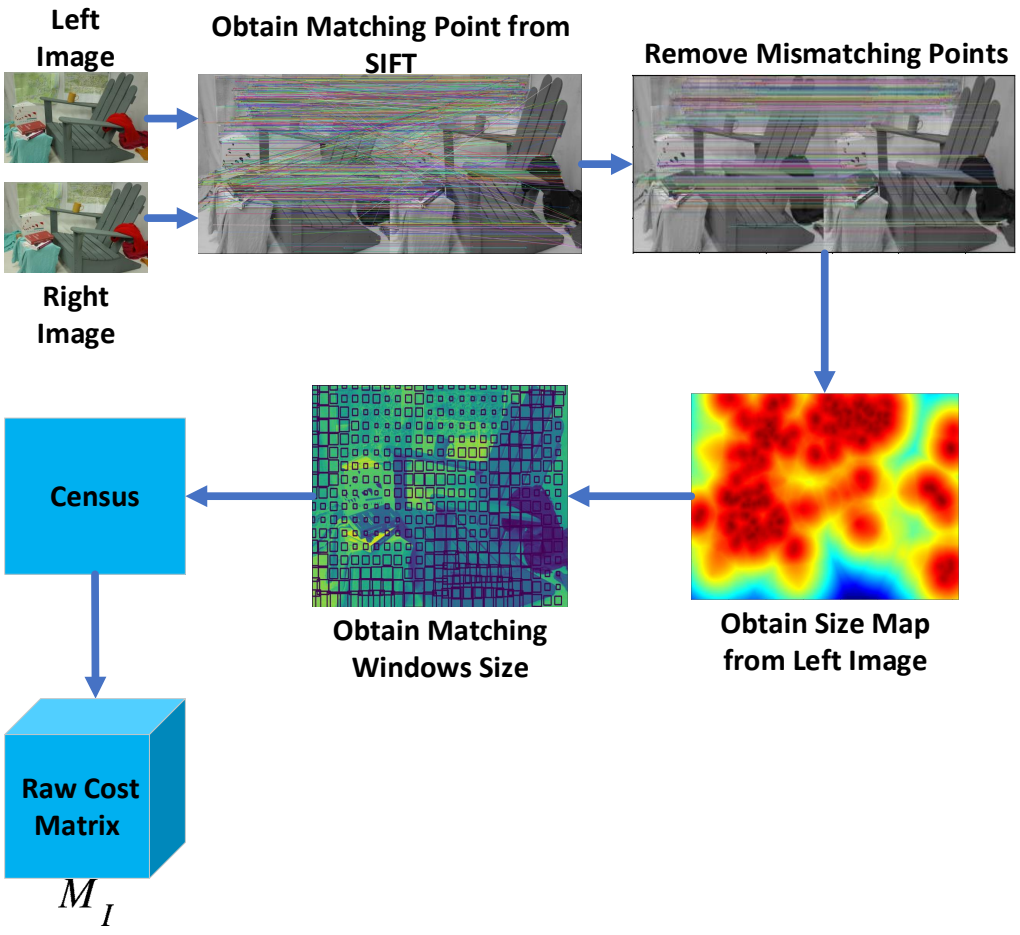


Figure 3.1: Details of proposed SIFT-Census.

A new approach named SIFT-Census is proposed, which combines SIFT features and the Census matching method. SIFT-Census is a census-based approach with adaptive window sizes, which uses the matching points based on SIFT as a cue to adjust the size of the matching windows.

Before details of SIFT-Census are discussed, the Census transform is first introduced. Census transform is a pixel-wise image operation that associates a binary strong with each pixel of a grayscale image. The idea of Census transform is to generate a bit string for each pixel by comparing the centre pixel p_c with a surrounding pixel p'. If the surrounding pixel p' is greater than the centre pixel p_c, the bit on this pixel is set to 1, otherwise, it is set to 0. Details of the Census string are shown in Figure 3.2. The formula of the Census bit string is defined as:

$$\xi\left(p_c, p'\right) = \begin{cases} 0 & \text{if } p_c > p' \\ 1 & \text{if } p_c \leq p' \end{cases} \tag{3.1}$$

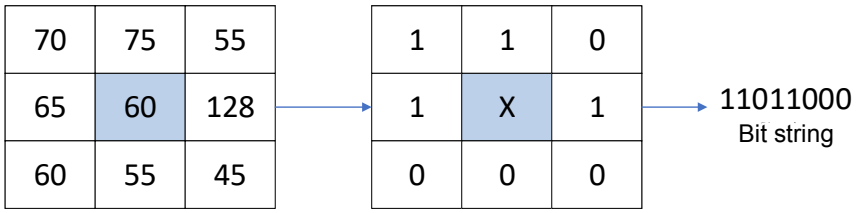


Figure 3.2: Details of Census bit string. Pixels with value less than the centre pixel are set to 0. Otherwise, they are set to 1.

After the pixel-wise Census string is obtained, the Census string of the right image is compared with that of the left image according to different disparity levels. To compare the Census strings the Hamming distance is used, which counts the number of different bit values on the left and right Census strings and divides it by the length of the string.

Using Census as the cost computing method in stereo matching can achieve high performance and easy parallelisation, facilitating computation on some edge devices.

As shown in Figure 3.1, the first step is to obtain a set of SIFT matching points between

the left and right images. Because both the left and right images have been rectified, the matching points p_{ml} and p_{mr} in the left and right images from the SIFT matching operation should be on the same row, which means that the correct matching points should have the same y coordinate. The second step is to remove those matching points with different y coordinates. The third step is to compute the size map. For each pixel p on the left image, the value of p_d on the size map is given by:

$$p_d = avg(top3(dis(p, p_{ml}))) \tag{3.2}$$

where dis is the point to point distance, $top3$ is the function that selects the top three shortest distance points, and avg is the average function.

A mapping function is designed to map p_d in the size map into size d of the Census matching windows. In the final step, for each pixel at location (x, y), the Census matching windows with adaptive sizes are used to obtain the cost value. Here d is given by:

$$d = s_b + p_d/f \tag{3.3}$$

where s_b is the size of the base window of the Census matching windows and f is the scale factor.

The window size is an essential hyperparameter to assure the quality of the matching result, especially in textureless areas. Due to a lack of information on textureless areas, it is possible to obtain the same cost value at different positions, leading to mismatches. One of the possible solutions is to utilise larger matching windows to capture more information rather than small window sizes.

However, large windows could lead to over smoothing in fully textured areas and on object boundaries. Therefore, to achieve a better matching result, the size of the matching windows should be adjusted dynamically.

For dynamic size adjustment of the matching window, cues are required to indicate whether

an area is fully textured or not. One of the cues are the matching points from SIFT obtained with a brute-force matcher. After removing mismatched points from the brute-force matcher, compared with those pixels located on the fully textured areas, matching points on the textureless areas always have a larger distance between them.

Compared to fixed window size methods, the proposed SIFT-Census method can dynamically adjust the window sizes. When the centres of the matching windows are located on textureless areas, the window sizes will be enlarged to reduce mismatching. When the centres of the matching windows are located on fully textured areas, small windows will be applied to retain more object details. For example, in Figure 3.1, the value d is directly used on the size map as the matching window size.

It is noted that the propsoed adaptive matching window method can be used in Census as well as in other windows based matching methods. In this chapter, other matching cost computation methods combined with the proposed method are also discussed.

3.2 Combining with Other Cues

Although SIFT-Census significantly reduces mismatches, some noise still exists due to noisy or challenging areas in images. In order to minimize the noise, one possible solution is to combine the SIFT-Census with other cues. Inspired by AD-Census [22], SIFT-Census is combined with other cost computation methods.

The first step of a combination algorithm is to normalise the cost volume MO from other methods:

$$MO_{\text{norm}} = \frac{MO - MO_{\text{min}}}{MO_{\text{max}} - MO_{\text{min}}} \tag{3.4}$$

where MO_{min} is the least cost value in the other cost matrix, MO_{max} is the largest value and MO_{norm} is the normalized cost volume from other cost computation methods.

After MO_{norm} is obtained, it is combined with the raw cost volume using point-wise addition:

$$M'_{\text{com}} = MO_{\text{norm}} + M'_c \tag{3.5}$$

After combination, the new cost volume is in the range of 0-2. To obtain the initial disparity map, $Argmin$ is applied to the new cost matrix:

$$D_{\text{com}} = Argmin(M'_{\text{com}}) \tag{3.6}$$

3.3 Experiments

To show the effectiveness of the proposed adaptive matching windows size method, SIFT-Census is compared with traditional Census with regular as well as large matching window size. The first experiment is designed to compare SIFT-Census with Census on the exact maximum matching window sizes, by using the maximum matching window size as the controlled variable. The third experiment uses large matching windows to show the advantage of keeping the edge and small object sharp. The second experiment uses average matching window size as the controlled variable in the regular window size part, in order to compare SIFT-Census with Census on the same average matching window sizes. In this part of the experiment, the matching windows of regular Census and SIFT-Census will be set to the same average and maximum value.

3.3.1 Datasets and Evaluation Metrics

The Middlebury Stereo 2014 dataset discussed in section 2.6 is a high-accuracy dataset for indoor scenarios. The hyperparameters s_b and f will be adjusted to achieve different average and maximum matching window sizes. During testing, the pixel gap greater than 2.0 between the ground truth and the raw disparity is considered to be a mismatched pixel.

3.3.2 Implementation Details

Implementation was within Python 3.7 environment. The SIFT implementation in OpenCV 3.4.2 Python version was used to obtain the SIFT matching points. Multiprocessing was applied in all parts of the implementation, including obtaining the window sizes and Census transformation. In addition, Numba, a library to accelerate Python code, was also widely adopted to further speed up computations.

3.3.3 Window Size: Maximum

In this experiment, SIFT-Census is compared with the ordinary Census, and the size of the base window is set to 7×7. In Table 3.1 the accuracies achieved by Census and SIFT-Census are shown, with different parameters on the training set from the Middlebury Stereo 2014 dataset.

The largest window size from SIFT-Census is selected as the fixed matching window size for Census for fair comparison. As can be seen, SIFT-Census produces more accurate results than the Census, even with smaller average matching window sizes than those of Census.

Max window size	**Census**	**SIFT-Census**
39*39	0.610	0.624
43*43	0.601	0.631
45*45	0.607	0.610
61*61	0.564	0.620

Table 3.1: Accuracy comparison of Census and SIFT-Census.

Clearly, combining Census with the proposed adaptive window size matching method, ie SIFT-Census, leads to better accuracy on the same maximum matching size. SIFT-Census is slightly better when the window size is less than 39x39, however it can significantly outperform Census when the matching window size is larger than 43x43. For example,

SIFT-Census obtains an improvement of 0.03 when the maximum matching window size is $43 * 43$, and the improvement is even more noticeable for window size $61 * 61$.

The underlying reason for the performance gap between Census and SIFT-Census when using large matching window sizes is that the average window size of SIFT-Census is significantly smaller than that of regular Census. In spite of this, the proposed adaptive windows method (SIFT-Census) is better than fixed-matching windows.

3.3.4 Regular Windows Size: Average

Here, the experimental setup is identical to that of Section 3.3.3: the same base window size and matching algorithm are used, except that the maximum matching window size is replaced with the average window size. An accuracy comparison appears in Table 3.2.

Average window size	**Census**	**SIFT-Census**
7*7	0.316	0.316
11*11	0.425	0.469
13*13	0.454	0.489
15*15	0.476	0.491

Table 3.2: Accuracy of Census and SIFT-Census compared on same average window size.

As clearly shown in Table 3.3, the proposed method SIFT-Census produces better performance than regular Census for the same average window size. Specifically, both models obtain the same accuracy with small window size $7 * 7$ as they have the same matching window size. When the average window size is increased to $11 * 11$, SIFT-Census outperforms regular Census by 4%. The improvements of SIFT-Census over regular Census for average window sizes $13 * 13$ and $15 * 15$ are 3.5% and 2%, respectively.

To summarise, the main reason for the improvement is that for the same average window size, SIFT-census allocates larger windows on textureless areas and smaller ones on richly textured areas so that the accuracy on both textureless areas and detailed areas improve.

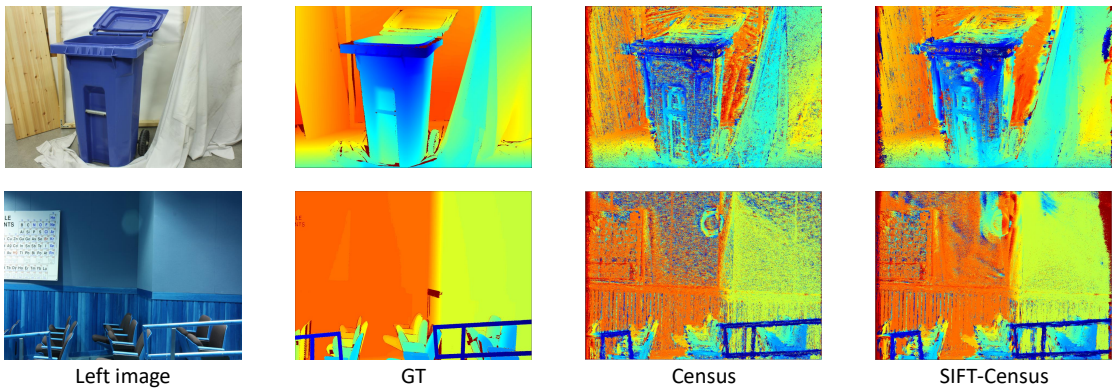


Figure 3.3: SIFT-Census example.

3.3.5 Application on Other Cost Compution Methods

In this section, our third experiment applies our adaptive windows method with other cost computation methods. The proposed adaptive windows method can be used not only in the Census transformation but also in other windows-based matching methods. The Sum of Squared Differences (SSD) is selected as the cost computation method to replace Census in this experiment. The comparison results are in Tables 3.3 and 3.4.

Average window size	**SSD**	**SIFT-SSD**
11*11	0.60	0.62
13*13	0.62	0.64
15*15	0.63	0.65
35*35	0.63	0.63

Table 3.3: Accuracy comparison of SSD and SIFT-SSD on same average window size.

Maximum window size	**SSD**	**SIFT-SSD**
25*25	0.55	0.68
37*37	0.60	0.66
39*39	0.59	0.67
45*45	0.56	0.65

Table 3.4: Accuracy comparison of SSD and SIFT-SSD on same maximum window size.

As can been seen from Tables 3.3, equipping SSD with the proposed adaptive windows matching method achieves better accuracy on the same average window size in general. Yet, the improvements are comparatively subtle. For example, the accuracy boosts are between 0 to 0.02 across different average window sizes. In addition, accuracy remains similar when increasing the average window size for both SSD and SIFT-SSD.

Similarly, SIFT-SSD was also tested on large window size and the results are shown in Table 3.4. The following observations may be made. Firstly, in general the enhancement of SIFT on SSD is larger than that on Census. Secondly, the performance of SSD and SIFT-SSD drop when the maximum window sizes are larger than 37*37 and 39*39 respectively. Thirdly, SIFT-SSD obtains the best performance at maximum window size 25*25, outperforming regular SSD by 0.15. The improvements at other maximum window sizes range from 0.06 to 0.09. This ascertains the effectiveness of the proposed approach.

Even though the accuracy enhancement is not significant when using large window sizes, there are some noticeable differences in small objects and edge details. Some examples are shown in Figure 3.4 where the matching window size is set to 35*35. Clearly, SIFT-SSD provides a sharper raw disparity map with the same accuracy as regular SSD. Significant differences can be found in two areas, including the chair and wheel. A reasonable explanation for this observation is that the sizes of the matching windows from SIFT-SSD are not evenly distributed. The matching windows around the rich textured areas like the chair and the wheel are smaller than those in less textured areas, which is beneficial as smaller matching windows sizes can avoid information losses on these areas.

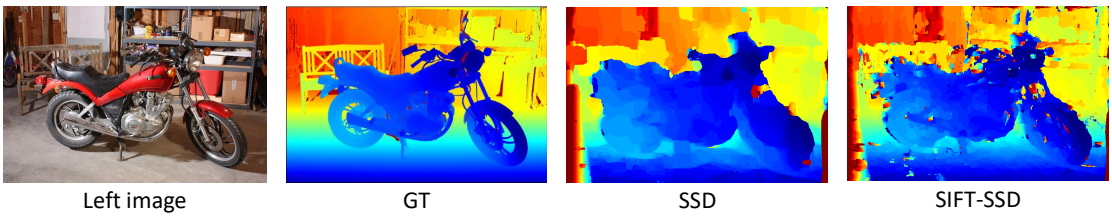


Figure 3.4: SIFT-SSD example.

3.3.6 Combining with Other Cues

In this section, the best results from SIFT-Census are combined with SIFT-SSD. Specifically, the SIFT-SSD with an average window size of 15*15 is adopted, and its normalized results are integrated with SIFT-Census.

As expected, better accuracy is obtained by combining the benefits of both methods. A major improvement of the combination is that it can reduce discrete noise as shown in Figure 3.5.

In Figure 3.5, noise in the initial disparity map is reduced after combining the results from SIFT-Census with SIFT-SSD. On the Motorcycle image, before the combination there are some mismatched points caused by the texture edge on the map, however it is significantly reduced after combination.

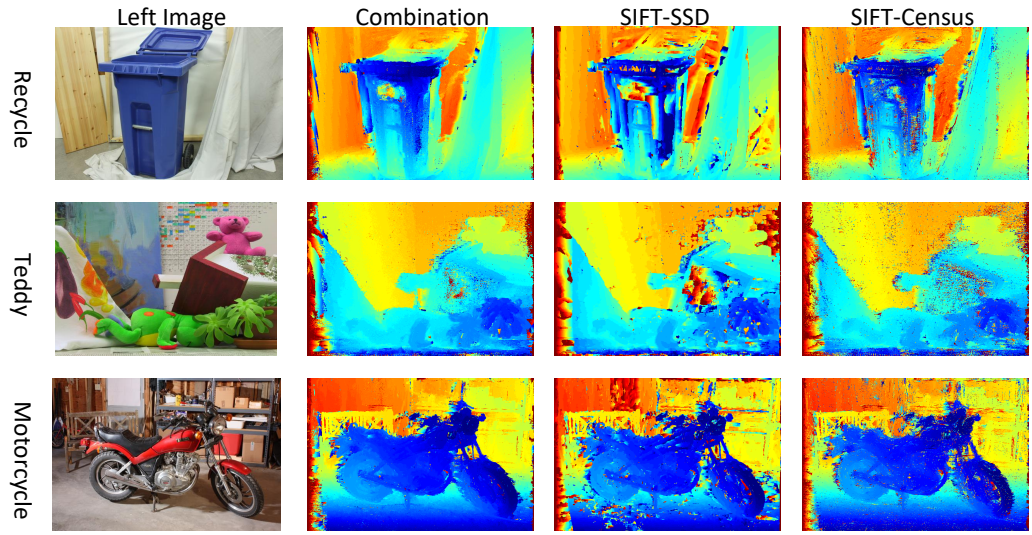


Figure 3.5: Combined example. After combination, noise in the textureless areas is reduced, however some mismatched areas still remain.

On textureless areas, the combination of SIFT-Census and SIFT-SSD also achieves a notable improvement. One of the examples is Teddy, which has large areas of a textureless wall, and SIFT-Census has reduced some mismatched points in the textureless regions.

There are some remaining mismatched points on the textureless wall. Similarly, SIFT-SSD also has a large mismatched region on the book. After combination, noise on the textureless book is reduced, but some mismatching areas still remain.

3.4 Summary

In this chapter, an adaptive windows cost computation method has been proposed. After introducing the details of this approach, it was applied to regular Census and SSD, and experiments conducted on benchmark datasets. Experiments show that the adaptive matching windows method can boost the performance of both Census and SSD. Finally, a combination of SIFT-Census and SIFT-SSD was explored and results demonstrate improved performance over single SIFT-Census or SIFT-SSD alone.

Chapter 4

Few-Shot Stereo Matching

Pre-training a model on a simulated dataset and fine-tuning it on target datasets is a popular paradigm in training deep learning stereo matching models. Although this approach to domain adaptation can alleviate the problem of data paucity, it requires labels in the new domain, which is not always obtainable at low cost, especially in production environments. Therefore, methods to develop a generalisable model that requires less training data are urgently needed.

To enhance model robustness and generalisation capability, recurrent 3D convolutional networks (recurrent 3D CNN) may be utilised. It has to notice that the recurrent 3D convolutional networks [92] are different from our work with the same name. The main idea of recurrent 3D CNN is to transfer the aggregate step to the step-by-step selecting stage. This idea force the model to extract more generalized features by adding restrictions.

In This chapter, we propose one of our contributions: recurrent 3D convolutional networks, which achieve competitive results using the few shot setting. In addition, our recurrent 3D convolutional networks keep stable performance on data from the different domains without any fine-tuning and retaining.

Unlike common methods [15,48] which directly aggregate the 3D cost matrix, in this work the input of the recurrent 3D CNN is the output of the previous layer. This step-by-step method increases the complexity of the model to obtain the final result using unrelated features. Also, in each step the aggregation process is decomposed into two goals. The first goal is to select the correct cost value from the cost matrix. Because some challenging areas will cause the same cost value to appear at different disparity levels, the second goal is to select the correct disparity based on the selected cost value from the first goal. This reduces the errors at those positions due to multiple correct cost values.

In this chapter, the pipeline of our few shot stereo matching is introduced in Section 4.1. Firstly, the details of transferring the regular cost volume into our two-channel cost volume are proposed in Section 4.1.1. Secondly, in Section 4.1.2, we propose our few shot cost aggregation methods with three components: recurrent Structure 3D CNN, recurrent structure 3D block, Compression Layer. Thirdly, our confidence measurement is introduced in Section 4.2.

In the experiment section, we firstly show the result from KITTI 2012 dataset and Middlebury Stereo 2014 dataset in Section 4.3.4 and Section 4.3.5. After that, we submit our result to the Middlebury Stereo 2014 and KITTI 2015 dataset benchmark. The benchmark result is shown in Section 4.3.7 and Section 4.3.8. Next, the ablation experiments to verify each component in our few shot pipelines are in Section 4.3.9, and the fine-tuned result is in Section 4.3.10. In addition, four metrics: robustness, generalisation, and interpretability, are presented in Section 4.3.11, Section 4.3.12 and Section 4.3.13. Finally, our confidence measurement result is shown in Section 4.3.14.

4.1 Proposed Methods

In this section, the pipeline and details of the two main contributions are outlined. Our pipeline is shown in Figure 4.1.

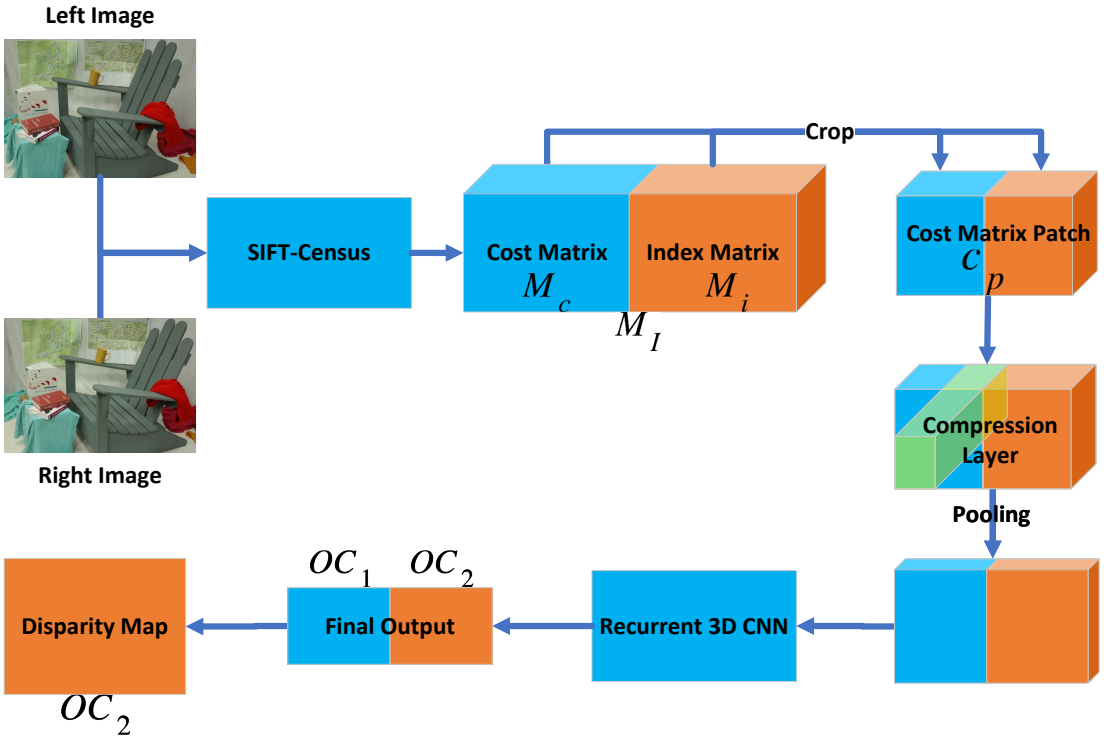


Figure 4.1: Our few shot pipeline follows the four steps pipeline of stereo matching.

In the first step, the matching cost computation method with adaptive windows proposed in Section 3.1 is used to obtain the 3D cost volume M_c. After M_c is computed, an index volume M_i with the same size as M_c is concatenated with the cost volume M_c to form a new two-channel 3D volume M_I. The third step is to decompose the cost volume M_I into a block consisting of two patches from different channels c_p. Then c_p is fed into a compression layer to adjust the size of the depth dimension. Finally, c_p is fed into arecurrent 3D CNN to obtain a two-channel output, where the second channel is the disparity map.

4.1.1 Cost Volume

To adapt different cost computation methods to the recurrent 3D CNN, two rules are proposed to build the cost volume M_c and input volume M_I. The cost volume M_c is obtained by inverting the raw cost volume M_c'. Given a raw cost volume M_c', the formula

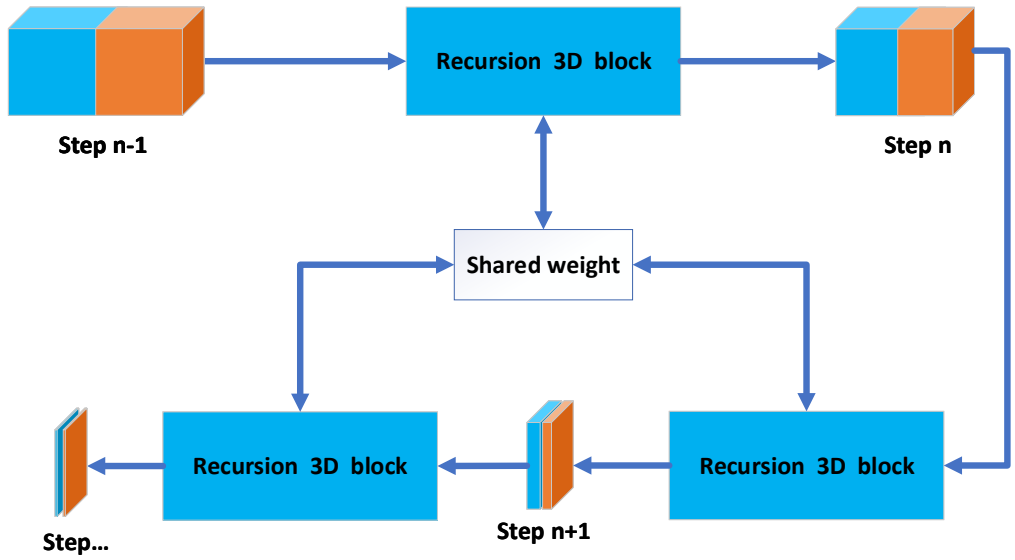


Figure 4.2: Recurrent 3D CNN uses the same block to handle inputs of different steps.

for building cost volume M_c is:

$$M_c = 1 - M_c'/\max(M_c') \tag{4.1}$$

where M_c' is the raw cost volume from any cost computation method and max() is the max value in the raw cost volume M_c'. $M_c'/\max(M_c')$ is a normalisation step and $1 - M_c'/\max(M_c')$ transforms the minimum value into the maximum value.

In order to retain the correct matches in 3D CNN with max pooling, the proposed rule for building a raw cost volume is different from the regular cost volume, where the index of min-cost value is the correct disparity. The proposed cost volume takes the index on the disparity value with the max value of the cost as the correct disparity. The value in the raw cost volume should be normalized to a value between 0 and 1.

After the raw cost volume is obtained, in order to avoid the ambiguity of the index of the corresponding value in the raw cost volume, an index volume M_i with the same shape

(D, W, H) as M_c' is created by following this rule:

$$\begin{aligned} M_i(d, x, y) = d, \forall d \in \{0, \ldots, D\}, \\ x \in \{0, \ldots, W\} \\ y \in \{0, \ldots, H\} \end{aligned} \tag{4.2}$$

In the final step, the raw cost volume and index volume are concatenated together to form a two-channel cost volume M_I.

$$M_I =< M_c, M_i > \tag{4.3}$$

where M_I is the input volume of our recurrent 3D CNN.

After cost volume M_I is obtained, the cost volume M_I of the left image is decomposed into different patches c_p. The sliding windows strategy is applied to the decomposition step. The two-channel cost volume M_I is divided into different patches c_p of size $(D, 128, 128)$ of both channels according to the hyper-parameters of step length and window shape.

4.1.2 Cost Aggregation

In the pipeline, the cost aggregation step consists of two-components: recurrent 3D CNN and compression layer. Before the cost volume patch c_p is fed into the recurrent 3D CNN block, one option is to apply a compression layer to reduce GPU memory consumption and speed up inference. Therefore, a compression layer is discussed in Section 4.1.2.3.

4.1.2.1 Recurrent Structure 3D CNN

The structure of the recurrent 3D CNN is shown in Figure 4.2. The recurrent 3D CNN relieves the overfitting problem by decomposing the aggregation method into different

steps. In each step, the same block is used to handle different inputs from previous steps, and each step relies on the output from the previous step. The structure for recurrent 3D CNNs is as follows:

$$\begin{aligned} O_i &= recurrent\_block(O_{i-1}), \\ O_{i-1} &= recurrent\_block(O_{i-2}), \\ &\vdots \\ O_1 &= recurrent\_block(input), \\ \forall i &\in \{0, \ldots, \log_{factor} D\} \end{aligned} \tag{4.4}$$

where O_i is the output of the recurrent 3D CNN block, $recurrent\_block$ is a recurrent 3D CNN block, the input is the cost volume patch c_p and the shape of the output from $step_i$ is $(D/factor, H, W)$, where $factor$ is the scaling factor used to reduce D. Usually, $factor$ should make sure that the size of each channel in the final output is $(1, 128, 128)$.

A recurrent 3D CNN may be used to avoid overfitting, instead of just one CNN. Although dividing the cost volume into different patches can alleviate the overfitting problem to an extent, it still exists when using a small amount of training data. The main reason for this is that the neural network uses some unrelated features from the training data to map the result directly. The recurrent 3D CNN uses different inputs from the previous outputs to address this problem. In addition, the recurrent 3D CNN has two goals. The first is to select the correct cost value on the cost volume. The second goal is to recover the disparity. The second goal relies on the first, which also relieves the overfitting problem.

The formula of the output of the recursion 3D CNN block is defined as follows:

$$O =< OC_1, OC_2 > \tag{4.5}$$

where the final output O is a two-channel 3D volume, OC_1 is the output from the first channel and OC_2 is the output from the second channel. The output from the first channel

is the selected cost volume, which is the remaining matching cost value of the first channel in the previous input, and the second channel is the index of the corresponding matching cost value of the first output channel.

The details of the recurrent 3D block are listed in Table 1. The base structure of the recurrent 3D CNN block is an encoder-decoder structure, where the layers only reduce the sizes of W and H. In the final layer of the recurrent 3D CNN block, the size of the convolution kernel and stride on the D dimension is $factor$.

Although fine-tuning is the most popular method to deploy a pre-trained stereo matching model in a new environment, there are cases where fine-tuning is not applicable when most of the data is unlabelled. Furthermore, fine-tuning is only helpful for input with one fine-tuned distribution, which means that the model needs to be fine-tuned again if the distribution of input data changes, leading to increased cost and reduced robustness. However, the excellent performance of fine-tuning on a target input domain shows that deep learning models can utilise some familiar cues to improve disparity results on different domains, and particularly cues from an input domain causing overfitting and reducing the robustness. Therefore, to reduce high training data requirement and enhance robustness, a possible solution is to use the critical and practical cues to recover the disparity from the cost matrix.

To this end, a recurrent 3D CNN is proposed in this work, which improves model robustness in two ways. Firstly, the final goal is decomposed into a few different goals handled by the same block. In each step, the input data is compressed by the same block. Therefore, the same block is used to handle various steps of input, enabling the network to extract more general features. Secondly, the disparity map is obtained via two steps that are dependent on each other. Specifically, the first step is to recover the corrected cost value from the cost matrix, and the second step is to recover the disparity value according to the cost value from the first step. The purpose of the second step is to reduce ambiguity. In the ideal cost matrix, the corrected disparity would only correspond to a single cost value.

However in practice, multiple cost values correspond to the disparity due to textureless areas or noise. In the experiments, the accuracy decreases by almost 15% when only using the cost value to recover the disparity,compared with the second channel of output from our recurrent 3D CNN.

In Section 4.3.9.1, an experiment is conducted based on the assumption that if a model extracts more general features and the test data is slightly different, the model with less accuracy decrease on the test data would extract more general features.

The behavior of the proposed recurrent block is similar to searching the indices of the optimal values on the first input channel to obtain the corresponding values in the second channel with the indices.

4.1.2.2 Recurrent Structure 3D Block

In this section, the details of the proposed recurrent structure 3D block are introduced. The basic structure of the recurrent structure 3D block is an encoder-decoder structure. The recurrent structure 3D block transfers 2D U-Net into the 3D version with skip connection. In the experiment, in order to speed up convergence and improve accuracy, an instance normalization layer is integrated into the block. The reason to use instance normalization instead of the more popular batch normalization is that the same block is used to handle the inputs of different steps whose distribution varies.

In the last layer of the block, a 3D convolution layer is added to reduce the depth dimension of the 3D output matrix. Different compress ratios are selected on the output layer according to the input 3D matrix shapes and layers. The first half of the block is the encoder part. In this part, the first 3D convolution layer followed by an activation function accepts a two-channel 3D matrix as input. After that, the instance normalization layer is used following the pooling layer, which only reduces the width and height of the 3D matrix.

In the decoder part, similar to 2D U-Net, the transport convolution layer is used to enlarge the width and height dimension of the 3D matrix. Following the transport convolution layer, a 3D convolution layer accepts the output from the previous encoder layer and the transport convolution layer to achieve the skip connection.

Detail of the recurrent structure 3D block are shown in Table 1.

4.1.2.3 Compression Layer

For adapting to different resolution images with maximum disparity values, a compression layer is utlised before passing the patch of the cost volume into the recurrent structure 3D CNN. The compression layer is defined as follows:

$$CP = P(K = (t_d/o_d, 1, 1), S = (t_d/o_d, 1, 1)) \tag{4.6}$$

where P is the 3D max pooling layer, K is the shape of the pooling kernel, S is the stride, t_d is the maximum disparity of the output of the compression layer and o_d is the compression ratio.

When deploying the model on a low-resolution image, the value of o_d needs to be adjusted to scale the input into the shape of the training data. Because the cost volume uses the index of the maximum value from the D dimensions as the correct matching index, a simple resizing method can directly apply pooling on the D dimensions. Simple resizing can obtain the same result without any additional downsampling methods such as those used in a Gaussian pyramid. Experiments show that employing the compression layer and deploying a small image model on larger images only leads to a small drop in accuracy. Also, directly using the compression layer in the training process can reduce the training time without losing accuracy.

In order to retain the correct output from the model with the compression layer, the final output OC_2 of the original model trained without the compression layer should be scaled up. The scale-up formula of OC_2' is given by:

$$OC_2' = OC_2 * o_d \tag{4.7}$$

where the OC_2' is the scaled new result.

4.1.3 Loss Function

Guidance by the whole ground-truth (GT) disparity maps is imposed on the predicted disparity maps and the corresponding label disparity cost value in the first channel of the input cost volume. In addition, the gradient loss between the output and the ground-truth is added to the loss to obtain a better result on object edges. Therefore, the proposed loss function consists of the cost value, the disparity and the gradient. In this work, these three components (cost value, disparity, and gradient) are all supervised by L1 loss. The proposed loss function is:

$$\begin{aligned} CV_l(x,y) &= M_c(x,y,GT(x,y)) \\ Grad_{\text{gt}} &= Sobel(GT) \\ Grad_{\text{output}} &= Sobel(OC_2) \\ Loss &= |OC_1 - CV_l| + |OC_2 - GT| + |Grad_{\text{gt}} - Grad_{\text{output}}| \end{aligned} \tag{4.8}$$

where CV_l is the volume on cost value selected from the cost volume M_c based on the ground-truth disparity GT. OC_1 is the first channel of the output O, OC_2 is the second channel of the output O and $Sobel$ is the Sobel operator.

The Sobel operator is applied to the ground-truth disparity map and the output disparity map to obtain the gradient map. In experimental tests, adding the gradient as supervised information can enhance model performance on edges and reduce the noise in flattened areas.

4.2 Confidence Measurement

Although cost aggregation can remove a significant number of errors in the disparity map, there are still some errors that the aggregation step cannot remove. Therefore, some confidence measurement methods are employed to detect the points in the disparity map that have a higher risk of being an outlier. As mentioned in Chapter 2, the most recent approach to achieve this directly takes the cost matrix as the network input and creates the confidence map as output. These supervised learning-based confidence measurement methods perform well but also incur additional computation burdens.

In this work, a new confidence measurement is introduced that does not use any new networks. The proposed approach achieved 75% accuracy on the confidence measurement task on the Middlebury Stereo 2014 datasets without additional training with the same model from Section 4.1.2.

4.2.1 Confidence Estimation

Although the proposed recurrent 3D CNN reduces a large number of errors on the final result, there are still some remaining. Some of the remaining errors are caused by over smoothing and occlusion. Therefore, a confidence measurement method is proposed that does not require additional learning nor increase in computational resources. The basic idea is to compare the cost value from the first channel with the cost value selected based on the second channel output. There is a restriction that the good areas should have only a single cost value in the cost matrix corresponding to the valid matching point under ideal conditions. If there is another cost value with the same value as the correct matching point, this pixel is likely located in challenging areas. In addition, textureless areas should correspond to flattened areas near the valid matching point in the cost matrix, which means that the error caused by the textureless regions should have the same cost value. Because the model can detect and handle textureless areas, the remaining error

always has a different cost value between the first and second channels. Therefore, when measuring the difference between the cost values from the first channel output and the second channel output, larger differences indicate lower confidence. Details of the proposed confidence measurement method are shown in Figure 4.3.

The first step of the confidence measurement method is to take the two-channel output from the recurrent 3D CNN as input. The second step is to transfer the second channel to the disparity map by turning floating point numbers into integer numbers. Then, the integer disparity map is used as the index to locate the corresponding cost value. Finally, the corresponding cost value from the second output channel is compared with the first channel. As discussed, larger differences between these two channels mean lower confidence.

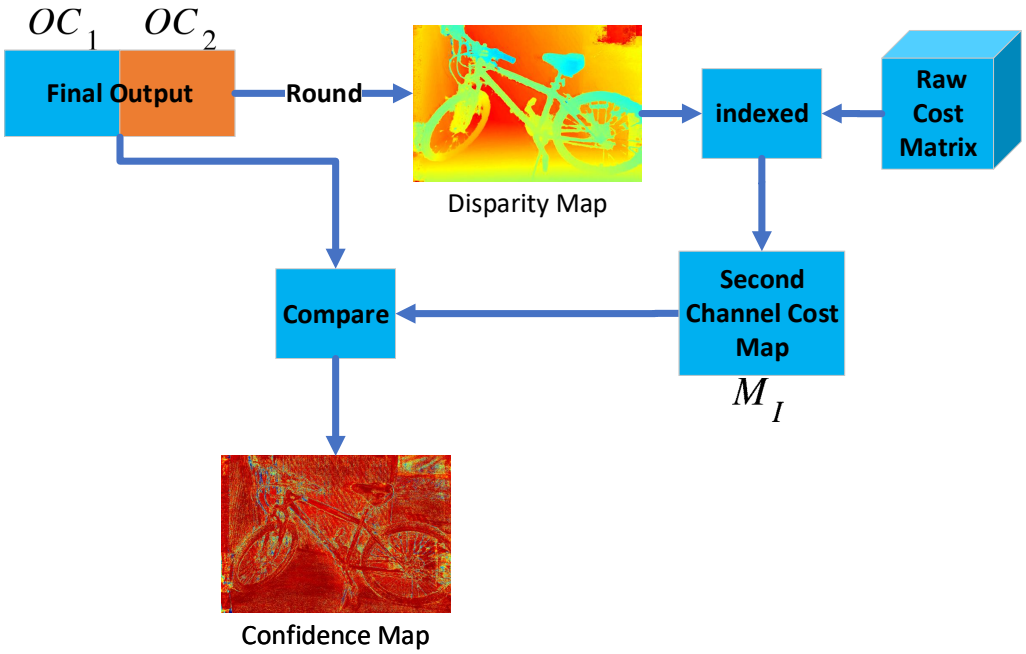


Figure 4.3: Details of the confidence measurement method. The confidence map is obtained by comparing the two disparity maps from the first channel output and indexed from the raw cost matrix.

4.3 Experiment

Three datasets are selected for training and testing the model. In order to test the ability to deal with cases of extremely small sized training set, no simulated datasets are used for per-training as is commonly done in other works.

Firstly, the 11-shot model, without any fine-tuning and retraining, was tested on the Middlebury Stereo 2014 datasets and KITTI 2015 dataset and is compared with SOTA models which use simulated datasets for pre-training and target datasets for finetuning.

First and foremost, the proposed recurrent structured 3D CNN was compared with the regular 3D U-Net from 1 to 11 training images to verify the efficacy of the proposed method on the few-shot setup. Secondly, ablation experiments are performed to show the impact of each component of the model. Next, the model with compression layers is compared to one without compression layers. The influence of different compression ratios is also listed. Finally, the result of a model trained on Middlebury Stereo 2014 datasets and fine-tuned on three images from the KITTI 2012 dataset is shown.

4.3.1 Datasets and Evaluation Metrics

The model was trained with no more than 11 images from the Middlebury Stereo 2014 datasets [1] and tested on the KITTI 2012 and KITTI 2015 datasets [93] without any fine-tuning and retraining. The Middlebury Stereo 2014 datasets are high accuracy datasets on indoor scenarios. Both KITTI 2012 and KITTI 2015 datasets are real-world datasets in outdoor scenarios, where only sparse ground truths are provided.

4.3.2 Implementation Details

The model was implemented in PyTorch and utilized Adam as the optimizer. Eleven half resolution images on the Middlebury Stereo 2014 datasets were selected as the training set. During testing, when the maximum disparity on the 3D cost volume is over 256, a compression layer was applied on the D-dimensions before being fed into the aggregation layer. For all the datasets, the SIFT-Census hyperparameter s_b was 7 and f was 3 in Eq. (3.3).

4.3.3 Results and Analysis

First, the results of the model trained on 11 images from the Middlebury Stereo 2014 datasets and directly tested on the KITTI 2012 dataset without any fine-tuning or re-training are shown. As the results of the KITTI 2012 dataset, even though the model was trained on indoor scenes, it can still perform with high accuracy on the outdoor dataset. To validate the effectiveness of each component proposed, controlled experiments were conducted on the test sets of the Middlebury Stereo 2014 and the KITTI 2012 validation datasets. Removal of the proposed recurrent structure 3D CNN led to a significant performance drop with the minimal training set. Compared with other models pre-trained on a large dataset, the training set used was very much smaller.

Training set size	Middlebury (recursion)	Middlebury (no recursion)	KITTI (recursion)	KITTI (no recursion)
1	68.93	44.56	65.80	58.73
2	71.08	54.81	74.02	72.36
8	78.01	70.24	78.88	77.65
11	81.25	69.88	92.82	85.96

Table 4.1: Percentage of pixel error less than 2.0 on the Middlebury Stereo 2014 dataset and less than 3.0 on the KITTI 2012 dataset, with recursion and non-recursion structures.

4.3.4 Result on KITTI 2012 Dataset

To test its generalisation capability, the model trained on the Middlebury Stereo 2014 dataset was directly tested on the KITTI 2012 dataset without a left-right consistency check and any post-processing. The accuracies achieved are shown in Table 4.1 and some examples are shown in Figure 4.5, where the first 3 columns are the best examples and the last 3 columns are the worst. The major errors from the model on the KITTI 2012 dataset come from the sky areas because the model is only trained on indoor scenes.

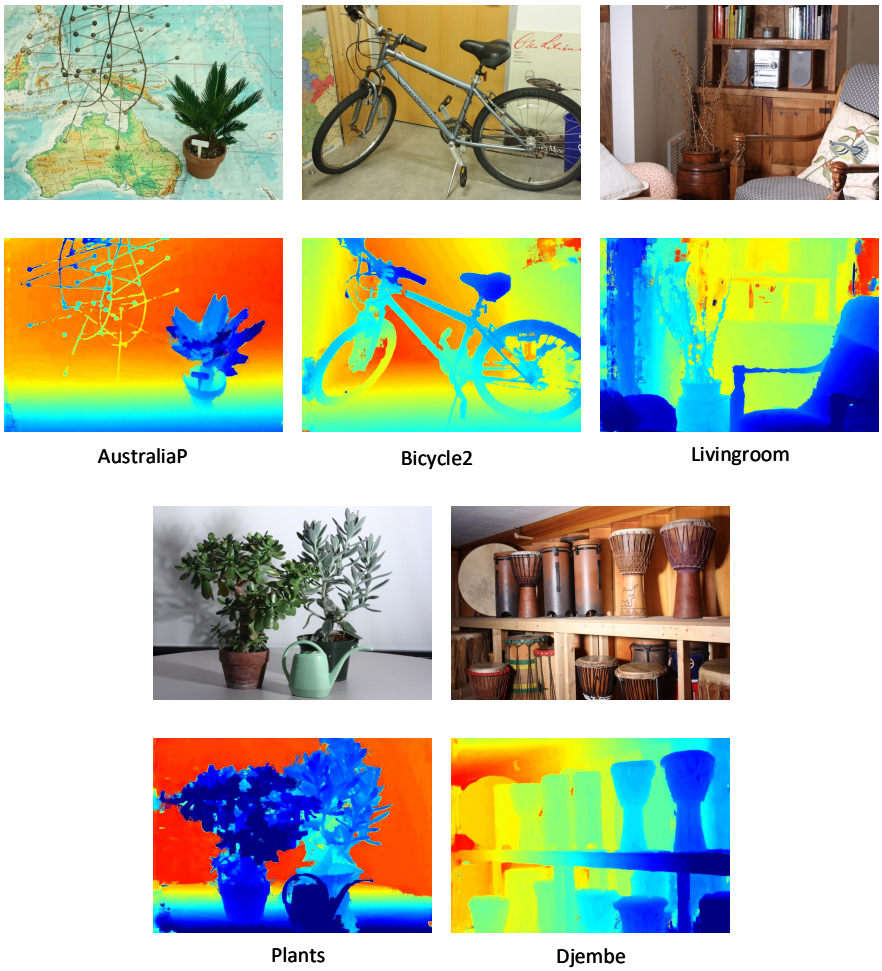


Figure 4.4: Results on the Middlebury Stereo 2014 test set based on the proposed 11-shot model. No ground-truth was provided by the Middlebury Stereo 2014 test set on benchmark.

Methods	bad-4.0-error	Rank	bad-2.0-error	Rank
iResNet [40]	22.1	1	31.7	1
R3DCNN (Proposed)	**24.2**	**2**	**38.7**	**3**
AANet	25.8	3	31.8	2
SPPSMNet [94]	27.5	4	46.8	4
PSMNet	29.2	5	47.2	5

Table 4.2: Leaderboard of Middlebury Stereo 2014 dataset.

4.3.5 Results on Middlebury Stereo 2014 Dataset

The results from the test set of the Middlebury Stereo 2014 dataset are shown in Figure 4.4, where the model obtains a competitive result even when using a smaller training dataset.

Error	**D1-bg (%)**	**D1-fg (%)**	**D1-all (%)**
All / All	8.00	26.91	11.15
All / Est	8.00	26.91	11.15
Noc / All	7.20	25.14	10.16
Noc / Est	7.20	25.14	10.16

Table 4.3: Leaderboard of KITTI 2015.

4.3.6 Benchmark Results

For benchmarking, the proposed model trained on 11 images in the Middlebury Stereo 2014 dataset without fine-tuning, was submitted to the Middlebury Stereo 2014 and KITTI 2015 leaderboards, and the details are listed in Tables 4.2 and 4.3.

4.3.7 Benchmark Results from Middlebury Stereo 2014

Details of each test set from Middlebury Stereo 2014 on bad-4.0-error metric are in Table 4.4.

In Table 4.2, the “bad-4.0-error” and “bad-2.0-error” are error pixels whose errors are

Image name	Accuracy	Image name	Accuracy
Australia	16.9	CrusadeP	42.6
AustraliaP	7.83	Djembe	6.87
Bicycle2	10.7	DjembeL	27.7
Classroom2	28.8	Hoops	36.6
Classroom2E	42.7	Livingroom	23.4
Computer	20.2	Newkuba	24.9
Crusade	45.0	Plants	15.2
Staircase	30.4		

Table 4.4: Leaderboard from Middlebury Stereo 2014 on bad-4.0-error.

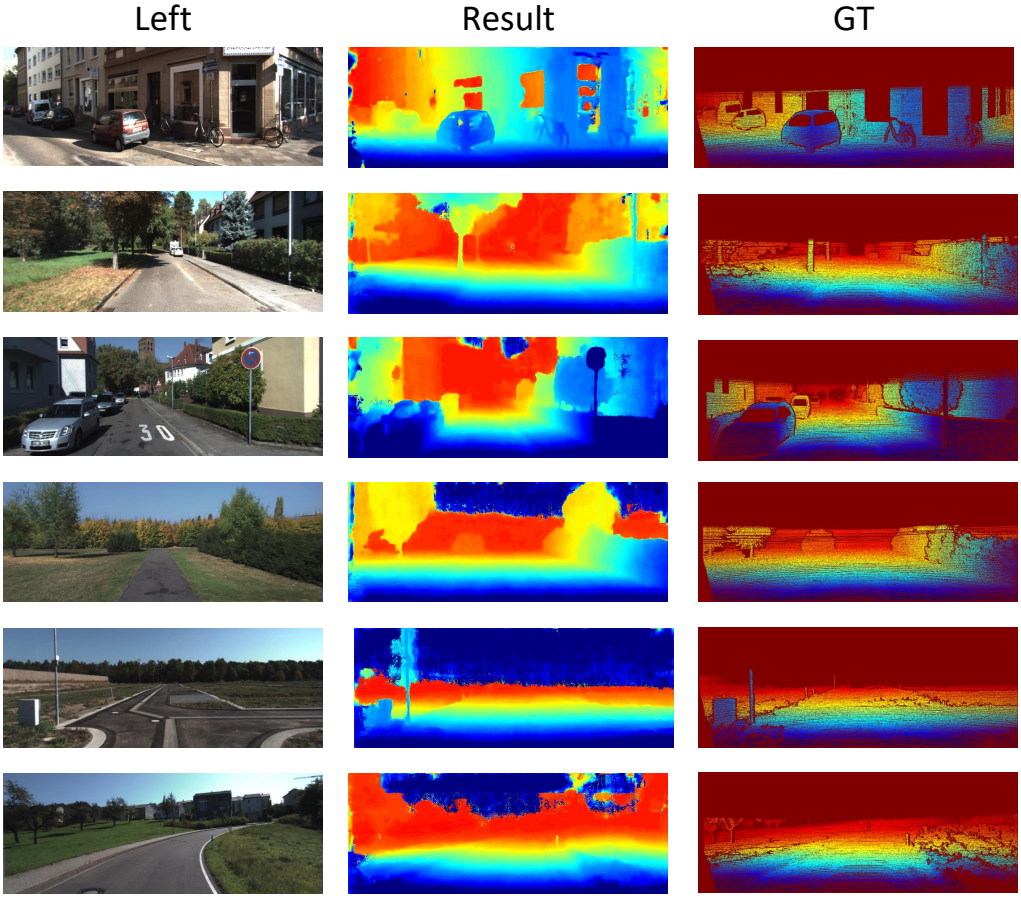


Figure 4.5: KITTI 2012 examples. The main errors are in the sky areas, which are out of the disparity range of the training set.

greater than 4.0 and 2.0 respectively.

In Table 4.3, D1 is the percentage of stereo disparity outliers in the left images. In particular, D1-bg is the percentage of outliers averaged only over background regions and D1-fg is the percentage of outliers averaged only over foreground regions. D1-all is the percentage of outliers averaged over all ground truth pixels. Also, in the first column, "All" means that all the labelled pixels are estimated, and "Noc" means that only the pixels not located in the occluded areas are estimated.

The model submitted to the Middlebury Stereo 2014 benchmark was only trained with 11 images from the Middlebury Stereo 2014 training set, without pre-training on any simulated datasets. However, most of the deep learning models from the leader-board use a large simulated dataset for pre-training, including the listed models such as AANet, GA-Net and PSMNet. According to their work, AANet was pretrained on the Scene Flow dataset, containing more than 39,000 synthetic sequence stereo frames of 960x540 pixel resolution. Similarly, GA-Net and PSMNet both used the SceneFlow dataset to pretrain their models.

As Table 4.2 shows, the proposed few-shot model still achieves a competitive result even though it is compared with non-few-shot models directly on the Middlebury Stereo 2014 test set. The proposed model achieves a "bad-4.0-error" of 24.2, which slightly outperforms AANet [16]. In addition, the proposed model is slightly better than PSMNet with "bad-4.0-error" 5 and outperforms SPPSMNet with "bad-4.0-error" 3.3. It is also noteworthy that the proposed model under-performs iResNet with "bad-4.0-error" 2.1.

4.3.8 Benchmark Results from KITTI Datasets

The proposed model achieves almost 90% accuracy on all labelled pixels. The major reason for errors on the KITTI dataset is the disparity range. Major error areas are those objects with small disparity values, which are smaller than the smallest disparity value from the Middlebury Stereo 2014 dataset. This is because the model submitted to the

KITTI leaderboard was only trained on the Middlebury Stereo 2014 dataset without any fine-tuning. In Section 4.3.10, the accuracy of the model is significantly improved after using three images from the KITTI 2012 dataset for fine-tuning.

4.3.9 Ablation Experiments

In order to verify the effectiveness of the various model components, some ablation experiments were run on the recurrent 3D CNN and the compression layer, which are are discussed in this section.

4.3.9.1 Comparison of Non-recurrent vs Recurrent Structure

In order to verify the effectiveness of the proposed recurrent structure in reducing overfitting, two models were trained with the same base 3D CNN block structure, with one model using the recurrent structure and the other not.

To obtain the disparity map from the non-recurrent model, Softmax was used as the activation function in the last layer of the non-recurrent model. Therefore, given a pixel on the output matrix of the non-recurrent mode, the probability value of the correct disparity of this pixel will be the highest. Ideally, the value of the correct disparity on the disparity dimension will be 1, and 0 otherwise.

As shown in Table 4.1, when the number of training images is extremely small, the proposed recurrent structure can alleviate the overfitting problem. For example, when using an image for training, the difference of "bad-2.0-error" between the recurrent and non-recurrent model on the Middlebury Stereo 2014 dataset is 24.367. After adding one more image for training, the difference is reduced to 16.27. With the number of training images growing, the accuracy difference between the recurrent and non-recurrent models decreases. For instance, the difference of training on 8 and 11 images is reduced to 7.77

and 11.371 respectively.

As for the KITTI dataset, the accuracy on it is generally higher than on the Middlebury Stereo 2014 dataset with the same number of training images. Comparing the proposed recurrent model with the non-recurrent model on the KITTI dataset, the former is better than the latter for all numbers of training images. The difference decreases the number of images for training is increased. For example, the difference is quite large when a single image is used for training, but the difference is significantly reduced if there is more than one training image.

The differences in the results on the KITTI dataset and the Middlebury Stereo 2014 dataset are mainly because

1. the scenes in the KITTI dataset are simpler than the Middlebury Stereo 2014 dataset. In the KITTI dataset, roads occupy the larger portion of the images, which are easilyhandled by the network
2. the labels of the KITTI dataset were collected using LiDAR, resulting in sparse ground truth which reduces the difference between the non-recurrent and recurrent models.

4.3.9.2 Compression Layer vs Non-Compression Layer

To probe the influence of the compression layer, a model was trained without a compression layer, using 11 quarter resolution images from the Middlebury Stereo 2014 dataset and tested on half-resolution images with the compression layer.

In the experiment, the model trained without the compression layer has an accuracy of 91.07%, which is nearly 5.17% higher than the model trained directly on half-resolution images without compression and directly tested with compression.

4.3.10 Fine Tuned Results

To show the results after fine-tuning on the KITTI dataset, eight images from the KITTI 2012 dataset were used for fine-tuning and trained for 20 epochs, then submitted to the leaderboard of KITTI 2015. The results are shown in Table 4.5.

Error	D1-bg (%)	D1-fg (%)	D1-all (%)
All / All	4.97	12.90	6.29
All / Est	4.97	12.90	6.29
Noc / All	4.58	11.70	5.75
Noc / Est	4.58	11.70	5.75

Table 4.5: Leaderboard from KITTI 2015 after fine-tuning.

In Table 4.5 a slight improvement is observed after fine-tuning. On the D1-all metric, there is a 5.53% improvement. As for the D1-bg and D1-fg metrics, there is 3.65% and 13.86% improvements. The major improvements of the fine-tuning focus on the foreground.

4.3.11 Robustness

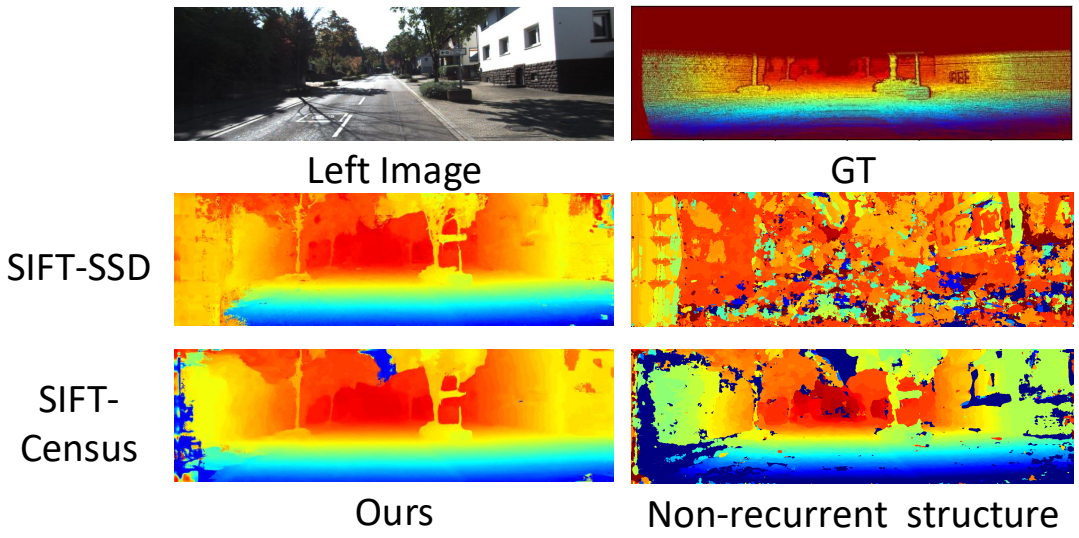


Figure 4.6: Results from the non-recurrent and recurrent model tested with different computation methods.

Before introducing the results of the robustness experiments, the experimental setting is presented. The recurrent structure 3D CNN and non-recurrent structure 3D CNN trained on SIFT-Census are directly tested on SIFT-SSD without fine-tuning and retraining. While testing SIFT-SSD, the assumption is that a robustness cost aggregation model can also perform well on other cost computation methods other than the cost computation method used for training. SIFT-Census and SIFT-SSD have the same maximum cost value and cost computation method in the experiments. In addition, the accuracy difference between SIFT-Census and SIFT-SSD in Section 3.3.4 and Section 3.3.5 is small. Therefore, a robustness model trained on SIFT-Census only suffers from a small accuracy decrease when it is directly tested on SIFT-SSD. In these experiments, SIFT-SSD is normalized to the range of 0 to 1 for each cost value.

Another experiment was designed where both the non-recurrent and recurrent models were trained with SIFT-Census on the Middlebury Stereo 2014 dataset and directly tested on the KITTI 2012 dataset with SIFT-SSD. As shown in Figure 4.6, the results from the recurrent structure model tested on SIFT-SSD are almost identical to that of SIFT-Census, which has the same cost computation on the training data. In contrast, the non-recurrent model collapses on different computation methods of the training set.

Through these experiments, it was found that the recurrent structure model extracts more generalized features to recover the cost matrix disparity. The experiments of the SIFT-SSD show that 3D CNN with non-recurrent structure is easy to overfit special matching cost curves from the cost computation method used for training. The overall accuracy of the raw cost matrix of SIFT-SSD used for testing is higher than that of SIFT-Census, however SIFT-SSD leads to more noise than SIFT-Census. Therefore, the experiments of SIFT-SSD show that the recurrent structure model does not overfit any curve shape used for training.

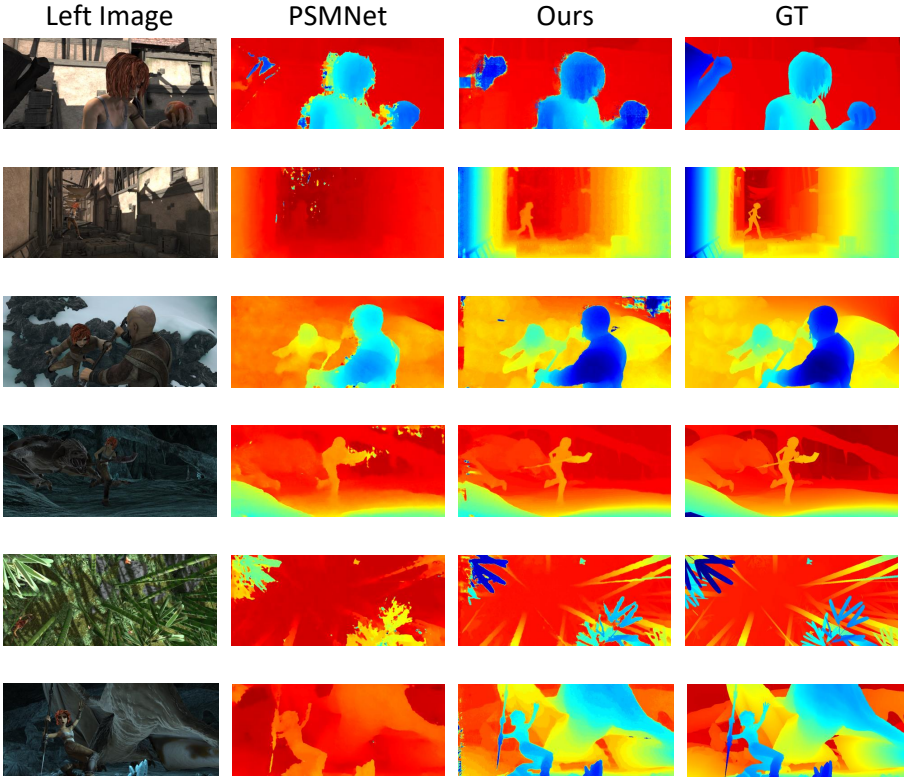


Figure 4.7: Results from the MPI Sintel dataset. The proposed 11-shot model trained on the Middlebury Stereo 2014 dataset was directly tested on the MPI Sintel dataset.

4.3.12 Generalisation

Many stereo matching models based on deep learning do not perform satisfactorily when the domain is considerably different. In particular, the difference between simulated and real scenes is the main factor leading to the domain adaptation problem.

In addition, the main reason for the overfitting problem is that the neural network uses some unrelated features from the training data to map the result directly. Therefore, there is an assumption that models with better robustness and generalisation capability can perform better both on real and simulated datasets.

Another experiment was conducted where the proposed 11-shot model trained on the

Middlebury Stereo 2014 dataset was directly tested on the MPI Sintel dataset [76], a simulated dataset created using scenes from a 3D movie. The results from this experiment are shown in Figure 4.7. Evidently, the proposed model achieves high accuracy on the MPI Sintel dataset. However, there are still errors that may be caused by the disparity range. For example, objects far away from the cameras, such as objects near the sky areas, have disappeared. As for those pixels within the disparity range, most of them are correct. Given these observations, it may be conclude that the proposed model has a better generalisation ability than other models, and it performs more stably in new domains. Additionally, the assumption that recurrent structures enable extraction of more generalized features is also justified.

Finally, other methods using our few shot settings are compared with our method. In this section, PSMNet [15], a representative SOTA model, is trained with 11 images from the Middlebury Stereo 2014 dataset and directly tested on the MPI Sintel dataset. Our model gains 83% accuracy on the "bad-1.0-error" with the same setting, which has a performance improvement of 19% over PSMNet . In order to show more intuitive results, some of the results from PSMNet are listed in Figure 4.7.

4.3.13 Interpretability

In this section, intermediate results from the trained model are analysed. Specifically, the feature map from the second last layer is selected. The test set from the Middlebury Stereo 2014 datasets is used. Because the intermediate results are a two-channel 3D matrix, the second channel (disparity channel) of the 3D matrix is sliced into different 2D channels according to depth. After the 2D intermediate results are obtained, some of the Values less than 0 in these feature maps. Therefore, all the 2D feature maps are normalised. Finally, all the feature maps are transformed into grayscale maps, where lighter pixels indicate higher values. There are three feature maps for each image. Some examples are shown in Figure 4.8.

Figure 4.8: (a) left image (b) right image (c) first feature map (d) second feature map (e) third feature map (f) final output of 11-shot model.

Observing the three feature maps in Figure 4.8, in general before the feature maps are combined to obtain the final result, it is clear that different feature maps respond differently

to different parts. The first feature map shows the disparity trend, which gives the larger disparity point a higher response value. The second feature map shows confidence. Unlike the first feature map, the value of this feature map does not change with distance but gives a higher response value to the easy-to-match areas. The third feature map distinguishes the foreground and background, and the value of the foreground areas is significantly larger than that of the background.

To support these conclusion, multiple examples are shown and analysed. Firstly, the AustraliaP image is analysed. In the first feature map from the AustraliaP image, most of the values on the first feature map have the same trend as the final disparity map. The closer a point is to the camera, the larger its value. In the second feature map, the values do not change with depth. As the grayscale maps show, some areas have a dark colour, which means that the value is small. Comparing the left image with the right image, most of those dark areas correspond to occluded areas, which is one of the challenges for stereo matching. For example, the dark areas near the iron mesh object and the flower pot correspond to occluded areas. In the third feature map, the model gives the foreground areas a large value. In the third feature map of the AustraliaP image, three main areas receive a large value: the iron mesh object, flowerpot and the background closer to the camera. The iron mesh object and flower pot can be seen as the foreground areas. For the background, only those areas closer to the camera are assigned correct values.

On the Bicycle2 image, the first and third feature maps exhibit similar behaviours to AustraliaP. The text on the second feature map has a larger value than surrounding points. That is because the word is surrounded by textureless areas, where it is harder to obtain correct matches compared to text. In addition, by comparing Djembe and DjembeL images, the second feature map focusses on confidence measurement. In the second feature map of Djembe, except for the occluded areas, the value of other areas is higher. On DjembeL, due to the different lighting conditions, some mismatching occurs. The mismatches correspond to the dark areas in the second feature map of DjembeL.

To summarise, through comparison with the test set of the Middlebury Stereo 2014 datasets, it is clear that the proposed model decomposes the final result into different features. Before the final step of gathering all the intermediate results, the experiment shows that the intermediate result clearly responds more to some cues than others.

4.3.14 Confidence Measurement

In this section, the propsoed method is tested on the Middlebury Stereo 2014 dataset and no more than 11 images are used for training.

The confidence measurement method is tested. In order to compare the confidence map with the error map, the point which has a value larger than a 2 times of the mean on the confidence map is set to 1.

The confidence map from the Middlebury Stereo 2014 test set is now demonstrated. In the confidence map, brighter pixels mean lower confidences. As Figure 4.9 shows, most of the lowest confidence areas correspond to wrong areas on the disparity map. To obtain the accuracy of the confidence map, the error map is compared with the confidence map on two test set images from the Middlebury Stereo 2014: DjembeL and Pipes, which achieve around 75% accuracy. During the test phase, the top 20% large values in the confidence map are considered as the low confidence points. If the difference between the GT and the final disparity map is larger than 2, it will be considered as an error point in the error map.

By further inspecting the error map, it was found that one of the major errors in the model arises from edges. For example, the error map of Bicycle2 shows that the model achieves high accuracy on flat areas of the foreground and background except for the edges of the bicycle. Obtaining a high accuracy disparity value on object edges is a challenging part of the stereo matching problem. The propsoed confidence measurement method, which can identify most of the mismatching edges, can overcome this problem.

Figure 4.9: Confidence measurement example, dark points are error / low confidence points.

Another cause of mismatches is from the occluding areas. There are some popular methods to detect occluding areas, such as the left-right consistency check. The proposed confidence measurement method, as an alternative tool, can also point out the occluded areas. Two examples Pipes and ArtL are shown in Figure 4.9. The major advantages of the proposed confidence measurement approach are the following:

1. Compared with the left-right consistency checking methods, the proposed method does not need to infer the right image repeatedly, therefore computational cost is reduced

2. the left-right consistency checking methods do not work well with most deep learning models. The cost matrix based on the right image is slightly different from the cost matrix used for training, which causes the accuracy of the model trained on the cost matrix of the left image to drop when directly applied to the cost matrix of the right image. The decreasing accuracy on the cost matrix of the right image prevents the traditional left-right consistency checks from being used on deep learning-based models.

The remaining mismatches are mainly caused by small objects or oversmoothing, which are also challenging issues in many stereo-matching models. The proposed confidence measurement method also shows oversmoothing areas. For example, on the error map of AustraliaP, most of the error comes from the thin and small parts of the iron mesh object. Therefore, those areas are marked as low confidence areas on the confidence map.

Last but not the least, differing illumination also leads to mismatches. An example of large area mismatching caused by lighting changes is the DjembeL image. The confidence map of DjembeL shows three mismatched areas caused by different lighting conditions. The error map of DjembeL also having three corresponded low confidence areas. Clearly, the proposed confidence measurement method can detect the errors caused by changes in illumination.

4.4 Remarks

This chapter introduced a few-shot stereo matching method based on a proposed recurrent structured 3D network. A method to transfer the regular cost matrix into the adapted in-

put format of the propsoed model was presented. Then, the recurrent structured 3D CNN, a core innovation of this work, was described in detail. In order to reduce the memory requirement and enhance efficiency, a compression layer is added to reduce unnecessary information in the input data. A gradient guided loss function was also introduced as part of the method.

The experiments show the performance of the proposed method in a few-shot setting. Competitive results can be obtained on the Middlebury Stereo 2014 and KITTI 2015 test sets. In addition, results from the benchmark also demonstrate that the proposed few-shot learning model without any fine-tuning achieves almost the same level of performance as that of recent deep learning models that are pre-trained using large simulated datasets and fine-tuned on the target datasets.

Moreover, to verify the efficacy of each component of the model, ablation studies were conducted. We find that the recurrent structure leads to better performance on small training sets, and while the compression layer just slightly reduces the model accuracy, it can also significantly reduce memory usage and computational cost.

Model robustness and interpretability were also discussed. Experiments show that the proposed model can adapt to different cost computation methods and domains without requiring retraining or fine-tuning, and the model is able to decompose the final result into different features.

FInally, a novel confidence measurement method was introduced. Without requiring additional training, the propsoed confidence measurement method can still indicate most of the mismatching areas.

Chapter 5

Conclusion

This thesis aims to reduce reliance on large amounts of training data of current deep learning based stereo matching methods. In this thesis, we makes novel contributions to adaptive cost computation, reduction of memory and time requirements, few-shot cost aggregation and confidence measurement.

According to our experiment result, the propsoed model was trained with 11 real indoor images on the benchmark and obtained the same accuracy level compared to non- few-shot models. In addition, the propsoed aggregation model can adapt to different cost computation methods, scenes (indoor / outdoor) and domains (simulated / real) without fine-tuning and retraining.

In Chapter 1, the motivation and challenges are addressed. After that, some recent works, including the traditional algorithm and deep learning methods, are reviewed in Chapter 2. Our first contribution, adaptive cost computation, is proposed in Chapter 3. The other contributions, few-shot cost aggregation, confidence measurement, reduction of memory and time requirements, are introduced in Chapter 4. Finally, we conclude our work and export the potential improvement methods in Chapter 5.

5.1 Thesis Contributions

5.1.1 Adaptation Cost Computation

The primary purpose of adaptive? cost computation is to find the right window size automatically. Generally, a small window has less smoothing effects with better retaining of object detail, but incurs mismatches on textureless areas. Opposite of small windows, large windows reduce noise but cause oversmoothing problems. Textureless and richly textured areas often exist in the same image. The proposed adaptive cost computation method detects the textures and richly textured areas to automatically adjust window sizes using SIFT cues. Details of the method were presented in Chapter 3.

5.1.2 Few-shot Cost Aggregation

Overfitting and domain shift are two main barriers to the applicability of SOTA deep learning models. A new structure of the 3D convolutional network, called recurrent 3D convolutional network (recurrent 3D CNN), was introduced which reduces the training set size. In addition, the recurrent 3D CNN shows strong domain adaptation ability. In experiments, the recurrent 3D CNN trained on a few real indoor data sets could stably generalize to outdoor and simulated datasets without fine-tuning. Details of the recurrent 3D CNN are described Chapter 4.

5.1.3 Reducing Memory and Time Requirement

Reducing memory consumption and enhancing inference speed are two main research goals in deep learning-based stereo matching. In this thesis, a simple compression method combined with the proposed robust model obtained competitive results. Details of the proposed compression method are in Chapter 4.

5.1.4 Confidence Measurement

Measurement of low confidence areas provides information for refinement. In addition, the ideal confidence measurement methods should not increase computational requirements. In Section 4.2, a confidence measurement method that does not require additional training and inference is introduced.

5.2 Limitations and Future Work

Although the performance of the proposed model is about the same as that of SOTA deep learning stereo matching models, there is still much capacity for improvement. Three main extensions are possible in future research: reducing training data size, improving compression ratio and improving accuracy while employing more complex structures.

5.2.1 Reducing Training Data Size

In experiments, training data greatly influences the model performance. Disparity range and object types are two main factors in stereo matching experiments. On disparity range, deep learning models only produce correct disparity values within the disparity range of the training set. Although the proposed model achieves competitive results using a limited number of training data samples, the image scenes still influence model performance during one-shot learning. For example, using an image only containing flattened areas as the training setleads to oversmoothing of small objects in the test set. Therefore, using a training image consisting of flattened areas as well as detailed areas and within a wide disparity range is likely to improve the performance with lesser training data.

5.2.2 Improving the Compression Ratio

In experiments, directly using pooling as a compression method significantly reduces memory and time consumption without sacrificing accuracy. However, more efficient compression methods may be applied to cost volume, because cost volume only contains one correct disparity value.

5.2.3 Improving Accuracy Using Complex Structures

The basic recurrent block uses a simple encoder-decoder structure network in the proposed model. More complex structures may improve model accuracy. One of the potential structures to explore is the residual connection between each step of the recurrent 3D convolutional network. Another potential structure is dilated 3D convolution, which can enlarge receptive fields and reduce computational resources.

5.3 Concluding Remarks

This thesis aims to solve the issues preventing the deep learning based stereo matching methods applied to the production environment: limited high quality labeled data, generalisation, and robustness. Our model achieves the competitive result without relying on the simulated dataset. In addition, our method can adapt the data from different domains without retraining and fine tuning. Our few shot aggregation methods can adapt different cost computation methods. This makes our aggregation method can be plugged and played in the production environment.

Appendix

Supplementary Materials

Input shape ($C * D * H * W$)	Layer	Output shape ($C * D * H * W$)
2*128*128*128	Conv3D (3,3,3), $1F$, stride=1, padding=1	$1F$*D*128*128
$1F$*128*128*128	Pooling (1,2,2), stride (1,2,2), padding=0	$1F$*D*64*64
$1F$*128*64*64	Conv3D (3,3,3), $2F$, stride=1, padding=1	$2F$*D*64*64
$2F$*128*64*64	Pooling (1,2,2), stride (1,2,2), padding=0	$2F$*D*32*32
$2F$*128*32*32	Conv3D (3,3,3), $4F$, stride=1, padding=1	$4F$*D*32*32

$4F$*128*32*32	Pooling (1,2,2), stride (1,2,2), padding=0	$4F$*D*16*16
$4F$*128*16*16	Conv3D (3,3,3), $8F$, stride=1, padding=1	$8F$*D*16*16
$8F$*128*16*16	Pooling (1,2,2), stride (1,2,2), padding=0	$8F$*D*8*8
$8F$*128*8*8	Conv3D (3,3,3), $16F$, stride=1, padding=1	$16F$*D*8*8
$16F$*128*8*8	Pooling (1,2,2), stride (1,2,2), padding=0	$16F$*D*4*4
$16F$*128*4*4	Conv3D (3,3,3), $32F$, stride=1, padding=1	$32F$*D*8*8
$32F$*128*4*4	ConvTrans3D (3,3,3), $32F$, stride=1, padding=1	$32F$*D*8*8
(32+16)F*128*8*8	Conv3D (3,3,3), $16F$, stride=1, padding=1	$16F$*D*8*8
$16F$*128*8*8	ConvTrans3D (3,3,3), $16F$, stride=1, padding=1	$16F$*D*16*16

(16+8)F*128*16*16	Conv3D (3,3,3), 8F, stride=1, padding=1	8F*D*16*16
8F*128*16*16	ConvTrans3D (3,3,3), 8F, stride=1, padding=1	8F*D*32*32
(8+4)F*128*32*32	Conv3D (3,3,3), 4F, stride=1, padding=1	4F*D*32*32
4F*128*32*32	ConvTrans3D (3,3,3), 4F, stride=1, padding=1	4F*D*64*64
(4+2)F*128*64*64	Conv3D (3,3,3), 2F, stride=1, padding=1	2F*D*64*64
2F*128*64*64	ConvTrans3D (3,3,3), 2F, stride=1, padding=1	2F*D*128*128
(2+1)F*128*128*128	Conv3D (3,3,3), 1F, stride=1, padding=1	1F*D*128*128
1F*128*16*16	Conv3D (factor,3,3), 2, stride (factor,1,1), padding (0,1,1)	2*D/factor*128*128

Table 1: The detail of the recurrent structured 3D block.

References

[1] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *German Conference on Pattern Recognition*. Springer, 2014, pp. 31–42.

[2] 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*, 2020.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[5] M. Menze, C. Heipke, and A. Geiger, "Joint 3D estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[6] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[7] Z. Xiong, J. Zhang, and J. Tian, "Solving the depth of the repeated texture areas based on the clustering algorithm," in *MIPPR 2015: Automatic Target Recognition and Navigation*, vol. 9812. International Society for Optics and Photonics, 2015, p. 98120O.

[8] G. Saygili, L. Van Der Maaten, and E. A. Hendriks, "Improving segment based stereo matching using SURF key points," in *2012 19th IEEE International Conference on Image Processing*. IEEE, 2012, pp. 2973–2976.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[10] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[11] N. Carlini and D. Wagner, "MagNet and "efficient defenses against adversarial attacks" are not robust to adversarial examples," *arXiv preprint arXiv:1711.08478*, 2017.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1765–1773.

[13] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *arXiv preprint arXiv:1912.02292*, 2019.

[14] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5695–5703.

[15] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[16] H. Xu and J. Zhang, "AANet: Adaptive aggregation network for efficient stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1959–1968.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[18] Ö. Çiçek, Ahmed, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: learning dense volumetric segmentation from sparse annotation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 2016, pp. 424–432.

[19] S. Vassiliadis, E. A. Hakkennes, J. Wong, and G. G. Pechanek, "The sum-absolute-difference motion estimation accelerator," in *Proceedings. 24th EUROMICRO Conference (Cat. No. 98EX204)*, vol. 2. IEEE, 1998, pp. 559–566.

[20] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 353–363, 1993.

[21] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *European Conference on Computer Vision*. Springer, 1994, pp. 151–158.

[22] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang, "On building an accurate stereo matching system on graphics hardware," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2011, pp. 467–474.

[23] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1592–1599.

[24] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[25] H. Park and K. M. Lee, "Look wider to match image patches with convolutional neural networks," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1788–1792, 2016.

[26] J. Zbontar, Y. LeCun *et al.*, "Stereo matching by training a convolutional neural network to compare image patches." *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, 2016.

[27] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3279–3286.

[28] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1180–1202, 2010.

[29] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang, "A deep visual correspondence embedding model for stereo matching costs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 972–980.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*, 2016, pp. 770–778.

[31] K. “He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[32] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, “Deep ordinal regression network for monocular depth estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2002–2011.

[33] H. Hirschmuller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2007.

[34] P. Knobelreiter, C. Reinbacher, A. Shekhovtsov, and T. Pock, “End-to-end training of hybrid CNN-CRF models for stereo,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2339–2348.

[35] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi, “StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 573–590.

[36] R. Chabra, J. Straub, C. Sweeney, R. Newcombe, and H. Fuchs, “StereoDRNet:: Dilated residual stereonet,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 786–11 795.

[37] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[38] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, “FlowNet: Learning optical flow with convolu-

tional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[39] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan, "Cascade residual learning: A two-stage convolutional neural network for stereo matching," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 887–895.

[40] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang, "Learning for disparity estimation through feature constancy," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2811–2820.

[41] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan, "Recurrent mvsNet for high-resolution multi-view stereo depth inference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5525–5534.

[42] G. Yang, J. Manela, M. Happold, and D. Ramanan, "Hierarchical deep stereo matching on high-resolution images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5515–5524.

[43] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 66–75.

[44] G.-Y. Nie, M.-M. Cheng, Y. Liu, Z. Liang, D.-P. Fan, Y. Liu, and Y. Wang, "Multi-level context ultra-aggregation for stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3283–3291.

[45] Y. Zhong, Y. Dai, and H. Li, "Self-supervised learning for stereo matching with self-improving ability," *arXiv preprint arXiv:1709.00930*, 2017.

[46] Z. Wu, X. Wu, X. Zhang, S. Wang, and L. Ju, "Semantic stereo matching with pyramid cost volumes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7484–7493.

[47] L. Yu, Y. Wang, Y. Wu, and Y. Jia, "Deep stereo matching with explicit cost aggregation sub-architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[48] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "GA-Net: Guided aggregation net for end-to-end stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 185–194.

[49] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. Van Der Maaten, M. Campbell, and K. Q. Weinberger, "Anytime stereo image depth estimation on mobile devices," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5893–5900.

[50] S. Tulyakov, A. Ivanov, and F. Fleuret, "Practical deep stereo (PDS): Toward applications-friendly deep stereo matching," *arXiv preprint arXiv:1806.01677*, 2018.

[51] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5515–5524.

[52] Y. Zhang, S. Khamis, C. Rhemann, J. Valentin, A. Kowdle, V. Tankovich, M. Schoenberg, S. Izadi, T. Funkhouser, and S. Fanello, "ActivestereoNet: End-to-end self-supervised learning for active stereo systems," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[53] C. Chen, X. Chen, and H. Cheng, "On the over-smoothing problem of CNN based disparity estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8997–9005.

[54] Z. Jie, P. Wang, Y. Ling, B. Zhao, Y. Wei, J. Feng, and W. Liu, "Left-right comparative recurrent model for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3838–3846.

[55] A. Seki and M. Pollefeys, "Patch based confidence prediction for dense disparity map." in *BMVC*, vol. 2, no. 3, 2016, p. 4.

[56] S. Gidaris and N. Komodakis, "Detect, replace, refine: Deep structured prediction for pixel wise labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5248–5257.

[57] R. Haeusler, R. Nair, and D. Kondermann, "Ensemble learning for confidence measures in stereo vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 305–312.

[58] A. Spyropoulos, N. Komodakis, and P. Mordohai, "Learning to detect ground control points for improving the accuracy of stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1621–1628.

[59] M.-G. Park and K.-J. Yoon, "Leveraging stereo matching with learning-based confidence measures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 101–109.

[60] M. Poggi and S. Mattoccia, "Learning from scratch a confidence measure." in *BMVC*, 2016.

[61] A. S. Wannenwetsch, M. Keuper, and S. Roth, "ProbFlow: Joint optical flow and uncertainty estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1173–1182.

[62] K. Batsos, C. Cai, and P. Mordohai, "CBMV: A coalesced bidirectional matching volume for disparity estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2060–2069.

[63] A. Shaked and L. Wolf, "Improved stereo matching with constant highway networks and reflective confidence learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4641–4650.

[64] F. Tosi, M. Poggi, A. Benincasa, and S. Mattoccia, "Beyond local reasoning for stereo confidence estimation with deep learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 319–334.

[65] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[66] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano, "Unsupervised adaptation for deep stereo," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1605–1613.

[67] J. Pang, W. Sun, C. Yang, J. Ren, R. Xiao, J. Zeng, and L. Lin, "Zoom and learn: Generalizing deep stereo matching to novel domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2070–2079.

[68] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 270–279.

[69] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3D human pose estimation in the wild: a weakly-supervised approach," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 398–407.

[70] Z. Zhang, C. Xu, J. Yang, Y. Tai, and L. Chen, "Deep hierarchical guidance and regularization learning for end-to-end depth estimation," *Pattern Recognition*, vol. 83, pp. 430–442, 2018.

[71] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "Towards real-time unsupervised monocular depth estimation on CPU," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5848–5854.

[72] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano, "Real-time self-adaptive deep stereo," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 195–204.

[73] Y. Zhong, H. Li, and Y. Dai, "Open-world stereo video matching with deep RNN," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 101–116.

[74] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2800–2810.

[75] C. Zheng, T.-J. Cham, and J. Cai, "T2Net: Synthetic-to-realistic translation for solving single-image depth estimation tasks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 767–783.

[76] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

[77] J.-H. Lee, M. Heo, K.-R. Kim, and C.-S. Kim, "Single-image depth estimation based on Fourier domain analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 330–339.

[78] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 118–126.

[79] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6647–6655.

[80] C. Zhou, H. Zhang, X. Shen, and J. Jia, "Unsupervised learning of stereo matching," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1567–1575.

[81] Y. Gao, Y. Liu, H. Zhang, Z. Li, Y. Zhu, H. Lin, and M. Yang, "Estimating GPU memory consumption of deep learning models," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1342–1352.

[82] E. Albert, S. Genaim, and M. Gómez-Zamalloa, "Parametric inference of memory requirements for garbage collected languages," *ACM Sigplan Notices*, vol. 45, no. 8, pp. 121–130, 2010.

[83] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[84] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convNets," *arXiv preprint arXiv:1608.08710*, 2016.

[85] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[86] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," *arXiv preprint arXiv:1802.00124*, 2018.

[87] N. Lee, T. Ajanthan, and P. H. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[88] J.-H. Luo, J. Wu, and W. Lin, "ThINet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5058–5066.

[89] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[90] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[91] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[92] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4207–4215.

[93] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[94] F. Yang, Q. Sun, H. Jin, and Z. Zhou, "Superpixel segmentation with fully convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 964–13 973.