Design of Binocular Stereo Vision System Via CNN-based Stereo Matching Algorithm

by

Yan Jiao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Stereo vision is one of the representative technologies in the 3D camera, using multiple cameras to perceive the depth information in the three-dimensional space. The binocular one has become the most widely applied method in stereo vision. So in our thesis, we design a binocular stereo vision system based on an adjustable narrow-baseline stereo camera, which can simultaneously capture the left and right images belonging to a stereo image pair.

The camera calibration and rectification techniques are firstly performed to get rectified stereo pairs, serving as the input to the subsequent step, that is, searching the corresponding points between the left and right images. The stereo matching algorithm resolves the correspondence problem and plays a crucial part in our system, which produces disparity maps targeted at predicting the depths with the help of the triangulation principle. We focus on the first stage of this algorithm, proposing a CNN-based approach to calculating the matching cost by measuring the similarity level between two image patches. Two kinds of network architectures are presented and both of them are based on the siamese network. The fast network employs the cosine metric to compute the similarity level at a satisfactory accuracy and processing speed. While the slow network is aimed at learning a new metric, making the disparity prediction slightly more precise but at the cost of spending way more image handling time and counting on more parameters. The output of either network is regarded as the initial matching cost, followed by a series of post-processing methods, including cross-based cost aggregation as well as semi-global cost aggregation. With the trick of Winner-Take-All (WTA), the raw disparity map is attained and it will undergo further refinement procedures containing interpolation and image filtering. The above networks are trained and validated on three standard stereo datasets: Middlebury, KITTI 2012, and KITTI 2015. The contrast tests of CNN-based methods and census transformation have demonstrated that the former approach outperforms the later one on the mentioned datasets.

The algorithm based on the fast network is adopted in our devised system. To evaluate the performance of a binocular stereo vision system, two types of error criteria are come up with, acquiring the proper range of working distance under diverse baseline lengths.

Acknowledgements

I would like to firstly express the sincere gratefulness to my supervisor, Professor Pin-Han Ho, for his patient guidance and support on my research.

I would like to give my heartfelt thanks to my co-supervisor Quentin Tang, for his professional instruction on camera calibration and rectification.

I would like to thank my dearest grandparents and parents for their spiritual and financial support.

I also would extend my gratitude to all of the colleagues who help me in the process of studying and doing research.

Finally, my genuine thanks to all of the worldwide frontline workers who are fighting with the COVID-19 pandemic. Their steady efforts make me stay home safely and finish this thesis successfully.

Dedication

This thesis is dedicated to my beloved grandparents and parents who encourage me to devote myself to conducting research.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 3D Camera Technologies

Machine vision is a technology, which is the combination of hardware and software, guiding the devices to execute specific functionalities based on the processing results of captured images. It's an important component in the advancements in logistics, robotics, automatic inspection, as well as autonomous driving. Within the industrial community, manufacturers have two choices of camera applied in machine vision, 2D or 3D camera[47].

The 2D camera is appropriate if this task asks for colour and structure information from the objects[14]. At present, it is the predominant measure when dealing with image processing problems. Applications for this technology can almost be found in all industrial scenarios, such as defective detection, dimensional measurement, and object positioning. Since it only visualizes a two-dimensional map of reflected intensity, a 2D camera misses the third dimension of objects, leading to certain difficulties under some circumstances.

Unlike the 2D camera, the 3D camera provides the depth information of the captured objects, which is applied to analyze the volumes, shapes, or spatial locations. This creates more opportunities for us to handle complicated requirements. For instance, depth values can be used to distinguish diverse types of defects that are similar in length and width for the 2D camera but reveal evident distinction in heights. The 3D camera is becoming an increasingly crucial role of many image processing applications, particularly in obstacle/presence detection, portioning of food, and navigation of self-driving vehicles in the

factory environment. Although it seems that 2D camera has won a bigger market share in the realm of image processing up to now, there's a continually growing trend towards the 3D camera. In the foreseeable future, 3D technology will play a significantly vital role due to the rapid growth of industrial automation and upgrading to Industry 4.0.

Various technologies have been developed in a 3D camera. Right now, the most commonly used approaches are stereo vision, structured light, and Time-of-Flight (ToF). Each of these methods follows different principle and they have respective strengths and weaknesses. On account of the complementary characteristics of these methods, the most proper solution will differ from the needs of the application.

Stereo vision is a technique aimed at predicting depth from two or more cameras. One of the most popular research topics is the binocular stereo vision system, working like a pair of human eyes that observe the same scene from two distinct perspectives. Image pairs are rectified based on the relative position of two calibrated cameras and the theory of epipolar geometry. After the rectification process, a stereo matching algorithm, which is the core module of the whole system, is implemented to find the corresponding pixels from the left and right images. Then the depth information is acquired with the assistance of the triangulation principle. The range of working distance varies from the baseline, which is the distance between the optical centres of the two cameras.

One possible way of enhancing the performance of the stereo vision is to introduce the structured light into this system. When an optical source projects special light patterns onto the objects, the depth values will be more accurate and the shortcomings of stereo vision (because of the lack of light and the existence of untextured regions) are dramatically mitigated.

Both two schemes above are likely to achieve high precision in a small range and they perform well in sunlight. But the problem lies in the computational complexity, making it hard to satisfy the demand for real-time ability. In contrast, ToF technology offers us depth data efficiently, which has been adopted by state-of-the-art mobile devices in augmented reality (AR) scenarios. There are two approaches taken by ToF, continuous wave and pulsed ToF. The former way measures the length of the phase from the brightness-modulated light. While the pulsed ToF gets distances through recording the travel time for a great number of light pulses. However, the feature of resorting to a particular media makes it too sensitive to be adopted under the sunlight condition.

Table 1.1 shows the comparison result among the mentioned methods in terms of several key criteria for an application. We must specify our target and expectation at the beginning to select the optimal technique. In this thesis, we are pursuing constructing a low-cost 3D camera system under natural sunlight condition, obtaining the depth information of

	Stereo Vision	Structured Light	Time-of-Flight
Range of working distance	Medium to far	Medium	Far
Resolution	Medium	Medium	High
Depth accuracy	Medium to very accurate in short working ranges	Medium to very accurate in short working ranges	Medium
Software complexity	High	Medium	Low
Real-time capability	Low	Low to medium	High
Behaviour in low light	Weak	Good	Good
Outside area	Good	Weak	Weak
Compactness	Medium	Medium	Very compact
Material costs	Medium	High	Medium to high
Total operating cost	Medium	Medium to high	Medium to high

Table 1.1: Comparison of three major 3D techniques

the objects at a range of one meter as fast and precise as possible. Out of the above considerations, our attention will be put on how to employ the binocular stereo vision technology to build a depth measurement system.

1.1.2 Binocular Stereo Vision

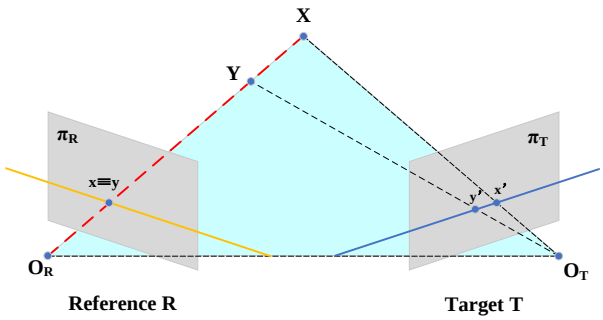


Figure 1.1: Binocular stereo rig under epipolar constraint

Figure 1.1 demonstrates a simplified graphic of the binocular stereo rig in an arbitrary position and direction[33]. There are two images delivered by the left and right cameras simultaneously. One is referred to as the reference image and the other becomes the target

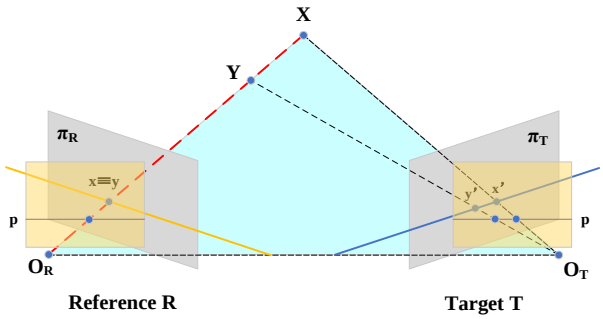


Figure 1.2: The standard form of binocular stereo rig

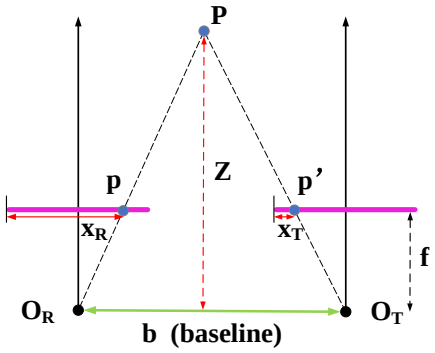


Figure 1.3: Relationship between depth and disparity

image. For every pixel in the reference image R, the corresponding pixel should be found in the target image T. Thanks to the epipolar constraint, the search area won't be the entire 2D image space. Let's suppose two points X and Y are on the same line of the sight of the reference image R, which means both points project onto the same image point $x \equiv y$ on the image plane π_R of the reference image R. The epipolar constraint tells us that the correspondence for a point lying on the line of sight (red dash line) is on the blue line from the image plane π_T of the target image, thereby decreasing the computational time and search range. Now that the search scope can be narrowed from 2D to 1D, we can rearrange the stereo rig more conveniently like figure 1.2, namely a standard form. Two cameras with the same focal length are aligned parallel from each other such that the corresponding pixels are restricted on the same epipolar line.

Figure 1.3 illustrates the relationship between the depth and disparity from the view

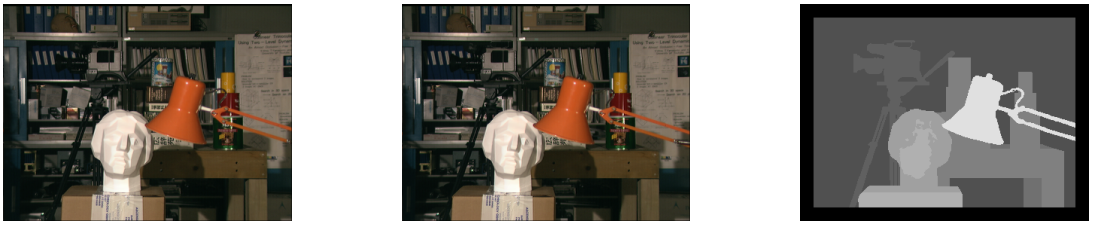
of geometry when applying the standard form of the stereo rig. By considering similar triangles PO_RO_T and Ppp', we derive the triangulation principle as the following:

$$\frac{b}{Z} = \frac{(b + x_T) - x_R}{Z - f} \Rightarrow Z = \frac{b \cdot f}{x_R - x_T} = \frac{b \cdot f}{d}, \tag{1.1}$$

where

Z: depth, $x_R - x_T, d$: disparity, b: baseline, f: focal length of each camera.

The displacement in pixels between the x coordinate of two corresponding points is called disparity. According to 1.1, points closer to the camera seem to be brighter (higher disparity value) as the result of the inverse relationship between the depth and disparity. After getting disparity values of all pixels in the reference image, a dense disparity map, which is typically encoded with a grayscale image is generated. Figure 1.4 indicates the well-known stereo image pair Tsukuba and its groundtruth disparity map. Given a stereo



(a) Left image (b) Right image (c) Groundtruth disparity map

Figure 1.4: Example of a stereo image pair and its groundtruth disparity map

rig with a fixed baseline b and focal length f, the depth range of this system is constrained by the disparity range $[d_{min}, d_{max}]$. Usually, we take integers for disparity values. In most situations, we assign the left image to be the reference image while the right one to be the target. However, for the sake of reducing occlusion errors and getting better accuracy in the final disparity map, we prefer treating the left and right images as both the reference and target images and running the stereo matching algorithm twice.

1.2 Contributions

The main contributions of this thesis are listed as follows:

- A novel CNN-based stereo matching algorithm is proposed, in which we learn to measure the similarity of image patches by training a convolutional neural network.

- An implementation of the binocular stereo vision system is conducted with an adjustable narrow-baseline USB stereo camera, providing a platform to verify the feasibility of the raised algorithm.
- Two metrics for evaluating the performance of a binocular stereo vision system are presented, offering fundamental guidance on how to determine the range of working distance under distinct baseline lengths for our system to perceive the depth information in 3D space.

1.3 Organization

The rest of this thesis is divided into four sections. Chapter 2 focuses on reviewing other related disparity map generation algorithms in stereo vision, and it reveals how their outcomes are distinguished from one another. In consideration of their advantages and disadvantages, the best one of these methods will be chosen as the benchmark approach in our later contrast experiment.

The procedures for building this system are detailed in Chapter 3. Above all, we briefly describe the stereo camera platform, listing out all its relevant parameters, software development tools, environments, and a general idea of the development. Then we thoroughly explain how to get rectified image pairs utilizing calibration and rectification. Finally, a CNN-based stereo matching algorithm made up of two parts is elaborated. Concerning the CNN part, we construct multiple datasets based on the existing standard ones. Owing to our goal of attaining the depth results fast and accurately, two types of network structures are put forward. To improve the quality of the disparity map produced by the first part, the preliminary result will go through a series of traditional post-processing stereo methods.

Chapter 4 highlights the details of public datasets and depicts the entire model training process as well. To validate the effectiveness of the proposed solution, we conduct both experimental and analytical studies on the benchmark method as well as our approaches. In particular, we investigate how different baseline distances impact the system performance on the strength of two new metrics, figuring out its valid range of working distance.

At last, we summarize this thesis, discuss the limitations, and outline the future work in Chapter 5.

Chapter 2

Literature Review

For a binocular stereo vision system, the central part is the stereo correspondence step. It's a challenging task because of all sorts of practical difficulties, including occlusion, transparent objects, discontinuities, photometric/perspective distortion, image noise, specular reflection (*e.g.* windows), foreshortening, large uniform/ambiguous regions (*e.g.* walls, sky *etc.*), and repetitive/indistinguishable patterns.

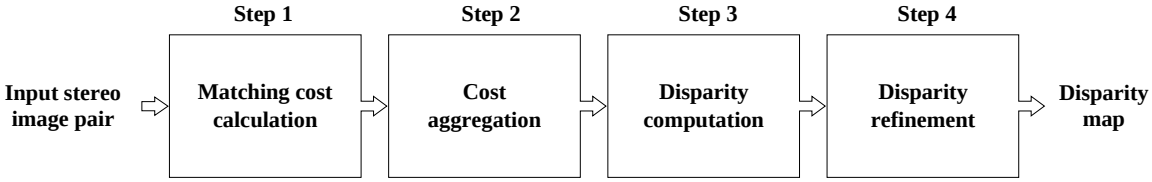


Figure 2.1: The flowchart for the implementation of a stereo matching algorithm

The taxonomy addressed by Scharstein and Szeliski[45] provides a systematic discussion of the processing stages in most of the stereo matching algorithms. These algorithms are composed of four major steps as indicated in figure 2.1. Normally, the stereo rig is set in standard form which captures the left and right images at the same time. For techniques analyzed in this chapter, we suppose that the input stereo image pair is rectified, which means the search scope has been confined to one-dimensional space. Then the rectified image pair will undergo all of the chunks in turn, from Step 1 to Step 4, producing a qualified disparity map. Each of the chunks consists of one or multiple algorithms and their results can be evaluated in the light of the groundtruth disparity maps.

2.1 Stereo Matching Algorithms

To have a basic understanding of the stereo matching algorithm, we first start with the simplest possible one. It's sound to presume that the colour of corresponding points in the reference and target images looks alike, named as photo consistency assumption. Moreover, the corresponding points are on the same epipolar line since we assume that the image pairs are rectified. Therefore, for every pixel p from the reference image, a naive algorithm will find along the corresponding epipolar line in the target image. Then the mapping pixel p', which is most similar to p from the view of colour, should be chosen. However, the high fuzziness of determining the mapping pixel leads to a noisy disparity map via this simple approach. For instance, if we are going to find the mapping point of a blue pixel in the reference image, there are frequently a fairly large number of correspondences in the target image. The problem due to ambiguity is usually resolved by exerting a smoothness assumption, which signifies that disparities of pixels belonging to spatial neighbours are probably to be similar. The way of implementing the above assumption varies from different stereo matching algorithms, which brings about the distinction between the following two types of approaches.

Traditionally, stereo matching algorithms can be classified into two categories, local or global methods. Instead of mapping individual pixels, the local method, referred to as the area-based or window-based method, considers the pixels in a preset window. It's noteworthy that we are taking advantage of the smoothness assumption implicitly by utilizing windows. In other words, all of the pixels within a window share the same disparity. Hence, this kind of approach performs efficiently with low computational complexity. Local methods contain all four steps mentioned before, preferred by some researchers in their works[13][34][51]. For every pixel, the calculated matching cost based on various metrics is aggregated through summing up or taking the average in the window. The disparity map is generated by choosing the Winner-Take-All (WTA) strategy, that is, the disparity of any pixel is determined by selecting that with the minimum matching cost.

Conversely, global methods aim at seeking the disparity assignment minimizing an energy function over the whole image. The typical form of an energy function is made up of two terms, a data term as well as a smoothness term. The former term, an explicit form of representing the smoothness assumption, gives penalties to disparity assignments that don't fit the matching cost concerning the entire image. While the latter term is responsible for favouring disparity maps that are spatially smooth as much as possible, but it penalizes disparity changes unless great variations happen at the depth borders. By making the disparities closer to that of neighbouring pixels, the disparity map is achieved. Global methods bring satisfactory results at the cost of a heavy computational burden,

which makes them hard to be adopted in real-time scenarios. Commonly, global algorithms only carry out three steps without performing the cost aggregation, which is Step 2 of the flowchart[39][41][50].

Apart from the local and global approaches, it's also well known that the optical flow-based stereo matching algorithms can fulfill the disparity estimation quickly, which assumes that the pixel intensity remains the same for homologous points in the stereo image pair. The optical flow explores the obvious movement between two consecutive images based on their intensities. Due to the epipolar constraint, the flow field is confined to the horizontal direction Differing from the window-based method, there's no particular window and the searching range varies from diverse features of the image. Sub-pixel accuracy[17][24] can be satisfied with this type of method at the expense of $O(N)$ computational complexity[16], where N indicates the size of the stereo image pairs. But the challenges related to texture-less or occluded regions[24] still retain in most of the optical flow-based techniques.

To further know more technical details and do a comparison among those algorithms, we will discuss each step separately in the following sections.

2.1.1 Matching Cost Computation

All of the stereo matching algorithms need a metric to measure the degree of similarity between two pixels. The function of the first stage is to judge whether two pixels correspond to the identical point in an image pair, contributing to calculating the disparity of each point in the reference and target image[16]. In the early days, the pixel-based matching cost was taken by many stereo matching algorithms in the step of calculating matching cost[30]. These metrics comprise absolute differences (AD) and square differences (SD). Furthermore, to limit the influence of outliers, more robust versions of the pixel-based metrics are developed, such as the truncated absolute differences (TAD) as well as dissimilarity measure which is insensitive to image sampling[15]. Besides, these metrics can process either gray or colour images.

Compared with the pixel-based matching costs, area-based matching costs offer more abundant information. This trait enhances the matching accuracy on account of adding neighbouring pixels to the target area. The area-based methods incorporate the sum of absolute differences (SAD), the sum of squared differences (SSD), the sum of truncated absolute differences (STAD), normalized cross-correlation (NCC), census transform (CT), and so on. The area-based methods compute the matching cost in a specific area. This area, which is known as the window as well, is likely to be square, rectangular, cross-shaped, or any other shapes with fixed or adaptive size. The primary drawback of the area-based

approaches lies in our implicit smoothness assumption, assigning a constant disparity to pixels in a window. The assumption is violated at depth discontinuities when pixels from the foreground and background are enclosed with the same window. So it's a thorny problem to pick the best-fit window shape and size for every pixel, which considerably impacts the quality of our disparity map.

A disparity space image (DSI) is produced regardless of whatever strategy we take for calculating the matching cost of each pixel. DSI is a $W \cdot H \cdot (d_{max} - d_{min})$ tensor, where W and H stand for the width and height of the image, respectively. Each element $C(x, y, d)$ in DSI, implying the confidence of correspondence, is the matching cost of pixel $I_R(x, y)$ in the reference image I_R and the corresponding pixel $I_T(x - d, y)$ in the target image I_T. (x, y) represents the coordinate of each pixel and d is the disparity.

Absolute Differences (AD)

This metric computes the intensity difference between every pixel in the reference image I_R and the corresponding pixel in the target image I_T as follows:

$$C_{AD}(x, y, d) = |I_R(x, y) - I_T(x - d, y)|. \tag{2.1}$$

The AD metric, which is the easiest one among those metrics, was applied to the real-time stereo matching situation and accelerated with the graphics processing unit (GPU) by Wang et al. in [49]. This metric works nicely in areas with a small amount of texture but it's not able to produce an eligible disparity map for multi-texture regions. To deal with this limitation, the modified version of the AD metric, i.e., truncated absolute differences (TAD), was put forward and then conducted by Pham[40] and Min[37], decreasing the error rate in the disparity maps.

Squared Differences (SD)

The SD metric sums up the squared differences between pixels in I_R and correspondences in I_T, as given by:

$$C_{SD}(x, y, d) = |I_R(x, y) - I_T(x - d, y)|^2. \tag{2.2}$$

To maintain the sub-pixel accuracy of the input image, Yang et al.[54] adopted this metric when constructing the matching cost volume. But the SD metric also resulted in some additional noises at the depth discontinuities of their disparity maps. Thus an extra post-processing technique was implemented through a bilateral filter to preserve the edges and

smooth the regions adjacent to depth borders. Likewise, diverse matching cost metrics were assessed in the context of intelligent vehicle applications[38]. Unfortunately, the SD metric brought the maximum error in their experiments as this metric can't tolerate the noise and brightness variation, particularly in a real-time system.

Sum of Absolute Differences (SAD)

The SAD metric, written as 2.3, denotes the summation of absolute differences between the intensity value of every pixel from the window centring at (x, y) in I_R, as well as the counterpart centring at $(x - d, y)$ in I_T.

$$C_{SAD}(x, y, d) = \sum_{(x,y)\in w} |I_R(x, y) - I_T(x - d, y)|. \tag{2.3}$$

This metric not only is the most famous one for figuring out the matching cost but also is capable of meeting the requirement of real-world applications thanks to its relatively less computational cost. This characteristic has been taken advantage of by Tippetts et al.[48] in human pose analysis for a resource-limited system. Meanwhile, a novel idea efficiently combining SAD metric and two correlation windows of different sizes was come up with by Gupta and Cho in [22]. In the first phase, the initial disparity map was estimated via the bigger correlation window and refined at discontinuities through a smaller window in the second phase. Although the implementation of the SAD metric is rapid, the output disparity map is unsatisfactory as a result of the heavy noise in textureless regions and depth boundaries.

Sum of Squared Differences (SSD)

Equation 2.4 defines the SSD metric, which aggregates the squared differences of intensity values over the window in the reference image and that from the target image. This metric was first used in computing matching cost in [20]. Researchers have employed the SSD metric in the multi-windows scheme with a fixed size, reducing the occurrence of occlusion errors. The reason for selecting several windows is to find the minimum SSD error and to choose a proper pixel in the disparity map. Similarly, Yang and Pollefeys[55] depended on the alike method which was proposed in [20], realizing the stereo matching algorithm on a platform running with GPU. Also, they claimed a better performance about processing speed in contrast to the former work.

$$C_{SSD}(x, y, d) = \sum_{(x,y)\in w} |I_R(x, y) - I_T(x - d, y)|^2. \tag{2.4}$$

Normalized Cross-correlation (NCC)

Alternatively, the NCC is another type of stereo matching metric, searching the matchup between two windows around a pixel from the image pair. The normalization operation for all pixels in the window makes up for the gap in bias and gain[28]. This metric is formulated in 2.5:

$$C_{NCC}(x, y, d) = \frac{\sum_{(x,y)\in w} I_R(x, y) \cdot I_T(x - d, y)}{\sqrt{\sum_{(x,y)\in w} I_T^2(x, y) \cdot \sum_{(x,y)\in w} I_R^2(x - d, y)}}. \tag{2.5}$$

Nevertheless, some studies demonstrated that the NCC metric brought about a higher extent of blurriness in areas near the depth borders compared to other metrics[28]. The underlying cause is that any exceptional values will give rise to serious errors during computing NCC values. A new technique for applying low-dimensional image features with the NCC metric to the matching task was raised by Satoh[42]. The NCC metric was opted, owing to its tolerance to intensity offsets and contrast variation. In terms of accuracy, this work reached an exciting result but it was very computationally expensive. Again, an advanced version of the NCC metric, called zero-mean normalized cross-correlation (ZNCC), was utilized by Cheng et al. in [18]. The pixels on the edges are processed in a way of multi-windows. These tricks offer higher accuracy but still have an extra computational load.

Census Transform (CT)

With CT metric, the comparative result between a pixel of interest and its neighbours located in a window is embedded in a sequence of bits like this:

$$\text{Census}(x, y) = \text{BitVector}_{(i,j)\in w}(I(i, j) \geq I(x, y)). \tag{2.6}$$

Then the Hamming distance is introduced to figure out the difference between two census bit vectors from the reference image and target one, as specified in 2.7:

$$C_{CT}(x, y, d) = \sum_{(x,y)\in w} \text{Hamming}(\text{Census}_R(x, y) - \text{Census}_T(x - d, y)), \tag{2.7}$$

where Census_R depicts the census vector of the reference image I_R and Census_T describes that of the target image I_T. The CT metric shows better results at discontinuities attributed to its high tolerance to abnormal values, as mentioned in [29] and verified by a

comparative experiment between SAD and CT. The shortcoming of this metric is that more errors tend to occur in areas with repeated patterns. To some extent, this disadvantage was overcome through some adjustments to the original CT metric[32]. They imported more bits to express the difference between the centre pixel and its neighbours. The experimental results proved their influence on mitigating the mismatching issue and achieving higher accuracy of the output disparity map as well. Additionally, the improved CT metric has definite immunity to a noisy input in contrast with the traditional CT metric.

By synthetic evaluation and comparison among the above six kinds of matching cost metrics, we decide to choose the census transform metric as the benchmark method, which will be compared with our proposed CNN-based stereo matching algorithm.

2.1.2 Cost Aggregation

Cost aggregation is the second stage of the entire stereo matching algorithm, aiming at increasing the probability of correct matching and having a significant impact on the overall effect of the disparity map. It's of great necessity to execute this step because the matching cost of merely one pixel isn't reliable for accurate matching. As for the local methods, the matching costs are often aggregated over the windows. Generally, the windows are set in a square shape centred at a pixel of interest. The most direct way of aggregating the matching cost is to consider using a low pass filter over a square window. The disparity map's error rate caused by the fixed-size window strategy will worsen when the window size exceeds a particular value. In addition, we have to adjust the optimal values of the required parameters of this approach, varying from diverse input image pairs. Or else the object contours are more easily to be blurry[52]. To overcome the fattening defects nearby the depth boundaries, more superior methods are developed such as multi-windows or shifting windows, as well as approaches applying adaptive windows (AW) with adaptive weights or sizes[23].

2.1.3 Disparity Computation

Currently, a general stereo matching algorithm pertains to one of the two main approaches, either the local or global method. For the local ones, the disparity of each pixel p in an image is substantially determined by taking the Winner-Take-All (WTA) strategy as shown in 2.8:

$$d_p = \underset{d \in D}{\operatorname{argmin}} C(p, d), \tag{2.8}$$

where $C(p, d)$ denotes the aggregated matching cost, and D is the set of predefined disparities. The disparity d_p associated with the minimum matching cost is selected. The WTA has been employed in local methods in [19], [31], and [58]. Whereas their implementation results suggest that the generated disparity maps still produce some occlusion errors and mismatching. Since the aggregation step is carried out through windows, which only takes the pixels surrounded by the interest one into account, any noises or blurred areas are likely to damage the accuracy. As a consequence, the performance of local methods relies on the previous two stages. Afterward, part of the errors will be eliminated in the final stage with certain post-processing techniques.

2.1.4 Disparity Refinement

To upgrade the image quality, the disparity refinement is an essential foundation for degrading noises and outputting a decent disparity map. Mostly, the final stage involves regularization and interpolation. Using a wide variety of spatial filters in the regularization, variations and inconsistencies among the pixel intensities are removed. In the meantime, the image noise will be suppressed. The interpolation procedure, also called occlusion filling, plays the part of estimating the disparities in the regions where disparities are indistinguishable. Typically, the occluded areas are identified with left/right consistency check, performed in [26] and [53]. After that, the occlusion problems are solved by filling those regions with disparities from the background or non-textured regions. If there exists a confidence level evaluating the reliability of each disparity, the algorithm can refuse undependable pixels. Then they will be replaced by the estimated values with the interpolation process, working out reasonable disparities based on the neighbouring pixels. In summary, with the aid of measuring the degree of confidence, the ultimate refinement stage integrates the adjacent pixels from the neighbours of the pixel of interest.

Chapter 3

System Design

In this chapter, the framework of the binocular stereo vision system is given. We will briefly introduce the design procedures in terms of platform establishment and development environments, which lay the foundations for executing the central stereo matching algorithm. Subsequently, calibration and rectification process are presented, as the premise of acquiring rectified stereo image pairs. Ultimately, the CNN-based stereo matching algorithm is put forward. Details about dataset construction, network architectures, and post-processing methods are elaborated in the chapter as well.

3.1 Introduction

3.1.1 Stereo Camera Platform

To configure the system platform and verify the feasibility of our proposed stereo matching algorithm, we purchased an adjustable narrow-baseline stereo camera at a reasonable cost. Table 3.1 demonstrates some crucial camera parameters. It's a USB 2.0 camera, whose imaging distance varies from the focal length and focus distance. Focal length, an inherent characteristic of a lens, is the distance between the lens and image sensor when the object is in focus, measuring the degree of light convergence or divergence. A positive shorter focal length suggests that the lens converges the rays more sharply, making it focus at a narrower range. In the field of photography, there exist two choices for lens selection: zoom lens or prime lens. Zoom lens with a zoom ring has a variety of focal lengths, and it's unnecessary to manually change the distance between the object and camera. Several kinds

Module No.	KS861		
Baseline range	35mm-169mm		
Focal length	3.9 mm		
Focus mode	Manual focus		
Interface	Micro USB 2.0		
Power	USB bus power		
Operating current	300 mA		
Sensor size	1/2.5″		
Sensor type	CMOS		
Pixel size	$3.75\mu m \times 3.75\mu m$		
Image compression format	MJPG/YUY2		
Imaging distance	0-50m		
Temperature (Operation)	-20℃ to 70℃		
Temperature (Stable image)	0℃ to 50℃		
Operating system requirement	Windows/Linux/Android		
Active array size video rate	Resolution	MJPG /(fps)	YUY2 /(fps)
	640×240	60	10
	1280×480	60	10
	2560×720	60	5
	2560×960	60	5
	1280×960	60	10

Table 3.1: Detailed specification of KS861 stereo camera[7]

of zoom lenses are provided such as wide zoom lens, telephoto zoom lens, or multi-purpose zoom lens. A prime lens, on the other hand, has the fixed focal length and there's no zoom ring on the lens. So if you want to get closer to or far away from your object, you need to physically move your camera back and forth. There are various prime lenses available, from ultra-wide-angle to wide as well as telephoto prime lens. The ultra-wide-angle lens, whose focal length is the minimum among those lenses, are usually applied at close object distances. Therefore, a $3.9mm$ lens was chosen as we aim to extract depth information of the objects in a range of one meter. Differing from the focal length, the focus distance is associated with the object, referring to the distance between the object and camera when the object is perfectly in focus. Since this simple lens only has manually focus mode, we have to twist the lens to focus the target object located at diverse distance ranges.

When it comes to a camera, another significant property is the resolution, determining the image quality and having a great impact on the performance of the disparity map as

Figure 3.1: Binocular stereo camera rig

well. This stereo camera offers us multiple resolutions, including 720p and 960p. One thing we should point out is that this stereo camera adopts a USB hub that expands a single USB into two so that two ports connect to each camera separately, which not only guarantees that the USB bandwidth is sufficient whenever transferring images at the maximum frame rate but also assures the synchronized outputs of two cameras. What we receive from the personal computer (PC) end are the results of two images which are spliced transversely. Here the resolution displaying in table 3.1 is equal to twice the camera's maximum resolution. So the maximum resolution of an individual camera is 1280×960. To remain the details of the scene as much as possible and elevate the matching accuracy of the stereo matching algorithm, we prefer using the maximum resolution in MJPG format, ensuring that the high-quality frames are transmitted in real-time via USB within the bandwidth limitation.

Pixel is the smallest controllable unit in digital imaging. To get the actual depth value based on the disparity value, correspondence needs to be set up between these two values through the camera pixel size.

Two categories of stereo cameras are sold on the market: fixed-baseline type and adjustable-baseline type. For a stereo camera, the baseline is an important parameter. From 1.1 we derive that the baseline is proportional to the depth given a fixed focal length and disparity value, proving that the baseline is likely to affect the range of working dis-

tance and the precision of produced disparity maps. To explore the relationship between the baseline and other factors, we decide to employ an adjustable-baseline stereo camera. The baseline ranges from $35mm$ to $160mm$, which is a relatively wide dynamic range for further study and experiments. Figure 3.1 depicts the stereo rig placed on the table. It's convenient for us to tune the height and angle of the stereo camera according to the positions of the objects in the scene, as it's equipped with a camera tripod.

3.1.2 Tools, Environments and Development Idea

Our system is mainly built with two scientific programming languages, Python and Lua. Python is a widely used integrated, multi-purpose and high-level language[10], serving as the basis of scientific and numerical computing, software development as well as business scenarios. Within the image processing community, basic manipulations encompass image display, cropping, rotating, shifting, etc. Python is a brilliant option for these kinds of common tasks owing to its rapid growth in the popularity and free accessibility of abundant advanced image processing tools in its libraries. Python libraries offer an intuitive and easy way to analyze the potential data and transform the images. It's embedded with a rich set of efficient libraries helping for data analysis and image processing such as Numpy, OpenCV-Python, and so on. Numpy[1] is one of the kernel libraries in Python and it assists in array operations. An image is substantially a Numpy array made up of pixels. With the help of fundamental Numpy operations like stacking, slicing, and masking, they allow us to manipulate pixel values in an image. OpenCV (Open Source Computer Vision Library)[8] is one of the most commonly utilized open-source libraries including hundreds of computer vision algorithms. Python provides users with the application programming interface (API) of this library, namely OpenCV-Python. OpenCV-Python works very quickly thanks to its background written in C/C++ and the ease with coding and calling (because of the foreground wrapped with Python). This feature makes it capable of running computer vision algorithms that require large amounts of calculations.

On the other hand, Lua is a lightweight, robust, leading scripting language, primarily supporting embedded use and data-driven approach[2]. It has an excellent reputation for its wonderful processing speed in the area of interpreted programming languages. Lua acts rapidly in certain benchmark codes and real-life situations as well, particularly in considerable numbers of large applications. In consideration of Lua's merits, Torch is embraced by our thesis as the machine learning library. Torch[12] is an open-source scientific computing framework satisfying the needs of a wide range of mainstream machine learning algorithms. It behaves productively and user-friendly for beginners, due to taking advantage of the virtue of fast implementation on the top of Lua, and a latent usage of C. There

are some dramatic features with Torch, which are the prerequisites for customizing your algorithms with the maximum adaptability and high speed. It supplies us with flexible arrays or tensors of arbitrary dimensions, which can be devoted to indexing, cloning, slicing, resizing, etc. Also, Torch has basic routines for linear algebra and numerical optimization, making it possible to build neural networks. Attributed to the support for GPUs, it's able to devise any structures of neural networks in a fast and efficient manner.

To code with Python and Lua, we rely on an integrated development environment (IDE) called PyCharm, offering code analysis and graphical debugger especially for the Python language[9]. While for Lua, it provides extra plugin EmmyLua, specifically for Lua programming. The development process concerning the PyCharm part is implemented on a Windows 10 laptop with Intel (R) Core (TM) i7-7660U CPU @ 2.50GHz and 16GB RAM, however, without NVIDIA GPU. As a result of limited computational resources on a laptop, we must realize the algorithm on a high-performance server. Our server is pre-installed the Ubuntu OS and equipped with NVIDIA GeForce RTX 2080 Ti 11GB GPU, Intel (R) Xeon (R) Silver 4114 CPU @ 2.20GHz and 15GB RAM. CUDA (Compute Unified Device Architecture) is a popular parallel computing platform and API model advanced by NVIDIA[5]. It encourages us to use a CUDA-enabled GPU to develop the post-processing stereo methods accelerated by a mass of concurrent threads running on GPU. Besides, the NVIDIA CUDA deep neural network library (cuDNN) is exerted for speeding up the training process of machine learning frameworks in the context of the Torch environment.

To create a secure file transmission tunnel from the local device and the remote server, a free and open-source basic file manager named WinSCP[6] is installed on the local end. Additionally, a tool for accessing the server from any other terminals is required. PuTTY[4] is a free terminal emulation program, aiming at configuring and controlling other devices from the Windows system.

The overall development ideas are summarized in the following steps:

- The first two offline procedures: calibration and rectification, are carried out by calling relevant functions in OpenCV-Python.
- Stereo image pairs are captured and then rectified based on the acquired parameters in the first step.
- Stereo matching algorithm is formulated and fulfilled in PyCharm.
- Executable program files containing the raised algorithm are going to be uploaded onto the server with WinSCP. Commands sent by PuTTY from the local PC will trigger the training/testing process and the final refined models are generated.

- Rectified stereo pairs are transferred onto the server and processed with the stereo correspondence algorithm.
- Produced disparity map returns to the local end.

To ensure the repeatability of our work and avoid version conflicts between various programs, table 3.2 illustrates the version information of major packages/software applied during the development process.

Package/Software name	Version
CUDA	10.0
cuDNN	7.6.4
Torch	7.0
OpenCV	3.1.0
OpenCV-Python	4.2.0
Lua	5.1
PyCharm	2020.1
PuTTY	0.73
WinSCP	5.17.2

Table 3.2: Version information of used packages/software

3.2 Camera Calibration

3.2.1 Methodology

Normally, camera calibration is indispensable for common cameras to correct lens distortion, which is a process of specifying the intrinsic and extrinsic parameters, as well as the distortion coefficients. To get an estimation of these parameters, we need to establish the mapping relation between the 3D points in the world and their corresponding 2D points on the images. We could get this relationship by capturing photos of a certain calibration pattern, *e.g.*, a checkerboard. The camera parameters can be derived from the above correspondences. After the calibration procedure, we can assess the accuracy of the estimated parameters by computing the reprojection errors.

Based on the model brought up by Jean-Yves Bouguet[3], the pinhole camera calibration algorithm has emerged. This model consists of the pinhole camera model[59] and

lens distortion[25]. Because a naive pinhole camera has no lens, the pinhole camera model doesn't take the lens distortion into account. To build a model for a practical camera, the radial and tangential lens distortion were imported into the camera model.

A pinhole camera is an easy device that has no lens but merely one aperture[11]. The light beam goes through the aperture and forms a reverse image on the other side of the camera. The camera matrix in 3×3 dimension stands for the pinhole camera parameters, mapping points in the 3D world onto the 2D images. With the intrinsic and extrinsic parameters, the calibration algorithm can figure out the camera matrix. The extrinsic parameters focus on the camera's location and pose in the world, representing a rigid transformation from 3D world coordinates to 3D camera coordinates. While the intrinsic parameters, meaning a projective transformation from 3D camera coordinates to 2D image coordinates, are associated with the camera only. As a consequence, they can be saved for future use once computed. The extrinsic parameters include two components, a rotation matrix R, and a translation vector T. The origin of the camera coordinates is at the optical centre. The intrinsic parameters cover the focal length, principal point (optical centre), and skew coefficient. The camera matrix K is denoted as:

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where f_x, f_y are the focal length in pixels, c_x, c_y are the principle point in pixels. Moreover, according to the pixel size, we can evaluate the calibration result from the aspect of true focal length F in millimetres, as shown in 3.2:

$$f_x = \frac{F}{p_x}, f_y = \frac{F}{p_y}, \tag{3.2}$$

where p_x and p_y are the pixel size mentioned in table 3.1.

On account of the existence of lens for a real camera, two types of lens distortion are possible to occur. When light beams curve more around the border of a lens than that happens at the centre, we refer it as the radial distortion, which seems greater for a smaller lens. The radial distortion coefficients describe this kind of lens distortion. We represent the distorted pixel with its location on the image, i.e., $(x_{distorted}, y_{distorted})$. The relationship between distorted and undistorted pixels are defined as:

$$x_{distorted} = x(1 + k_1 \times r^2 + k_2 \times r^4 + k_3 \times r^6), \tag{3.3}$$

$$y_{distorted} = y(1 + k_1 \times r^2 + k_2 \times r^4 + k_3 \times r^6), \tag{3.4}$$

where

x, y: location of undistorted pixel, k_1, k_2, k_3: radial distortion coefficients, r: $x^2 + y^2$. Generally, k_1, k_2 are well enough to calibrate a camera lens. k_3 will be chosen to handle severe lens distortion.

The tangential distortion coefficients are responsible for dealing with another type of lens distortion if the lens isn't set to be parallel to the image plane.

$$x_{distorted} = x + [2 \times p_1 \times x \times y + p_2 \times (r^2 + 2 \times x^2)], \tag{3.5}$$

$$y_{distorted} = y + [2 \times p_2 \times x \times y + p_1 \times (r^2 + 2 \times y^2)], \tag{3.6}$$

where p_1 and p_2 are tangential distortion coefficients.

In general, there are totally five distortion coefficients, forming a vector as follows:

$$\text{Distortion coefficients} = [k_1\ k_2\ p_1\ p_2\ k_3]. \tag{3.7}$$

3.2.2 Procedure

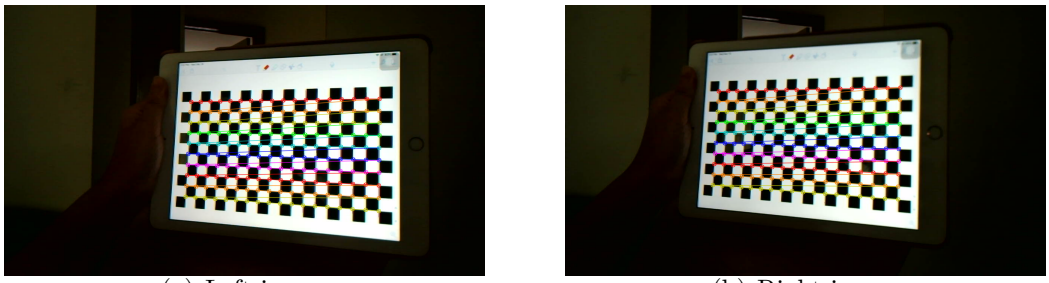
At the beginning of the calibration, we need to determine the pattern (the number of rows and columns) of the checkerboard. A fewer number of squares in either dimension may degrade the performance of rectification. Furthermore, any tiny physical distortions of the checkerboard image will notably worsen the calibration error. So we decide to display the checkerboard image in full-screen mode on an iPad. The checkerboard has 11×19 squares with 10×18 internal corners and the checker size is 10 mm.

Due to applying a stereo camera comprising two lenses, we need to calibrate the two cameras respectively. To calibrate each camera, a set of 3D world points and their corresponding 2D image points are necessary. 2D image points are the intersections of two black squares in the checkerboard image but 3D world points are unknown to us. So for simplicity, we suppose that they are on XY plane ($Z = 0$) with coordinates $(0, 0), (1, 0), (2, 0), \cdots$, which means the camera is moving around the checkerboard. But in fact, the checkerboard is the moving one, instead of the camera. Since we know the size of each square, the coordinates can be scaled by multiplying the square size in millimetres. Under the circumstances, the 3D world points on the checkerboard are mapped to the 2D image points based on the camera's location and pose at this moment.

Collecting a series of qualified stereo image pairs is the key to camera calibration. To make the calibration result more precise, we need to move the checkerboard in a particular way:

- making sure the entire pattern is fully visible to both cameras.
- moving it left and right as well as up and down, such that it will be detected at the edges of the field of view horizontally and vertically.
- rotating it from diverse angles.
- moving it back and forth so that the size of the checkerboard in the images varies.

For every possible camera pose, an image pair is taken and saved into image files. There's a trade-off between the number of image pairs and calibration precision. If we apply an excessive amount of image pairs, it will take the calibration function from OpenCV too much time to figure out the calibration parameters, or even not converge to acceptable values. Through our experiments, we noticed that taking 10 image pairs are sufficient to give us a reasonable accuracy in the calibration process. Then for every captured photo, we need to convert it to a grayscale image and pass it to *cv2.findChessboardCorners*, extracting the detected 2D coordinates of the checkerboard internal corners. If all the expected points are found out, we can call another function *cv2.cornerSubPix* to further enhance the accuracy of checkerboard corners' coordinates to sub-pixel level. The detected checkerboard pattern can be visualized via *cv2.drawChessboardCorners*. Once we get ready for all the ten groups of 3D world points and 2D image points from two cameras, the calibration step can be implemented through calling function *cv2.calibrateCamera* twice, which calculates the camera matrix K, distortion coefficients vector, rotation matrix R, and translation vector T for each camera separately. Also, there's a crucial return argument named reprojection error, giving us a good estimation of the precision of computed parameters. This error should be as close as possible to zero. Figure 3.2 implies the sample calibration checkerboard image pair and its detected corners. However, it's noteworthy that we need to double-check the



(a) Left image (b) Right image

Figure 3.2: Sample calibration checkerboard image pair with the drawn detected corners

order of these corners for an image pair, ensuring that the corners' order from the left image is consistent with that of the right image.

After harvesting the parameters from calibrating the individual camera, we still need to calibrate them together. Through treating those pre-calculated parameters as the input of *cv2.stereoCalibrate*, we can work out the rotation matrix R and translation vector T between these two camera coordinate systems like follows:

$$R_2 = R \cdot R_1, \tag{3.8}$$

$$T_2 = R \cdot T_1 + T, \tag{3.9}$$

where $(R_i, T_i), i = 1, 2$ are the computed poses of the checkerboard with respect to the first and the second camera. Taking the absolute value of the first component of T and scaling it with the actual checker size, the theoretical baseline length b can be achieved, providing an indicator for evaluating the performance of camera rectification. (R, T) helps us to calculate the position of one camera relative to another, which can be used to link the coordinate systems of two cameras. Similarly to *cv2.calibrateCamera*, we are pursuing minimizing the reprojection error for all points from diverse perspectives of two cameras.

3.3 Stereo Rectification

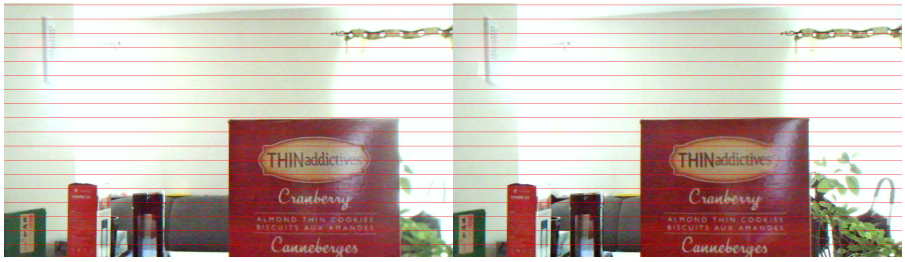
Based on the estimated camera parameters from calibration, we'll proceed with stereo rectification transformation, guaranteeing that image planes of both cameras are on the same plane. Accordingly, this makes the epipolar lines parallel to each other, which simplifies the later stereo matching algorithm as the corresponding pixels from the left and right cameras are on the same epipolar line, narrowing the searching range from 2D space to 1D. *cv.StereoRectify* allows us to calculate the rotation matrices for two cameras. As long as those matrices are applied to a stereo image pair, they will become co-plane.

Figure 3.3 shows the comparison results of the rectification procedure. Stereo image pairs are cropped and rotated such that evenly spaced red horizontal lines go through the corresponding pixels, signifying this image pair is well rectified and it will become a reliable input of the subsequent stereo matching algorithm.

3.4 CNN-based Stereo Matching Algorithm

Inspired by the SAD matching cost calculation formula in 2.3, we can obtain the matching cost of two image patches, one is from the left image (reference image) centred at (x, y)

(a) Stereo image pair before rectification



(b) Stereo image pair after rectification

Figure 3.3: Comparative result of stereo image pair before/after the rectification process

while the other is from the right image (target image) centred at $(x - d, y)$. We are aiming at getting a low matching cost if the centres of these two patches correspond to the identical point from the object, and a high cost if they don't.

Given that the correct and wrong matches can be acquired through some public datasets, a supervised learning model will be employed to resolve the stereo correspondence problem. Since the convolutional neural network (CNN) has been successfully applied to the image processing field, we attempt to utilize this approach to measure the similarity level of an arbitrary pair of image patches.

3.4.1 Dataset Construction

We adopt two popular public datasets called Middlebury and KITTI, using their groundtruth disparity maps to build a binary classification dataset. For every pixel whose true disparity value is available, a correct match sample (a positive sample) and a wrong match sample (a negative sample) can be established, avoiding constructing an imbalanced dataset. A

positive sample contains two image patches, one is from the left image while the other is from the right image, and their centre pixels correspond to the same point on the objects. But for a negative sample, there is no such correspondence.

Here's the detailed description of the dataset construction process. Two image patches are given by: $(P^L_{m\times m}(\boldsymbol{c_L}), P^R_{m\times m}(\boldsymbol{c_R}))$, where $P^L_{m\times m}(\boldsymbol{c_L})$ is an $m \times m$ image patch centred at $\boldsymbol{c_L} = (x, y)$ from the left image, and $P^R_{m\times m}(\boldsymbol{c_R})$ is an $m \times m$ image patch centred at $\boldsymbol{c_R}$ from the right image. d defines the true disparity value at $\boldsymbol{c_L}$. A negative match sample is attained through setting the centre pixel of the right image patch as:

$$\boldsymbol{c_R} = (x - d + d_-, y), \tag{3.10}$$

where d_- is an offset leading to the image patch not centring at the same point from the object. It's selected from either $[-neg_{low}, -neg_{high}]$ or $[neg_{low}, neg_{high}]$, obeying the uniform distribution. A positive sample is achieved by letting:

$$\boldsymbol{c_R} = (x - d + d_+, y), \tag{3.11}$$

where d_+ is opted from $[-pos, pos]$ at random, allowing some error tolerance for a positive example. We don't make the offset to be zero but permit the approximate good matches within certain limits (pos was set to be less than one pixel), in terms of the performance of the later post-processing stereo methods, specifically for the cross-based cost aggregation step, which shows better disparity map under this situation.

3.4.2 Network Architectures

Two types of network architectures, which are the fast network and the slow network, are come up with to measure the similarity level between two image patches. The first type performs faster than the second but at the cost of generating a bit less accurate disparity map than the second type. The inputs of both two networks are two image patches and the output is a measurement of the similarity level. The kernel part of these networks lies in the feature extraction step that is used to encode an image patch with a vector. Consequently, the similarity level of two image patches is evaluated by two feature vectors, other than purely referring to the intensity values of image patches. The fast one depends on a traditional similarity metric for assessing two feature vectors, while the slow one tries to learn a new similarity metric for measuring the feature vectors.

Fast Network

The fast network takes the siamese architecture, using the same weights within two network branches. Figure 3.4 indicates the architecture of our fast network. Each branch is made up of multiple convolutional layers accompanied by the rectified linear unit (ReLU) except for the last layer. A feature vector extracting the characteristics of an image patch is outputted from each sub-network. Then the conventional cosine similarity metric makes a comparison between two output feature vectors, generating the similarity score as the final output of this network. The computation of cosine similarity is divided into two steps: normalization and dot product. This strategy helps to save the implementation time because the normalization step only needs to be executed once for per pixel in the image.

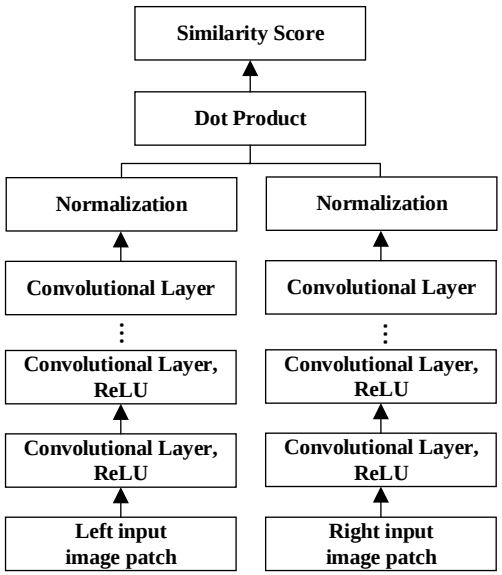


Figure 3.4: Architecture of the fast network

The loss function of this network is a triplet loss, considering a positive sample and a negative sample from an image patch centred at the same location. Suppose o_+, o_- are the output similarity scores of a positive sample and a negative sample, respectively. M, which is the margin loss, is a number greater than zero. The triplet loss for a pair of samples is denoted as $max(0, M + o_- - o_+)$, which means we want the similarity score of the positive

sample is greater than that of the negative sample by at least M. The margin M regulates the boundary between the positive sample and negative sample.

Some hyperparameters are introduced when training this network, including the number of convolutional layers, kernel size, the number of feature maps, and the size of the image patch.

Slow Network

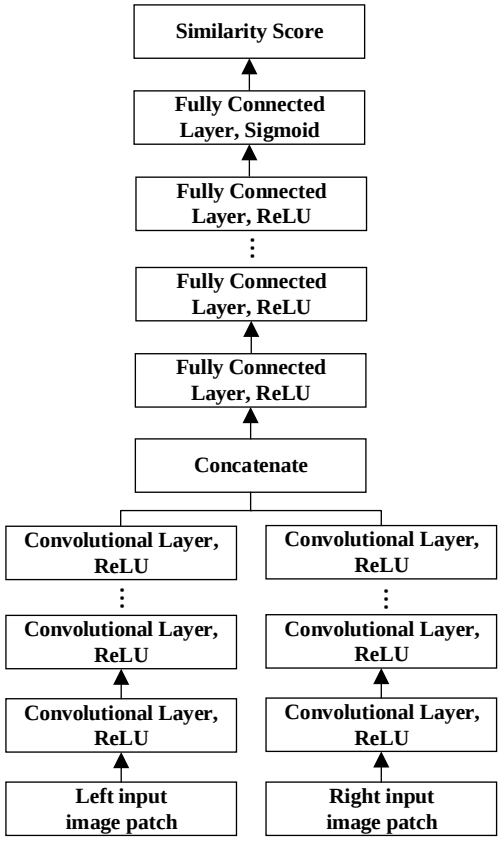


Figure 3.5: Architecture of the slow network

The slow network is motivated by the fast one, indicated by figure 3.5. Instead of applying the commonly used cosine similarity metric, we adopt a couple of fully connected

layers, getting a slightly higher accuracy at the expense of increasing the number of parameters and training time. Multiple convolutional layers with the ReLU activation function are taken in every branch network. Two feature vectors are then concatenated and passed through several fully connected layers with ReLU as well. Finally, the last fully connected layer is ended up with a sigmoid function, producing a number ranging from 0 to 1, which can be regarded as the measurement of similarity level for a pair of image patches.

Because this network is similar to a binary classifier, we recommend choosing the binary cross-entropy loss. Here we set o_i to be the output similarity score and y_i to be the label of the input sample. y_i is 1 if the input belongs to the positive sample while y_i equals to 0 when the input is a negative sample. The loss for this sample is denoted as $y_i \cdot log(o_i) + (1 - y_i) \cdot log(1 - o_i)$. Since the cross-entropy loss isn't suitable for calculating the cosine similarity in the fast network, we determine to select different loss functions under two types of network architectures.

Apart from the mentioned hyperparameters in the fast network, the number of fully connected layers and the number of neurons in each fully connected layer are brought in as well.

3.4.3 CNN-based Matching Cost Calculation

The matching cost between two image patches is obtained based on the output of the network:

$$C_{CNN}(\boldsymbol{c_L}, d) = -O(P^L_{m\times m}(\boldsymbol{c_L}), P^R_{m\times m}(\boldsymbol{c_L} - \boldsymbol{d})), \tag{3.12}$$

where the RHS is the outcome of the network when comparing two image patches $P^L_{m\times m}(\boldsymbol{c_L})$ and $P^R_{m\times m}(\boldsymbol{c_L} - \boldsymbol{d})$. Taking the opposite value of the similarity score is equivalent to calculating the matching cost.

To get the matching cost tensor $C_{CNN}(\boldsymbol{c_L}, d)$ of the entire stereo image pair, we need to pass through every location in the image and consider each possible disparity value. Some technical tricks are elaborated to make this algorithm feasible to process an image in limited time.

- The feature vector of an image patch simply needs to be computed once for every location in the image, regardless of the value of disparity.
- The similarity score calculation can be implemented by feeding the whole image into the network at a time rather than image patches. Processing a complete $w \cdot h$ image with the network once is faster than dealing with $w \cdot h$ image patches thanks to reusing the intermediate results.

- The bottleneck of the slow network is due to a lack of the reusability of temporary results. We need to go through the fully connected layers under different disparity values. Assume the maximum disparity is d_{max}, the fully connected part has to be carried out d_{max} times. By contrast, the part which needs to be run d_{max} times for the fast network is the dot product between two feature vectors. d_{max} is 228, 400 for the KITTI, Middlebury dataset separately.

Therefore, no matter which kind of network we prefer, to gain the matching cost of a stereo image pair, the branch network will run for each image once, along with performing the following fully connected layers/dot product step for d_{max} times.

3.4.4 Post-processing Techniques

The raw matching cost derived from the CNN-based network can't satisfy the precision need of an eligible disparity map, producing remarkable errors in untextured regions and occlusions. To improve the quality of the disparity map, a set of post-processing techniques consisting of cost aggregation and disparity refinement are realized. Stimulated by [35] and [56], these techniques cover cross-based cost aggregation, semi-global cost aggregation, interpolation, and image filtering operations.

3.4.5 Cost Aggregation

Cross-based Cost Aggregation

For local methods in the stereo matching algorithm, a fixed-size aggregation window is frequently constructed in order to abstract the features from the adjacent pixels. Nevertheless, this trick doesn't work around depth borders attributed to the smoothness assumption. So we need to customize the range of a window for every pixel in the image to ensure that the neighbouring pixels are approximately from the same subject in the scene. For the cross-based cost aggregation approach proposed in [57], a special-shape window around each pixel is built via enclosing pixels whose intensity values are close to that of the centre pixel. We presume that pixels with similar intensities are more likely to belong to the same object.

For every pixel in the image, a vertical cross will be established to construct the special window. The left branch centred at $\boldsymbol{c}$ grows to the left at $\boldsymbol{c_l}$, provided that two conditions are met as below:

- $|I(\boldsymbol{c}) - I(\boldsymbol{c_l})| < \boldsymbol{threshold}$: the absolute intensity difference between pixels at $\boldsymbol{c}$ and $\boldsymbol{c_l}$ needs to be no more than $\boldsymbol{threshold}$.
- $\|\boldsymbol{c} - \boldsymbol{c_l}\| < \boldsymbol{distance}$: the horizontal distance (or vertical distance if it is upward or downward branch) between $\boldsymbol{c}$ and $\boldsymbol{c_l}$ is less than $\boldsymbol{distance}$.

Arms from other directions (up, down, right) are built similarly. Then we can work out the special-shape window $W(\boldsymbol{c})$, which is the union of horizontal branches of all the $\boldsymbol{c_0}$ located on the vertical branch of $\boldsymbol{c}$, like figure 3.6 demonstrated. Furthermore, the construction of

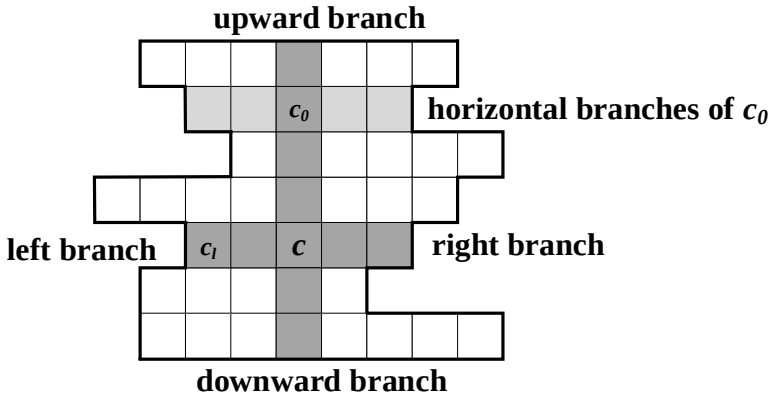


Figure 3.6: The special-shape window $W(\boldsymbol{c})$

$W(\boldsymbol{c})$ will necessitate combining the windows from both images in a stereo pair. W^L and W^R represent the windows from the left and right images individually. The union of two windows are expressed as:

$$W(\boldsymbol{c}) = \{\boldsymbol{c_0} | \boldsymbol{c_0} \in W^L(\boldsymbol{c}), \boldsymbol{c_0} - \boldsymbol{d} \in W^R(\boldsymbol{c} - \boldsymbol{d})\}. \tag{3.13}$$

The aggregated cost will be the average result within the joint window:

$$\begin{aligned} C_W^0(\boldsymbol{c_L}, d) &= C_{CNN}(\boldsymbol{c_L}, d), \\ C_W^j(\boldsymbol{c_L}, d) &= \frac{1}{|W_d(\boldsymbol{c_L})|} \sum_{\boldsymbol{c_0} \in W_d(\boldsymbol{c_L})} C_W^{j-1}(\boldsymbol{c_0}, d), \end{aligned} \tag{3.14}$$

where j stands for the iteration index. The averaging operation will be executed a couple of times on account of the intersection of different windows, varying the matching cost every time. Considering this approach has an expensive computational cost, it is only utilized on the slow network to increase the accuracy of the disparity map. To expedite the process of matching cost calculation, we overlook this step in the fast model.

Semi-global Cost Aggregation

To supplement the early smoothness assumption, an extra condition is set by giving a penalty to variation of ambient disparities. According to [27], an energy function $E(D)$ is introduced by denoting:

$$E(D) = \sum_{\boldsymbol{c}} \Bigg(C_W^{i_{max}}(\boldsymbol{c}, D(\boldsymbol{c})) + \sum_{c_0 \in N_{\boldsymbol{c}}} P_1 \cdot 1\{|D(\boldsymbol{c}) - D(\boldsymbol{c_0})| = 1\} + \sum_{c_0 \in N_{\boldsymbol{c}}} P_2 \cdot 1\{|D(\boldsymbol{c}) - D(\boldsymbol{c_0})| > 1\} \Bigg), \tag{3.15}$$

where $1\{\cdot\}$ is the indicator function and i_{max} represents the maximum number of iterations. The first term is the summation of matching costs for all disparities in D. The second term gives penalty P_1 to neighbouring pixels of $\boldsymbol{c}$ whose offset disparity is one. While a larger penalty P_2 is added for pixels whose disparities differ greater than one.

Our goal is to refine the matching cost in a manner of finding the disparity map D that minimizes energy function $E(D)$ with 1D dynamic programming instead of the global optimization. However, this method would suffer from the streaking problem because of the trouble with building relationships among optimized rows in a 2D image. Semi-global method aggregates the 1D minimum matching costs from multiple directions, and take the average to get the matching cost. Despite sixteen cost paths were chosen in the original work, we only care about the horizontal and vertical directions. The matching cost $C_{\boldsymbol{r}}(\boldsymbol{c}, d)$ along $\boldsymbol{r}$ direction has the below recursive relationship:

$$\begin{aligned} C_{\boldsymbol{r}}(\boldsymbol{c}, d) = C_W^{i_{max}}(\boldsymbol{c}, d) + \min \Big\{ & C_{\boldsymbol{r}}(\boldsymbol{c} - \boldsymbol{r}, d), C_{\boldsymbol{r}}(\boldsymbol{c} - \boldsymbol{r}, d-1) + P_1, \\ & C_{\boldsymbol{r}}(\boldsymbol{c} - \boldsymbol{r}, d+1) + P_1, \min_m C_{\boldsymbol{r}}(\boldsymbol{c} - \boldsymbol{r}, m) + P_2 \Big\} - \min_m C_{\boldsymbol{r}}(\boldsymbol{c} - \boldsymbol{r}, m). \end{aligned} \tag{3.16}$$

The last term, which is the minimum matching cost of the previous step along the path $\boldsymbol{r}$, is deducted from the total cost to avoid the overflow of $C_{\boldsymbol{r}}(\boldsymbol{c}, d)$ without affecting the disparity values.

The penalty variables P_1 and P_2 are determined by intensity gradient since disparities vary with the edges in the stereo image pairs. Intensity difference between two adjacent pixels located on the path we are minimizing can be written as $\Delta I_1 = |I^L(\boldsymbol{c}) - I^L(\boldsymbol{c}-\boldsymbol{r})|$ and $\Delta I_2 = |I^R(\boldsymbol{c}-\boldsymbol{d}) - I^R(\boldsymbol{c}-\boldsymbol{d}-\boldsymbol{r})|$. P_1 and P_2 are defined as the following:

$$\begin{array}{lll} P_1 = P_{10}, & P_2 = P_{20} & \text{if } \Delta \mathrm{I}_1 < \mathrm{I}_0,\ \Delta \mathrm{I}_2 < \mathrm{I}_0; \\ P_1 = P_{10}/P_{31}, & P_2 = P_{20}/P_{31} & \text{if } \Delta \mathrm{I}_1 \geq \mathrm{I}_0,\ \Delta \mathrm{I}_2 \geq \mathrm{I}_0; \\ P_1 = P_{10}/P_{30}, & P_2 = P_{20}/P_{30} & \text{otherwise.} \end{array} \tag{3.17}$$

P_{10} and P_{20} are the initial penalty at discontinuous depth borders. They will be divided by various factors when both ΔI_1 and ΔI_2 or one of them suggests a large intensity gradient. Again, P_1 will be decreased by dividing P_V when $\boldsymbol{r}$ appears in the vertical directions. By analyzing the groundtruth disparity map, we found out that the probability of disparity changing vertically is higher than that happens horizontally and thus a smaller penalty is reasonable.

The eventual semi-global matching cost $C_{SG}(\boldsymbol{c}, d)$ is calculated through averaging all four cost paths:

$$C_{SG}(\boldsymbol{c}, d) = \frac{1}{4} \sum_{\boldsymbol{r}} C_{\boldsymbol{r}}(\boldsymbol{c}, d). \tag{3.18}$$

3.4.6 Disparity Map Calculation and Refinement

The disparity map $D(\boldsymbol{c})$ is obtained with WTA strategy, which calculates each value by finding the corresponding disparity d with the minimum $C(\boldsymbol{c}, d)$:

$$D(\boldsymbol{c}) = \arg\min_{d}\ C(\boldsymbol{c}, d). \tag{3.19}$$

Interpolation

The purpose of interpolation is to solve the disparity conflict phenomenon between two predicted disparity maps based on the left/right image. $D^L(\boldsymbol{c})$, $D^R(\boldsymbol{c})$ are the disparity maps regarding left/right image as the reference image. Following our consistent setting, we have $D^L(\boldsymbol{c}) = D(\boldsymbol{c})$. For a particular pixel with known groundtruth disparity, D^L and D^R bring contradictory predictions occasionally. So left/right consistency check is accomplished to detect these conflicts. We classify every pixel $\boldsymbol{c}$ in the image into three

categories obeying some rules:

$$\begin{aligned} &\text{if } |\mathrm{d} - \mathrm{D}^{\mathrm{R}}(\boldsymbol{c} - \boldsymbol{d})| \leq 1 \text{ when } \mathrm{d} = \mathrm{D}^{\mathrm{L}}(\boldsymbol{c}), && \textit{correct}, \\ &\text{if } |\mathrm{d} - \mathrm{D}^{\mathrm{R}}(\boldsymbol{c} - \boldsymbol{d})| \leq 1 \text{ when } \mathrm{d} \neq \mathrm{D}^{\mathrm{L}}(\boldsymbol{c}), && \textit{mismatch}, \\ &\text{otherwise}, && \textit{occlusion}. \end{aligned} \tag{3.20}$$

For occluded pixels, we will alternatively choose a proper disparity with interpolation, moving to the left till the first *correct* pixel is found and replacing the occluded disparity with this value. To interpolate the *mismatch* pixels, the median disparity of pixels which are nearest to the *correct* pixel are carefully selected from sixteen directions. After realizing the interpolation operation, the intermediate disparity map D_I is generated.

Image Filtering Operations

At last, our disparity map is refined by means of a median filter with 5×5 kernel and a bilateral filter defined as:

$$D_B(\boldsymbol{c}) = \frac{1}{Q(\boldsymbol{c})} \sum_{\boldsymbol{c_0} \in \boldsymbol{N_c}} D_I(\boldsymbol{c_0}) \cdot G(\|\boldsymbol{c} - \boldsymbol{c_0}\|) \cdot 1\{I^L(\boldsymbol{c}) - I^L(\boldsymbol{c_0}) < T\}, \tag{3.21}$$

where $G(\|\boldsymbol{c} - \boldsymbol{c_0}\|)$ is a Gaussian function with zero mean and σ standard deviation. $Q(\boldsymbol{c})$ represents the normalization term, denoted as:

$$Q(\boldsymbol{c}) = \sum_{\boldsymbol{c_0} \in \boldsymbol{N_c}} G(\|\boldsymbol{c} - \boldsymbol{c_0}\|) \cdot 1\{I^L(\boldsymbol{c}) - I^L(\boldsymbol{c_0}) < T\}. \tag{3.22}$$

The median filter plays the role of removing the noise in the disparity map. While the bilateral filter smooths the disparity map and preserves the edges at the same time. D_B is the eventual disparity map processed by all of the post-processing techniques mentioned above.

Chapter 4

Experiments and Results

4.1 Datasets

We select three major stereo datasets for training and validating purposes in our experiments, including Middlebury, KITTI 2012[21], and KITTI 2015[36]. Each dataset consists of the training set and test set, with available groundtruth disparities for the training set. For the test set, their groundtruth disparity maps are hidden in the server online in order to compare with other existing stereo matching algorithms according to some metrics. Therefore, we split the training set into a smaller training set as well as a validation set, doing performance analysis under self-defined metrics. The error rate is denoted as the percentage of pixels whose difference results between the groundtruth disparity and the predicted one is greater than 3 pixels. If we convert the pixel unit into the actual depth, that is, we allow the measurement error to be no more than 3 centimetres if the true distance between the subject and camera lens is 2 meters.

4.1.1 KITTI Dataset

KITTI dataset is a series of rectified stereo image pairs captured by two high-resolution video cameras installed on the roof of a car with a baseline around 54 centimetres. The scenes are recorded in rural areas and on highways when driving around the city of Karlsruhe. The maximum resolution of the taken images is 1242×375. The accurate groundtruth depth is provided by a 360° Velodyne laser scanner, producing depth information for around 30% of the total image pixels. There are two KITTI datasets, KITTI

2012 and KITTI 2015, with subtle differences between these two data sources for the stereo matching task. The dataset from 2012 contains 194 stereo pairs for training and 195 for testing, while the one from 2015 is made up of 200 training pairs and 200 test pairs separately. Compared with the earlier dataset in 2012, the newer dataset in 2015 introduces window glass, focusing on evaluating the method's ability to handle the situation of reflection surfaces.

4.1.2 Middlebury Dataset

The stereo image pairs in the Middlebury datasets are mainly caught in the indoor scenes under well-defined lighting conditions. To get the groundtruth disparity maps, structured light was applied to collect the depth data more precisely and densely than that sampled in the KITTI dataset. The datasets were released in five respective papers in 2001[45], 2003[46], 2005[44], 2006[28] and 2014[43]. In our thesis, the combination of the above five datasets is collectively known as the Middlebury dataset. Some detailed descriptions are listed in table 4.1. The stereo image pairs will only include samples whose groundtruth disparity maps are accessible. To boost the robustness of our model, we would apply imperfect rectified stereo image pairs into the training stage. The latest three stereo datasets were constructed under several different illuminations and exposures for the same scene. Besides, the datasets offer full-size, half-size, and third-size resolution. The disparity map error is measured in the full-size image. As a result, if a disparity map based on the half-size or third-size stereo pair is produced, it needs to be upsampled before the error calculation. We determine to implement our algorithm on the half-size images on account of the limitation of GPU memory.

Dataset Name	Number of Stereo Image Pairs	Maximum Resolution	Maximum Disparity
2014 dataset	23	3000×2000	800
2006 dataset	21	1400×1100	230
2005 dataset	6	1400×1100	230
2003 dataset	2	1800×1500	220
2001 dataset	8	380×430	30

Table 4.1: Details about five stereo datasets from Middlebury

4.1.3 Dataset Augmentation

Data augmentation is a commonly used approach to improve the network's performance and its ability to generalize. Various image transformations (e.g. rotation, translation, scaling, contrast, and brightness) will be exerted on the extracted image patches, without influencing the groundtruth disparities.

Name of Transformation	Transformation Parameters	
	Left Patch (P^L)	Right Patch (P^R)
Rotation	θ	$\theta + \Delta\theta$
Scaling	s	$s \cdot \Delta s$
Horizontal Scaling	s_h	$s_h \cdot \Delta s_h$
Horizontal Shearing	hs	$hs + \Delta hs$
Vertical Translation	None	d_v
Brightness Change	br	$br + \Delta br$
Contrast Adjustment	con	$con \cdot \Delta con$

Table 4.2: Procedures of data augmentation carried out on a pair of image patches

For a pair of image patches, a set of transformations will be imposed on this pair with distinct parameters. The parameters of each transformation are randomized for image patches, varying from different training epochs. For instance, the left patch is scaled by a coefficient of 0.9, while the right patch is scaled by another coefficient of $0.9 \cdot 0.8$. However, it's a remarkable fact that diverse datasets have their corresponding suitable choices for data augmentation. Improper image transformation or parameters may lead to higher training errors.

On account of the available images under multiple illuminations and exposures in the Middlebury dataset, we can utilize this feature in the data augmentation. For two datasets from KITTI, we keep the same parameter scope for all image transformations owing to their similar shooting environments.

To overcome the difficulty of imperfect rectification in the Middlebury dataset, vertical translation was introduced between a pair of image patches. The procedures of data augmentation are elaborated in table 4.2, where the multiplication and addition are operated in an element-wise manner.

Table 4.3 shows the augmentation hyperparameters taken by the training sets.

Hyperameters	KITTI 2012/2015	Middlebury
	Range	Range
θ	[-7, 7]	[-28, 28]
s	[0.9, 1]	[0.8, 1]
s_h	[0.9, 1]	[0.8, 1]
hs	[0, 0.1]	[0, 0.1]
d_v	None	[0, 1]
br	[0, 0.5]	[0, 1.2]
con	[1, 1.4]	[1, 1.2]
$\Delta\theta$	[-1, 1]	[-2, 2]
Δs	[0.9, 1]	[0.8, 1]
Δs_h	[0.8, 1]	[0.9, 1]
Δhs	[0, 0.1]	[0, 0.2]
Δbr	[0, 0.2]	[0, 0.6]
Δcon	[1, 1.3]	[1, 1.1]

Table 4.3: Hyperparameters of augmentation technique applied to the datasets

4.2 Details of Training

Every stereo image was pretreated through deducting the mean and dividing by the standard deviation of the overall intensity values. We prefer using the greyscale images instead of the RGB ones because the subsequent experimental results have proven that additional colour channels even worsened the disparity maps occasionally. We built the dataset so that it's suitable for a binary classifier from all of the attainable training sets. For the KITTI dataset, we removed the corresponding pixels that appear to be non-visible, occluded as well as white in the disparity maps. Analogously, we eliminated some invalid areas in the Middlebury dataset on the basis of the given mask images marking the non-occluded areas. The entire dataset comprises 24 million samples or so from KITTI 2012, 16 million samples from KITTI 2015, and 37 million samples from the Middlebury stereo dataset.

In the training stage, we fed a batch composed of 128 pairs of image patches into the network. While during the testing, a full stereo image pair was regarded as the input to the network. Although we can adopt the full-scale images as the training input, there are a few advantages of choosing image patches. It's not only more convenient to regulate the batch size but also better to balance the quantity of positive and negative samples in a batch. Additionally, the application of the shuffling technique within a batch makes it

possible to bring in diverse stereo image pairs in a batch.

We optimize the loss function with stochastic gradient descent, accompanying with a momentum parameter adjusted to be 0.9. The number of epochs is set to be 16 with an initial learning rate of 0.003 for the fast network and 0.004 for the slow one. A divisor of 10 was imported from the 13^{th} epoch to avoid the overfitting problem. With the help of K-fold cross-validation, these hyperparameters (epochs, learning rate, and their way of adjustment) can be tuned to achieve the optimal values. Table 4.4 demonstrates the hyperparameters we finally employed in the training process. It's noteworthy that these hyperparameters are targeted at the preprocessed stereo images rather than the raw images.

As for the training time, it varies from the size of the dataset and the network's complexity. Even for the more sophisticated slow architecture with larger dataset Middlebury, we spent no more than two days finishing the training process.

Hyperparameters	Middlebury		KITTI 2012		KITTI 2015	
	fast	slow	fast	slow	fast	slow
Image patch size	11×11	11×11	9×9	9×9	9×9	9×9
# of convolutional layers	5	5	4	4	4	4
# of feature maps	64	112	64	112	64	112
kernel size	3	3	3	3	3	3
# of fully connected layers		3		4		4
# of neurons in each fc layer		384		384		384
neg_low	1	1	3	3	3	3
neg_high	5	17	9	9	9	9
pos	1	1	1	1	1	1
threshold		0.03		0.1		0.04
distance		12		6		6
P_{10}	2.6	1.6	3	1.38	2.8	2.8
P_{20}	57	19.2	220	30	41.2	54.7
P_{30}	3.5	4	3	3	3	3
P_{31}	7	8	6	6	6	6
P_V	1.6	2.6	1.5	2	1.36	1.76
I_0	0.08	0.13	0.03	0.06	0.06	0.06
T	3	3	5	6	5	5
σ	6	1.8	7.5	6.5	4.5	6

Table 4.4: Hyperparameters of the fast and slow network

4.3 Traditional Method versus CNN-based

In Chapter 2, we made a comprehensive comparison among six types of approaches to calculating the matching costs and decided to use the census transform as the benchmark. Table 4.5 exhibits the error rate and runtime of the fast, slow network as well as census transform on the validation set from KITTI 2012, KITTI 2015, and Middlebury. For each dataset, we randomly chose 40 stereo image pairs from the training set as the validation set. The runtime is measured based on the below image size and maximum disparity: 1242×375 with 228 disparity levels for KITTI dataset, 1500×1000 with 200 disparity levels for Middlebury dataset. The runtime and error rate in the diagram are obtained by averaging these 40 samples.

Method Name	KITTI 2012		KITTI 2015		Middlebury	
	Error Rate (/%)	Runtime (/s)	Error Rate (/%)	Runtime (/s)	Error Rate (/%)	Runtime (/s)
CNN-based Fast	2.95	0.39	3.57	0.37	9.45	0.84
CNN-based Slow	2.26	31.47	2.66	32.07	7.03	34.04
Census Tranform	12.40	0.47	11.60	0.47	25.28	1.21

Table 4.5: Error rate and runtime contrast among two CNN-based methods and census transformation approach

Apparently, our CNN-based methods have an overwhelming advantage in the accuracy of the disparity map. There's a trade-off between the error rate of disparity prediction and processing time, which is the major performance distinction between these two kinds of architectures. Compared with the fast network, the slow one generated more precise results in every used dataset, but at the expense of heavy computational load due to the structure of multiple fully-connected layers, which brought in way more parameters that need to be tuned. While the fast network has a higher processing speed with a reasonable error rate. The fast network is at least 40 times faster than the slow one. Furthermore, the error rate difference between the slow network and the fast one isn't significant. The worse error difference is 2.42% for the Middlebury dataset, which seems to be the most challenging case for all three methods. The best case happens when we move to the KITTI 2012, merely having a 0.69% error rate difference.

Figure 4.1, 4.2 and 4.3 display the example of stereo image pairs, groundtruth, and their corresponding predicted disparity maps, which were chosen from the three datasets respectively. To make it easier to distinguish small changes in disparity levels (grayscale intensities) in the same image, a simple way to pseudocolour a grayscale image utilizing OpenCV's predefined colourmaps was taken. We applied colourmap *COLORMAP_JET* to the images. Lower disparities, which means greater depths, are replaced by blue while higher values tend to be red. We can observe that the census transformation technique caused more errors in untextured regions and lost many details of the objects. Whereas the two CNN-based methods depicted the contours of main targets more accurately.

4.4 Result of Transfer Learning

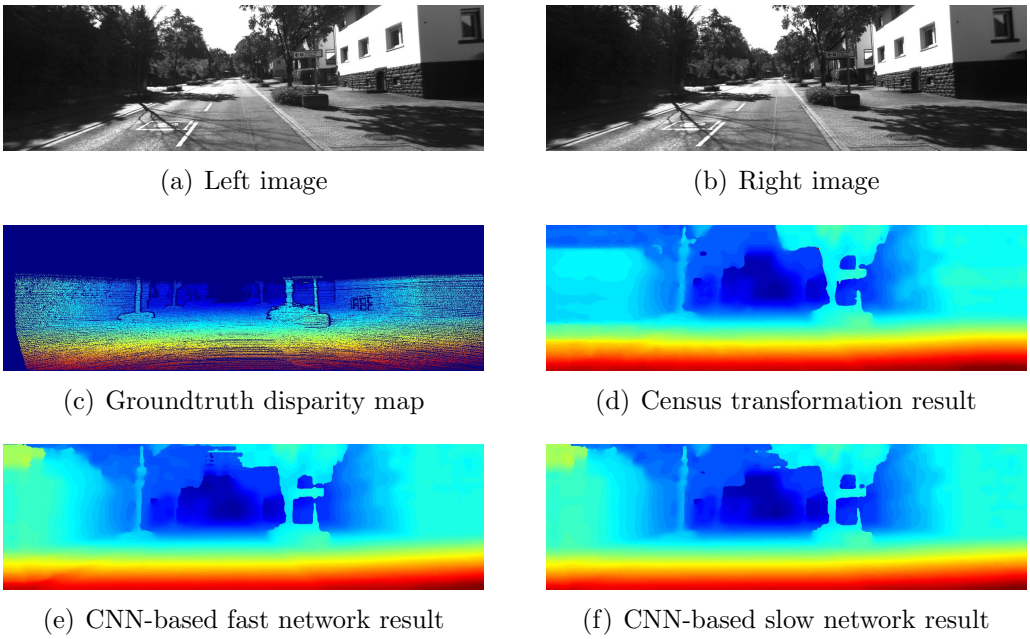
Error Rate (/%)		Validation Set					
		KITTI 2012		KITTI 2015		Middlebury	
		Fast	Slow	Fast	Slow	Fast	Slow
Training Set	KITTI 2012	2.95	2.26	3.73	3.58	12.21	11.21
	KITTI 2015	3.58	3.91	3.57	2.66	13.29	14.38
	Middlebury	3.02	2.92	3.96	3.94	9.45	7.03

Table 4.6: Summary of validation error when training set and validation set are different

So far all the results derived from the proposed methods are based on the same training set and validation set, which isn't consistent with the real circumstances since it's unlikely to train a specific network without the groundtruth disparity maps. Hence we execute our algorithm on the dataset that wasn't used to train the model. For example, the KITTI 2012 dataset served as the training dataset but the validation error was figured out on the KITTI 2015 dataset. The relevant experimental results are shown in table 4.6, revealing some exciting outcomes. When we appraise the pre-trained models based on an arbitrary dataset, they performed equally well on two KITTI validation sets, comparing with the results of using the same dataset for training and validation. For the KITTI 2012 validation set, the slow network based on Middlebury got even a lower error rate than that using the fast model trained on KITTI 2012.

Also, we found that if we treat the Middlebury as the training set, the validation errors on the KITTI are very close to the error we got when using KITTI as the training set as well. So the networks trained on the Middlebury dataset can be migrated well to all KITTI datasets.

When both accuracy and runtime are considered together, we prefer the fast network, which is more likely to satisfy the goal of producing a precise disparity map as soon as possible. Thus in the later section, we will adopt the CNN-based fast network as the framework of the stereo matching algorithm, applying it into our system.



(a) Left image

(b) Right image

(c) Groundtruth disparity map

(d) Census transformation result

(e) CNN-based fast network result

(f) CNN-based slow network result

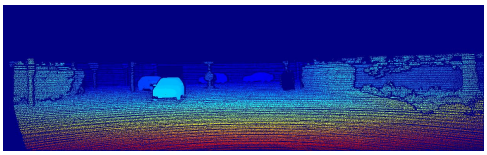
Figure 4.1: Sample stereo image pair and prediction results from KITTI 2012 dataset

4.5 System Performance Analysis

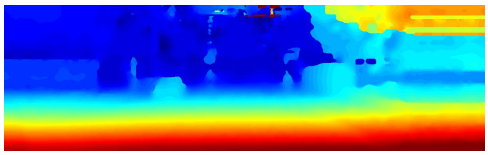
In previous sections, we concentrated on analyzing the network on the base of standard stereo vision datasets, from the aspects of runtime and error rate. To further verify the feasibility of the presented approach, we performed the CNN-based stereo matching algorithm on the captured image pairs with our stereo camera rig and investigated the performance of our system.

(a) Left image

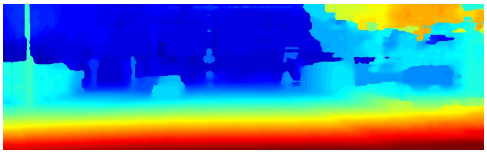
(b) Right image



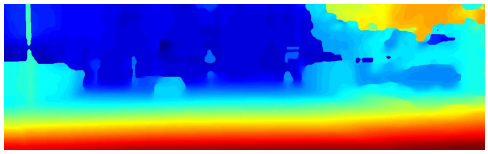
(c) Groundtruth disparity map



(d) Census transformation result



(e) CNN-based fast network result



(f) CNN-based slow network result

Figure 4.2: Sample stereo image pair and prediction results from KITTI 2015 dataset

4.5.1 Two Types of Error Criteria

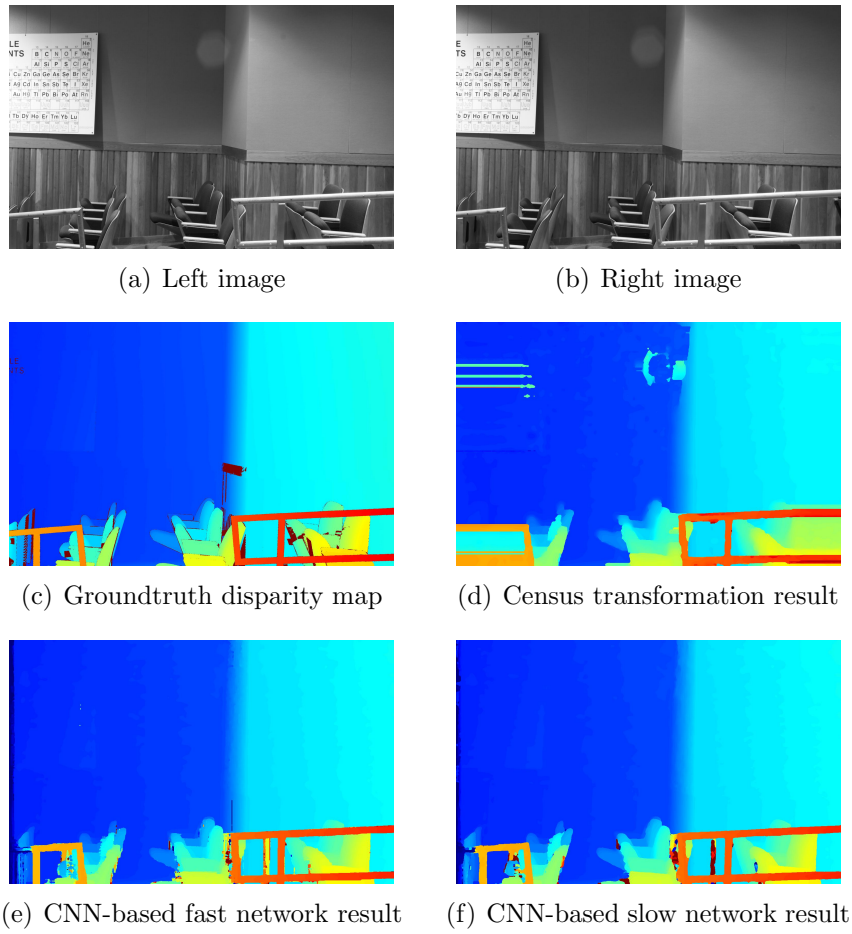
To assess the behaviour of the binocular stereo vision system, the error rate relying on the percentage of wrongly predicted pixels has some limitations. The percentage can't tell us how serious is the disparity miscalculation or the error distribution among various disparity levels. As a consequence, two new types of criteria evaluating the quality of the disparity map are put forward.

To begin with, the histogram of the groundtruth disparity map is required to be the reference of computing the disparity errors. Note that in reality both the groundtruth and predicted disparity values are floating numbers instead of integers. Therefore, we divide the disparity levels into d_{max} intervals, and all pixels are counted according to their corresponding intervals. Meanwhile, the disparity masks labelling the locations of pixels whose disparities are in the same interval can be generated.

For every pixel, the absolute and relative disparity error are defined like the following:

$$Err_a = |d_{pred} - d_{real}|, \qquad Err_r = \frac{|d_{pred} - d_{real}|}{d_{real}}, \tag{4.1}$$

where d_{pred} and d_{real} stand for the predicted as well as real disparity values. The absolute error describes the direct difference between the achieved and expected disparity value. While the relative error expresses the significance level of error difference with respect to the truth value. Then these two types of errors belonging to the same disparity interval can be accumulated and taken the average separately, acquiring the average absolute/relative error per pixel in various disparity intervals. From 1.1, we realize that the disparity is



(a) Left image (b) Right image

(c) Groundtruth disparity map (d) Census transformation result

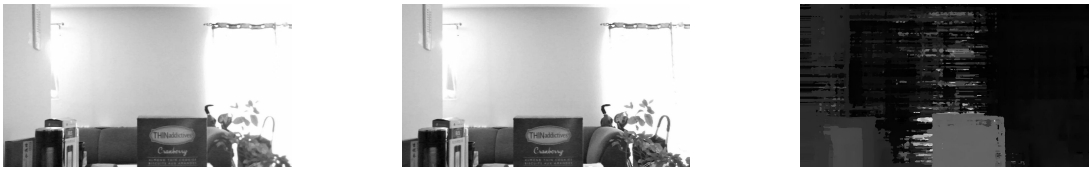
(e) CNN-based fast network result (f) CNN-based slow network result

Figure 4.3: Sample stereo image pair and prediction results from Middlebury dataset

inversely proportional to depth under the same set of camera parameters, which suggests that analyzing the disparity error is equivalent to depicting the depth error distribution. So suppose we fix the baseline and focus distance of the stereo camera, the average absolute/relative error gives us a rough estimation about the appropriate range of working

distance for detecting the depth. Moreover, we can attempt to explore the relationship between the most accurate depth range and its baseline setting under certain error tolerance. One more thing that's necessary to be explained is that since the camera parameters were dynamically changed with the positions of target objects in the scenes, it's hard to utilize these two indicators to assess the system performance on the mentioned standard datasets.

4.5.2 Results of Stereo Camera Rig



(a) Left image (b) Right image (c) Disparity map using fast network

Figure 4.4: A rectified stereo image pair and its disparity map when the baseline is 47.8mm

Given that the groundtruth disparity/depth maps are unavailable for any scenes, we consider building a stereo scene such that the distance between the object and camera lens is known. For simplicity, we seek to place a plane in front of the camera lens. The surface pattern of this plane shouldn't be repetitive or untextured but it had better with some irregular characteristics. Aimed to maintain a constant distance between any point from this plane and the plane where the camera lens lies in, this plane is supposed to be perpendicular to the desktop. To minimize the measurement errors, we try to take a relatively regular-shaped box with multi-coloured letters as the object, ensuring that the front of the box is parallel to the camera lens as much as possible and using the measuring tape to figure out the actual depths.

In the premise of fixing the baseline and focus distance, we capture a series of stereo image pairs at different distances by manually moving the box back and forth, followed by going through the CNN-based stereo matching algorithm and obtaining the disparity maps. For every stereo image pair, the area of interest (AOI) of the box is determined by fitting it with a rectangular box and taking down the coordinates of its four corners. In the case of an image pair with the minimum depth a_1, we artificially set its groundtruth disparity d_1 via averaging the intensity values within the drawn rectangle from the disparity map. Thanks to the constant baseline b and focal length f, the product value of depth Z and

disparity d is invariable, denoted as M. Consequently, the groundtruth disparity d_n under other depth a_n is derived by letting M divided by a_n.

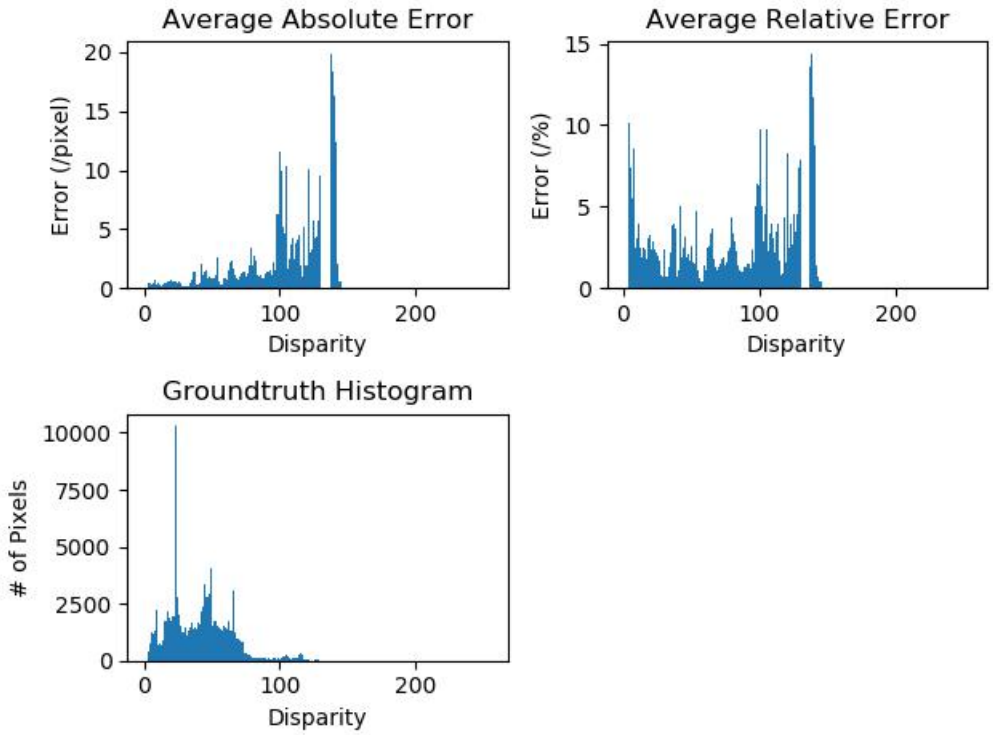


Figure 4.5: The distribution of average absolute/relative error per pixel and the histogram of groundtruth disparity map based on a sample stereo image pair

Figure 4.4 indicates a sample of rectified stereo image pairs taken by our stereo camera and the predicted disparity map. Unfortunately, the quality of disparity map coming out of the self-designed scene recorded by our stereo camera rig is worse than that from the standard datasets. Incorrect disparity predictions not only happen on partial of the box's front surface but also in most of the background part. A couple of potential reasons may account for this phenomenon. The resolution and performance of our stereo camera is virtually the low-configuration of the camera used in standard datasets, bringing about blurriness and noises in the images. Again, we hope that the exposure of the left image is ideally consistent with the right one, which enhances the probability of finding out the correct disparity with our stereo matching algorithm. Whereas it's big trouble for us to control the indoor lighting condition as we took the photos in the daytime. Concerning

the camera lens, the focal length is another factor that affects image sharpness. Since our goal is to acquire the depth information within one meter, a short-focus lens was selected and hence the disparity values which are far from the lens are unreliable. What's more, the white wall and curtain where the light goes constitute the major component of the background in our scenes, which brings difficulty of attaining the disparities, attributed to lack of textures.

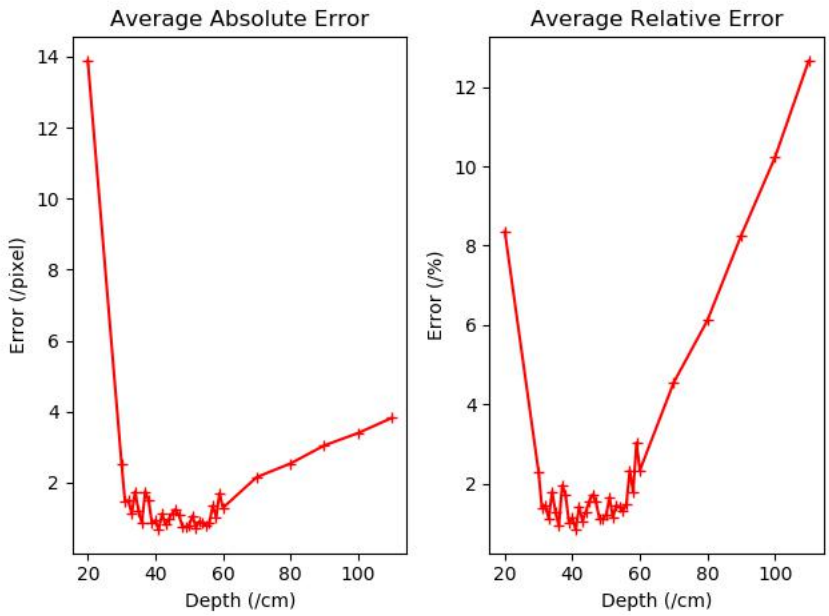


Figure 4.6: Average absolute/relative error per pixel-depth curve depending on the whole box surface when the baseline is 35.17mm

We designed a variety of experiments to seek the proper range of working distance for depth perception. Three baseline lengths (35.17mm, 47.8mm, and 66mm), which correspond to three groups of camera parameters were selected for testing purpose. A baseline length was reached by manually changing the distance between two optical centres of the camera lenses. After that, the camera calibration and rectification procedures were carried out and a set of camera parameters were computed. By continuously altering the distance between the front side of the box and camera lens, we sampled a cluster of stereo image pairs. The distance varies from 20cm to 100cm and the physical interval of per capturing is initially set to be 10cm. Following the trend of the error-depth curve, the depth range where the working distance most probably lies in is determined. Afterward, we continue

to collect the stereo image pairs in this range at 1cm intervals. Based on the existing camera parameters, the stereo image pairs were rectified and considered as the input of the fast network. Our stereo matching algorithm dealt with these data and generated the final disparity maps, which contributed to establishing the relationship between the actual depths and groundtruth disparities. Then we can analyze the average absolute/relative error per pixel under different disparity intervals for a specific baseline length.

(a) Left image (b) Right image (c) Disparity map using fast network

Figure 4.7: Example of a stereo image pair and its predicted disparity map affected by lens reflection

For the purpose of demonstration, a validation set including 40 stereo image pairs from KITTI 2012 was employed to calculate the average absolute/relative error per pixel with the aid of the groundtruth disparity maps. From figure 4.5, which exemplifies the error-disparity relation, the groundtruth disparities are mainly distributed in the range of less than 100, where the maximum average absolute error per pixel is no more than ten pixels. The corresponding relative error is lower than 10% as well. Through observing the fluctuation of our diagrams, although the average absolute error isn't obvious when the disparity value is under 100, its significance level becomes comparatively high when considering the average relative error.

Figure 4.6 exhibits the error-depth curves when we adjust the baseline distance to be 35.17mm. The results of two types of errors in the early experiments suggest that the working distance is likely to be located in the interval of 30cm to 60cm. This sketchy range is refined by sampling more stereo image pairs. Despite the curves appear great fluctuation, we can still infer that the working distance is in the scope of 30cm to 60cm if we set the rules that the average absolute error is less than two pixels and the relative one is restricted to be 2%. The values of these two kinds of error criteria begin to increase beyond this range. An interesting discovery is that a higher error occurs when the object is a bit closer to the camera lens. As shown by figure 4.7, by further analyzing the distinction within the same set of stereo image pairs and their produced disparity maps, we found out that the effect of box's reflection is non-negligible at a close distance, which caused striking

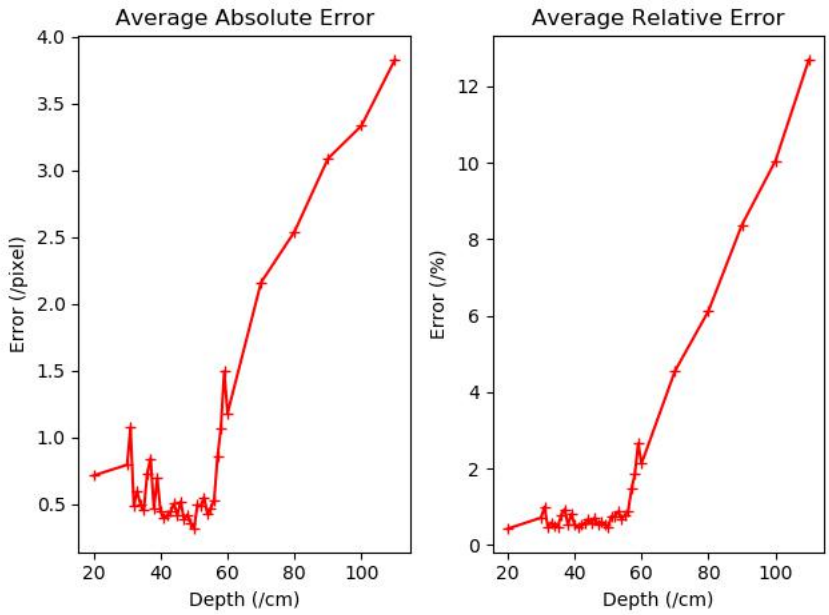


Figure 4.8: Average absolute/relative error per pixel-depth curve counting on non-reflection part of the box surface when the baseline is 35.17mm

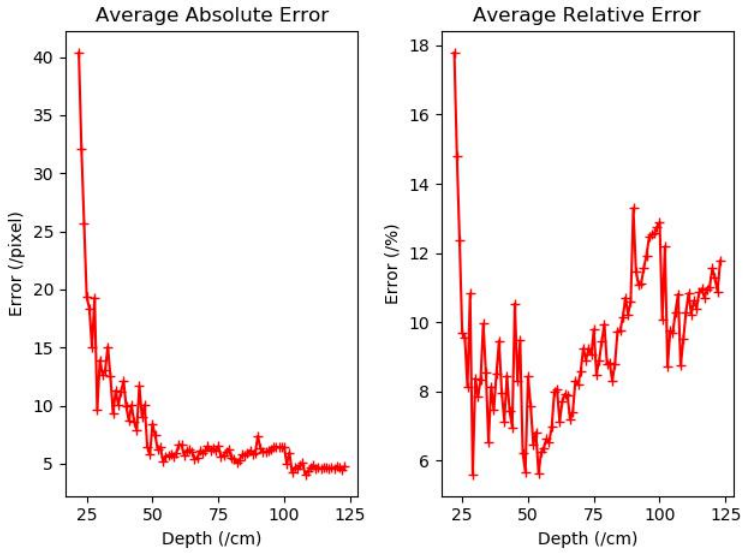
differences in somewhere of the box's front surface between the left and right input image. To some extent, the diversity of lighting conditions of stereo image pairs made our stereo matching algorithm fail.

To inquire about the degree to the impact of the reflection problem on the error per pixel, we modified the range of AOI. Rather than enclose the whole front side of the box in the disparity map, the part without the influence of reflection was regarded as the new AOI, which was constructed by arbitrarily circling a rectangular area as large as possible within the scope of the front surface of the box, avoiding the effect of reflection in the meantime. Figure 4.8 displays the results of the average absolute/relative error curve under the newly defined AOI. Clearly, both two errors dropped dramatically in the close-up range if the depth was smaller than 30cm, even they were approximate to the error level of the working distance range. But when the depth was longer than 30cm, the trend and values of average absolute/relative error were similar to that when deeming the complete front surface of the box as the AOI. For one thing, this finding validates our speculation that the illumination difference is adverse to predicting disparities. For another thing, the proposed stereo matching algorithm behaves relatively stable when there's no interference from reflection.

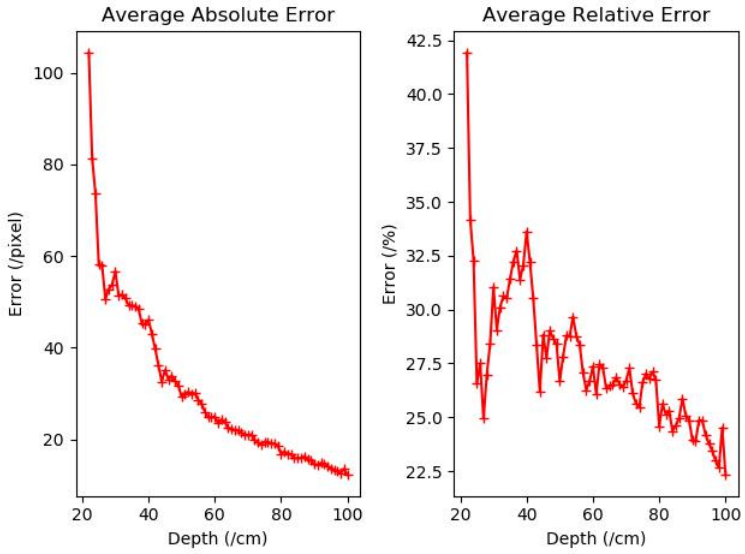
The working distance range stays the same as the former AOI definition, with no impact on the eventual result. So in other cases of baseline lengths, we were concerned about the entire front surface of the box instead of its part without the reflection issue.

Figure 4.9 illustrates the error situation when the baseline lengths are 47.8mm and 66mm, respectively. Similarly, higher error emerges at close range. Nevertheless, differing from the consequence in figure 4.6, both two error curves approximately present the decreasing property in the depth from 20cm to 100cm, which marks the best working distance didn't emerge in the anticipated range. With the rise of the baseline length, the tendency of our curves changed, leading to the shift of the working distance range to a further location. In addition, the average absolute/relative error per pixel is also growing at a particular depth. In view of the principal objective for our system is to gain the depth value within one meter, we reckon that the most feasible baseline setting is between 35mm to 47.8mm for our stereo camera whose baseline distance is limited to 35mm to 169mm. Within the shortened range of baseline, we could carry on doing more experiments about searching the best working distance range under diverse baseline lengths.

As you might imagine, there are some latent problems with the aforementioned experimental scheme. It's impractical to guarantee that the exposure conditions show no difference during the collection procedure whether for the same/distinct stereo image pairs, which would result in prediction errors for our stereo matching algorithm. Of course, taking into account the nonparallel relationship between the stereo camera lenses and the target box, there must be some measurement errors when wondering the distance between the front surface of the box and the plane where two cameras lenses are located. Besides, it's also unrealistic to adjust the baseline distance to be the expected value. Because this estimated value is firstly accomplished through measuring the distance by hand and then it will get further verification with the assistance of the stereo rectification step. In fact, we had hoped to get the most accurate depth value for the specified baseline value. And yet, we turned to hunt for the best working distance range under the preset fault tolerance, in consideration of the error fluctuations caused by the foregoing factors.



(a) Baseline = 47.8mm



(b) Baseline = 66mm

Figure 4.9: Average absolute/relative error per pixel-depth curve counting on the entire box surface when the baseline lengths are 47.8mm and 66mm

Chapter 5

Discussions

5.1 Conclusions

In this thesis, we've fulfilled a couple of desired outcomes as below:

- An integrated binocular stereo vision system using narrow adjustable-baseline cameras is established, which can acquire the left and right images of a stereo image pair simultaneously.
- Stereo camera calibration and rectification steps have been applied to get the rectified stereo pairs for any narrow-baseline stereo cameras.
- A CNN-based stereo matching algorithm has been proposed. We focus on the stage of calculating the matching cost, by the means of measuring the similarity level of two image patches. It consists of two kinds of network architectures. Both of them are inspired by the siamese network, outputting the extracted feature vector of the left/right image patch from each sub-network. The fast network utilizes the cosine similarity metric as the tool of measuring the similarity level between different image patches, obtaining the similarity score at an acceptable speed and accuracy level. Instead of employing the cosine similarity, the slow network tries to learn new similarity metric with several fully connected layers, achieving a little bit higher accuracy but at the expense of increasing the processing time and the number of training parameters.
- Considering that the initial matching cost based on the CNN network can't meet the accuracy requirement, a series of post-processing techniques are also taken into

account, including the cross-based cost aggregation as well as semi-global cost aggregation. After applying the WTA strategy, the final disparity map is produced, followed by additional refinement approaches containing interpolation and image filtering operations. The presented networks are trained in three standard datasets (Middlebury, KITTI 2012, and KITTI 2015) and the training details are given. They performed well on transfer learning and the validation set, in the context of choosing the census transformation as the benchmark method.

- Two types of error criteria, which are the average absolute/relative error per pixel under diverse disparity intervals, are put forward in order to assess the performance of the entire system, informing us about the appropriate working distance range under different choices of baseline distance.

5.2 Limitations and Future Work

The following issues deserve our further exploration and study in future work:

- Although the experimental results have proven the effectiveness of our raised network in the aspect of accuracy, it's a little far away from generating a high-quality disparity map in real-time. There still could be room for improvement about the time spent on handling a stereo image pair from the view of network architecture designing.

- Limited by the precision of our stereo camera, poor lighting condition during the capturing and other factors, there is significant image quality difference between the standard dataset and the recorded scenes, leading to quality distinctions of the attained disparity maps. Therefore, it's worth considering selecting other camera platforms. Despite the high-resolution cameras embedded in the mobile phones are playing an increasingly important role in our daily life, the APIs associated with cameras, which make two camera lenses under control so as to work simultaneously, behave quite differently due to various versions of Android/IOS system, as well as whether the camera is fully open to developers.

- Attributed to lacking the groundtruth disparity maps, we have to devise a set of redundant experiments by manually moving the box for acquiring known distances and doing analysis on system performance, but it's very likely to get unreliable results in the end. Hence, alternative ways of getting the actual depth maps in advance are needed, like taking advantage of a ToF camera or the newly released iPad Pro equipped with the LiDAR sensor.

References

[1] 10 python image manipulation tools. https://opensource.com/article/19/3/python-image-manipulation-tools. Accessed: 2020-04-20.

[2] About lua. https://www.lua.org/about.html. Accessed: 2020-04-21.

[3] Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. Accessed: 2020-04-23.

[4] Download putty - a free ssh and telnet client for windows. https://www.putty.org/. Accessed: 2020-04-22.

[5] An even easier introduction to cuda. https://devblogs.nvidia.com/even-easier-introduction-cuda/. Accessed: 2020-04-22.

[6] Introducing winscp. https://winscp.net/eng/docs/introduction. Accessed: 2020-04-22.

[7] Ks861 stereo camera introduction page. https://item.taobao.com/item.htm?spm=a1z10.5-c-s.w4002-6695776477.17.13a728a7UCEec0&id=556150206198. Accessed: 2020-04-18.

[8] Opencv. https://opencv.org/. Accessed: 2020-04-20.

[9] Pycharm - wikipedia. https://en.wikipedia.org/wiki/PyCharm. Accessed: 2020-04-21.

[10] Python (programming language). https://en.wikipedia.org/wiki/Python_(programming_language). Accessed: 2020-04-20.

[11] What is camera calibration? https://www.mathworks.com/help/vision/ug/camera-calibration.html#buvr2qb-1. Accessed: 2020-04-23.

[12] What is torch? http://torch.ch/. Accessed: 2020-04-21.

[13] A Arranz, Á Sánchez, and M Alvar. Multiresolution energy minimisation framework for stereo matching. *IET computer vision*, 6(5):425–434, 2012.

[14] Jana Bartels. 2d or 3d camera? which 3d camera technology fits your application? Technical report, Basler AG, 2016.

[15] Stan Birchfield and Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, 1998.

[16] Myron Z Brown, Darius Burschka, and Gregory D Hager. Advances in computational stereo. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):993–1008, 2003.

[17] Jinhai Cai and Rodney Walker. Robust video stabilisation algorithm using feature point selection and delta optical flow. *IET computer vision*, 3(4):176–188, 2009.

[18] Feiyang Cheng, Hong Zhang, Ding Yuan, and Mingui Sun. Stereo matching by using the global edge constraint. *Neurocomputing*, 131:217–226, 2014.

[19] Cevahir Cigla and A Aydın Alatan. Information permeability for stereo matching. *Signal Processing: Image Communication*, 28(9):1072–1088, 2013.

[20] Andrea Fusiello, Umberto Castellani, and Vittorio Murino. Relaxing symmetric multiple windows stereo using markov random fields. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 91–105. Springer, 2001.

[21] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[22] Raj Kumar Gupta and Siu-Yeung Cho. Window-based approach for fast stereo correspondence. *IET Computer Vision*, 7(2):123–134, 2013.

[23] Rostam Affendi Hamzah and Haidi Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016, 2016.

[24] M Hatzitheodorou, Evaggelia-Aggeliki Karabassi, Georgios Papaioannou, Alexander Boehm, and Theoharis Theoharis. Stereo matching using optic flow. *Real-Time Imaging*, 6(4):251–266, 2000.

[25] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1106–1112. IEEE, 1997.

[26] Yong Seok Heo, Kyoung Mu Lee, and Sang Uk Lee. Joint depth map and color consistency estimation for stereo images with different illuminations and cameras. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1094–1106, 2012.

[27] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[28] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[29] Martin Humenberger, Christian Zinner, Michael Weber, Wilfried Kubinger, and Markus Vincze. A fast stereo matching algorithm suitable for embedded real-time systems. *Computer Vision and Image Understanding*, 114(11):1180–1202, 2010.

[30] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of stereo vision algorithms: from software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[31] Zucheul Lee, Jason Juang, and Truong Q Nguyen. Local disparity estimation with three-moded cross census and advanced support weight. *IEEE Transactions on Multimedia*, 15(8):1855–1864, 2013.

[32] Li Ma, Jingjiao Li, Ji Ma, and Hanyue Zhang. A modified census transform based on the neighborhood information for stereo matching algorithm. In *2013 Seventh International Conference on Image and Graphics*, pages 533–538. IEEE, 2013.

[33] Stefano Mattoccia. Stereo vision: algorithms and applications. Seminar Notes, May 2012.

[34] Stefano Mattoccia, Simone Giardino, and Andrea Gambini. Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering. In *Asian Conference on Computer Vision*, pages 371–380. Springer, 2009.

[35] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 467–474. IEEE, 2011.

[36] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[37] Dongbo Min, Jiangbo Lu, and Minh N Do. A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy? In *2011 International Conference on Computer Vision*, pages 1567–1574. IEEE, 2011.

[38] Alina Miron, Samia Ainouz, Alexandrina Rogozan, and Abdelaziz Bensrhair. A robust cost function for stereo matching of road scenes. *Pattern Recognition Letters*, 38:70–77, 2014.

[39] Jesús M Pérez and Pablo Sánchez. Real-time stereo matching using memory-efficient belief propagation for high-definition 3d telepresence systems. *Pattern recognition letters*, 32(16):2250–2253, 2011.

[40] Cuong Cao Pham and Jae Wook Jeon. Domain transformation-based efficient cost aggregation for local stereo matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(7):1119–1130, 2012.

[41] Jan Salmen, Marc Schlipsing, Johann Edelbrunner, Stefan Hegemann, and Stefan Lüke. Real-time stereo vision: making more out of dynamic programming. In *International Conference on Computer Analysis of Images and Patterns*, pages 1096–1103. Springer, 2009.

[42] Shin'ichi Satoh. Simple low-dimensional features approximating ncc-based image matching. *Pattern Recognition Letters*, 32(14):1902–1911, 2011.

[43] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014.

[44] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[45] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[46] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE, 2003.

[47] HERMARY Machine Vision Technology. *3D OR 2D MACHINE VISION? Benefits of Using 3D Scanners*, 2018 (accessed April 3, 2020).

[48] Beau J Tippetts, Dah-Jye Lee, James K Archibald, and Kirt D Lillywhite. Dense disparity real-time stereo vision algorithm for resource-limited systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(10):1547–1555, 2011.

[49] Liang Wang, Mingwei Gong, Minglun Gong, and Ruigang Yang. How far can we go with local optimization in real-time stereo matching. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 129–136. IEEE, 2006.

[50] Yu-Chih Wang, Cheng-Ping Tung, and Pau-Choo Chung. Efficient disparity estimation using hierarchical bilateral disparity structure based graph cut algorithm with a foreground boundary refinement mechanism. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(5):784–801, 2012.

[51] Lingfeng Xu, Oscar C Au, Wenxiu Sun, Lu Fang, Ketan Tang, Jiali Li, and Yuanfang Guo. Stereo matching by adaptive weighting selection based cost aggregation. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 1420–1423. IEEE, 2013.

[52] Qingqing Yang, Pan Ji, Dongxiao Li, Shaojun Yao, and Ming Zhang. Fast stereo matching using adaptive guided filtering. *Image and Vision Computing*, 32(3):202–211, 2014.

[53] Qingxiong Yang, Liang Wang, Ruigang Yang, Henrik Stewénius, and David Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):492–504, 2008.

[54] Qingxiong Yang, Ruigang Yang, James Davis, and David Nistér. Spatial-depth super resolution for range images. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[55] Ruigang Yang and Marc Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE, 2003.

[56] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32, 2016.

[57] Ke Zhang, Jiangbo Lu, and Gauthier Lafruit. Cross-based local stereo matching using orthogonal integral images. *IEEE transactions on circuits and systems for video technology*, 19(7):1073–1079, 2009.

[58] Ke Zhang, Jiangbo Lu, Qiong Yang, Gauthier Lafruit, Rudy Lauwereins, and Luc Van Gool. Real-time and accurate stereo: A scalable approach with bitwise fast voting on cuda. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(7):867–878, 2011.

[59] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.