Final Project Report: Stereo Disparity

1st Tianyu Huang
Faculty of Engineering and IT
University of Melbourne
Melbourne, Australia
thhuang@student.unimelb.edu.au

2nd Ziyang Huang
Faculty of Engineering and IT
University of Melbourne
Melbourne, Australia
ziyangh@student.unimelb.edu.au

***Abstract*—This document is the report of University of Melbourne COMP90086 final project. In this project, we work on the problem of finding correspondences and compute the disparity map between stereo image pairs taken from a moving vehicle.**

***Index Terms*—Stereo Algorithm, Disparity Map**

I. Introduction

Stereo matching is the problem of finding correspondences between two images taken simultaneously by two cameras that are parallel to one another and separated apart along their x-axis. A disparity map, which displays the disparity as x - x' for every pixel x in the left image and how many pixels to the left its equivalent x' is in the right image, is produced as a result of stereo matching.

In this project, we work on stereo image pairs taken from a moving vehicle. We calculate a disparity map for the left image and compute some statistics using the given ground truth depth images in the dataset. In this report, we will discuss how to handle this problem, how we improve the performance of our method and provide some evaluations on our algorithm.

II. METHODOLOGY

A. Scan the whole line

First, to begin with a simple approach, we follow the method mentioned in lecture 13 where we have a scanline to scan through to find the best matching pixel block.

Fig. 1. Example of Scan Line.

In detail, for every pixel in the left image, we select a pixel block around it. Since two cameras are placed on the same y-axis, we can iterate over the same y-axis line to find the best matching pixel block in the right image. To find the best matching block, several matching functions are used like Normalized Cross Correlation (NCC), Sum of Square Difference (SSD) and Sum of Absolute Difference (SAD). These matching methods will be discussed in the following section. Also, we have a hyper-parameter which is the size of the pixel block, e.g. 3 mean the pixel block is 3x3. The performance of different pixel block size would be analysed later. After finding the best match blocks, the disparity value would be calculated by subtracting their x coordinates.

This is a brute way to compute the disparity map since we are checking every point on that scanline which makes the runtime of the algorithm very slow. It takes about twenty minutes to compute the disparity map for one image pair which is not very satisfying for a problem like this. It could be very dangerous if this algorithm is applied on an autonomous car.

B. Search Box

To improve the performance of the previous algorithm, we come up with a method to shorten the runtime: instead of scanning through the whole line, we only scan through part of the scanline.

Fig. 2. Example of Search Box.

It is obvious that we don't need to scan the whole line since the corresponding pixel block can only appear around the similar pixel location on the other image, e.g. a pixel block on the very left of the left image cannot match with the pixel block on the very right of the right image. So, we only need a small range to find a good match. We have another hyper-parameter here which indicates the length of the search box, e.g. 30 means the search box starts from x-30 and ends at x+30 where x is the coordinate of the original pixel in the left image. This method gives us some improvement on the runtime since we only check a small part of the scanline. For each image pair, we need around 3 minutes to compute the disparity map.

However, this is still not satisfiable enough. We find that the correct correspondence is always located on the left side of the search box mentioned above since images taken from the right camera, so we further improved this algorithm by only looking at the left side of the search box. By applying this change, we now need about 1.5 minutes to get the disparity map. The following evaluations are all experimented on this scanning algorithm.

C. Matching Functions

1) Sum of Absolute Differences (SAD):

$$\sum_{i,j} \left| w_{i,j} - w_{i,j}^{'} \right| \tag{1}$$

To begin with an easy approach, we start with an intuitive metric for similarity which is calculate sum of absolute difference between pixel values. It is very clear to see that 3 and 6 are more similar than 3 and 9 since $|3-6| < |3-9|$. Pixel blocks with lower SAD are more similar than pixel blocks with higher SAD. This matching function is a good start for this project since it is efficient and very easy to implement.

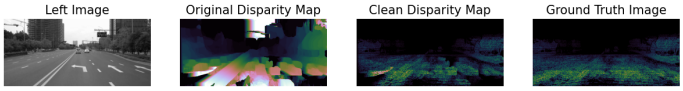

Fig. 3. Block Match using SAD.

2) Sum of Square Differences (SSD):

$$\sum_{i,j} (w_{i,j} - w_{i,j}^{'})^2 \tag{2}$$

Sum of Squared Differences (SSD) is a metric for comparing two images based on pixel-by-pixel intensity differences [1]. Instead of summing up the absolute differences in SAD, SSD calculates the summation of squared differences. This method is more advantageous than SAD since it is more sensitive to large differences. For example, we have differences (1,9) and (5,5), SAD(1,9) = SAD(5,5) = 10. However, SSD(1,9) = 82, SSD(5,5) = 50. SSD selects (5,5) to be the better correspondence. This method is a better version of SAD since it is more sensitive to pixel differences.

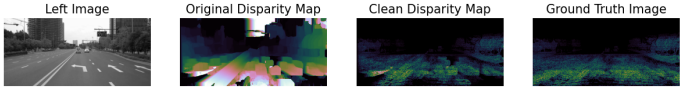

Fig. 4. Block Match using SSD.

3) Normalized Cross Correlation (NCC):

$$\frac{\sum_{i,j} w_{i,j} w_{i,j}^{'}}{\|w_{i,j}\| \left\|w_{i,j}^{'}\right\|} \tag{3}$$

Though SSD has had a better performance, we further implement another match function Normalized Cross Correlation (NCC). In each pixel block we compute the sum of the product of corresponding points and then divide their norm. The norm of each pixel block is calculated by square root of the sum of the square of pixel value in the block. The normalized correlation result cannot exceed one and we are trying to maximise the correlation to find the best matching block. Normalized Cross Correlation is more robust in the light change of the image, it allows differences in brightness of two images if we can't control the light gain of two cameras.

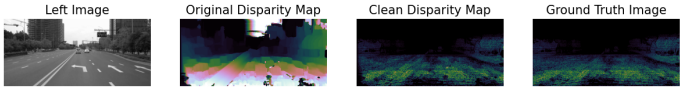

Fig. 5. Block Match using NCC.

III. Evaluation

To evaluate the performance of the three matching functions, we calculate the following three statistics using the given ground truth depth images. The ground truth image file is a 16 bit png where the value it contains is 256*disparity, to get the true disparity value, we first read image as uint16 then divide by 256. And since some of the disparity is absent and filled in with zero. Our evaluation on statistics are restricted to pixels for which there are valid depths. Later experiments would be run on the first image pairs in dataset.

- Root Mean Square Error (RMSE)
  We use Root Mean Square Error(RMSE) to assess the over all performance of our result. The smaller RMSE means our estimate disparity map more fit the true disparity value.

$$\sqrt{\frac{\sum_{i,j}(w_{i,j} - w_{i,j}^{'})^2}{N}} \tag{4}$$

- Fraction of pixels with error under thresholds
  Using RMSE have the risk being affected by some noise value, so we also calculated the fraction of pixels error comparing to ground truth within some thresholds to evaluate the accuracy. We have 4 thresholds here which are 4, 2, 1 and 0.25. But giving some tolerance to the algorithm, we will mainly use pixel error under 4 as our evaluation method in the following sections.
- Runtime
  Another aspect considered is the efficiency, we record the runtime of each matching method using python library tqdm.

TABLE I
EVALUATION STATISTICS (AVERAGE ON ALL DATA)

	NCC	SSD	SAD
RMSE	10.535	16.600	16.784
Pixel error under 4	0.841	0.694	0.681
Pixel error under 2	0.666	0.559	0.562
Pixel error under 1	0.457	0.397	0.410
Pixel error under 0.5	0.274	0.240	0.253
Pixel error under 0.25	0.154	0.136	0.144
Runtime	3min 44s	1min 21s	1min 20s

After running experiment for all the pictures in data set and taking the average result of each evaluation method we get Table 1. It is shown that NCC takes longest run time around 3min 44sec while Using SSD and SAD both take less than half of NCC around 1min 20sec. The reason would possibly be that NCC need extra calculation time when normalizing kernels. At the meantime, taking the longest computation time,

NCC provides the best result over other two match functions with the lowest RMSE 10 and the highest pixel fraction in all scale. Thus, there is a trad-off between runtime and disparity accuracy for our match functions.

A. Window size and search box size

Regarding to our block match algorithm, there are two hyperparameters we could tune: search box length and match block size. Each of them has different effect on our disparity map output.

As defined in previous section, search box length represent how far we need to check the pixel block from the corresponding location in the other image. As the search box increases, more blocks would be compared. However, if there is a very similar block occurs very far from the match block, it will be wrongly considered as the same object and result in a very high disparity value. For example, if two same model cars lie on the two side of the road, two matching blocks on two cars would easily be miss matched. But if the search box is too small, the true matching block would possibly lie out of the search boundary and underestimate the disparity value. Moreover, the length of search box effects the runtime, the larger search box means we need longer check time for each block in every iteration.

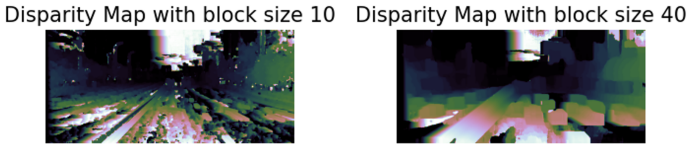

Fig. 6. Comparison of different Block Size.

On the other hand, block size determines the resolution of the output. With a small block size, the difference between blocks of neighboring pixels would be larger and match with different blocks on the other images hence a sharper disparity map would be presented. While a larger block size makes neighboring pixels share a very similar block and match with same block on other image hence gives a blur output. However, each choice has limitations. Too small block sizes would lead to lot of miss match blocks and contain too much noise, since it's not large enough to capture object information, instead, it captures more texture information. Too larger block size would lose details and show pieces of same disparity value.

B. Grid Search

In order to find the best search box length and block size, we present a grid search which is a common tuning technique that attempts to compute the optimum values of hyperparameters. Consider the runtime difference between three match functions, we decided to present a larger range experiment on SSD and smaller range on NCC. The result in figure 7 shows that for SSD, with search length of 70 and block size of 38, we get the lowest RMSE 15.871. For NCC we reach 7.923 RMSE with search length of 70 and block size of 28. Regarding to pixel accuracy, we get 81.3% pixels error under 4 using NCC with search length of 70 and block size of 18; 68.5% pixels error under 4 using SSD with search length of 70 and block size of 26. The best search length seems to converge on the similar value using search length of 70. It is because in the ground truth the maximum disparity value is 68.734 so the result fit to a similar range. And for block size, the RMSE decrease as the block size increase because it favors a smooth output and don't care the details. However, the decrease of RMSE exceed loss of pixel error under 4 fractions too much, so we might want to limit the block size to 25 as it balances two evaluation methods.

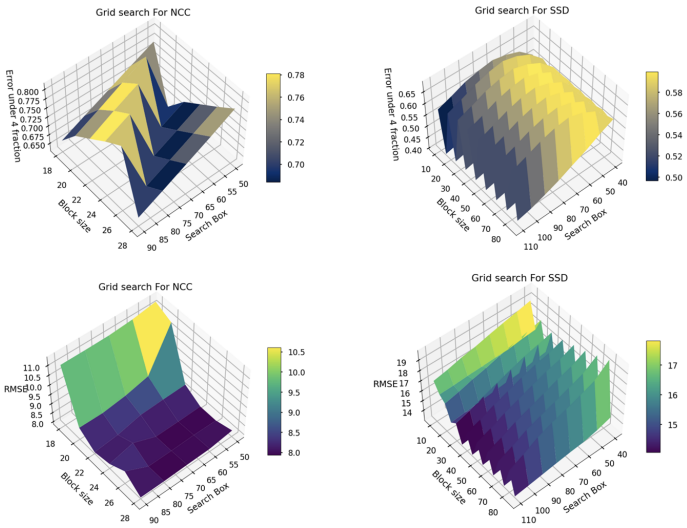

Fig. 7. Grid Search Results.

C. Subpixel accuracy

As discuss the in previous section, the RMSE and pixel error fraction are all calculated if the ground truth pixel at that pixel location is non-zero. That is sub-pixel accuracy, as we're not getting all pixels' error because there are too many zeros in ground truth. However, In the real world, there is no holes to infinity on the objects, thus the zero values among a group of non-zeros are not reasonable. We decide to blur the ground truth disparity using median filter to impute some of the zero values. This method takes the median of all the pixels under the kernel area and the central element is replaced with this median value. This is highly effective against salt-and-pepper noise in an image. After comparing the blurring result with our estimate disparity map using the best model tuning in the previous section, the result shows that we can further decrease the RMSE to 4.651 and have 0.839 disparity value error under 4 with the best hyperparameters tuned before.

D. Smoothness and Pixel importance

Even we have already reached a decent result through above techniques, we still want to improve our performance by smoothing the output. The first thing we tried is apply constrains and use Energy Minimization method where

$$E(D) = \sum_i (W_1(i) - W_2(i + D(i)))^2 + \lambda \sum_{i,j} \rho(D(i) - D(j)) \quad (5)$$

The first part is match equality term which is calculated by the matching function that we have discussed, the second part is smoothness term which will consider the distance from middle pixel to adjacent pixels in the block. However, defining a proper smooth term is difficult. So, we decided to combine two terms that apply the different weights on pixels according to their distance, so that pixels in the matching block have different importance. After running experiments with different block size and SSD matching function, it turns out that with small block size like 20, the RMSE using weighted function decrease from 17.275 to 17.037 and pixel error under 4 fraction decrease from 0.671 to 0.670. With larger block size like 40, the RMSE using weighted function increase from 15.244 to 15.333 and pixel error under 4 fractions increase from 0.639 to 0.658. The result shows that there is little impact on the smaller block size since there is not much differences on the distance between pixels and the pixels contribute in a similar weight. With larger block size, weighted matching function would focus more on the center pixels therefore has an increase regard to pixel error under 4 fractions.

E. Acceleration

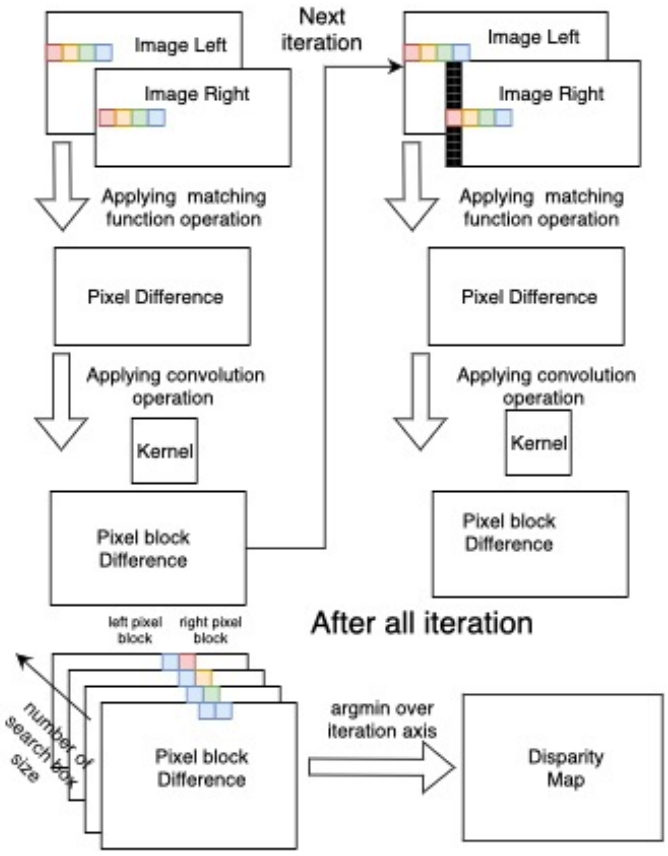

Fig. 8. Acceleration Procedure.

In order to accelerate the process computing the disparity map, we have tried to convert the loop operation to matrix operation. Taking the matching function SSD as example, the core criteria using to find matching pixel block is the difference between corresponding pixels. So instead of loop through every pixel, creating blocks and find best match of the block, we first applying the match function to every corresponding block and store the result in size of (height, width) same as the original image.

After that, we apply a convolution process with the kernel of all ones and the size is same as previous match block size. This would have same result as we apply the matching function to blocks, the difference is instead of calculating the dissimilarity of corresponding pixels and do the summation for each block, now we calculate the dissimilarity of corresponding pixels in the whole image and use convolution filter to simulate and accelerate previous summation process.

So far, we finish the first iteration, and we get an image size of (height, width) with each pixel value represent left image same position pixel block difference to pixel block in right image, left pixel block at (0, 0) match right pixel block (0,0). Then we shift the right image to right for one pixel in next iteration and fill in constant in the original first column.

Now we repeat the process in first iteration, still we get an image size of (height, width) with each pixel value represent left image same position pixel block difference to pixel block in right image, but now the value represents left pixel block at (0, 0) match the constant border pixel block and left pixel block at (0, 1) match right pixel block (0,0). After iterating the number of search box size (N) times, we get a buffer in size of (height, width, N) that stores all the block differences we need. Over the axis 2 we have pixel block differences over search box, then we simply find the minimum block pixel difference value which means it has the highest similarity and its index will be the disparity value.

One problem is that after shifting right image, some of the left pixel block in the left image would match the constant pixel block, this will the result in a very large difference between blocks. However, this will not affect our result since we are picking the minimum difference value index, and this matches with previous algorithm that finding search box to the left direction will only have few match block for the pixel block on the left image.

IV. Conclusion

In conclusion, we have utilized the stereo matching algorithm to calculate the disparity map for the left image and compute some statistics using the given ground truth depth. We have discussed the basic matching algorithm how we implement it as well as a comprehensive evaluation on our algorithm and how we improve the performance of our method. Our best result using NCC matching function give a robust performance achieving over 80% pixels disparity value error under 4 with under 8 RMSE.