Final Project: Stereo Disparity

Leting Zhou
lzhouli@student.unimelb.edu.au
University of Melbourne
Melbourne, Australia
1240313

Yu Weng
ywweng1@student.unimelb.edu.au
University of Melbourne
Melbourne, Australia
1235216

***Abstract*—This report compared the fundamental block-matching approach to the kernel-based accelerated method with three distinct matching cost functions SAD, SSD and NCC. And in order to compute a better disparity map, grid search is utilized for finding the superior searching window size and matching window size (kernel size). Afterward, the pre-processing and post-smoothing methods are implemented for achieving better performance of the computed disparity map.**

I. Introduction

The usage of stereo images for retrieving depth perception is significant in computer vision applications.[1] The aim of the stereo block matching method is mainly used for reconstructing a 3D model acquired from different angles.[2]. And in this project, our team implemented three main similarity-measuring methods for stereo matching which are the Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), and Normalized Cross Correlation (NCC). The traditional Block matching method is implemented first to generate a basic understanding of the stereo block matching method. However, due to the efficiency of the traditional stereo block matching method, a kernel-based method is introduced to further improve the block matching process. The corresponding detailed methods, evaluation results, and experiments will all be demonstrated below.

II. Implementation of the disparity map using basic SAD, SSD and NCC

Stereo matching requires locating the corresponding positions in two images. As the left image is taken from the left angle and the right image is taken from the right angle, the corresponding points x' in the right image are definitely to the left of the same position as x in the left image. The block matching steps is shown below,

1) For each pixel on the left image, we find the corresponding epipolar line, and we construct a window around that pixel.
2) As for the right image, we also construct a window with the same window size
3) The window on the right image will slide along the epipolar line to compare the contents of that window with the referenced window on the left image, And the matching cost is mainly based on SSD, SAD, and NCC.

For the basic stereo block matching method, we adopted three layers of loops: the loop of rows, the loop of columns, and the loop of search window size. Overall the values of matching difference, the minimum was selected and the index of the position x' in the right image is kept to calculate x-x' for the disparity map.
The performance of the three different matching cost methods is listed in the table provided below.

TABLE I
RMS error and running time for three basic BM methods

Methods comparison	Matching cost methods		
	SAD	***SSD***	***NCC***
Wall time	1min 23s	1min 22s	2min 45s
Root Mean Square Error	17.8	17.7	13.2

Window size: 19, Searching window size: 70

III. Acceleration of running time with 3D data model and kernel convolution

In real-time distance detection, the running time is one of the most important criteria for judging the matching cost functions. Therefore, we accelerated the running time by building a 3D Numpy array model and applying kernel convolution then. The general steps for kernel convolution are shown below:

1) Building a 3D array which shape is (searching window size, number of rows, number of columns)
2) Looping the searching window size:
   a) Padding the right image on the left side with the iterated searching window size i
   b) Reconstructing a new same-size right image by including the padded columns on the left side and cropping the same number of columns on the right side of the image
   c) Utilizing the newly created right image and left image to do the convolution(Methods may be different with different matching cost methods)
   d) Saving the computed disparity image to the previously designed 3D array
3) Utilizing the Numpy **argmin()** method to find the minimum index for each pixel within the searching window size. In another word, just focusing on one pixel, there are in total searching window size number of convolutions between the left and right images. And

the corresponding index of the minimum value is exactly the disparity value that we want.

And the detailed implementations will be discussed below, for every pixel in the left image, we have to calculate the matching differences to the right image for the number of times the same as the searching window size. Theoretically, we shift the right image one column to the right every time, and pad the missing left parts with the border reflect. Then we use the generated image and the original left image to calculate the difference. For the difference matrix, we conducted a kernel convolution to calculate the matching cost matrix. As the searching window goes, we generated the number of cost matrices the same as the number of searching window size. With all the cost matrices, we selected the minimum one, and the index is the disparity value $x - x'$.
As for the pre-computed 3D array, the first dimension as searching window size, the second dimension as rows, and the third dimension as columns. The most important dimension is the first one, which records the matching difference for every position in the search window.
Regarding SAD matching cost method, the matching difference matrix is simply the absolute value of the left image subtracting the generated right image. The cost matrix is the convolution output of the matching difference matrix.
Similarly, in the SSD method, the matching difference matrix is the element-wise squared value of the left image subtracting the generated right image. The cost matrix is also the convolution output of the matching difference matrix.
As for the NCC method, it is a little bit different when compared with the previous two methods. We have to calculate the numerator and denominator separately in the formula. The numerator is the convolution output of the element-wise product of the left image and the generated right image. The denominator is the square root of the convolution output of the squared left image times the convolution output of the squared generated right image.
The performance of the accelerated methods is as follows, using the same search window size of 70 and kernel size of 19 as basic methods (Wall time will be compared). After implementing the kernel-based methods, the wall time for each matching cost method has been reduced a lot. This is mostly because the two for-loops of the basic Block matching method are replaced by the convolution operation between the all-one kernel matrix and the stereo images.

TABLE II
WALL TIME COMPARISON BETWEEN THE BASIC METHODS AND KERNEL-BASED METHODS

Methods comparison	Matching cost methods		
	SAD	***SSD***	***NCC***
Kernel-based methods Wall time	469 ms	435 ms	1.18 s
Basic BM methods Wall time	1min 23s	1min 22s	2min 45s

IV. Experiment of best search window size and kernel size

In the process of the computing disparity map, two parameters have a noticeable influence on the final results: the search window size and the kernel size.
The search window size determines how many pixels in the right image we have to compare to the left image. Normally, the disparity x-x' should be in a range of numbers. The kernel size represents the size of the block that we used to compare the two images.
In order to find the best combination of search window size and kernel size, we looped the range of 120 in search window size and the range of 50 in kernel size to find the maximum fractions of pixels with errors less than four. The performance of the three methods is as follows:

TABLE III
BEST WINDOW SIZE AND SEARCHING WINDOW SIZE

Methods comparison	Matching cost methods		
	SAD	***SSD***	***NCC***
window size	29	29	27
Searching window size	65	65	67
Fractions of pixels with error less than four	71%	72%	87%

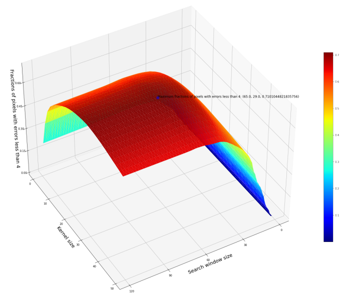

Fig. 1. Fractions of pixels with errors less than 4 for SAD

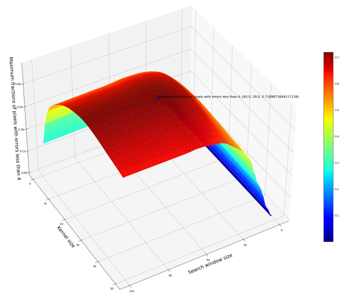

Fig. 2. Fractions of pixels with errors less than 4 for SSD

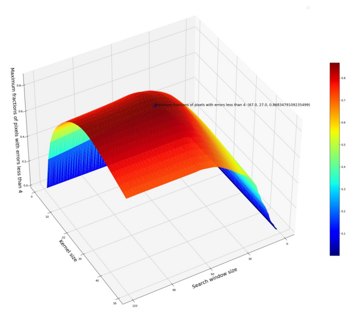

Fig. 3. Fractions of pixels with errors less than 4 for NCC

V. Importance of each pixel in the matching window

In the matching window, each pixel was assigned the same weight as 1 previously. However, it is worth well to do an experiment for checking whether some pixels are more important than others in the matching process. With the fixed best searching window size and window size retrieved above, we assigned different values to the center point of the kernel, and the result is shown in the figure below:

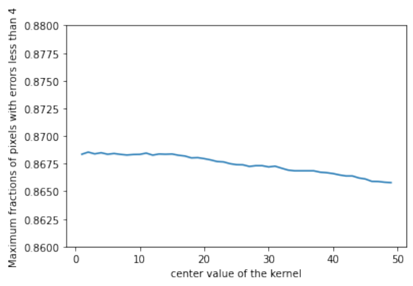

Fig. 4. Different center point value of the kernel Versus below 4-pixel error fraction

After implementing the experiment above, we found that increasing the center value of the kernel(Window) will basically lead to no impact on the fractions of pixel errors less than four (can be extended to other pixel error thresholds as well). Therefore, we will still keep the center value of the kernel to one.

VI. Sub-pixel accuracy

As the ground truth disparity map is a 16-bit PNG where the value it contains is 256 * disparity, we have to read it with full value and divided it by 256 to get the authentic disparity map. For some pixels in the ground truth disparity map, the disparity value is zero, which means there is no valid disparity value. Therefore, we have to find the positions that have zero disparity values in the ground truth map and set the value of the corresponding position in our generated disparity map to zero. The fractions of pixels with errors less than 4, 2, 1, 0.5 and 0.25 pixels will be calculated by counting the number of qualified pixels, divided by the total number of valid pixels in the ground truth map. The corresponding results are presented in table 4.

TABLE IV
Sub-pixel Accuracy

Kernel-based Methods comparison	**Matching cost methods**		
	SAD	***SSD***	***NCC***
Root Mean Square Error	15.4	15.1	7.7
Fraction of pixel error less than 4	71.0%	71.9%	86.8%
Fraction of pixel error less than 2	54.1%	53.4%	64.2%
Fraction of pixel error less than 1	37.7%	36.0%	42.7%
Fraction of pixel error less than 0.5	22.3%	21.0%	24.7%
Fraction of pixel error less than 0.25	12.1%	11.0%	13.1%

Hyperparameters are based on the previous best combination

VII. Performance improvement by pre-processing and post-smoothing

Stereo matching is largely dependent on the detection of the corresponding points. We experimented with several methods to conduct the pre-processing of the original image and post-smoothing of the generated disparity map, aiming at improving the disparity map.
For pre-processing, we implemented three different methods. The first one is the normalization of the original image, which may provide a more accurate matching cost. Another one is sharpening the original image with a filter, trying to emphasize the contrast of pixels. The third one is to use Gaussian blurring. We can see that normalization and sharpening actually couldn't improve the NCC performance, whereas Gaussian blurring can. The results are presented in table 5.

TABLE V
Pre-processing methods comparison

Pre-processing Methods	**NCC fraction of pixel error less than 4**
Image Normalization	82.2%
Image Sharpening	75.1%
Image Blurring (Gaussian)	88.1%
Baseline (No pre-processing)	86.8%

Hyperparameters are based on the previous best combination

For post-smoothing, we adopted three different filters to reduce the noise in the generated disparity map. The first one is median filter blur, which computes the median of the pixels within the kernel size area and replaces the center with this median value. The second filter is Gaussian blurring, which recomputes the pixel values to follow the Gaussian distribution. The last bilateral filter can effectively remove noise as well as keep sharp edges. The performance of the three smoothing methods is shown below.

TABLE VI
Post-smoothing methods comparison

Post-smoothing Methods	**NCC fraction of pixel error less than 4**
Image Blurring (Median)	89.4%
Image Blurring (Gaussian)	90.6%
Image Blurring (Bilateral)	90.0%
Baseline (No post-smoothing)	86.8%

As experimented with above, we found that Gaussian blurring can improve the output disparity map in both pre-processing and post-smoothing.

VIII. Critical Analysis, Result Conclusion, and Future Improvements

A. Critical Analysis

1) Matching cost function choosing and the functions of pre-processing and post-smoothing: NCC matching cost is selected as the main matching cost for this report. Mainly because both SSD and SAD matching costs could cause mismatches when there are intensity differences in the image. As for the smoothing methods, they improve the performance by providing smooth change for the computed disparity image.

2) Error Analysis: In comparison with the ground truth disparity image, we concluded that most of the sub-pixel differences occurred on the left side of the computed disparity image. Because of the translation of the image, the corresponding pixels on the left side of the left image will not be found on the right image. Therefore, the computed disparity values will definitely be invalid numbers.
And another possible reason for causing incorrect disparity map values is the similar matching window pixel values located on the same epipolar line causing invalid block matching(specularities, repeated patterns, and textureless regions). For instance, the road line and the side mirror of the car are located on the same epipolar line.

B. Result Conclusion

We conclude an overall process for generating a high-quality disparity map within a short time. Firstly, pre-process the left and right images with Gaussian blurring. Then use the convectional computation of NCC method, with a searching window size of 67 and kernel size of 27. With the draft disparity map generated, Gaussian blurring is applied again to get the final disparity map. This process could improve the fractions of pixels with errors less than 4 increased to **92.4%**. The corresponding final result is shown below:

TABLE VII
Final Result

	NCC
Root Mean Square Error	3.33
Fraction of pixel error less than 4	92.4%
Fraction of pixel error less than 2	79.4%
Fraction of pixel error less than 1	53.4%
Fraction of pixel error less than 0.5	30.1%
Fraction of pixel error less than 0.25	16.1%
Wall time	1.24 s

C. Future Improvements

Since a good stereo correspondence can be characterized by two primary factors, the first is the matching quality, and the second is smoothness. As for future improvement, energy minimization(combined data term and smooth term) via graph cuts can be implemented for improving performance.

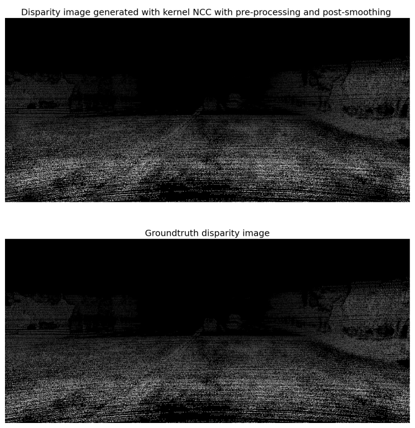
Disparity image generated with kernel NCC with pre-processing and post-smoothing
Groundtruth disparity image

Fig. 5. Final disparity map comparison

References

[1] M. Brown, D. Burschka, and G. Hager, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 993–1008, 2003. DOI: 10 . 1109 / TPAMI . 2003 . 1217603.

[2] N. Sabater, J.-M. Morel, and A. Almansa, "How accurate can block matches be in stereo vision?" *SIAM J. Imaging Sciences*, vol. 4, pp. 472–500, Mar. 2011. DOI: 10.1137/100797849.