Recovering Depth and 3D Reconstruction

Author: Yeziwei (Maggie) Wang

Contents

1 Introduction

A common method of conducting stereo matching is to use disparity map, which extracts depth information from a pair of stereo images. A basic algorithm for calculating disparity map is block matching algorithm. This algorithm finds the matching pixels in the image pair and calculates the displacement between each pixel in different image, which is called disparity. The depth of a 3D point can then be computed using triangulation. To make the computation more efficient, rectified images are usually used. All the images used in this report for testing the algorithm are rectified.

This report first explores the theory behind stereo vision. A description of the code implementation is then introduced, as well as testing metrics. Finally, testing results are investigated and discussed in comparison to some ground truth, as well as further improvements of the algorithm.

2 Theory

2.1 Epipolar Geometry

Depth recovery works with a pair of rectified images, where two image planes are coplanar and parallel to the baseline.

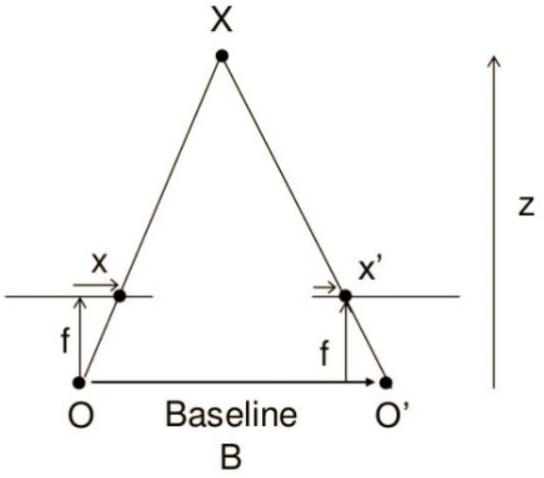


Figure 1: A Pair of Rectified Image Planes

According to epipolar constraint, for a point in the left image plane, the corresponding point in the right plane lies on the epipolar line. In a pair of rectified image planes, the two corresponding points lie on the same horizontal scan line. Considering left image as reference, the disparity of a point in the left image plane is:

$$disparity = x - x' = \frac{Bf}{Z}$$

where x and x' are the corresponding coordinate of 3D point X in the two image planes, B is the distance between two cameras and f is the focal length of cameras. In calibrated camera system, B and f are known.

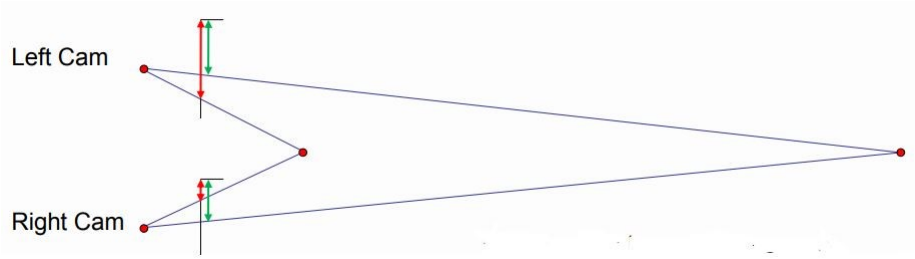


Figure 2: Disparity and 3D Object Point

If the 3D point is close, the disparity is large and if the point is far, the disparity in the two images is small, as shown in Figure 2.

2.2 Block Matching Algorithm

In order to recover the depth of the image, we need to calculate the disparity of each pixel in the reference image. For each pixel, a best match is found in the right image with the most similar pixel intensity. However, one pixel is not be able to withstand image noise, image blur and illumination changes. Therefore, a block of pixels are used to perform matching. Block matching algorithm takes a block of pixels in the left image and finds a matching block in the right image.

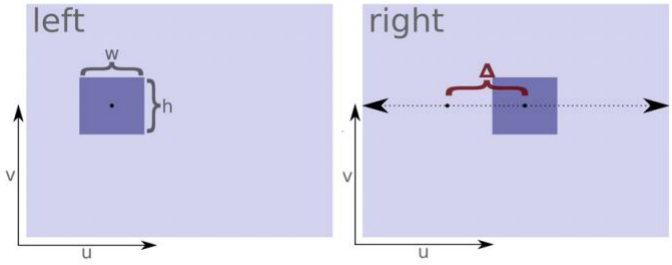


Figure 3: BM Algorithm

Since the images are rectified, we only need to search on the same horizontal scan line to find the best match block. A cost function is used to measure the similarity between two blocks. Common cost functions are sum of absolute difference (SAD), sum of squared differences (SSD) and normalised cross-correlation (NCC). Their expressions are shown as follows:

$$SAD = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} |I_1(i,j) - I_2(i,j)| \tag{1}$$

$$SSD = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} (I_1(i,j) - I_2(i,j))^2 \tag{2}$$

$$CC(u,v) = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} I_1(i,j) \times I_2(u+i, v+j) \tag{3}$$

$$NCC(u,v) = \frac{CC(u,v)}{\sqrt{\sum_{i=0}^{w-1} \sum_{j=0}^{h-1} I_1^2(i,j) \times \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} I_2^2(u+i, v+j)}} \tag{4}$$

When the cost function reaches a minimum value, a match is found. This is winner-take-all method. The displacement of the pixel coordinates is the disparity of the pixel in the reference image plane.

2.3 Sub-pixel Estimation

The disparity values calculated using block matching are all integers. Considering sub-pixel locations can help reduce the discontinuity within disparity map, where the neighbouring cost values around the minimum cost are taken into consideration. In order to perform sub-pixel estimation, the three costs: the cost of pixel to the left of minimum cost, minimum cost and the cost of pixel to the right of minimum cost, will be fitted on a parabola.

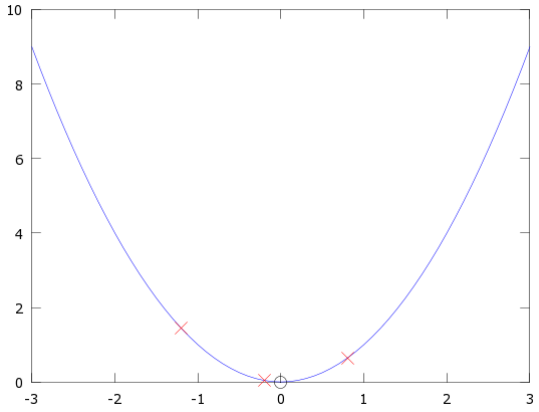


Figure 4: Three cost values fitting a parabola

The sub-pixel disparity esitimate can be calculated as follows:

$$d_{sub\_pixel} = d - \frac{1}{2} \times \frac{C_{left} - C_{right}}{C_{right} + C_{left} - 2C_{min}}$$

3 Implementation

The following arguments are needed for the algorithm:

1) the two stereo images;

2) the block size;

3) the number of disparity, which limits the searching range to avoid scanning the whole line;

4) the disparity map, which has the same size as the image.

The algorithm works on a pair of rectified images. Therefore, the cameras needs to be calibrated and image plane rectified. The algorithm is performed in the following 4 steps:
1) matching cost computation;
2) cost aggregation;
3) disparity computation;
4) disparity refinement.
The cost function used in the algorithm is SAD, implemented using formula (1). After calculating cost between the reference block in the left image and all the candidate blocks within search range in the right image, a minimum cost is found where the disparity lies. The disparity computed only contains integer numbers, thus can cause discontinuity. Finally, sub-pixel estimate is used to smooth out the disparity map.

3.1 Pseudo-code

To calculate disparity map of a pair of rectified images, my custom block matching algorithm is implemented following the pseudocode below:

Algorithm 1: Block Matching Algorithm for Calculating Disparity Map

Data: LeftImage, RightImage, numDisparity, blockSise
Result: Disparity map of left image
Initialise numberDisparity and blockSize;
for *Rows of left image* **do**
    Calculate row range of the block;
    for *Columns of the image* **do**
        Calculate column rage of the block;
        Calculate the search range;
        for *Within the search range* **do**
            Calculate SAD for each pixel;
        end
        Find minimum SAD;
        Store the disparity of minimum SAD;
    end
end

3.2 Quantitative Comparison Metrics

A number of different images from Middlebury Stereo dataset are used to test the algorithm. These images include different scenes, illuminations and qualities. The disparity maps produced by custom algorithm and OpenCV algorithm are compared against each other.

3.3 Qualitative Comparison Metrics

The method used by D.scharstein (D.scharstein and R.Szeliski, 2002) is used to examine the quality of the algorithm. The disparity map is compared against some ground truth disparity map. RMS(Root Mean Squared) error between the computed disparity map and the ground truth map can be calculated as follows:

$$R = \sqrt{\frac{1}{N}\sum_{(x,y)} |d_C(x,y) - d_T(x,y)|^2}$$

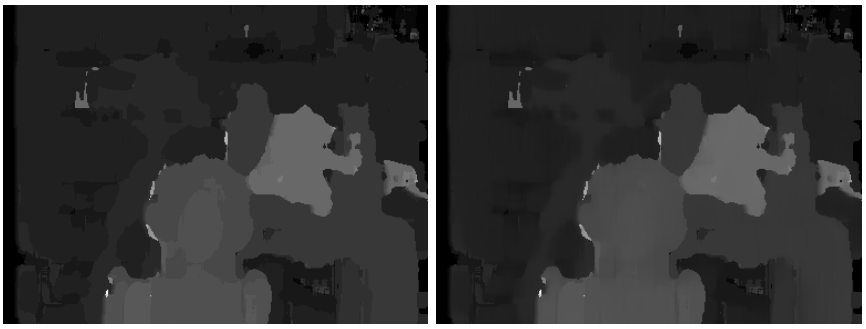
where $d_C(x,y)$ is the computed disparity map, $d_T(x,y)$ is the ground truth map and N is total number of pixels.
Another quality measure is the percentage of bad matching pixels:

$$B = \frac{1}{N}\sum_{(x,y)} (|d_C(x,y) - d_T(x,y)| > \delta_d)$$

where δ_d is the disparity error threshold.

4 Results

Sub-pixel estimation can smooth the changes between coherent pixels. Figure 5 shows the results from both integer pixel values and sub-pixel estimations. It is obvious that, the disparity map with sub-pixel estimate has smoother object boundaries. It also removes the contour effect in the disparity map with integer intensity values.



(a) MyBM with integer pixel value (b) MyBM with sub-pixel estimation

Figure 5: Comparison between integer pixel and sub-pixel values

The search range and block size are crucial to obtaining an optimal disparity map. If the search range is too small, the correct matching point can't be found causing errors in the disparity map. The block size controls how much details are included in th disparity map. The smaller the block size, the more details there is. However, it also means that the disparity map is more likely to be affected by noise. After trial and error, a search range of 64 pixels and a block size of 15×15 are chosen.

4.1 Runtime

In terms of running time, my custom algorithm takes much longer than OpenCV algorithm. The detailed time for each run are shown in Table 1. On average OpenCV takes about 10ms to run the algorithm, whereas my custom algorithm takes 1 to 2 minutes. Therefore, in terms of run time, OpenCV has a much better performance.

	middlebury	teddy	cones	art	books	dolls	laundry	moetuis	reindeer
OpenCV	6.9ms	9.2ms	10.0ms	12.1ms	10.3ms	11.1ms	9.9ms	12.1ms	10.5ms
MyBM	70s	116s	105s	111s	120s	125s	110s	120s	118s

Table 1: Algorithm Running Time

4.2 Quantitative Comparison

In total 9 images are chosen from `http://vision.middlebury.edu/stereo/data/` to test the algorithms. These images have different illuminations, colours, textures, different shapes and different level of details.

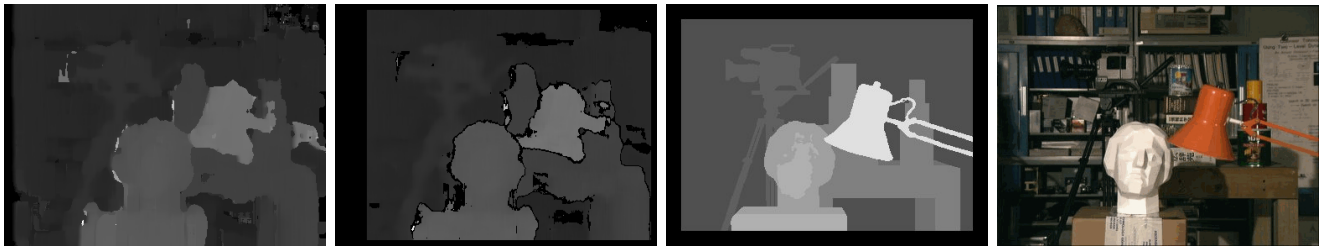
4.2.1 Disparity Maps

All the disparities are shown in comparison with ground truth map provided from the website. Truth map provides a very good disparity description. As for current algorithm performance, custom BM and OpenCV stereoBM, both can capture the main depth information in the image, while a lot of details are missed or mistaken. For example, in 'Art' image, a lot of details are shown in the truth map: closer to image plane there are paint brushes in the cup. Disparity maps produced by custom and OpenCV block matching algorithm fail to show that detail. Attempts have made to reduce the block size to include more details in the images. However, it didn't provide better disparity accuracy, but introduced more noise in the disparity map.

Furthermore, both algorithm don't deal with the edge of the image very well, although in different ways. For custom algorithm, the block size is differeent when the pixel is at the edge of the image. For the chosen block size 15, when pixel is located at (1,1) the block size is 8×8. When the pixel is located at (1,2), the block size is then 8×9. This means that pixels at the edges of the left image may not find the correct match in the right image. This problem is especially obvious at the left edge of the image. The reason for this is that pixels at the left edge of left image usually can't find the right matching block in the right image. The same matching block usually appears to shift slightly to the left in the right image, causing the matching blocks to be out of the image when the corresponding blocks in the left image is too close to the left edge. For OpenCV stereoBM, it discards left edge of the image for a certain width to ensure the disparity map have regular edges. In all the disparity maps produced by OpenCV, there is a certain width of black area to the left of the image.

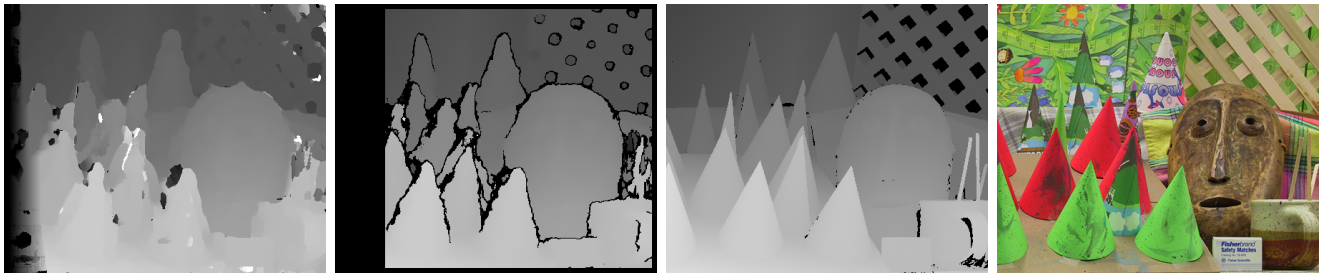
There is one major difference between disparity maps produced by OpenCV and custom block matching algorithm. In OpenCV disparity map, the boundary of different objects appear to be discontinuous, surrounded by a darker edge. This discontinuity is absence in the disparity map produced by custom block matching algorithm.

Occlusion view can be one of the reasons for the poor performance of both block matching algorithms. If a block of pixels is occluded in the right image, then there won't be a correct match found, and this pixels will appear as noise in the disparity map.



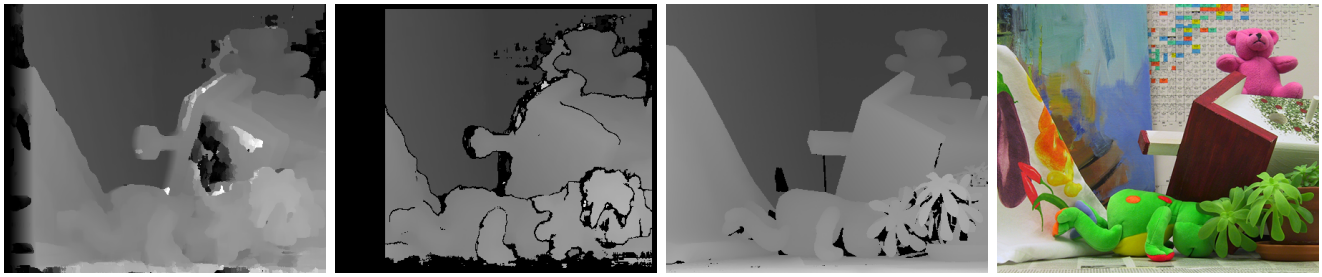
(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Truth Map

Figure 6: Middlebury-Disparity Map



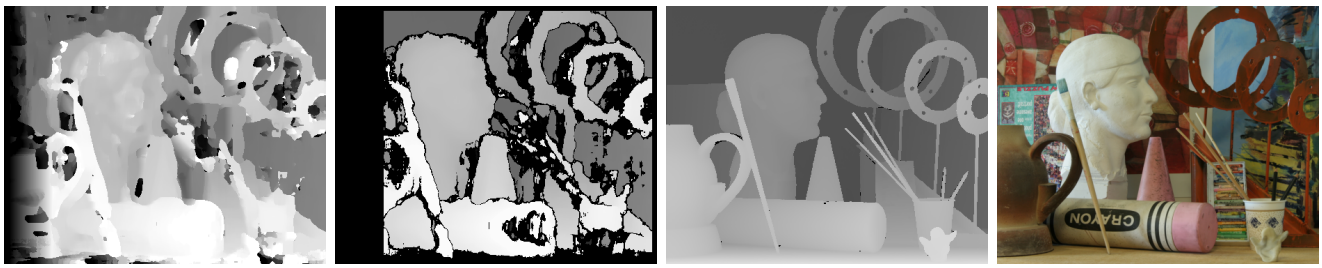
(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 7: Cones-Disparity Map



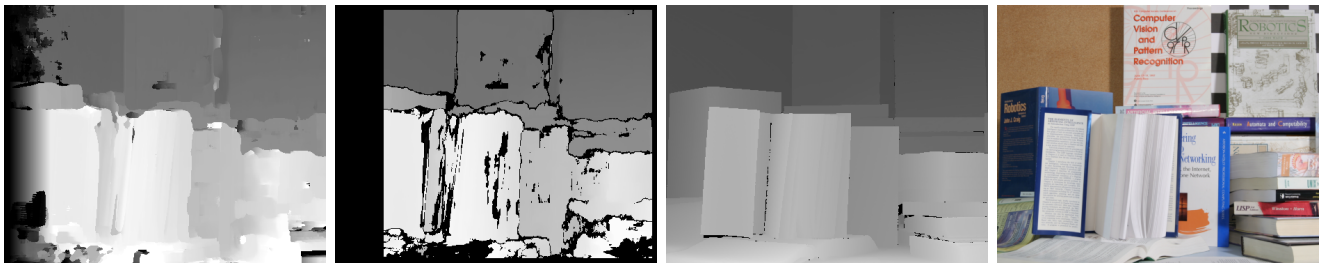
(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 8: Teddy-Disparity Map



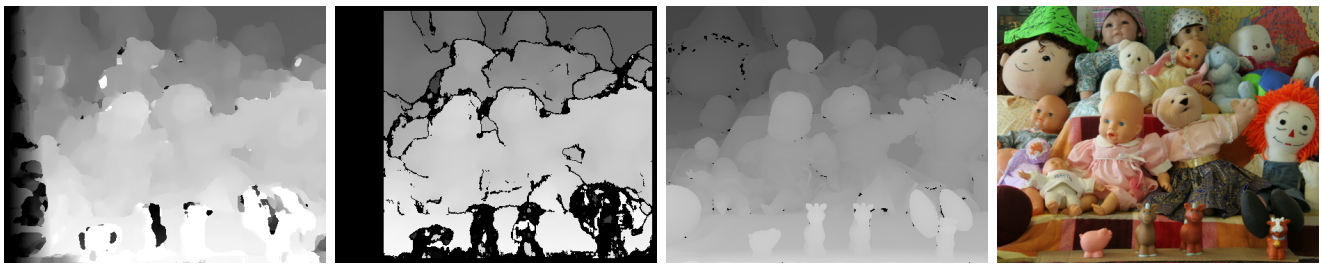
(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 9: Art-Disparity Map



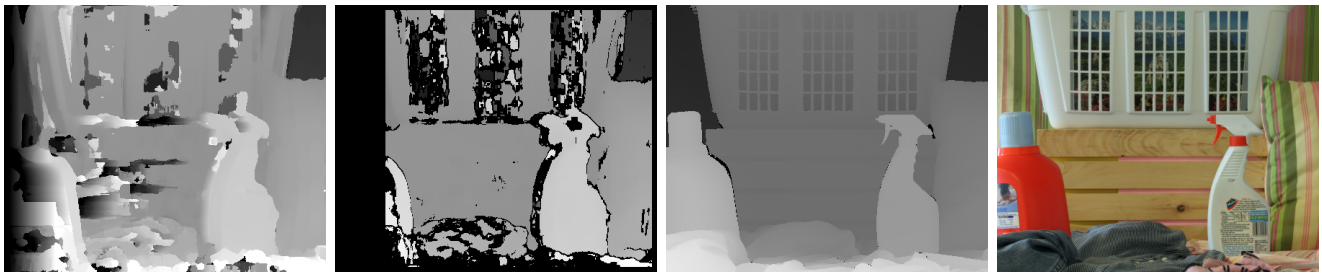
(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 10: Books-Disparity Map



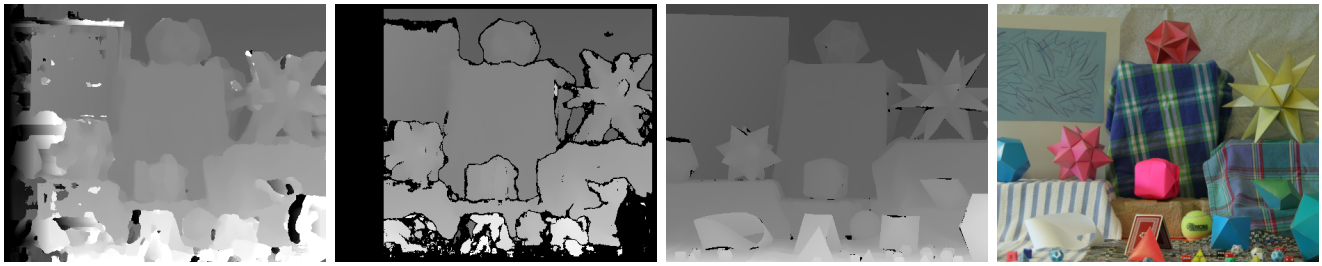
(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 11: Dolls-Disparity Map



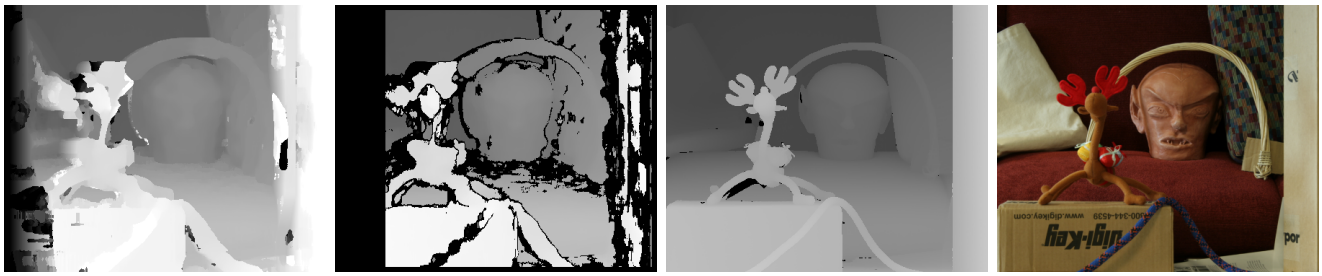
(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 12: Laundry-Disparity Map



(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 13: Moebius-Disparity Map

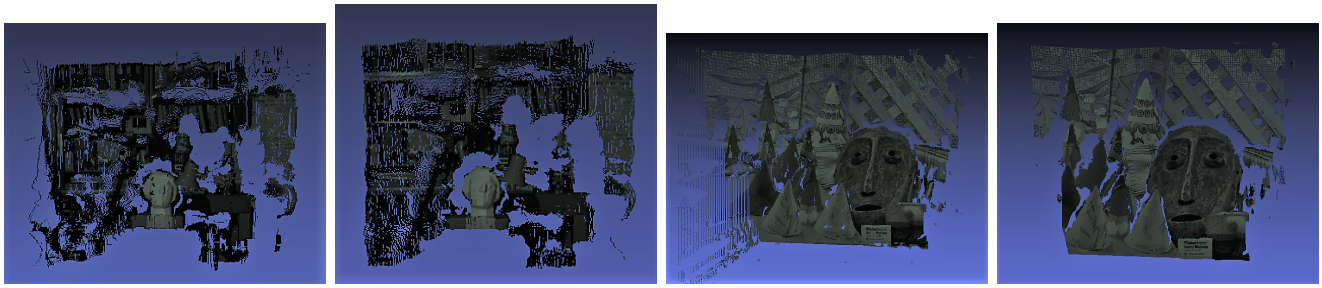


(a) My Custom BM (b) OpenCV StereoBM (c) Truth Map (d) Original Image

Figure 14: Reindeer-Disparity Map

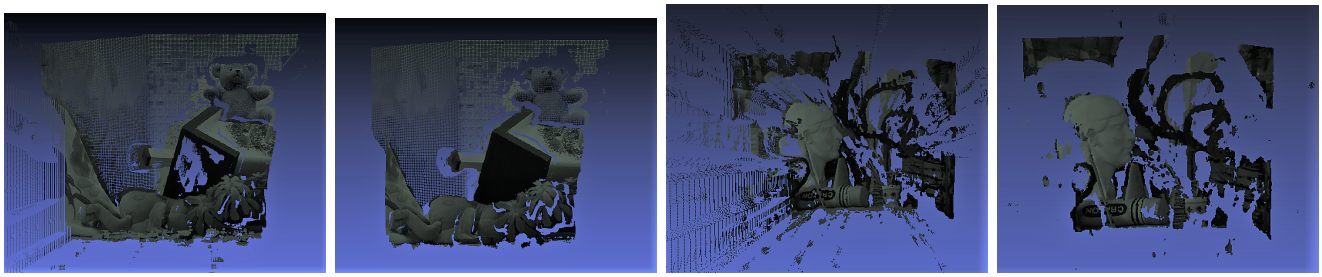
4.2.2 3D Points Cloud

Similar observations can be found from 3D points cloud. For all 3D points clouds generated, the center of the cloud contains better information from the images, while the edges are less accurate. The 3D points cloud corresponds well to the disparity map, where OpenCV algorithm discarded the points at the edges of the image that don't have a good match. My custom algorithm tries to keep all the points without including errors at the outlier of the 3D points cloud. In general, my custom algorithm and OpenCV algorithm have similar performances.



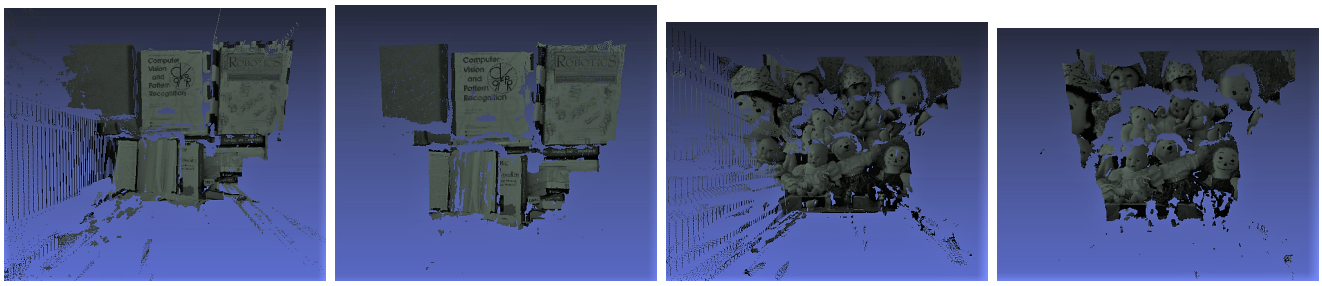
(a) My Custom BM (b) OpenCV StereoBM (c) My Custom BM (d) OpenCV StereoBM

Figure 15: (a),(b):Middlebury, (c),(d):Cones-3D Points Cloud



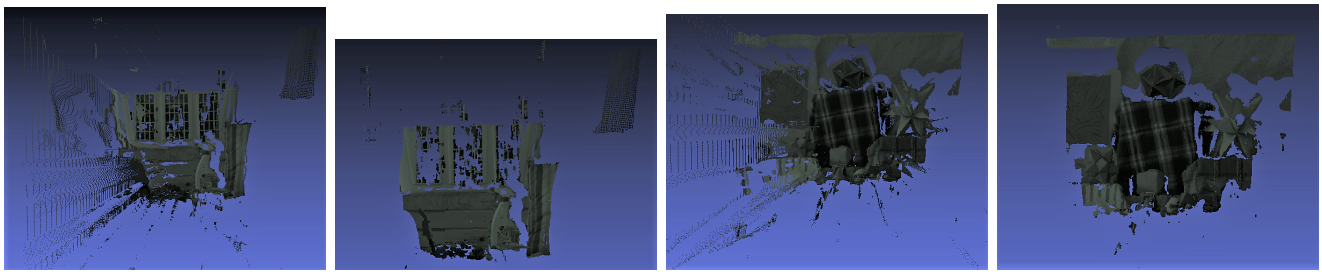
(a) My Custom BM (b) OpenCV StereoBM (c) My Custom BM (d) OpenCV StereoBM

Figure 16: (a),(b): Teddy, (c),(d): Art - 3D Points Cloud



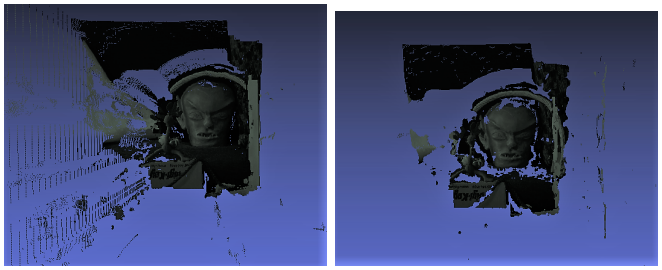
(a) My Custom BM (b) OpenCV StereoBM (c) My Custom BM (d) OpenCV StereoBM

Figure 17: (a),(b):Books, (c),(d):Dolls -3D Points Cloud



(a) My Custom BM (b) OpenCV StereoBM (c) My Custom BM (d) OpenCV StereoBM

Figure 18: (a),(b):Laundry, (c),(d):Moebius -3D Points Cloud



(a) My Custom BM (b) OpenCV StereoBM

Figure 19: Reindeer -3D Points Cloud

4.3 Qualitative Comparison

The following two metrics are used to measure the performance of the block algorithm. The first one is RMS of the pixel errors and the second one is the percentage of bad matching pixels. From RMS error, myBM and Opencv stereoBM implementation seem to have similar values for different images. For bad-matching percentage, the threshold is chosen at 50. For example, for image 'middlebury' both disparity maps have 35.57% of pixels that have an intensity error that's greater than 50.

		middlebury	teddy	cones	art	books	dolls	laundry	moetuis	reindeer
RMS error	OpenCV	57.76	49.55	38.63	42.40	43.21	42.40	39.67	34.90	44.63
	MyBM	57.90	49.25	37.72	41.81	41.64	40.70	38.44	32.61	43.16
Bad Matching	OpenCV	35.57%	31.82%	8.83%	24.34%	20.73%	31.55%	22.22%	11.48%	35.45%
Percentage	MyBM	35.57%	31.76%	8.41%	28.56%	15.32%	28.25%	21.34%	10.81%	40.81%

Table 2: Algorithm Performance using Cost Function SAD: RMS and Bad-matching Percentage

Different images contains different level of details and noise, therefore, performing block matching using the same parameters will yield different level of performance. Judging from bad-matching percentage, image 'cones' have the best result. This agrees well with the disparity map generated and the 3D points cloud, where we can see from Figure 7 and Figure 15(c),(d) that they contains most accurate information extracted form the stereo image pair. Looking at the same metric, the worst result is from the disparity map generated by myBM with the highest bad-matching percentage. The same can be observed from the corresponding disparity map and 3D points cloud in Figure 14(a) and Figure 19(a), where most points are missing.

In general, the two metrics agree well with each other. When the bad-matching percentage is high, the RMS error tends to have higher value as well. In order to see if the metrics reliable and if different cost function can improve the performance, cost function SSD is used for the same test. The results are shown in Table 3.

		middlebury	teddy	cones	art	books	dolls	laundry	moetuis	reindeer
RMS	OpenCV	57.76	49.56	38.62	42.40	43.20	42.40	39.67	34.80	44.63
	MyBM	57.87	49.28	37.75	41.94	41.72	40.75	38.43	32.66	43.17
Bad Matching	OpenCV	35.57%	31.82%	8.83%	24.34%	20.73%	31.55%	22.23%	11.48%	35.45%
Percentage	MyBM	35.56%	31.87%	8.44%	29.04%	12.76%	28.34%	21.35%	10.82%	40.90%

Table 3: Algorithm Performance using Cost Function SSD: RMS and Bad-matching Percentage

First of all, the evaluation metrics seem to be reliable as after changing the cost function the changes in the values are reasonable. There are only slight changes of values between the two tables. This also shows that changing cost function to SSD is not effective in terms of improving algorithm performance. The best result is still from image 'cones'.

The results demonstrate that the two implementations have equally poor performance. Because of the missing pixels in the disparity maps and global differences between images, an average pixel inten-

sity is subtracted from each pixel value before calculating the error.

5 Further Improvements

5.1 Pre and Post Processing

Based on current algorithm implementation, some pre and post processing can potentially improve the performance of the algorithm. Radiometric differences, which can be caused by noise, different camera settings, strength and position of the light source, have a major influence on the performance of the algorithm. For example, among all the images chosen for testing, 'middlebury' and 'reindeer' tends to have a darker overall illumination and the quality measurements show that the algorithm performed equally bad on them. Some pre-processing of the image may be able to correct the bias in lighting and provides a better input for the algorithm.

On the other hand, post processing can also help reduce overall errors in the results. For example, left-right consistency check can be used to invalidate mismatches and occlusions. Then invalid disparity areas can be filled by propagating neighbouring disparity values. From current results, it is obvious that a lot of the points are invalid causing empty regions in the 3D point cloud. If this method is applied after the algorithm, the missing areas in the 3D point cloud can be filled with estimated points.

5.2 Cost Functions

Different cost functions can also be used to improve the performance of the algorithm. SAD and SSD are both tested in this report and they both showed similar performance. Other cost functions, such as mutual information(MI) and NCC, can also be used. Some cost functions can be applied as filters, for example Laplacian of Gaussian(LoG), rank transform and mean filter. Hirscmuller and Scharstein tested all above 6 different cost functions using different stereo matching method. They concluded that the performance of cost functions depends on the stereo method used. For a local stereo method like block matching algorithm, rank transform appeared to be the best cost function.

5.3 Stereo Methods

Different stereo methods can be used to improve the performance of stereo correspondence matching. There are three different types of methods: local-based method, semi-global method and global method. Block matching algorithm implemented in this report is a type of local based method. Semi-global method can be SGM, dynamic programming(DP), etc. Global method includes graph cuts(GC). A global algorithm may be able to deal with some global noise and bias but it doesn't necessairlly perform better than the other two methods. In (H. Hirscmuller and D. Scharstein, 2007), global methods seem to produce the worst results. The best results are produced by semi-global method. Most importantly, to achieve a satisfying result, a suitable cost function should be chosen for a certain stereo method.

6 Conclusion

The custom stereo matching algorithm implemented is a block matching algorithm using sum of absolute difference as cost function. The performance of the custom algorithm is compared against OpenCV stereoBM algorithm. Both algorithms are tested on a set of different rectified images, outputting disparity map and 3D point clouds. Some ground truth disparity from middlebury database is used as comparison. Two quality metrics are used to judge the performance of the algorithm: root mean squared error between the computed disparity map and ground truth map, as well as bad matching percentage that calculates the number of pixels that have an error larger than a threshold.

In conclusion, my custom BM algorithm shows a similar performance as OpenCv BM algorithm quantitatively and qualitatively, but has much longer run time. Both performances are much worse than ground truth. To improve the accuracy of stereo matching, combinations of cost functions and stereo methods need to be further explored. Other than algorithm itself, pre and post processing of images can also help reduce the errors in general.

References

[1] Heiko Hirschm¨uller, Daniel Scharstein *Evaluation of Cost Functions for Stereo Matching*. Proc. IEEE Conf. Comput. Vis. Pattern Recognit. pp. 1-8 Jun. 2007.

[2] Middlebury stereo website. *www.middlebury.edu/stereo*.

[3] D. Scharstein and R. Szeliski. *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*. In International Journal of Computer Vision, volume 47, pages 7 42, 2002.

[4] Online tutorial *http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/*.