COMP3065 – Computer Vision Report

Please download the whole directory through the SharePoint:

https://nottinghamedu1-my.sharepoint.com/:f:/g/personal/scyzw1_nottingham_edu_cn/EkvRxjYF5SJLmAj_FPqvLn0BNrj7vaYaC3bifUYnjhMMEQ?e=XVepmW

Objective

In this project, the main objective is to compute the depth maps of several real-world scenes with stereo vision. To realize this objective, the program should be capable to solve several subproblems by implementing necessary functions. The first subproblem is the problem of finding corresponding pixels between pairs of images [1]. The second subproblem, reconstruction problem [1], deals with the calculation of disparity map provided with the corresponding pixel pairs. Apart from this, an additional problem is that the stereo camera should be calibrated to acquire intrinsic and extrinsic parameters of the camera and correct lens distortion.

I implemented several functionalities in this project.

1. The basic requirement, calculating the depth map (disparity map) after rectification.
2. Stereo camera calibration to acquire intrinsic parameters, distortion coefficients, fundamental matrix, and relative rigid transformation of the two cameras.
3. Real-time display of disparity map with tunning parameters by trackbar.
4. Draw of epipolar lines with some SIFT points to validate the result of rectification.
5. Disparity calculation algorithm using stereo SGBM algorithm (with or without filtering) provided by opencv library.
6. Several versions of disparity calculation algorithm implemented all by me of global searching scale and local searching scale, with SAD/SSD measurements according to lecture notes.

Implementation of Features

Stereo Camera Calibration

The aim of camera calibration is to estimate camera intrinsic parameters, extrinsic parameters, and distortion coefficients. This procedure is vital for computing the depth map as it relies on camera parameters to estimate and restore the structure of the scene in Euclidean space with multiple pictures shot by the stereo camera.

The projective transformation of a pinhole camera model is demonstrated below.

$$p = sA\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}[R|t]P \quad (1)$$

Where p and P are the 2D homogenous point in the projection plane and the corresponding 3D homogenous point respectively; $[R|t]$ is the rigid transformation from the world coordinate system to the camera coordinate system; A is the intrinsic matrix of the camera. This equation shows how a 3D point P is projected to a 2D point p in the camera's projection plane.

As this is a projection transformation with homogenous coordinates, the distance of the object relative to the camera is no longer an independent variable as homogenous coordinates can be scale arbitrarily with the coefficient s. Thus, multiple distinct objects can be projected and calculated as pairs of p and P in this one equation.

With an input chessboard image, the program can calculate corners of the chessboard using corner detection algorithm by calculating the gradients. Corners are utilized as p in the equation. The predetermined real-world chessboard points corresponds to P in the equation. The camera intrinsic matrix A can be estimated with adequate input of chessboard images.

Rectification

After camera calibration, the intrinsic parameters and distortion coefficients are acquired. On the input left and right images, rectification should be performed. The aim is to prepare rectified input images feasible for calculating disparity map. The reason is as follows.

Before calculating disparity, pairs of points in the left and right image should be corresponded. For unrectified images, for each pixel in the left image, whole right image must be iterated for finding correspondence. This operation is infeasible as the computational expenses can be massively large and error matching cases can regularly occur.

As in stereo camera system, there exists an epipolar constraint, which is that the correspondent point to a point in another image plane must lie in the corresponding epipolar line. Thus, this can be used for reducing search space for disparity calculation. In rectified image pair, epipolar lines in two images are horizontally parallel to each other. Thus, instead of searching the whole right image, only pixels in the right epipolar line that corresponds to the left epipolar line which contains the current left pixel need to be iterated. This can efficiently reduce computational expenses.

Disparity Map Calculation

After stereo rectification, corresponding epipolar lines become horizontally paralleled scanlines. They can be used to reduce calculation expenses of disparity effectively.

I implemented two types of disparity map calculation, stereo SGBM algorithm provided by opencv, and my self-implemented window-based algorithm using Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD) for similarity measurement. The library algorithm calculates quickly while self-implemented algorithm takes several minutes to finish.
For the library algorithm, I implemented two versions with or without filtering process.

In my implemented algorithm, it takes two grayscale images as input and calculate the disparity map as output. Initially, an empty disparity map is initialized with the shape of input images. Then all pixels in the left images are iterated, for each left pixel p_L, following steps are executed.

1. A patch of the left pixel W_L is created sliced from the left image with a specified size of window s.
2. The corresponding scanline in the right image is calculated.
3. Pixels in the right scanline are iterated and created with a patch W_R sliced from the right image with the same window size.

4. For each patch W_{R_i}, compute the Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD) with regard to W_L.

$$E(W_L, W_R) = \begin{cases} \sum_{p_1 \in W_L, p_2 \in W_R} |I_L(p_1) - I_R(p_2)| \\ \sum_{p_1 \in W_L, p_2 \in W_R} \left(I_L(p_1) - I_R(p_2)\right)^2 \end{cases} \quad (2)$$

Where I_L and I_R are the left image and right image, respectively.

5. Select the pixel p_R in the right epipolar line that is with the minimum SAD and calculate the Euclidean distance d with regard to p_L.
6. Fill the value of d into the disparity map at the location corresponding to the location of p_L.

After the iteration is finished, disparity map of the two images are calculated.

However, after experiments, it is found computational expenses are large and output of this algorithm is lack of precision. Therefore, I realized that it can be improved that rather than searching the whole scanline in the right image, only searching pixels within a limited searching scale can not only reduce computational expenses, but also improve precision. This is because matching points are more likely to be near each other, using local minimum cost can be more reasonable than using global minimum cost.

```
scanline_start_point <- 0
IF left_point_position > search_scale THEN
DO scanline_start_point <- left_point_position - search_scale
scanline_end _point <- img_width
IF img_width - left_point_position > search_scale THEN
DO scanline_end_point <- left_point_position + search_scale
```

I also implemented a version of this algorithm for improving accuracy, this version costs more computational expenses but may improve the accuracy for correspondence matching. In this case, instead of directly finding the scanline in the right image, the corresponding epipolar line is calculated again using the fundamental matrix F acquired from camera calibration. Then after discretization, points in the epipolar line are iterated. This may reduce error generated by rectification.

Depth Map Calculation

With the disparity map calculated, the depth map of one camera can be calculated supplemented with the focal length of the camera.

$$depth = \frac{B \cdot f}{disparity} \quad (3)$$

Where B is length of the baseline and f is the focal length of the camera.

Epipolar Lines with SIFT

After rectification, the epipolar lines in the left and right images become horizontally paralleled scanlines. To visualize the precision of rectification, I draw epipolar lines with some SIFT points on rectified left and right images. First, I find some key points and descriptors, and use FLANN algorithm to find matching pairs. Then, I import the fundamental matrix F from camera calibration and draw epipolar lines for each pair of matching points. Finally, the rectified images with some epipolar lines drawn on SIFT points are saved for examination.

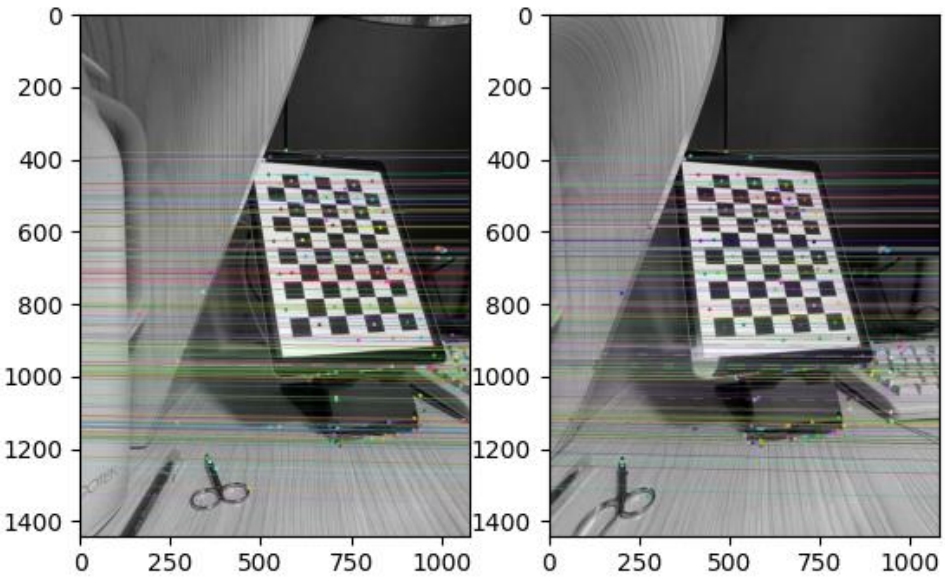

Figure 1 Epipolar Lines with SIFT Points

Dynamical Display with Tunning Parameters

In this feature, with input flag valued 1, the disparity map can be calculated and displayed dynamically while the users are tunning parameters of disparity calculation. For implementation, I created several track bars on the window and calculate disparity map according to parameters in the track bar repeatedly. Press `ESC` to

exit.

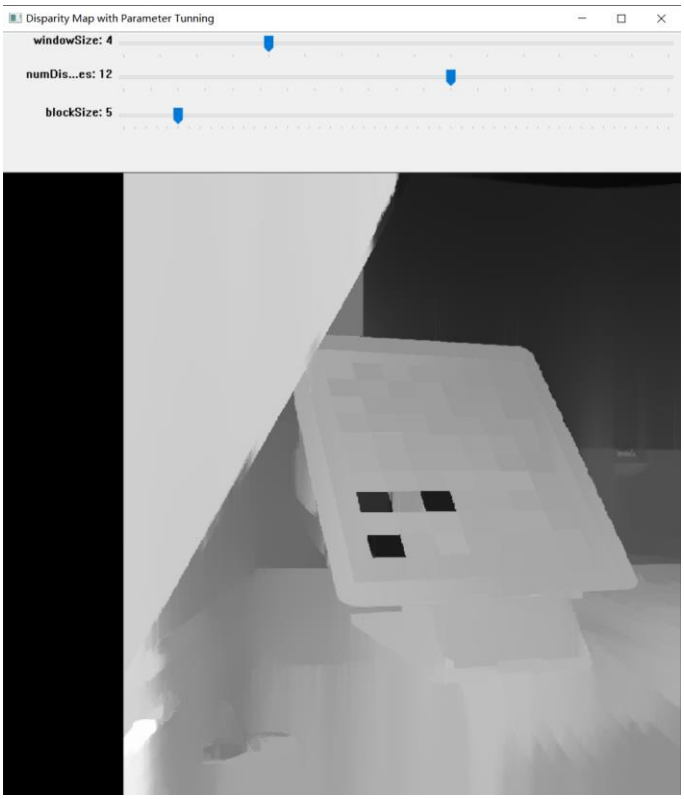

Figure 2 Dynamical Display of Disparity with Tunning Parameters

Results & Evaluation

For evaluation, comparations are divided into two sets, the comparation between SGBM algorithms with or without filtering, and the comparation between different versions of my algorithm. Two sets of input images are selected: image pair 1, my own images, and image pair 2, images downloaded from https://vision.middlebury.edu/stereo/data/.

Following are the original images.

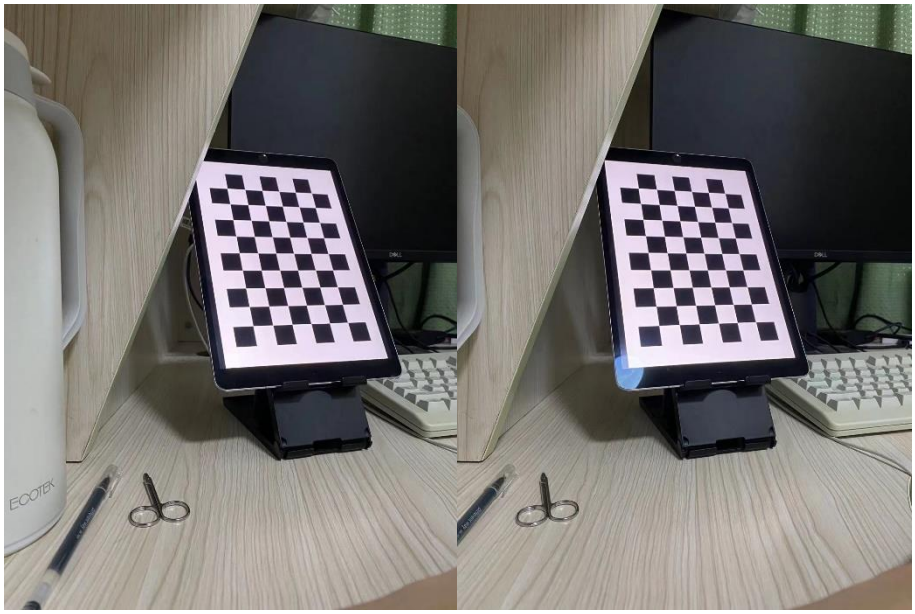

Figure 3 Original Image Pair 1

Figure 4 Original Image Pair 2

SGBM Algorithms

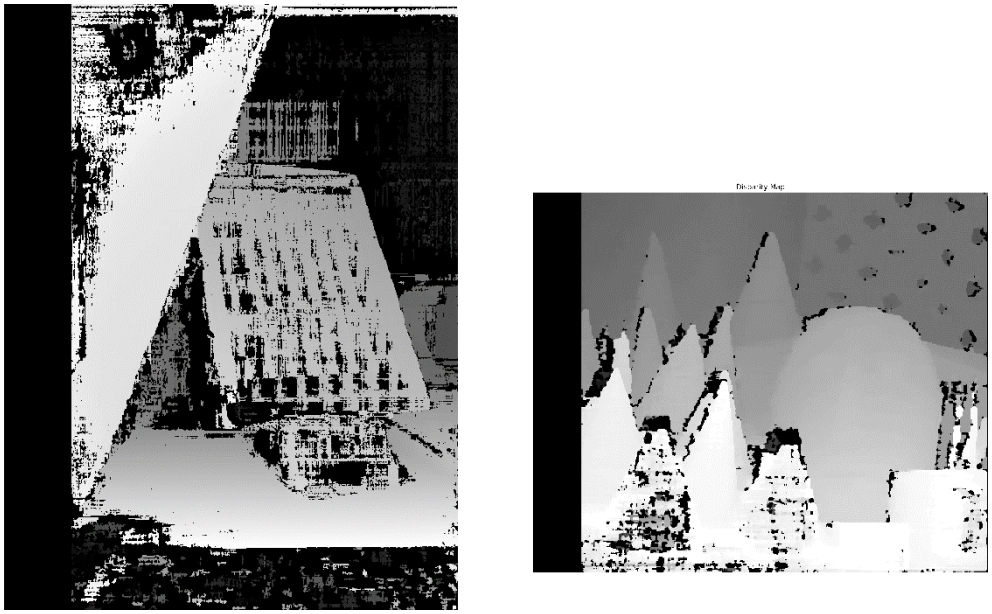

Figure 5 Output of Stereo SGBM Algorithm

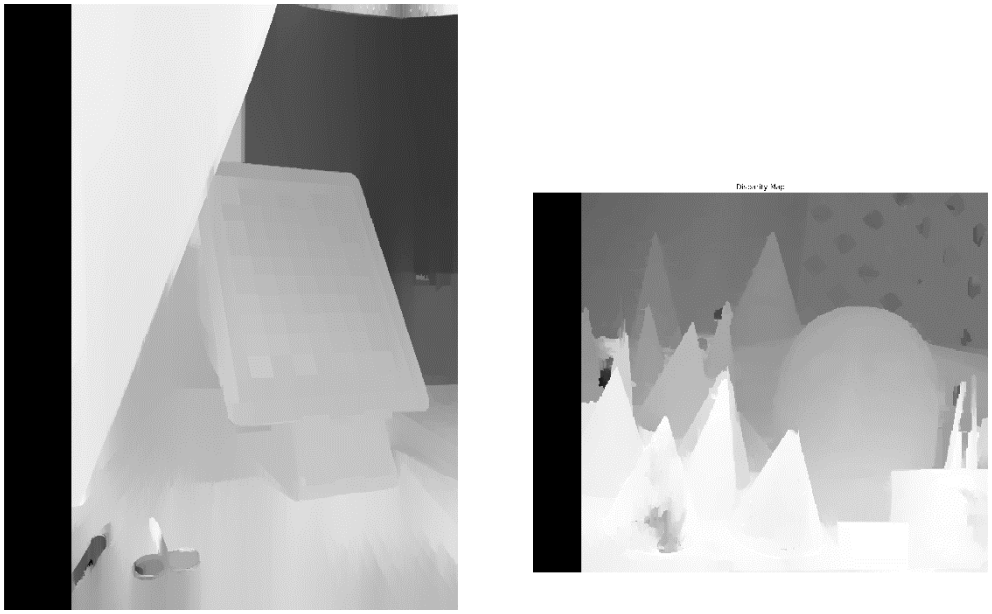

Figure 6 Output of Stereo SGBM Algorithm with Filtering

Assessment		**SGBM Algorithm**	**SGBM Algorithm Without Filtering**
Pros	1	Have robust matching at complicated textures.	
	2	\	Edges of objects are smooth.
Cons	1	Generate a black bar to the left of disparity map.	
	2	Have unrobust matching at plain textures or black objects.	
	3	Mismatch around left/right edges of objects.	\
	4	Pixel values can be extreme.	\

Table 1 Evaluation results for SGBM algorithms

Justification and analysis:

For pros 1, both algorithms successfully restore a plane shape for the background of image pair 2 which is complicated textured. Cons 2 is detected by examining the bottom area of image pair 1 and the black display screen/shades, where disparities are apparently calculated wrongly. This can be accounted that the cost of window-based matching is more likely being calculated wrongly as pixel values in the scanline are similar. Cons 3 can be intuitively detected and could be due that in one image, some object points are revealed and in the other, they are obscured. Cons 4 can be detected by comparing bottom area of image pair 1. This may be due to the calculation of cost function could possibly result in extreme value.

My Algorithms

Figure 7 Output of My Algorithm with Global Searching Scale (SAD left, SSD right)

Figure 8 Outputs of My Algorithm with Local Searching Scale (SAD left, SSD right)

Overall, the performance of self-implemented algorithms are noticeably less than library algorithms. There is rough distinction of disparities between closer and deeper points, with massive mismatches.

For the comparison with the two types of algorithms, overall, the algorithm with local searching scale produces less mismatches in terms of large area. This may be accounted that true matches are more likely to be near each other. Besides, the improved algorithm generates more white noises. This may be accounted to the limitation of searching scale.

For the comparison of SAD and SSD measurements, they generates similar results. But in the outputs of improved algorithm, result of SSD has more expansions and result of SAD have more corrosions. This can be detected by comparing the area of “DELL” logo.

Appendices

Following are outputs of the other two image sets.

Figure 9 Raw Image Set 2

Figure 10 Outputs of Image Set 2 My Algorithm, SGBM, SGBM with Filter in order

Figure 11 Raw Image Set 3

Figure 12 Outputs of Image Set 3 My Algorithm, SGBM, SGBM with Filter in order

1. Trucco, E. and A. Verri, *Introductory techniques for 3-D computer vision*. Vol. 201. 1998: Prentice Hall Englewood Cliffs.