Report Title:

Computer Vision Coursework Report
selected topic: Stereo Vision

Module Code: COMP 3065

Author: Wangkai Jin

Student ID: 20124870

Submission Date: 2022-04-29

Department of Computer Science
University of Nottingham Ningbo China

1 Report Overview and Project Objectives

This is my coursework report for Computer Vision Coursework, which I choose the Stereo Vision task. The project goal is to accurately calculate the disparity of input image pairs and estimate their corresponding depth map with the help of camera calibration. The report is outlined as follows. Section 2 shows the overview and Section 3 illustrates the details of my implemented functionalities/features. Section 4 reports the results obtained from my captured images and Section 5 presents the discussion on strength/weakness of my approach.

2 Implemented functionalities/features

This section presents the overview of implemented functionalities/features for stereo vision, which can be classified into five categories:

1. Stereo Camera Calibration
2. Block Matching algorithm
3. Semi-Global Matching algorithm (8-way aggregation)
4. Uniqueness Check, Sub-pixel interpolation, and Left/Right Check

3 Implementation Details

This section illustrates the implementation details of the above functionalities/features.

Figure 1: My stereo camera

3.1 Stereo Camera Calibration

The Stereo Camera Calibration is implemented with the support of OpenCV API and following the method introduced in [1]. Figure 1 shows my stero camera. The stereo camera calibration process work as follows. First, a printed chessboard is prepared for its clear patterns fro camera calibration. Then, multiple stereo images are taken by using a stereo camera. [1] requires a minimum of 8 image pairs to solve the least square problems and here I took 16 image pairs. Then the left and right images are separately loaded into the calibration program and I first calibrate the left and right camera respectively.

To calibrate cameras individually, I need to first find the object points coordinate in the 3D world. This is achieved by assuming the chessboard is placed still at the XY plane and the value along the z axis is 0 [2]. Therefore, we can assign grid-size (i.e., 20 in my work) value to the XY coordinates of different object points (e.g., (20,0), (40,0)). Then the images points are also needed for calibration. The images points on left/right images are retrieved by applying *cv2.findChessboardCorners()* and *cv2.cornerSubPix()* functions (shown in Figure 2 and Figure 3). Then all the object points and images are all utilized by *cv2.calibrateCamera()* to obtain the primary intrinsic and distortion coefficients of each camera. These intrinsic and distortion coefficients are further optimized by using *cv2.getOptimalNewCameraMatrix()*. The last step is to use the object points, image points, optimized intrinsic and distortion coefficients together to calibrate the stereo camera together to get their rotation matrix and translation vector, by using *cv2.stereoCalibrate()*. Therefore, the intrinsic, extrinsic and distortion coefficients of the stereo camera are obtained and we can use these parameters to rectify the stereo images.

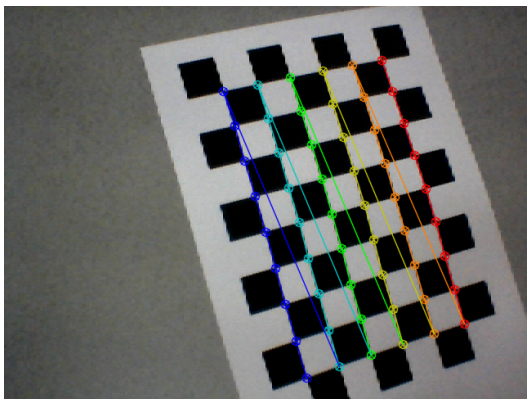

Figure 2: An example of a left image with detected corners.

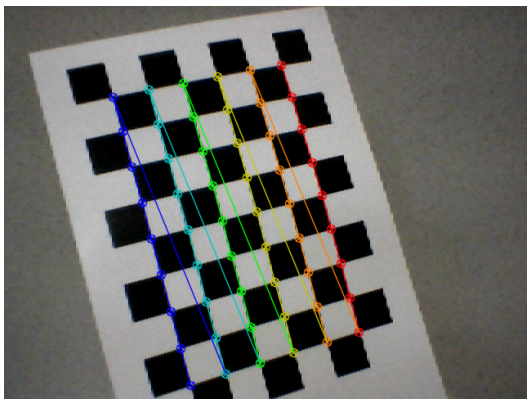

Figure 3: An example of a right image with detected corners.

3.2 Block Matching Algorithm

I implement the Block Matching (BM) algorithm with the Sum of Absolute Difference (SAD) as the cost for each block. The implementation steps of the algorithm are as follows. First, I extract a window of pixels (the size of window is 11 in my work) of the left images and extract a same-size window of pixels in the right images with the specified disparity ranges. Then, the SAD of these windows are calculated by using the following equation. The disparity of the right block which has minimum SAD with the left block is the current disparity in the pixel of the left image. The steps are iterated for all the pixels in the left images and the output is the disparity map.

$$SAD(I_1(i,j), I_2(x+i, y+j)) = \sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)|$$

3.3 Semi-Global Matching Algorithm

I also implement the Semi-Global Matching (SGM) to calculate the disparity map [3, 4]. Different from the cost model in the original paper [3, 4] (i.e., Mutual Information), I use

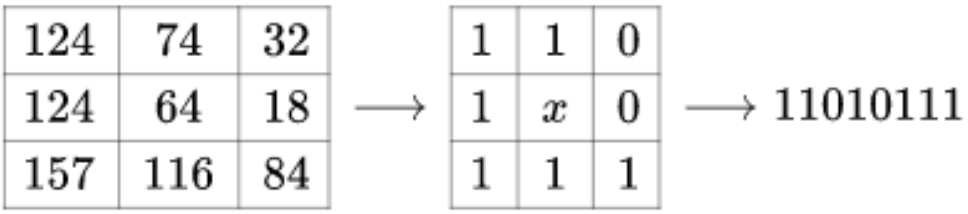

Figure 4: An example of Census Transform

the Census Transform (CT) as the default cost model for its robustness in matching cost for stereo vision in different local radiometric changes [5, 6].

First, the Census Transform (CT) is applied to the image pairs in a windowed manner. The Census Transform represents a pixel with a bit string computed from its neighbor pixels. An example is shown in Figure 4, where the neighbor bits are set to 1 if their values are greater than the central value or set to 0 if opposite. The bit strings are then formulated in a top-to-down and left-to-right order in the neighborhood window. The matching costs based on Census Transform is calculated by measuring the Hamming Distance of corresponding pixels (i.e., the number of different digits in the bitstrings), which is implemented by first xor matching bitstrings and then count the number of '1' in the xor outputs. In my implementation, I set the window size of Census Transform to 7 for better image characteristics extraction and compute the matching costs of corresponding pixels in different disparity levels using *numpy.bitwise_xor()* and *binn.count('1')*.

The second step for SGM is the cost aggregation, which adds two smoothness constraints to penalize different discontinuities and aggregates the costs in different directions. The calculation of the cost along on direction is shown in the following equation :

$$L_r(p,d) = C(p,d) + \min(L_r(p-r,d), L_r(p-r,d-1) + P1, \\ L_r(p-r,d+1) + P1, \min_i L_r(p-r,i) + P2) - \min_k L_r(p-r,k)$$

where $L_r(p,d)$ is the cost of current pixel in disparity d along direction r, $C(p,d)$ is the matching cost (i.e., cost of CT) of pixel p at disparity d, $L_r(p-r,d)$ is the previous cost of pixel p in r direction at disparity d, $L_r(p-r,d-1)$+P1 and $L_r(p-r,d+1)$+P1 are previous cost of nearby pixels with penalty P1, $L_r(p-r,i)$+P2 is the cost of faraway pixels with penalty P2, $\min_k L_r(p-r,k)$ is the minimum cost of pixels in r direction. The path cost calculation is implemented with a series of matrix operations for efficiency. First, an symmetric penalty matrix is created given to corresponding disparity ranges, where each rows specifies the penalty terms for corresponding disparity levels. An example penalty matrix is shown below, which has a disparity range of 4 and in each row, only positions that are before or after the 0s has penalty term P1.

$$\begin{bmatrix} 0 & P1 & P2 & P2 \\ P1 & 0 & P1 & P2 \\ P2 & P1 & 0 & P1 \\ P2 & P2 & P1 & 0 \end{bmatrix}$$

Accordingly, the cost of previous pixel in different disparity levels are formulated in a matrix of same size. Therefore, the second term in the above equation can be simply obtained by finding the min return of the sum of penalty matrix and the previous cost matrix. To calculate the last minimum cost of the previous pixel, I use an 2d array of size (path_length, disparity level) to store the previous computation results. So the last

term can also be obtained by finding the min value in this 2d array with proper index. In this coursework, I aggregate the 8-way costs for the disparity map due to the heavy memory consumption and latency in using the 16-way processing. After the costs along different paths are obtained, the last step is to find the disparity values in corresponding pixels that can minimize the following equation, which finds the disparity values that can minimize the overall cost.

$$S(p,d) = \sum_r L_r(p,d)$$

This is implemented by applying *numpy.armin()* on the aggregated results. Since the disparity level is between 0 and 64 in my implementation, the returned values are the disparity levels in the corresponding pixels. Therefore, We can obtain a preliminary disparity map.

3.4 Post Processing

After obtaining the preliminary disparity map, several processing techniques are applied to further improve the calculation accuracy.

Uniqueness Check: The first technique is the Uniqueness Check (UC) which aims to check whether the returned disparity is unique for each pixel (i.e., whether there are other disparity levels with similar cost). This is justified by finding another disparity level with a second minimum cost for each pixel sec_min_cost than the min_cost. The uniqueness stands if $sec\_min\_cost - min\_cost <= min\_cost * (1 - uniqueness\_ratio)$ where the $uniqueness\_ratio$ is set to 0.95 in my work. For pixels which fail the UC, it will be labeled as mismatch pixels and will be later filled with estimated values following the steps in [4], which are the median disparity values in the neighbor (i.e., in a 3x3 window). In addition, pixels that fail the UC will not proceed to the following two post processing techniques.

Sub-pixel Interpolation The Sub-pixel Interpolation is a common technique to introduce sub-pixel level disparity accuracy into the disparity map, aiming to improving the overall accuracy. It can be abstracted as fitting a curve near the best disparity level to reach a further minimum cost in sub-pixel accuracy. This is implemented for each pixel as follows. First, the current minimum cost c_0 and its corresponding disparity level d (i.e., the actual disparity in my work since the $min\_disparity$ equals 0) are retrieved based on the obtained disparity map. In addition, the cost c_1 at *d-1* and the cost c_2 at *d+1* are also retrieved. The output of Sub-pixel Interpolation at the current pixel's best disparity level is calculated as:

$$d_{sub} = d + \frac{c_1 - c_2}{2 * (c_1 + c_2 - 2c_0)}$$

Left/Right Check Left/Right Check is also a technique to improve disparity map accuracy by checking the consistency in the image pairs. For low latency, I use the costs of the left image to estimate the costs in the right image. This is implemented by first assign the costs of the left image at [i,j] in disparity d, $CostLeft[i, j_{left}, d]$ to the costs of the right image at $costRight[i, j_{left} - d, d]$. Then I apply SGM and sub-pixel Interpolation on the right image cost to obtain a right disparity map. First the corresponding disparity values $dispL$ at $[i, j_{left}]$ and $dispR$ at $[i, j_{left} - d]$ in the image pairs are compared with a threshold (i.e., 1). If the delta between the disparity values exceed the threshold, the

current pixel in the left image is a outlier and should be further classified into the occlusion or mismatch points. Such classification is achieved by inferring a disparity value in from the right view $dispL_{estimated} = dispLeft[i, j_{right} - d, d]$. If the $dispL_{estimated}$ is larger than $dispL$, then the current pixel is an occlusion point, otherwise it will be a mismatch points. For occlusion points, a fix is to fill the pixel with the second minimum disparity in a 3x3 neighbor and for mismatch points, a fix is to fill the pixel with the median disparity in a 3x3 neighbor.

4 Results

This section showcase the visual effects of my approach on public images and the captured images by my stereo camera.

4.1 Results on Public Images

Figure 5: Left Image

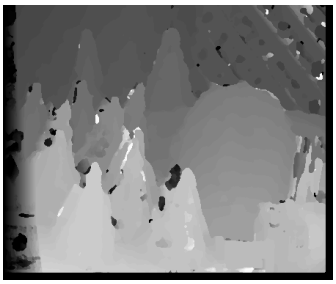

Figure 6: BM Result

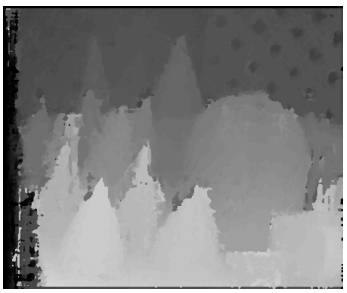

Figure 7: SGM Result

Figure 5, Figure 6 and Figure 7 shows the cone image from the Middle Burry dataset [7] and their corresponding disparity map from BM and SGM algorithms. Both disparity maps are obtained by naive BM and SGM (i.e., without any post processing techniques) based on census transform. From visual inspection, we can see that the naive algorithms can profile the disparity quite well. The visual effects of Figure 6 are inferior to that of Figure 7. This is because 1) the cost model of BM is less robust to that of the SGM and 2) SGM leverages cost aggregation to constrain the penalties of the costs on each pixel, which improve the performance.

Figure 8, Figure 9, Figure 10 and Figure 11 report the disparity obtained from naive SGM, SGM plus Sub-pixel Interpolation, SGM plus Uniqueness Check, SGM plus L/R Check. By comparing Figure 8 and Figure 9, we can see that the edges of cones in the down-left area are smoother. This is attributed to the Sub-pixel Interpolation that can delineate the shapes in the images clearly and smoothly. By comparing Figure 8 and Figure 10, we can see some of the mismatch points are cleaned, which also results in several black areas (i.e., the invalid points). By comparing Figure 8 and Figure 11, there are no notable differences from visual inspection. The possible explanation is that using the costs of right images to estimate the costs of the right images lead to high similarity in corresponding pixels in terms of disparity. Therefore, the L/R check has little effect in this case.

4.2 Results on My Stereo Images

Figure 12 to Figure 20 reports the left images captured by my stereo camera, their corresponding Census Transform outputs and disparity maps. The visual effects of their disparity maps are significantly poorer than that of the public images. An extensive tuning of penalty parameters, disparity ranges and re-capturing them in a brighter environment cannot fix such problem. One explanation of the poor performance is that my stereo camera inherently lacks denoising capability. This is observed from the CT outputs of the left images, which contains massive and unexpected black points. Such noises can interfere the cost matching and cost aggregation of the SGM, which lead to the poor performance in the disparity maps. Therefore, the poor performance of my approach on my stereo images is actually caused by the hardware limits, and the effectiveness of my approach with the additional post processing techniques on high-quality stereo images have already been justified in Section 4.1.

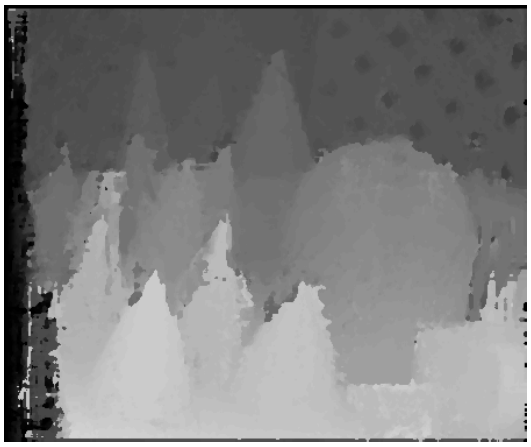

Figure 8: Naive SGM

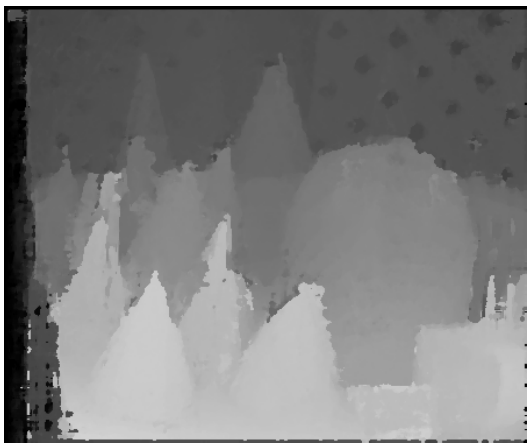

Figure 9: SGM plus Sub-pixel Interpolation

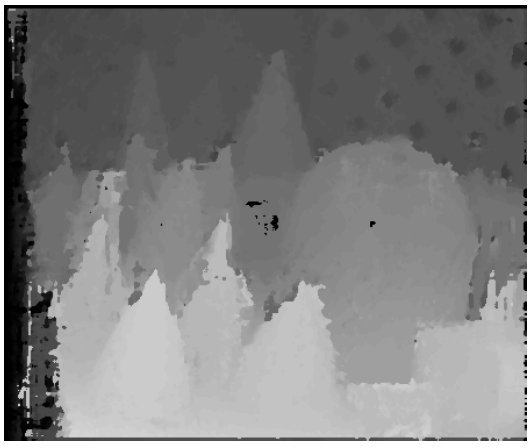

Figure 10: SGM plus Uniqueness Check

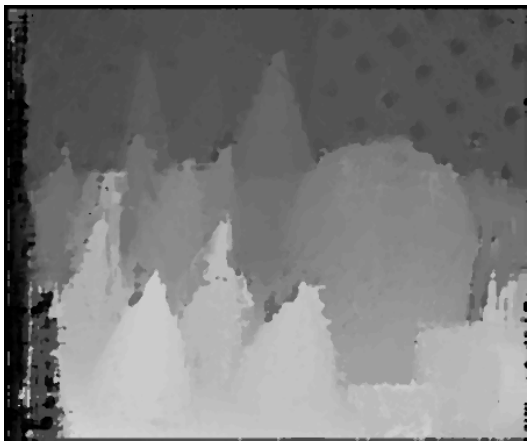

Figure 11: SGM plus L/R Check

Figure 12: Left Image

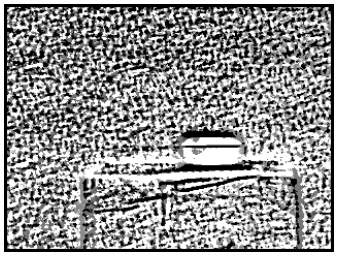

Figure 13: CT Output

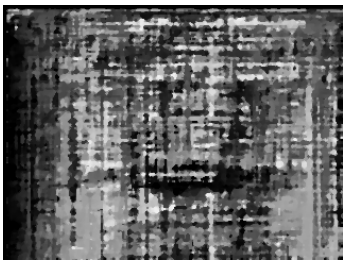

Figure 14: Disparity Map

Figure 15: Left Image

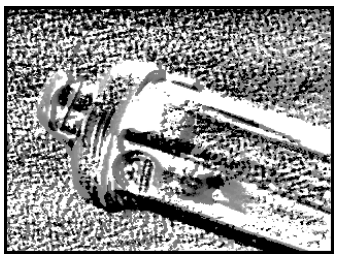

Figure 16: CT Output

Figure 17: Disparity Map

Figure 18: Left Image

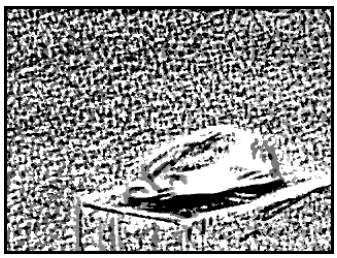

Figure 19: CT Output

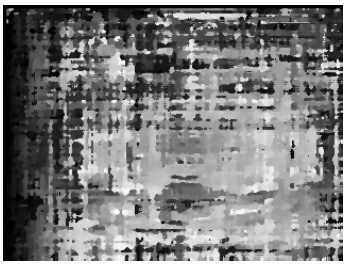

Figure 20: Disparity Map

5 Discussions

Strength: The strength of my approach is two folded. First, the algorithm has high recall rates on public stereo images (e.g., at best 82.5% for cone image pairs). This shows that my approach can effectively calculate most disparity levels in the public image pairs. Second, I estimate the right disparity by forecasting the costs of the left image to the right image during Left/Right check, rather than recomputing the disparity of the right image from the start. This can reduce the execution time and memory consumption to a great extent.

Weakness: The weakness of my approach is two folded. First, the processing speed is still slow despite my optimization techniques (e.g., dowmsample images). Since the speed correlates with the image size, disparity range, and the number of directions to aggregate costs. The processing speed can be further optimized by using tiled processing or SIMD vectorizations [8, 9]. Second, the currrent approach is still memory-hungry, as it the space complexity can reach $O(WHD)$ where W is the image width, H is the image height and D is the disparity range. This can be further optimized by variants of SMG like memory-efficient SGM [10].

References

[1] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

[2] OpenCV, "OpenCV:Camera Calibration." https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.

[3] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pp. 807–814, IEEE Computer Society, 2005.

[4] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, 2008.

[5] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Computer Vision - ECCV'94, Third European Conference on Computer Vision, Stockholm, Sweden, May 2-6, 1994, Proceedings, Volume II* (J. Eklundh, ed.), vol. 801 of *Lecture Notes in Computer Science*, pp. 151–158, Springer, 1994.

[6] H. Hirschmüller, "Semi-global matching: Motivation, development and applications," pp. 173–184, 09 2011.

[7] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16-22 June 2003, Madison, WI, USA*, pp. 195–202, IEEE Computer Society, 2003.

[8] S. K. Gehrig and C. Rabe, "Real-time semi-global matching on the cpu," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pp. 85–92, 2010.

[9] D. Hernandez-Juarez, A. Chacón, A. Espinosa, D. Vázquez, J. Moure, and A. López, "Embedded real-time stereo estimation via semi-global matching on the GPU," *Procedia Computer Science*, vol. 80, pp. 143–153, 2016.

[10] H. Hirschmüller, M. Buder, and I. Ernst, "Memory efficient semi-global matching," vol. I-3, 08 2012.