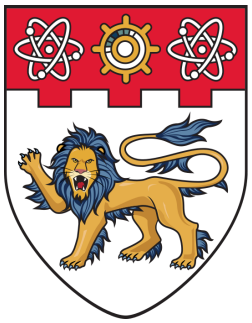


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

College of Computing and Data Science

Master of Science in Artificial Intelligence

AI6121 Computer Vision

Assignment 2 Report

Tan Jia Wei (Matric No.: G2403382B)

Tiffany Tan Ge Ru (Matric No.: G2404908D)

Fung Kwok Pong (Matric No.: G2405595A)

Zhou Chunxi (Matric No.: G2405212J)

Jin Zitong (Matric No.: G2403513F)

This document consists of **15** pages, including the cover page.

Contents

1 General Procedure of Disparity Computing

Disparity computing is the processing of calculating the difference in the positions of corresponding points in two rectified images of the same scene taken from different viewpoints (e.g., left and right cameras). This difference is used to estimate depth in a scene using stereo vision techniques. Disparity is closely related to parallax, where objects appear to shift positions between views based on their distance from the camera.

In stereo vision, disparity is crucial for generating a depth map, which allows us to perceive and reconstruct the 3D structure of the scene. Key concepts of disparity computing include:

- Rectified Images: Images are rectified to align the corresponding points on the same horizontal line (epipolar line), simplifying the correspondence search to a horizontal scan.
- Correspondence Problem: Disparity computing addresses the challenge of identifying corresponding points (pixels) in the two images.

The general procedure of disparity computing are:

- Image Rectification: First, the image pair must be rectified to ensure that corresponding lines lie on the same horizontal line. This alignment reduces the search for corresponding points to a one-dimensional scan along the row.
- Stereo correspondence: The goal of disparity computing is to find corresponding pixels in the left and right rectified images. For each pixel in one image (e.g., left image), search for the corresponding pixel in the same row in the right image. This search is performed by sliding a small window or patch around the pixel in the left image and comparing it with patches along the corresponding row in the right image. Common techniques for comparing patches include Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD) or Normalized Cross-Correlation (NCC).
- Disparity calculation: Once a corresponding pixel is found, the disparity is calculated as the horizontal distance (in pixels) between the matched pixel positions in the left and right images. Disparity = $X_{\text{left}} - X_{\text{right}}$, where X_{left} and X_{right} are the horizontal coordinates of corresponding points in the left and right images, respectively.
- Disparity map generation: The disparity values for all pixels are used to create a disparity map, which is a grayscale image where the intensity of each pixel represents the disparity value. Darker pixels indicate smaller disparities (farther objects), while brighter pixels represent larger disparities (closer objects).

1.1 Several Methods for Comparing Patches

In this section, we briefly introduce several similar methods for comparing patches. As we have mentioned before, common methods include Sum of Absolute Differences (SAD), Mean Absolute Differences (MAD), Sum of Squared Differences (SSD), Mean Squared Differences (MSD), and Normalized Cross-Correlation (NCC).

Let $S(x, y)$ be the image in which we search for the patch, and $T(x, y)$ be the patch we want to find in S. $m \times n$ and $M \times N$ are their size respectively. In these algorithms, we will take a subgraph of size $M \times N$ with (i, j) as the upper left corner of the subgraph, and calculate its similarity or distance D with the patch. We repeat the above calculation and traverse the entire image. Here are their formulas for computing distance

D. SAD and MAD compute the 1-norms of distance of pixels. (The only difference between them is that MAD compute the average value of 1-norms, while SAD simply sum up the 1-norms.)

$$\textbf{SAD: } D(i,j) = \sum_{s=1}^{M}\sum_{t=1}^{N} |S(i+s-1, j+t-1) - T(s,t)| \tag{1}$$

$$\textbf{MAD: } D(i,j) = \frac{1}{MN}\sum_{s=1}^{M}\sum_{t=1}^{N} |S(i+s-1, j+t-1) - T(s,t)| \tag{2}$$

Similarly, SSD and MSD compute the 2-norm. SSD is the 'sum' version and MSD is the 'average' version.

$$\textbf{SSD: } D(i,j) = \sum_{s=1}^{M}\sum_{t=1}^{N} (S(i+s-1, j+t-1) - T(s,t))^2 \tag{3}$$

$$\textbf{MSD: } D(i,j) = \frac{1}{MN}\sum_{s=1}^{M}\sum_{t=1}^{N} (S(i+s-1, j+t-1) - T(s,t))^2 \tag{4}$$

The formula of NCC is a bit more complicated. NCC computes the sum (normalized) of the cross correlation between subgraphs and the patch. Let $E(i,j)$ and $e(T)$ represent the average grayscale value of the subgraph at (i,j) and average grayscale value of the patch. We have,

$$\textbf{NCC: } D(i,j) = \frac{\sum_{s=1}^{M}\sum_{t=1}^{N} |S(i+s-1, j+t-1) - E(i,j)| \cdot |T(s,t) - e(T)|}{\sqrt{\sum_{s=1}^{M}\sum_{t=1}^{N} (S(i+s-1, j+t-1) - E(i,j))^2 \cdot (T(s,t) - e(T))^2}} \tag{5}$$

In this report, we will first apply SSD as the implementation of baseline criterion, and then discuss possible improvements by changing the comparison criterion and matching algorithm as well.

2 Code Implementation for Computing Disparity of Images

The pixel values of a disparity map is derived by solving the correspondence problem. It is the X positional gap between corresponding pixel on the left and right image, where the difference between the respective patches are at minimal. There are multiple techniques to compute this patch difference; in our implementation, we have used the SSD method.

2.1 Sum of Squared Differences (SSD)

The following is an excerpt from our implementation of the Sum of Squared difference:

```python
# Compute sum of square difference
def sum_squared_diff(first, second):
    first = np.array(first)
    second = np.array(second)
    squared_diff = (first - second) ** 2
    return np.sum(squared_diff)
```

A single tile from the left image is compared with tiles derived from the (whole horizontal stripe on the) right image. All SSD results are stored into a list. Since **correspondence is based on minimal regional distance**, by using *argmin*, we return the **corresponding X position** in the right image:

```
# Compute index that yields min diff for row
def idx_min_sum_square_diff(pixels, tile, reference):
    start = math.floor(pixels / 2)
    end = reference.shape[1] - math.floor(pixels / 2) - 1
    diff = []
    for step in list(range(start, end)):
        diff.append(sum_squared_diff(tile,reference[:, step-math.floor(pixels / 2): step + math.floor(pixels / 2)]))
    return np.argmin(np.array(diff)) + start
```

Finally, the **pixel value of the disparity map** is computed using the X **positional gap between corresponding pixel on the left and right image**:

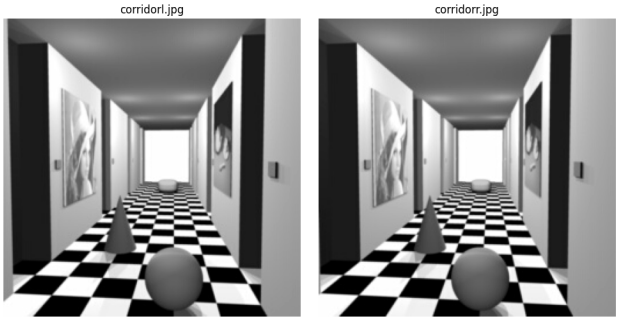
```
for step_y in list(range(start_y, end_y)):
  for step_x in list(range(start_x, end_x)):
      tile = l_image[step_y - math.floor(pixels / 2): step_y + math.floor(pixels / 2), step_x - math.floor(pixels / 2): step_x + math.floor(pixels / 2)]
      h_stripe = r_image[step_y - math.floor(pixels / 2): step_y + math.floor(pixels / 2), :]
      best_x = idx_min_sum_square_diff(pixels, tile, h_stripe)
      d_map[step_y, step_x] = abs(best_x - step_x)


d_map_tmp = np.uint8(d_map * 255 / num_disparity)
plt.imshow(d_map_tmp, cmap=color_map, interpolation='nearest')
plt.show()
```

Upon computing **all** correspondence and their respective X positional gaps, we have derived the **full** disparity map using the two rectified images.

3 Discussion of Adjustable Parameters

3.1 Original Dataset

The original dataset consists of 2 pairs of sample images, labeled corridor and triclopsi2, in Joint Photographic Experts Group (JPG/JPEG) format. We have included the tsukuba sample for additional comparisons. Figure 1 shows the original images.



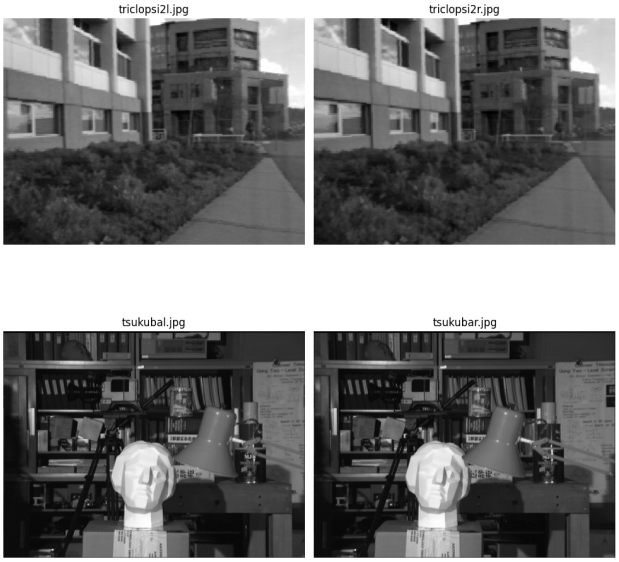


Figure 1: Dataset of original sample images.

In the next two sections we will discuss the results of the disparity maps from the SSD algorithm, choice of parameters and factors affecting disparity. The disparity map plays a crucial role in stereo vision by enabling the estimation of depth information from 2D images. It is a visual representation of the pixel-wise differences in position between corresponding points in two stereo images (left and right images taken from slightly different viewpoints for this report)

3.2 Comparison on Sliding Window Size

The window size (or block size) defines the number of pixels considered around each point in the image during the matching between the left and the right versions of each image. It acts as a local region to compare the pixel intensities. In the SSD algorithm, the window size is one of the parameters to compute disparity. The outputs of the difference between small and big window sizes will be shown.

3.2.1 Disparity Maps

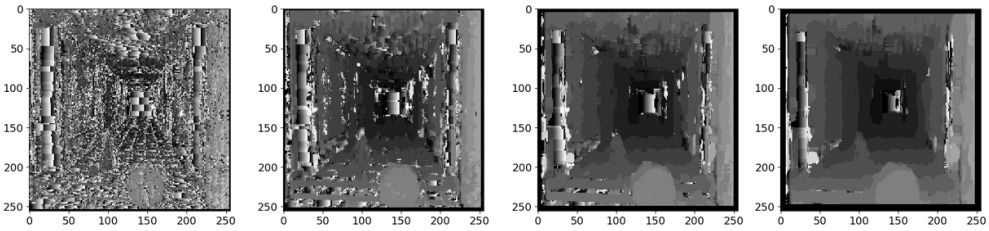


Figure 2: Corridor image - blockSize 3,7,11,15

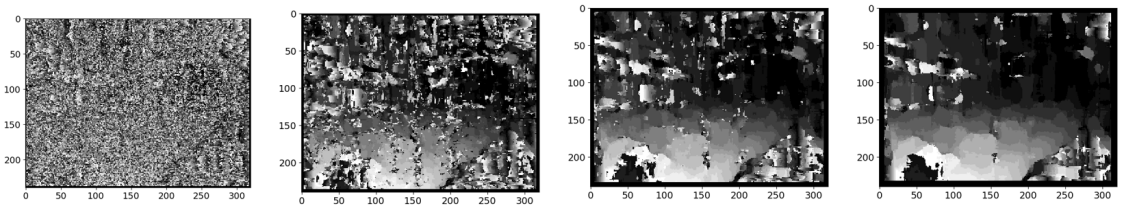


Figure 3: Triclopsi2 image - blockSize 3,7,11,15

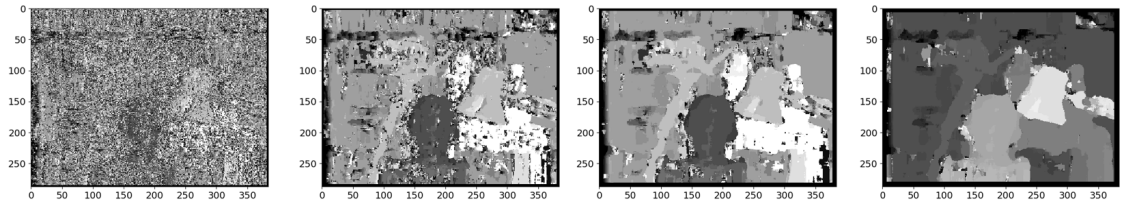


Figure 4: Tsukuba image - blockSize 3,7,11,15

3.2.2 Discussion on Disparity Maps

From the images, we can see that increasing the window size from 3,7,11 to 15 helped to smooth disparities but resulted in reduction of fine details. Smaller window captures details as they compare small regions of the image, which inevitably results in more noise. This is caused by incorrect matching when there are multiple optimal candidates, and is more apparent when using small windows on images with homogeneous regions. The sensitivity to noise manifest as the scattered high-disparity values (light gray or white) across the maps. This is especially noticeable in regions that should ideally be smooth (like walls of the corridor image and of triclopsi2). Increasing the window size resulted in averaging out depth information, leading to a loss of accuracy in areas where depth changes rapidly. The window size optimal for each image depends on its features.

Comparing the three scenes:

1. Corridor - This scene contains sharp depth images and fine details like object boundaries and distinct checkered floor. A smaller window size will preserve these details and provide more accurate disparity estimation, especially for the geometric shapes and the floor's texture.

2. Triclopsi2 - Overall this scene has relatively low textures. The building's walls and path are low-texture, and the path is another low-texture area. Hence a larger window size would be ideal as there are not too many transitions in depth.

3. Tsukuba - This scene is filled with a variety of textures, ranging from low textures like table surfaces to highly textured areas like bookshelves. Furthermore, it also has a distinct foreground and background. A moderate window size would be ideal to capture the details from the textured areas while also smoothing out some of the noise in low-texture areas.

3.3 Comparison on Maximum Disparities

Other than experimenting the sliding window size (blockSize), we also explored another parameter, which is the maximum number of disparities. It represents the maximum disparity value that the algorithm will test for each pixel in the left image while searching for its corresponding pixel in the right image. A larger number of disparities allows the algorithm to capture objects that are closer to the camera, since close objects have larger shifts between the images.

3.3.1 Disparity Maps

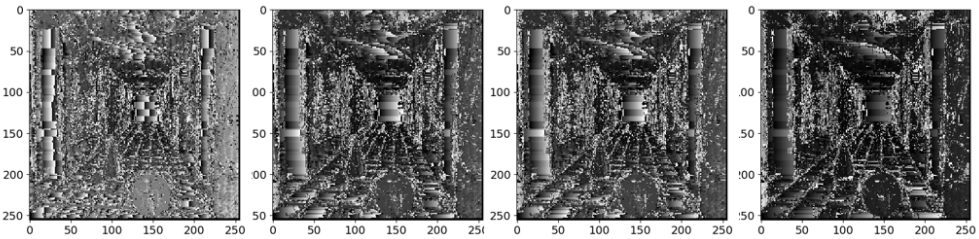


Figure 5: Corridor image (blockSize 3) - numDisparities 16, 32, 64, 255

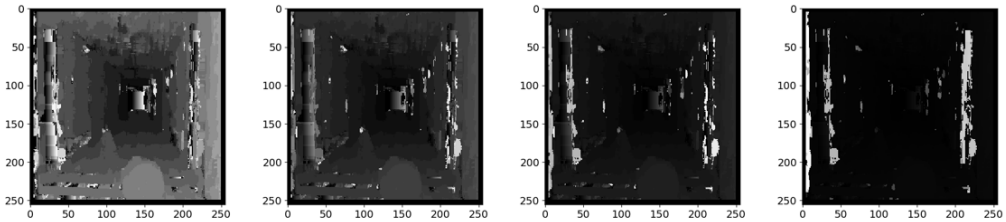


Figure 6: Corridor image (blockSize 11) - numDisparities 16, 32, 64, 255

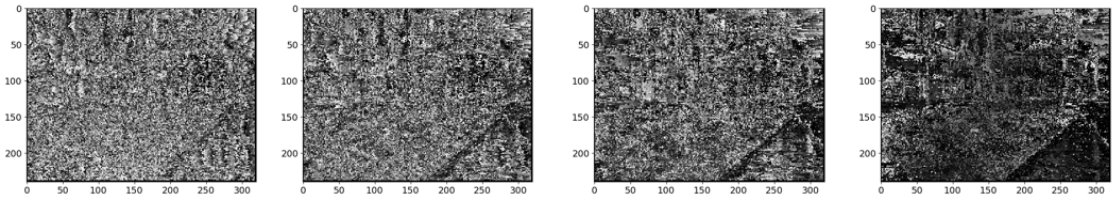


Figure 7: Triclopsi2 image (blockSize 3) - numDisparities 16, 32, 64, 255

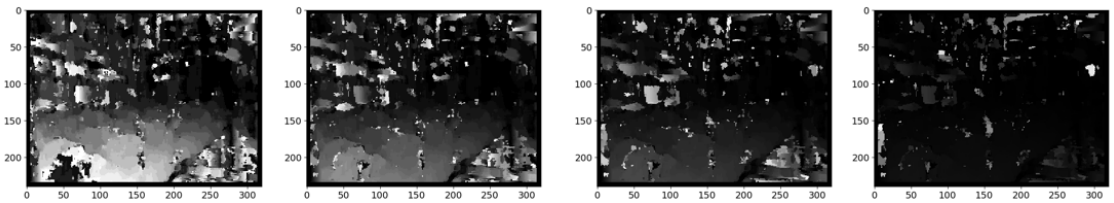


Figure 8: Triclopsi2 image (blockSize 11) - numDisparities 16, 32, 64, 255

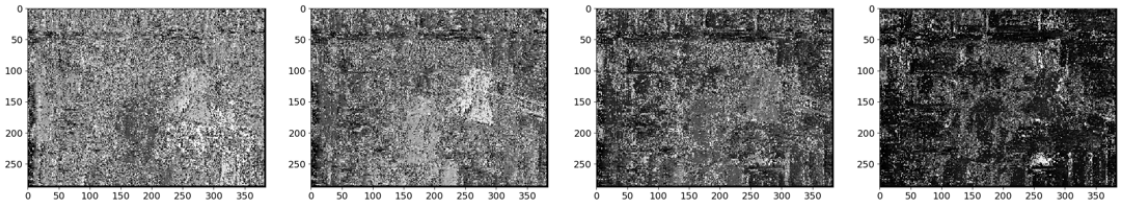


Figure 9: Tsubaka image (blockSize 3) - numDisparities 16, 32, 64, 255

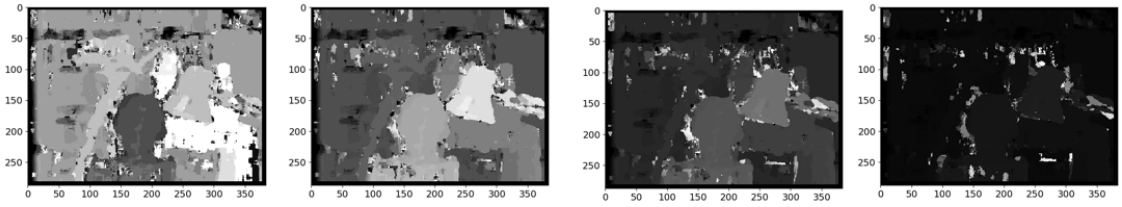


Figure 10: Tsubaka image (blockSize 11) - numDisparities 16, 32, 64, 255

3.3.2 Discussion on Disparity Maps

For this section, we explored increasing the parameter numDisparities from 16 to 255 for blockSize 3 and 11 for each of the three sample images. Small max disparities (16 and 32) tend to result in poor depth estimation across all block sizes and scenes. The limited range of disparities makes it difficult to capture depth variations, leading to noisy and unreliable maps. When max disparities were increased, particularly at numDisparities = 255, they captured more depth variation, providing clearer and more detailed depth information. This is especially true for larger block sizes (11), where the disparity maps are much smoother and have better structure. The best configuration of window size and number of disparities depends on the contents of image.

1. **Corridor**: Larger block sizes (11) and larger disparities (64, 255) produce the best results, as they capture the depth of the corridor more accurately and reduce noise in textureless areas like walls.
2. **Triclopsi2**: This scene benefits greatly from larger block sizes (11) and higher disparities. The small block size (3) results in too much noise, while larger disparities help bring out depth variation.
3. **Tsubaka**: Similar to the other scenes, the best results are achieved with larger block sizes (11) and larger disparities (64, 255). The small block size struggles with textureless areas and introduces noise, whereas the larger block size smooths out the map and reduces noise.

It is also to be noted that some images have black bars at the sides. This is because of how the algorithm handles boundary regions and window size during disparity computation. At the image edges, there are not enough pixels on one side of the image to create a full block. For instance, at the left edge of the image, there are not enough pixels to fully construct a block to compare with the other image. Also, if the number of disparities is too large relative to the scene, the algorithm runs out of pixels to compare, especially at the edges of the image, resulting in the black bars.

3.3.3 Other Factors

From the above 2 chapters, we can see how the parameters window size and maximum disparities affect the disparity maps. Additionally, SAD algorithm was also explored as a further improvement and will be

discussed in the later sections.

4 Improvements in Disparity Computing

In this chapter, possible improvements in disparity computing are revealed and investigated in five sections: cost volume formulation, matching algorithms, experiment results, discussions on the results and state-of-the-art results.

4.1 Cost Volume Formulation

In stereo matching, the cost volume, capturing the similarity of possible correspondences across two input images, is a key search space for cost aggregation where the disparity of a pixel is chosen by minimizing costs (Xu and Zhang, 2020). As introduced in Section 1.1, several costs can be chosen as the criterion of disparity, such as SAD, SSD and NCC. In this section, the three aforementioned costs are used to compute cost volumes. The general representation of cost volume is given by

$$\boldsymbol{V}(x, y, d) = \mathcal{C}(\boldsymbol{I}_l(x, y), \boldsymbol{I}_r(x - d, y)), \tag{6}$$

where $\boldsymbol{I}_l$ is the left image, $\boldsymbol{I}_r$ is the right image, $x \in \mathcal{W}$, $y \in \mathcal{H}$, $d \in \mathcal{D}$, $\mathcal{W} = \{0, 1, \cdots, W-1\}$ is the range of width of the image, $\mathcal{H} = \{0, 1, \cdots, H-1\}$ is the range of height of the image, $\mathcal{D} = \{0, 1, \cdots, D-1\}$ are all valid disparities, and D is the number of disparities and also the maximum disparity. In Equation 6, $\mathcal{C}(\boldsymbol{I}_l, \boldsymbol{I}_r)$ is the cost function chosen from Equation 1, 4 and 5. The cost volume formulation above is applied throughout the following disparity computations.

4.2 Matching Algorithms

The process of cost aggregation requires efficient matching algorithms. In this section, two algorithms are investigated to aggregate cost volumes and produce disparity maps.

4.2.1 Greedy Matching

Greedy Matching provides a computationally inexpensive way of obtaining the best disparity for each pixel by taking the point with the minimum in the cost volume along the disparity axis (in this section, the horizontal axis or x-axis). The mathematical form of Greedy Matching is given by

$$\boldsymbol{D}(x, y) = \underset{d}{\operatorname{argmin}} \boldsymbol{V}(x, y, d), \tag{7}$$

where $\boldsymbol{D}$ is the produced disparity map.

4.2.2 Semi-Global Matching

A different approach is Semi-Global Matching (SGM, Hirschmuller, 2008). Rather than looking for the best fit along the disparity axis, SGM takes surroundings into consideration. Each pixel with a corresponding unary cost given by the cost volume is assigned an additional pairwise cost that depends on wherever the

neighboring pixels have a similar depth value or deviate significantly. The objective of this algorithm is

$$\min_{d_{\boldsymbol{p}}} \left[\sum_{\boldsymbol{p} \in \mathcal{I}} \boldsymbol{V}(\boldsymbol{p}, d_{\boldsymbol{p}}) + \sum_{(\boldsymbol{p}, \boldsymbol{p}') \in \mathcal{E}} f(d_{\boldsymbol{p}}, d_{\boldsymbol{p}'}) \right], \tag{8}$$

where $\boldsymbol{p}$ is an image pixel, $d_{\boldsymbol{p}}$ is the best disparity of $\boldsymbol{p}$, $\mathcal{I}$ is the image, $\boldsymbol{V}$ is the cost volume, $\mathcal{E}$ is the edge containing $\boldsymbol{p}$, and the function f is the pairwise cost penalizing jumps between neighboring pixels, given by

$$f(d_{\boldsymbol{p}}, d_{\boldsymbol{p}'}) = \begin{cases} 0, & \text{if } d_{\boldsymbol{p}} = d_{\boldsymbol{p}'}, \\ L_1, & \text{if } |d_{\boldsymbol{p}} - d_{\boldsymbol{p}'}| = 1, \\ L_2, & \text{otherwise.} \end{cases} \tag{9}$$

In the experiment of this chapter, $L_1 = 0.025$ and $L_2 = 1.0$. Then, the disparity map $\boldsymbol{D}$ is calculated as

$$\boldsymbol{D}(x, y) = \underset{d}{\operatorname{argmin}} \left[\boldsymbol{V}(x, y, d) + \sum_{e \in \mathcal{E}} \sum_{y \in \mathcal{H}} \sum_{x \in \mathcal{W}'} \sum_{d \in \mathcal{D}} \boldsymbol{M}_e(x+1, y, d) \right], \tag{10}$$

where $\mathcal{E} = \left\{ \underrightarrow{H}^+, \underrightarrow{H}^-, \underrightarrow{W}^+, \underrightarrow{W}^- \right\}$ are the four summation directions, $\mathcal{W}' = \mathcal{W} - \{W-1\}$ to ensure the term $x+1$ would not overflow and

$$\boldsymbol{M}_e(x+1, y, d) = \min_{d' \in \mathcal{D}} \left[\boldsymbol{M}_e(x, y, d) + \boldsymbol{V}(x, y, d) + f(d, d') \right], \tag{11}$$

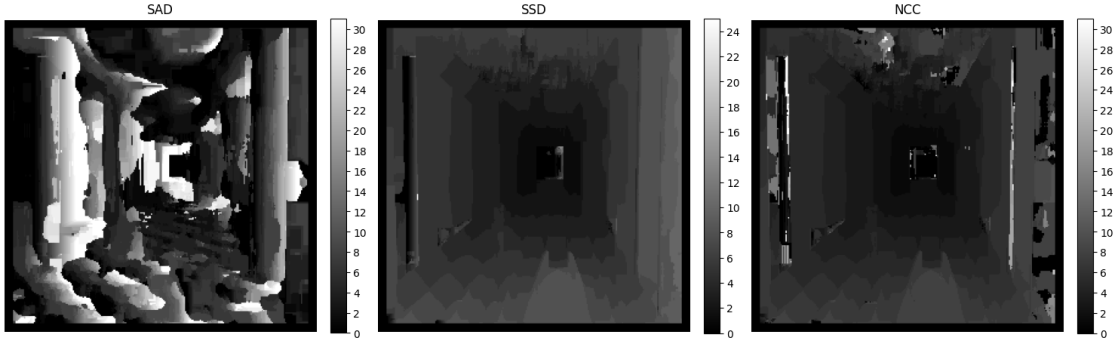
where $\boldsymbol{M}_e(0, y, d) = 0$.

4.2.3 Semi-Global Block Matching

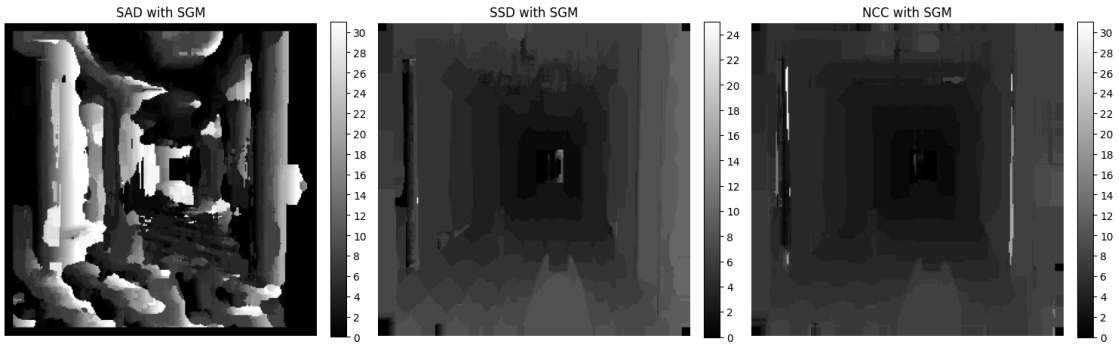
The Semi-Global Block Matching (SGBM) algorithm is introduced empirically by OpenCV (Bradski, 2000), where the difference from SGM is that the cost aggregation is not pixel-wise but block-wise. In this chapter, SGBM is investigated as well by directly using OpenCV's implementation.

4.3 Experiment Results

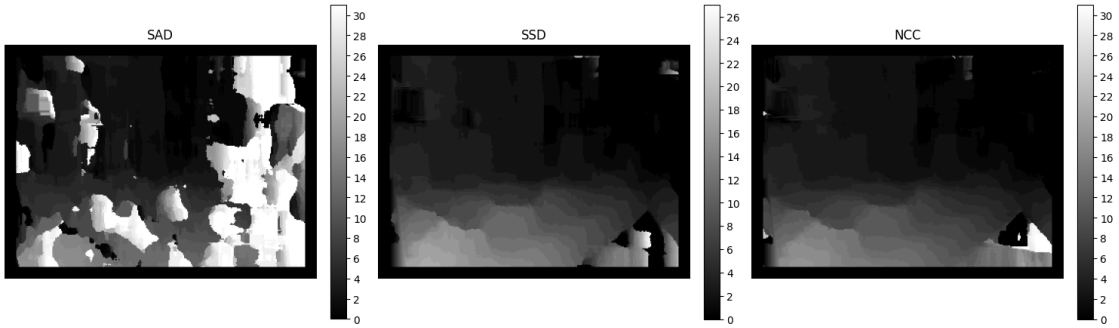
In this chapter, under the aforementioned formulation, there are two adjustable parameters which are maximum disparity $d_{\max}$ and window size of cost volumes $w_{\boldsymbol{V}}$. In the experiment below, the parameters are fine-tuned per image pair.



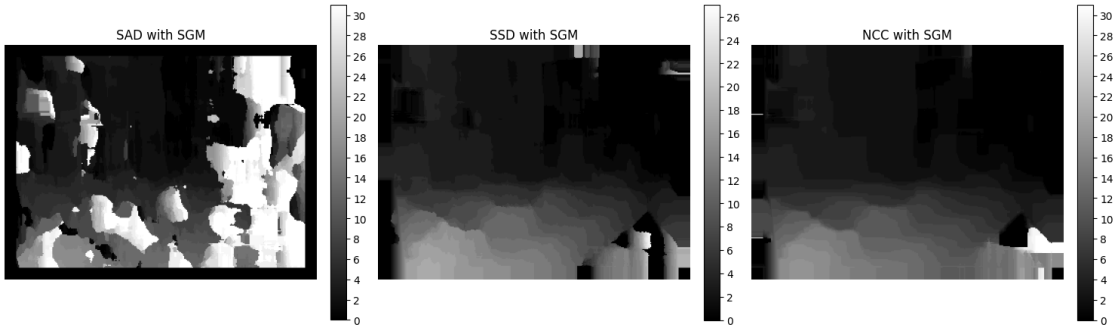
(a) Corridor image after Greedy Matching, $d_{\max} = 32$, $w_{\boldsymbol{V}} = 7$



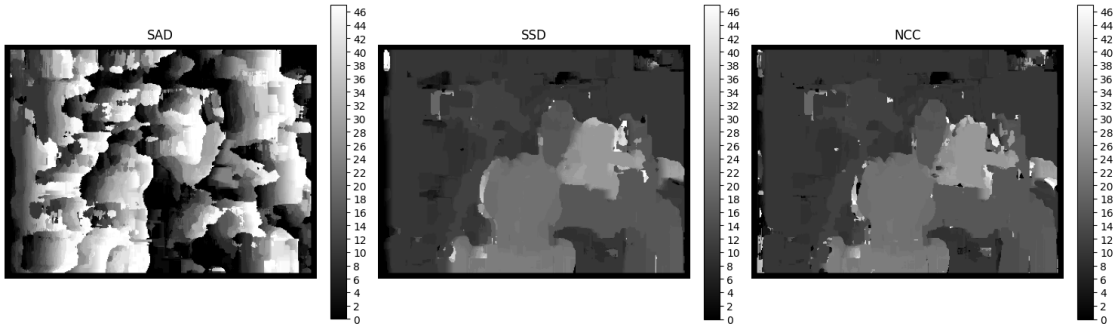
(b) Corridor image after SGM, $d_{\max} = 32$, $w_{\boldsymbol{v}} = 7$



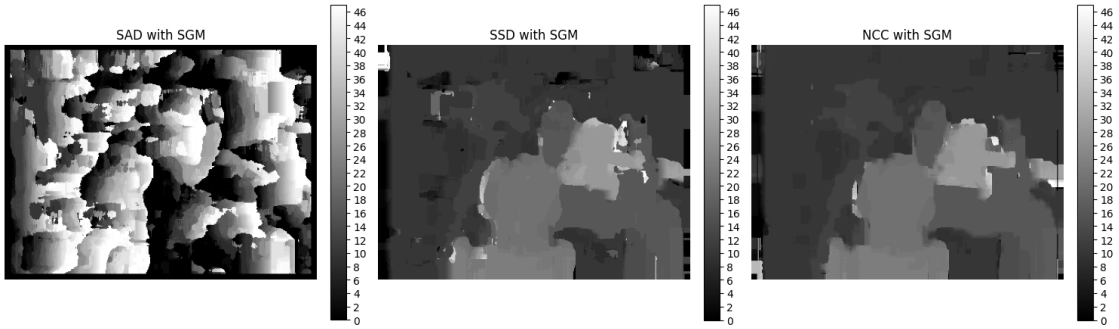
(c) Triclopsi2 image after Greedy Matching, $d_{\max} = 32$, $w_{\boldsymbol{v}} = 12$



(d) Triclopsi2 image after SGM, $d_{\max} = 32$, $w_{\boldsymbol{v}} = 12$



(e) Tsukuba image after Greedy Matching, $d_{\max} = 48$, $w_{\boldsymbol{v}} = 7$



(f) Tsukuba image after SGM, $d_{\max} = 48$, $w_{\boldsymbol{V}} = 7$

Figure 11: Experiment results of the improvements.

4.4 Discussion on the Results

From Figure 11, it can be seen that all outputs using SAD to construct cost volume are malformed. That is because the absolute difference of disparity of regional blocks is not limited and SAD would enlarge the difference and tends to reach more to zero disparity when the neighboring pixels of the pair have close luminance. The other cost volumes under SSD and NCC behave normally. Compared to results of Greedy Matching, SGM improves the smoothness of the disparity maps under NCC. For instance, in Figure 11(a) and (b), the ordinary NCC disparity of Corridor image contains noise points in the floor areas; Using SGBM, these noise points are cancelled and aggregated by surrounding disparity clues. However, there is also incompetence of SGM, which is found in the bottom-right corner of the Triclopsi2 image as shown in Figure 11(d). This is due to the same pattern of stripes appear in the same place of both input images (Figure 1). At the bottom-right corner, the same dark vertical stripes, most probably because of the stains from the camera, are causing the algorithm incorrectly considers this region has a disparity near zero and the region of sidewalk fails to match. Moreover, because there are no ground truths given, the accuracy of disparity computing is not calculable. It can also be seen that SGM algorithm would not change the disparity maps significantly from ordinary Greedy Algorithm, but instead try to improve regional flaws to make the maps look smoother.

4.5 State-of-the-Art Results

Semi-Global Block Matching (SGBM) algorithm implemented in OpenCV shows further improvements in disparity computing because several more processing algorithms are introduced to the vanilla SGM. SGBM is also more configurable with more adjustable parameters. The results of SGBM with fine-tuned parameters are shown in Figure 12 and the corresponding fine-tuned parameters are listed in Table 1. There are still imperfections, after all, for each output disparity map shows a black stripe on the left, the width of which depends on the number of disparities. One feasible way to alleviate this is to upscale the image to twice of its original size.

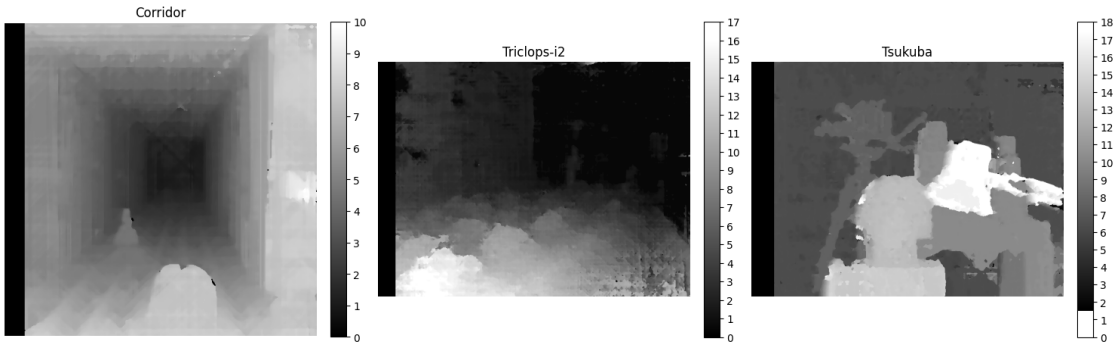


Figure 12: State-of-the-art results of OpenCV's StereoSGBM framework.

Table 1: Fine-tuned parameters of SGBM algorithm to process sample images.

Image	"Corridor"	"Triclops-i2"	"Tsukuba"
`minDisparity`	1	1	6
`numDisparities`	32	35	50
`blockSize`	6	2	6
`P1`	2675	435	450
`P2`	13700	14010	9450
`disp12MaxDiff`	10	32	60
`preFilterCap`	12	32	6
`uniquenessRatio`	1	1	1
`speckleWindowSize`	1	1	1
`speckleRange`	3	3	7

5 Conclusion

In this assignment, we implemented and analyzed several methods for computing disparity maps in stereo vision, examining both fundamental techniques and advanced algorithms. Using SSD as a baseline, we explored variations in window sizes and maximum disparities, observing that larger window sizes reduce noise at the cost of finer detail, while higher maximum disparities enhance depth detail for closer objects. Our exploration of cost volume formulations and matching algorithms, including Semi-Global Matching (SGM) and Semi-Global Block Matching (SGBM), demonstrated improvements in disparity map accuracy and smoothness.

SGM notably enhanced regional consistency, effectively smoothing noise in homogeneous regions without compromising overall depth quality. However, the presence of similar patterns in both images, such as the Triclopsi2 case, exposed limitations in matching algorithms, highlighting potential areas for further refinement. The SGBM implementation in OpenCV, with its configurable parameters, offered the most robust results, supporting state-of-the-art disparity map generation with optimized smoothness and precision. Future work could focus on refining adaptive parameters and integrating additional constraints to mitigate challenges with repetitive patterns. This study underscores the importance of parameter tuning and algorithm selection in optimising stereo vision for reliable depth reconstruction.

References

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 328–341. https://doi.org/10.1109/TPAMI.2007.1166

Xu, H., & Zhang, J. (2020). Aanet: Adaptive aggregation network for efficient stereo matching. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1956–1965. https://doi.org/10.1109/CVPR42600.2020.00203