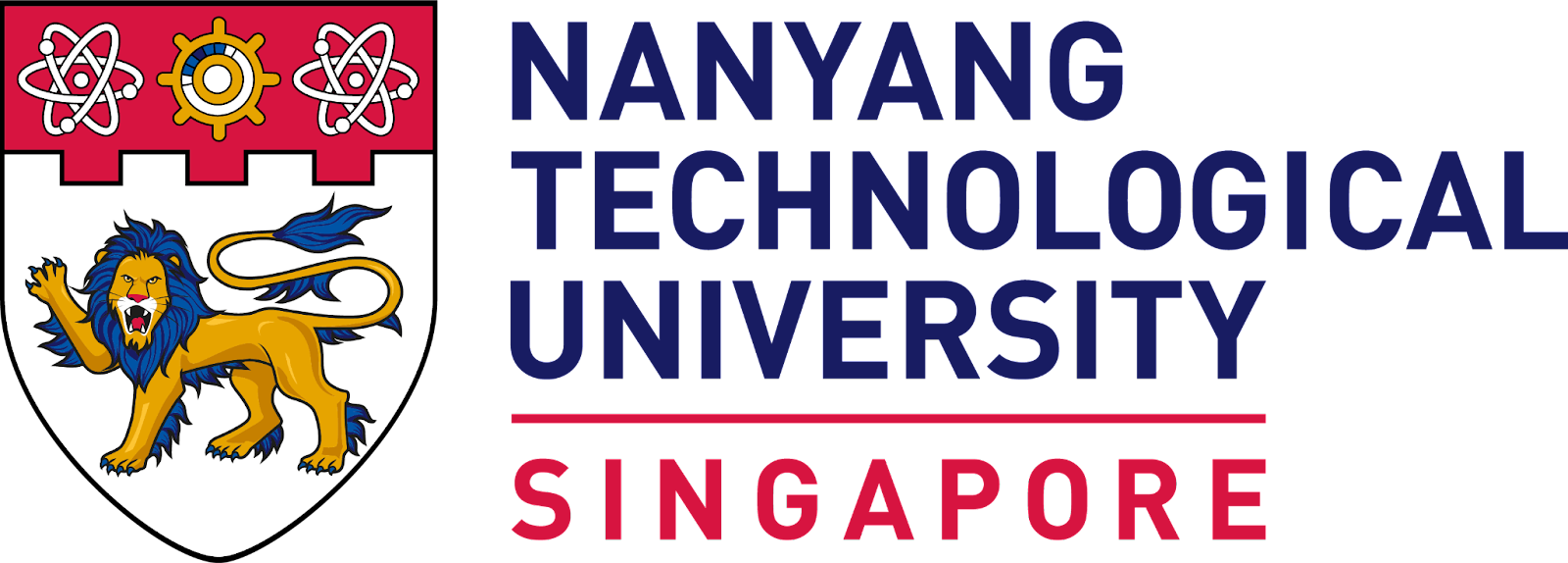
****

**AI6121 Computer Vision**

Assignment 2 Report

Members:

Chiam Jia Hao(G2104016B)

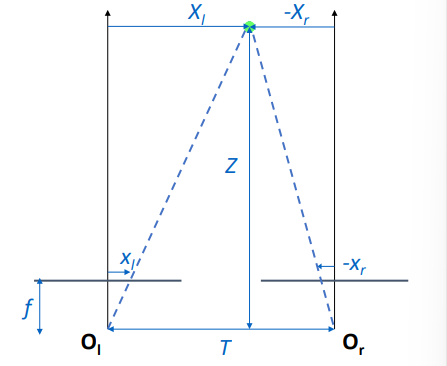
Li Xuemeng(G2202226E)

Ryan Tan Zhiyang(G2202174K)

## Part 1. Description of disparity computing



*Fig.1 example pairs*



*Fig.2 an example of camera translation from lecture slide*

What is the disparity? Assume there are at least two stereo cameras, and all of them are put in a horizontal line, and all of them take a picture at the same time. Given a system, the input of the whole system should be a pair of(or more) images taken by the cameras, pointing to one point on the left/right image, the algorithm should output the corresponding point on the other right/left image.

So, the key point of the algorithm is how to find the most matching point(pixel). Then the disparity is the difference between two points’ coordinates in the x-axis direction. For Fig.2, the disparity is . To solve the problem, the lecture slide provides two methods. The first one is Appearance-based matching and another one is Feature-based matching.

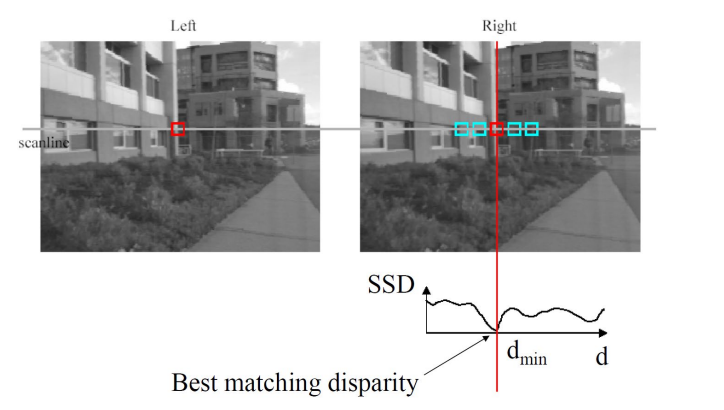
However, the provided images would not always be perfect to match, the illumination, some repeating parts on the image, covering, etc. will lead to a mismatching. So, if we got some images that look quite different, we have to first pre-process them before using them as inputs, for example, removing the noise, to make sure they have the same or similar quality. Additionally, one of the images should apply a stereo rectification since the original images are mirrored.

In order to process the computation of the disparity, the calculation of the cost is necessary. Basically, we have the cost function

Therefore, the Sum of Absolute difference can be seen as:

with represents the Left stereo image, and stands for the Right stereo image, the cost equals to the absolute difference between them. As mentioned, the disparity is the difference between two x-coordinates. The desired cost is always the minimum one. When the selected region is bounced in , and d is an integer, then it becomes the Winner Takes all(WTA) strategy. But, the output from WTA is much noisier compared to the ground truth.

According to the WTA and some other basic methods, it is hard to get a good result if we only consider a single pixel. A better solution would be thinking about all the values in a bounded window from a single pixel, this region is called the ‘Support Window’.



*Fig.3 SSD result from slide*

The smallest Sum-of-Squares Difference(SSD) is included for Appearance-based matching, the finding of patch center (x,y) for the second image(target), given the first image(reference), which also applied the smallest SSD is

As its name, it sums up all the squared differences between all two corresponding pixels. As mentioned, we now consider all the values in the widow, so the range becomes , and the amount of cost will be (dmax-dmin+1).

From the lecture, we can know that, when cameras have a small baseline, then appearance-based matching is the most accurate approach, also the processing of search is not robust.

So, when the cameras have a large baseline, feature-based matching will be more suitable. In this approach, feature points need to be selected because only sparse point correspondences are required for 3D reconstruction. Also, when having multiple answers, although unreliable, heuristics are used to select one of them. Proximity shows the most similar answer and order making sure that all matches are in the same order for both images etc. Overall, feature-based matching also can match feature points with the same properties that all images have. Additionally, Scale Invariant feature transform(SIFT) is a widely-used approach. The SIFT feature is a local feature of the image, which remains invariant to rotation, scaling, and brightness changes, and also maintains a certain degree of stability to viewing angle changes, affine transformations, and noise.

These two matching methods are two of the solutions to the pixel-matching problem. Then when multiple image points/pixels are matched, the disparity computation could be solved.

## Part 2. Implementation

Using the Appearance-based matching algorithm mentioned in part 1, we are able to get the disparity maps for the “corridor” and the “triclopsi2” images.

In the algorithm, we need to define 2 parameters - “window\_size” and “search\_window”. “Window\_size” is the size of the “Support Window” (in pixels) mentioned in part 1. The larger the “window\_size”, the smoother the disparity map will appear but it will lose details around the edges. “Search\_window” is the range that the algorithm will scan along the horizontal x-axis. For instance, when “search\_window” = 15, the algorithm will try to find the corresponding image patch 15 pixels to the left and right of the current “Support Window” position. If the “search\_window” is too small, it might not be able to find the corresponding image patch. On the other hand, when the “search\_window” is too large, it might match itself to the wrong image patch because the matching cost is the lowest by chance. In addition, a large “search\_window” will also slow down the computation.

Lastly, once we have computed the disparity maps, we will need to perform histogram equalisation for the difference in disparity values to have high contrast.

After experimenting with several values, we arrived at the “window\_size” = 9 and “search\_window” = 15 for the “corridor” images, which yielded reasonably good results.



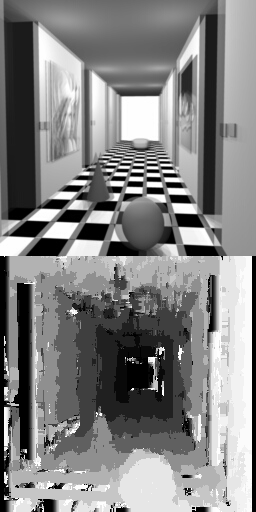
*Fig.4 Disparity map for “corridorl.jpg” and “corridorr.jpg”*

For the “triclopsi2” images, we arrived at “window\_size” = 15 and “search\_window” = 15.



*Fig.5 Disparity map for “triclopsi2l.jpg” and “triclopsi2r.jpg”*

## Part 3. Disparity Map



*Fig.6 Overlapping the left and right images and their respective disparity maps*

Disparity map refers to a 2-D image that stores the disparity value of all pixels with stereo correction in a single viewpoint. We can find the corresponding disparity value by the coordinate of a pixel in the 2D image since they have the same ordering. Also, with a disparity map, we can do retrieval between all neighbours and apply filters easier.

These are some of the observations we made from the obtained disparity maps:

1. Objects further away will have smaller disparity values between the left and right images, which are reflected as the darker regions (orange boxes) in the disparity maps. The area at the end of the corridor and the further building in the “corridor” and “triclopsi2” images respectively have shown smaller disparity values.
2. On the other hand, objects closer to the foreground have larger disparity values, which are reflected in the lighter regions (red boxes). From the overlapped “corridor” images, we can see that the rightmost wall and the sphere have shown a large displacement between the left and right images. This was correctly reflected in the bright regions observed from its disparity map.
3. Objects with homogeneous surfaces tend to generate noisy and random disparity values as we can see from the tiles in the foreground and the pathway highlighted in green boxes from the “corridor” and “triclopsi2” disparity maps respectively. This is because there is a higher chance that the matching cost happens to be lower at an erroneous part of the image by chance.
4. However, the algorithm also generates a lot of noise as can be seen from the random brighter regions on the left building in the “triclopsi2” disparity map when they are supposed to have similar disparity values as nearby regions.
5. The algorithm also could not generate smooth regions without losing too much detail at the edges, which is observed from the roof of the “corridor” disparity map.
6. One of the improvements could be: the noise may be removed by applying a median or some other filters, further outputting the new disparity map with reliability. Then we can assume the disparity values under 5 or 10 as noise, after the movement., we would get a smoother map.

Therefore, the possible improvements we can work on would be improving the smoothness of the disparity maps without losing details at the edges and reducing noise from the homogeneous regions.

## Part 4. Bonus Task (Estimating depth with SGM)

**4.1 Factors that affect disparity map computation**

Disparity is the motion of objects between two stereo images. With the use of the two stereo images, the disparity map is computed by matching every pixel in the left image to the corresponding pixel in the right image. The distance between each pair of left and right pixels is then computed. With the distance values, a disparity map is then represented in the form of an intensity image. Since depth is inversely proportional to disparity, if the geometric arrangement of the cameras that took the two stereo images is known, the disparity map can then be converted into a depth map using triangulation. Factors that affect disparity map computation would be the distance measure used and also the method of determining the pair of pixels in the stereo images that correspond to the projection of the same physical point in space. Simply, addressing the so-called correspondence problem.

**4.2 Improvements to current algorithm**

**4.2.1 Using another disparity estimation method**

As mentioned above, disparity estimation is a correspondence problem. To find the corresponding pixel of the pixel in the left image from the right image, we need to obtain unique information about the physical point to ensure that we match the correct point. One way of doing so is to try to use pixel intensity, however this may not be unique enough and may form many false matches. One could also use parametric measures meaning using statistical features of the pixels such as the mean or variance intensities. One downside in using parametric measures is that it could result in the minority of neighbourhood pixels having a larger impact on the value.

Therefore, to improve on our algorithm, we decided to use a non-parametric method named census transform. It relies on the order of neighbourhood pixel intensities rather than the values itself. Census transform creates a string of bits from the relative intensities of the neighbouring pixels. Take for example a centre pixel with intensity of 130 in a 3 by 3 block of neighbouring pixels with the values as seen in the matrix below.

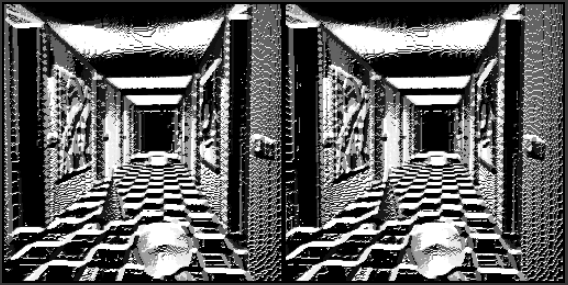


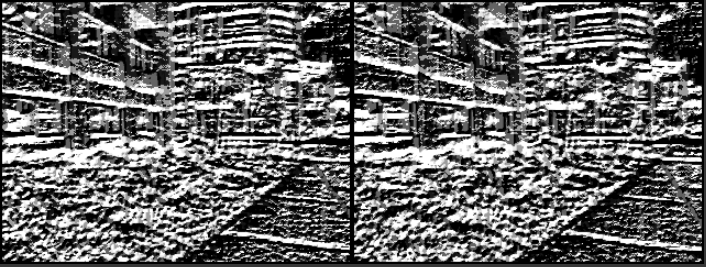


The corresponding census transformation would be as seen in the matrix below. This is due to the characteristic that if the neighbouring pixel intensity is less than the target pixel intensity, the value of 1 will be added to the bit string. On the other hand, if the neighbouring pixel intensity is more than the target pixel intensity, the value of 0 will be added to the bit string. The highlighted values are transformed into the values seen in the matrix below.



When flattened out into a bit string, it would then take the form of 00111000. The bit strings are then stored in the form of a numpy integer. Since the census bit strings are in the format of integers, they can be converted into uint8 for some visualisation tasks. As seen from the figure below, the census images are able to capture the edges of the two images.





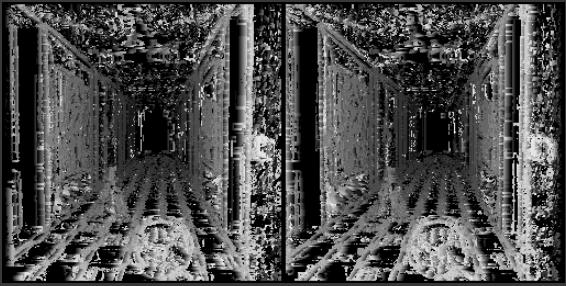
*Fig.7 Census value visualisation*

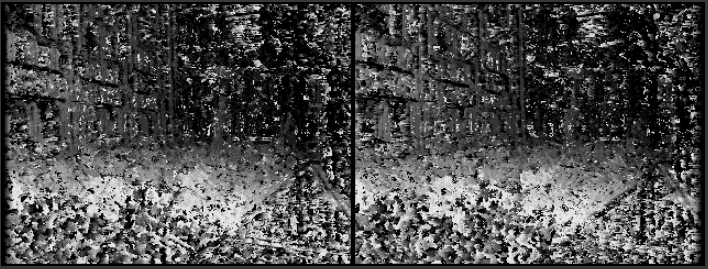
**4.2.2 Hamming Distance**

To measure the similarity measure of the pixels, hamming distance was used. It sums up the number of different bits in both pixel pit strings. If the hamming distance in lower, it corresponds to high similarity. For example taking two bit strings, 0011 and 1111, their hamming distance will be 2 since they differ in bit values in the first and second bit position.

**4.2.3 Cost computation and Cost Aggregation**

Using the census values together with the hamming distance, we are able to find the match with the highest probability in the corresponding image. This is done by measuring the matching costs of the pixel with its corresponding pixel in the other image for each disparity integer until the max\_disparity value is reached. The matching cost is calculated using the hammond distance. The costs for one disparity level is calculated for all pixels in the image in one pass. The costs for all pixels calculated at all disparity levels are then stored into an array called the cost volume. The disparity map can then be formulated from the cost volume by finding the disparity with the minimum cost for each pixel. However, this produces a very noisy disparity map as seen in the figures below.

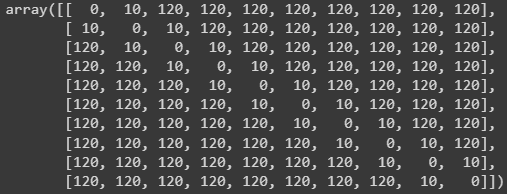




*Fig.8 Noisy disparity map*

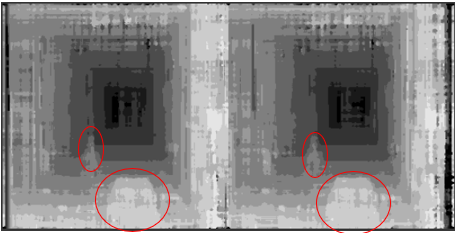
The disparity map is so noisy due to incorrect matches. To rectify this problem, cost aggregation is used. The Semi-Global Matching’s cost aggregation is applied to the cost volume computed earlier. Cost aggregation solves this problem by penalising changes in the neighbouring disparities. This will help smoothen the cost volume which in turn leads to a smoother disparity map.

Cost aggregation compares the costs of the neighbouring pixels in sequence along a single path. The current pixel costs is then updated based on the penalty term. This is iterated for all pixels along the single path. After this process, a cost volume with updated costs will be obtained and used for creating a new disparity map. However, this would result in a streaking effect in the direct of the path. To combat this problem, the process will be repeated for multiple paths and the cost volumes for each path are then concatenated into a single cost aggregation volume. The cost for all paths will be summed into a single cost volume and used for disparity map computation by finding the disparity index for each pixel containing the minimum cost. The penalty matrix can be seen below.



*Fig.9 Penalty matrix*

The columns in the matrix represent the current pixel disparity while the rows represent the previous pixel disparity. The diagonals are all zeros as this is where the current disparity and the previous disparity are the same. Thus, no penalty will be applied. Values that are 10 are due to the current and previous disparity differ by 1, everywhere else is 120 as the pixels differ by more than 1. The disparity maps are then displayed with a median filter to reduce the streaking as shown in the figures below.



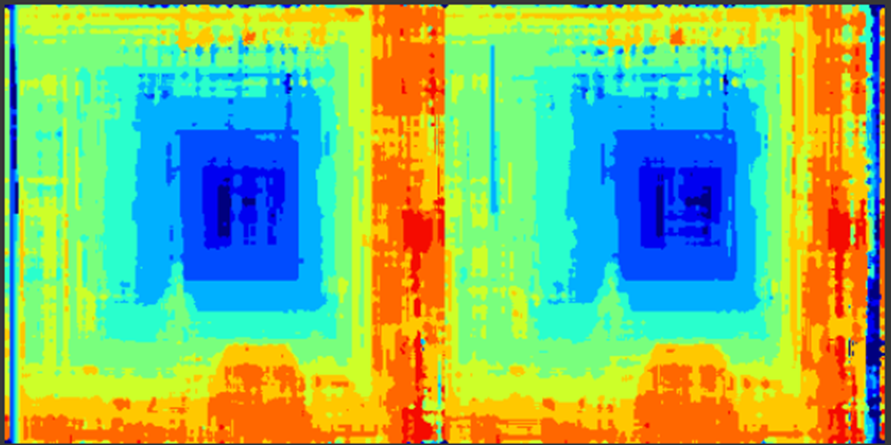


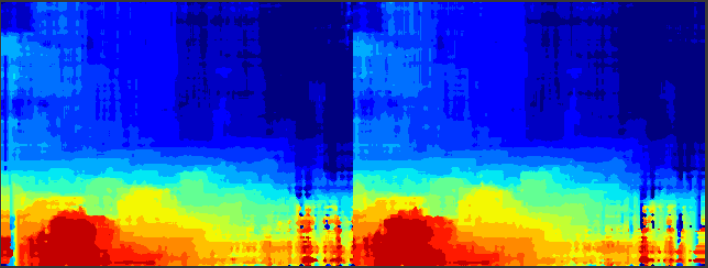
*Fig.10 Final disparity maps*

By comparing with the disparity maps produced by appearance based, it can be seen that the ones produced by using the SGM algorithm are less noisy and streaky. However, the features in the images such as the sphere and the cone in the corridor are harder to detect as seen circled in red in the figures above as an example.

**4.2.4 Adding colour to disparity maps**

Another step in improving our algorithm was to add colour to our disparity maps. This helps with better visualisation of the depth of the maps. This can be seen in the figures below.





*Fig.11 Final disparity maps with colour*

With the use of colours, it is easier to see the objects that are nearer (red) and objects that are further away (blue).