

**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**  

---

**SINGAPORE**

**School of Computer Science and Engineering**

**Nanyang Technological University**

**AY2024/25**

**AI6121 Computer Vision**

**Assignment 02**

**Stereo Reconstruction**

Group Members:

Kee Ming Yuan (G2304842E)

Yip Chen Fei (G2304486K)

Timothy Yeo Xiao Jing (G2406412E)

Lehan Gong (G2205030D)

Wong Seik Man (G2303494L)

# Content

Content.....	2
Introduction.....	3
Evaluation Metrics .....	4
Visual Analysis by team members .....	4
Bad Matched Pixels (BMP).....	4
Mean Squared Error (MSE) .....	5
Mean Absolute Error (MAE) .....	5
Mean Relative Error (MRE).....	6
Subpixel Zero Error (SZE).....	6
Summary of Evaluation Metrics .....	7
Factors affecting the Disparity Map generated .....	7
Input Image Pairs Quality .....	7
Preprocessing .....	8
Stereo Mapping Algorithm and Tuning of Algorithm parameters.....	10
Baseline Stereo Mapping method: Block Matching .....	11
Methodology .....	11
Result and Discussion .....	14
Semi-Global Matching (SGM) .....	17
Algorithm steps for SGM.....	17
Results and Discussion.....	18
Graph Cut Stereo Mapping .....	21
Explanation for energy function.....	21
Result and discussion .....	23
Best Stereo Mapping model.....	26
Best metrics of each Stereo Mapping model on Corridor image .....	26
Triclops2 image analysis.....	29
Best Stereo Mapping model by Visual Analysis voting .....	29
Conclusion .....	30
References.....	31

## Introduction

Disparity computing is a core concept in stereo vision, which involves estimating the depth of objects in a scene by comparing two or more images taken from slightly different viewpoints. The basic principle relies on the fact that objects appear at different positions in images captured by cameras separated by a baseline distance (like human eyes). The difference in the object's position between these images is referred to as **disparity**.

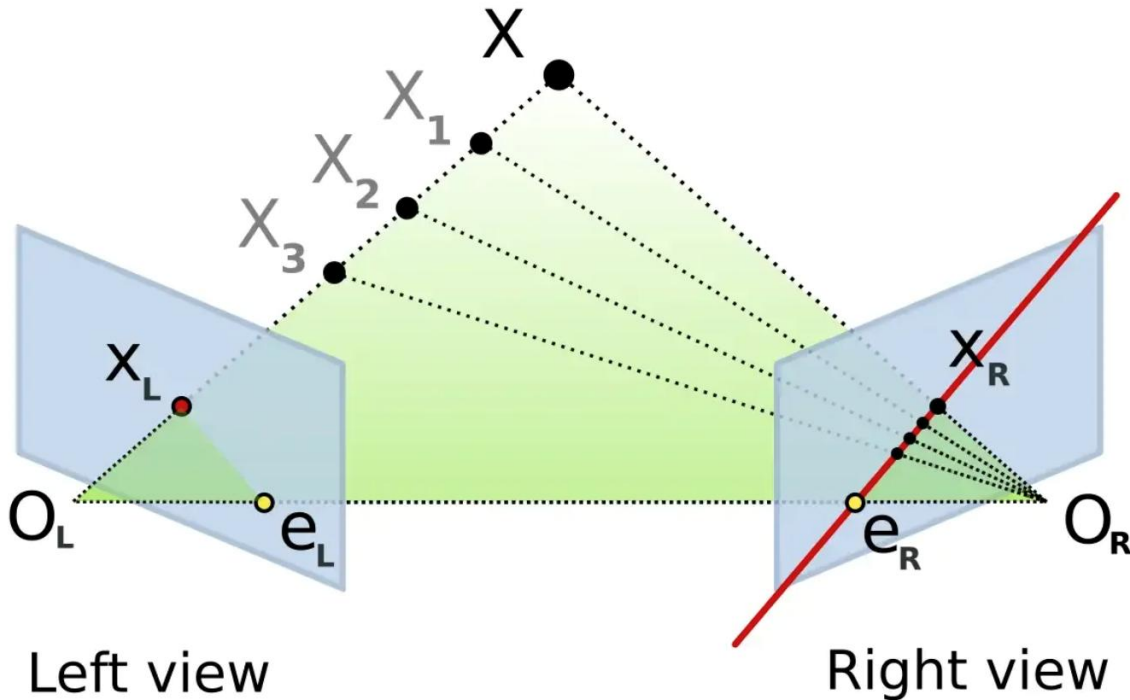


Fig 1: Illustration of a stereo camera model. [1]

Given a pair of rectified images of the same scene captured from two different viewpoints on the same baseline, the position of the same object at the left image ( $X_L$ ) will be different with the position at the right image ( $X_R$ ), the distance between  $X_L$  and  $X_R$  are inversely proportional to the distance between the object and the baseline ( $Z$ ).

$$Z \propto \frac{1}{|X_L - X_R|}$$

Which means, we can compare the distance of the same object in the rectified images, to estimate the how far is the object from our viewpoints, the larger the distance, the closer the object.

Based on this idea, we can compute the disparity map of two rectified images by applying a search algorithm to find corresponding pixels in both images. By measuring the distance between matching pixels, we calculate the disparity for each point, which is then used to generate the disparity map.

In practice, accurately matching corresponding pixels in rectified images is challenging. As a result, the most common approach is to compare blocks of pixels (Block matching) rather than individual pixels to determine the disparity more reliably.

## Evaluation Metrics

In this report, several Stereo Mapping algorithms will be applied on the two image pairs given and the disparity maps generated will be compared with each other using the evaluation metrics stated in this section to conclude the best Stereo Mapping algorithm.

### Visual Analysis by team members

A robust approach for evaluating disparity maps involves visual assessment by team members using results generated from different Stereo Mapping algorithms. In this process, the shape of each object in the image pair should be represented correctly in the disparity map. Objects that are farther from the camera are expected to appear in darker shades on the disparity map, while closer objects should be represented in lighter shades. This visual consistency helps ensure that the disparity map accurately reflects the relative depth of objects in the scene.

While visual assessment is undoubtedly straightforward, quantitative evaluation is necessary for some important reasons:

1. Subtle differences in the shape of the object in the image pair and disparity map cannot be detected easily visually, especially when tuning of hyperparameters are conducted to derive the best Stereo Mapping model where tiny changes such as a 0.1 to the hyperparameter might be hard for humans to tell.
2. Different teammates might have different opinions as to which disparity maps generated from which Stereo Mapping algorithm is more visually appealing
3. Inconsistency in the screen display and brightness of each team member's screen might affect the member's judgement of the disparity maps

Therefore, we will employ additional evaluation metrics commonly used in image enhancement to objectively measure the quality of our results.

### Bad Matched Pixels (BMP)

BMP is a measure of the percentage of pixels in a disparity map where the estimated disparity values are different than that of the ground truth disparity map by more than a user-defined threshold. In our report, the threshold is set as 1, which means the estimated disparity value need to be  $\pm 1$  for the pixel not to be considered as 'bad' pixel.

$$BMP (\%) = \frac{\text{Number of bad pixels in the disparity value range}}{\text{Total number of pixels}} \times 100$$

The lower the BMP percentage, the better as it means more pixels in the disparity map are closer to that of the ground truth disparity map. The BMP percentage is dependent

on the threshold value used. This meant that errors smaller than the threshold value is not captured.

### Codes for BMP

```
disparity = disparity.astype(np.float32)/16  
  
imgGroundTruth = imgGroundTruth.astype(np.float32)/16  
  
# BMP (Bad Matched Pixels)  
error_threshold = 1.0  
  
bmp = np.sum(np.abs(disparity[valid_mask] - imgGroundTruth[valid_mask]) >  
error_threshold) / np.sum(valid_mask)  
  
print(f"BMP (Bad Matched Pixels): {bmp * 100:.2f}%")
```

### Mean Squared Error (MSE)

MSE is the average squared of the difference in predicted and ground truth disparity value for each of the pixel in the disparity map given by the below formula:

$$MSE = \frac{1}{n} \sum_{i=1}^n (d_i - \hat{d}_i)^2$$

Where  $d_i$  is the ground truth disparity value for the  $i$ th-pixel,  $\hat{d}_i$  is the predicted disparity value of the same pixel and  $n$  is the total number of pixels in the disparity map.

The smaller the MSE, the better the Stereo Mapping performance. MSE is highly susceptible to outliers in the disparity prediction value as difference in outliers and ground truth values will be squared, making the MSE bigger.

### Codes for MSE

```
# MSE (Mean Squared Error)  
  
mse = np.mean((disparity[valid_mask] - imgGroundTruth[valid_mask]) ** 2)
```

### Mean Absolute Error (MAE)

MAE gives as more direct comparison of the predicted error. It is the average absolute error between the predicted and the ground truth values.

$$MAE = \frac{1}{n} \sum_{i=1}^n |d_i - \hat{d}_i|$$

Where all terms are the same as what was defined in MSE.

Large errors are not magnified in MAE unlike MSE and are less sensitive to outliers. The smaller the MAE, the better.

## Codes for MAE

```
# MAE (Mean Absolute Error)
```

```
mae = np.mean(np.abs(disparity[valid_mask] - imgGroundTruth[valid_mask]))
```

## Mean Relative Error (MRE)

MRE is the average relative difference the predicted disparity value and the ground truth value as per the equation below:

$$MRE = \frac{1}{n} \sum_{i=1}^n \frac{|d_i - \hat{d}_i|}{|d_i| + e}$$

Where all terms are the same as what was defined in MSE, and e is a small value to prevent numeric instability due to division by 0.

As per the equation above, MRE involves the normalization of the absolute difference in the estimated and ground truth values with the absolute value of the ground truth values. This is useful in the event where the ground truth values vary greatly. The smaller the MRE the better. A drawback to MRE is that the MRE value would be inflated if the ground truth values are very small.

## Codes for MRE

```
# MRE (Mean Relative Error)
```

```
mre = np.mean(np.abs(disparity[valid_mask] - imgGroundTruth[valid_mask]) /  
(np.abs(imgGroundTruth[valid_mask]) + 1e-5))
```

## Subpixel Zero Error (SZE)

SZE measures the number of pixels with an error of less than 0.5 compared to the number of valid pixels in the ground truth disparity map. The valid pixels in ground truth disparity map are defined as pixels with values greater than 0. This is because disparity map of value 0 often meant there are missing values. That said, we have checked that all the values in the ground truth disparity map are more than 0.

$$SZE (\%) = \frac{\text{Number of pixels with error} < 0.5}{\text{Total number of predicted pixel value with ground truth value}} \times 100$$

The bigger the SZE, the better. SZE is highly useful for tasks which requires high precision as the threshold for SZE is very small (less than 0.5 error). These tasks include autonomous driving or vision for robots.

## Codes for SZE

```
# SZE (Subpixel Zero Error)
```

```
size = np.sum(np.abs(disparity[valid_mask] - imgGroundTruth[valid_mask]) < 0.5) /  
np.sum(valid_mask)
```

```
print(f"SZE (Subpixel Zero Error): {size * 100:.2f}%")
```

## Summary of Evaluation Metrics

Table 1: Summary of ground truth information requirement amongst different metrics

	Visual Analysis	BMP	MSE	MAE	MSE	SZE
Ground truth needed?	No	Yes	Yes	Yes	Yes	Yes

Of all the evaluation metrics discussed previously in this section, all of them require ground truth data except for Visual Analysis.

## Factors affecting the Disparity Map generated

There are several key factors affecting the quality of disparity map generated.

### Input Image Pairs Quality

Calibrating the cameras used to capture the left and right images are important. Intrinsic parameters such as the focal length and optical centre as well as extrinsic parameters such as the position and orientation of the cameras need to be estimated for calibration to happen.

When taking the left and right images, it is important for the camera to lie on the same epipolar line such that the corresponding points of both images lie on the same horizontal line. Else, the images would need to be rectified so that the Stereo Mapping algorithm only need to search along the x axis for correspondences.

Having good lighting across the whole image can reduce noise, shadows and overexposure. This allows the matching algorithm to make accurate matches. It is best that the images taken contains a lot of textures like edges as it is easier for the matching algorithm to estimate disparity. The below is a sample good image:



Fig 2: Tsukuba image which is easy for the matching algorithm to produce a good disparity map [2].

For this report, two pairs of images were provided, and we will be using the images provided by this course to do analysis.

## Preprocessing

Before the left and right images are fed to the Stereo Mapping algorithm, proper image preprocessing is important. Contrast enhancement algorithms like histogram equalization can be used to increase the contrast of poorly contrasted images. Gaussian smoothing can be applied to reduce noise in the images. However, it is important to note this will reduce small details of images.



Fig 3: Left stereo images of the "Corridor" (left) and "Triclopsi2" (right) pairs provided for this report.

The diagram above shows the left stereo image of both image pairs provided to the team. The one on the left is a picture of a corridor while the one on the right is a picture of an urban environment. The team has found a ground truth disparity map online for the left picture shown in the above diagram as shown below [3]:

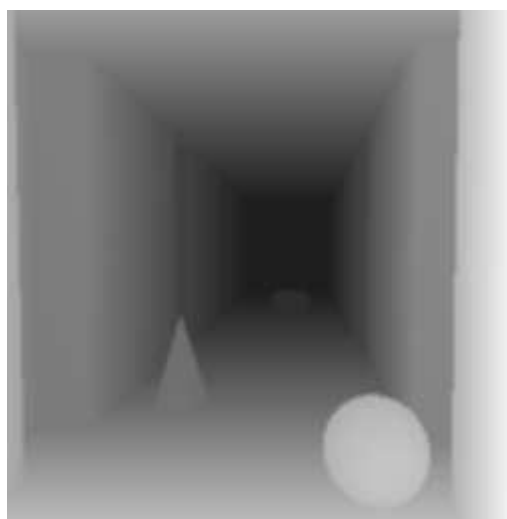


Fig 4: Ground truth disparity map for the "Corridor" stereo image pair.

With the ground truth image found from online, the team will be conducting pre-processing steps on the corridor image pair with respect to the ground truth.



## Codes for Gaussian smoothing (GS)

```
# Function to apply Gaussian smoothing
def apply_gaussian_smoothing(image, kernel_size=(5, 5), sigma=1):
    return cv.GaussianBlur(image, kernel_size, sigma)

imgL_smoothed = apply_gaussian_smoothing(imgL, kernel_size=(5, 5), sigma=1)
imgR_smoothed = apply_gaussian_smoothing(imgR, kernel_size=(5, 5), sigma=1)
```

## Codes for Histogram Equalization (HE)

```
# Function to apply Histogram Equalization
def apply_histogram_equalization(image):
    return cv.equalizeHist(image)

imgL_smoothed_he = apply_histogram_equalization(imgL_smoothed)
imgR_smoothed_he = apply_histogram_equalization(imgR_smoothed)
```

## Results from applying pre-processing steps on the baseline algorithm

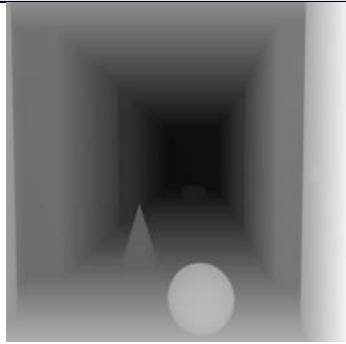
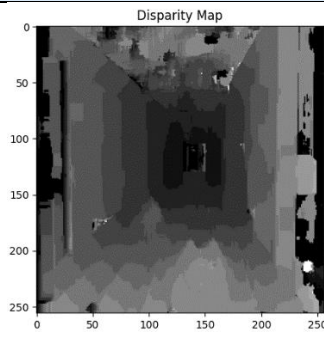
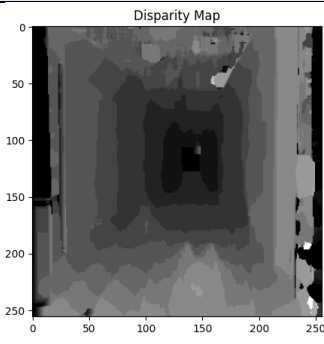
Table 2: Comparing Disparity Map results when different types of pre-processing methods are applied.

Index	Pre-processing methods	BMP	MSE	MRE	SZE	MAE
1	No pre-processing	57.88%	14.8727	0.2811	19.94%	2.2831
2	Histogram Equalization only	58.78%	15.5862	0.2941	19.18%	2.3693
3	Gaussian Smoothing only	59.25%	13.4046	0.2700	18.79%	2.2022
4	Both HE and GS	60.48%	16.8044	0.3027	18.52%	2.4672

From the above table, we see that applying Gaussian Smoothing only (pre-processing method 3) can bring the best results with the lowest MSE, MRE and MAE. This is intuitive as GS uses a Gaussian kernel to reduce noise and outliers from the image by blurring them away, making the image more homogeneous. With less noise, correspondence can be found more easily and more accurately. This allows depth estimation to be done more accurately, reducing the MSE, MRE and MAE. Nevertheless, introducing blur can cause edge details and small details to appear less prominent, affecting BMP and SZE which focus on pixel accuracy. This could be why the baseline without any pre-processing gives the lowest BMP and SZE values.

Since conducting gaussian smoothing only performs better at MSE, MRE and MAE but do worse at BMP and SZE, the team will be conducting visual analysis of the disparity maps generated with the ground truth.

Table 3: Baseline Disparity Map comparison with and without Gaussian Smoothing with Ground Truth

Ground Truth	Baseline	Baseline with Gaussian Smoothing
		

From the above table, the team members observed that the disparity map after GS looks more similar to the ground truth disparity map than that of the baseline as it has less noise. Hence, pre-processing step of GS is chosen over not conducting any pre-processing step for the baseline algorithm.

It is important to note that different pre-processing steps might bring the best results for each Stereo Mapping algorithm. Hence, pre-processing steps of HE and GS will be conducted in each of the Stereo Mapping to evaluate which combination of pre-processing steps gives the best results on the corridor image pair.

As no ground truth disparity map image is available for the urban environment image pair, for each of the Stereo Mapping algorithm, we will apply the chosen pre-processing steps combination for the corridor image on the urban environment image pair. This would mean that referring to Table 1 in the previous section, only Visual Analysis will be used to evaluate the urban image pair since it does not have a ground truth image.

## Stereo Mapping Algorithm and Tuning of Algorithm parameters

For the Stereo Mapping algorithm, the matching function used affects the accuracy of the depth prediction. The matching function serves to identify the correspondence between the left and right image in the image pair. It aims to find the pixel which is best matched to the other pixel from the other image. With good correspondence, the disparity values can then be calculated accurately. The matching function used as the baseline in this report will be explained later.

Moreover, important parameters such as block size and search range need to be tuned carefully for the algorithm to perform well on that image pair. The tuning of parameters will be conducted for the baseline algorithm in the subsequent sections.

# Baseline Stereo Mapping method: Block Matching

## Methodology

For the baseline Stereo Mapping method in this assignment, we adopted Block Matching (BM) to generate a disparity map from an image with dimensions (N x M), which can be broken down into the following steps.

We began by pre-processing the stereo image pair with Gaussian Smoothing, as discussed in the previous section. Following this, we applied Sobel filtering to further enhance the images. Sobel filtering is a popular edge detection technique that calculates the gradient of image intensity at each pixel, helping to highlight edges and transitions. The *SobelFilter()* function applies edge detection using the Sobel operator. It calculates horizontal (*cx*) and vertical (*cy*) gradients for each pixel by convolving the image with predefined Sobel matrices. The magnitude of the gradients is computed as the square root of their sum of squares, representing edge intensity. The output image stores these values, capped at 255, highlighting the edges in the image.

## Codes for Sobel Filtering

```
def SobelFilter(image):
    height, width = image.shape
    out_image = np.zeros((height, width))

    table_x = np.array([-1, -2, -1], [0, 0, 0], [1, 2, 1])
    table_y = np.array([-1, 0, 1], [-2, 0, 2], [-1, 0, 1])

    for y in range(2, width - 2):
        for x in range(2, height - 2):
            cx, cy = 0, 0
            for offset_y in range(0, 3):
                for offset_x in range(0, 3):
                    pix = image[x + offset_x - 1, y + offset_y - 1]
                    if offset_x != 1:
                        cx += pix * table_x[offset_x, offset_y]
                    if offset_y != 1:
                        cy += pix * table_y[offset_x, offset_y]
            out_pix = math.sqrt(cx**2 + cy**2)
            out_image[x, y] = out_pix if out_pix > 0 else 0
    np.putmask(out_image, out_image > 255, 255)
    return out_image
```

Before applying Block Matching algorithm, several parameters need to be determined: block size, search range, and similarity function. Block size defines the dimensions of the blocks used for matching ( $m \times n$ ), where smaller blocks capture more detail but are prone to noise, while larger blocks are more resistant to noise but may blur finer details. The search range represents the maximum distance for finding a matching block, which should be large enough to cover the depth variations in the scene but not excessively large, as this would increase computational time and noise. In this assignment, we applied the Sum of Squared Differences (SSD) as the similarity function to measure block similarity, with the best similarity score indicating the optimal match.

$$SSD(B^L, B^R) = \sum_{i=1}^n \sum_{j=1}^m (B_{ij}^L - B_{ij}^R)^2$$

Where  $B^L$  and  $B^R$  is the block of pixels in the left and right images respectively,  $B_{ij}^L$  and  $B_{ij}^R$  are the pixel intensity values at the  $i$ -th row and  $j$ -column of the left and right blocks respectively,  $n$  is the height of the block and  $m$  is the width of the block.

To generate a disparity map, the process begins by initializing an empty map of the same size as the image. For each pixel in the left image, a patch is selected with a defined block size, centred at coordinates  $(XL, YL)$ . A corresponding patch is placed on the right image at  $(XR, YR)$ . The similarity between the two patches is then calculated using a similarity function. The patch on the right image is moved horizontally, one step at a time, within a defined search range, and the similarity is recalculated at each step. The best match is determined based on the highest similarity score, and the disparity is calculated as the horizontal distance between the two patch centres,  $|XL - XR|$ . This disparity value is then placed in the corresponding location on the disparity map. The process is repeated for all pixels, completing the disparity map, which represents the depth information for each pixel in the image.

Finally, we normalized the disparity values to 0 and 255 to improve the visual representation of the disparity map, and this does not affect depth information.

## Codes for Block Matching

```
def ssd(left_block, right_block):
    return sum(sum(np.square(right_block - left_block)))

def disparity_map(left_img, right_img, max_disparity, block_size, matching_func=sad):
    height, width = left_img.shape
    res = np.zeros((height, width), dtype=np.float32)
    half_block = block_size // 2
    # Add padding to ensure that patches near the edges can be fully processed without
    # exceeding image boundaries
    left_img_padded = np.pad(left_img, ((half_block, half_block), (half_block, half_block)),
                             mode='edge')
```

```

right_img_padded = np.pad(right_img, ((half_block, half_block), (half_block, half_block)),
mode='edge')

max_height = height + half_block
max_width = width + half_block

for y in tqdm(list(range(half_block, max_height))):
    for x in range(half_block, max_width):
        best_similarity = float('inf')

        left_block = left_img_padded[y - half_block:y + half_block + 1, x - half_block:x + half_block
+ 1]

        # Keep looking to left until found best or reach max_disp
        for offset in range(max_disparity):
            right_x = x - offset
            if right_x < half_block:
                break # Out of bounds, stop searching

            # Get the right block
            right_block = right_img_padded[y - half_block:y + half_block + 1, right_x -
half_block:right_x + half_block + 1]

            # Calculate SAD of two blocks
            sim = matching_func(left_block, right_block)

            # Check if this is the best match, smaller the sim, the better
            if sim < best_similarity:
                best_similarity = sim
                best_offset = offset

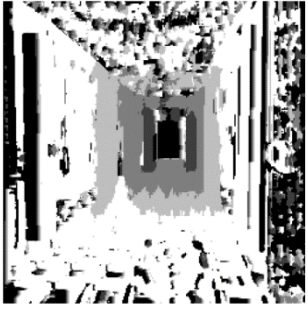
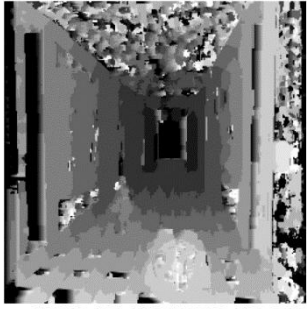
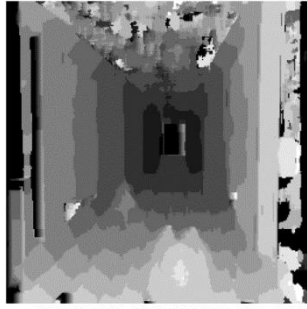
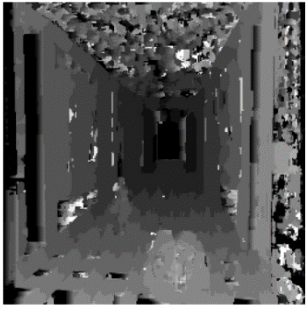
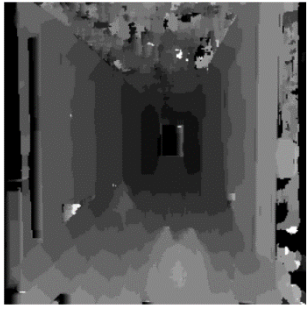
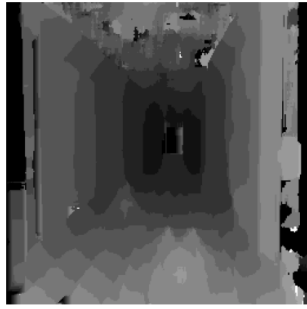
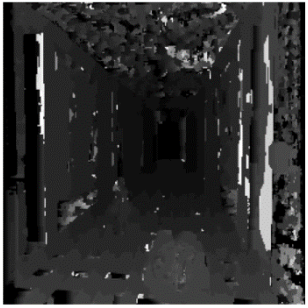
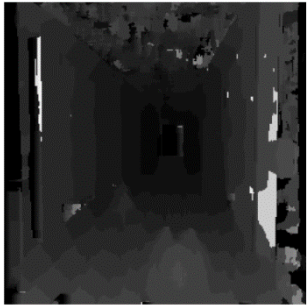
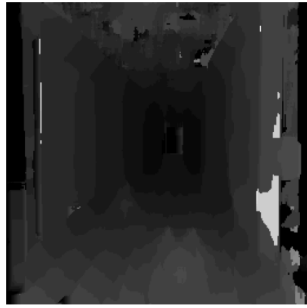
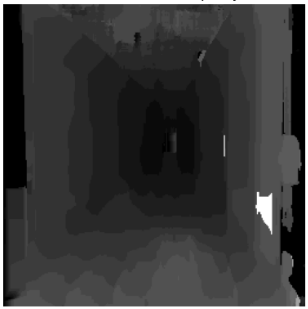
            # Assign the disparity value, (255/max_disparity) just for scaling
            res[y - half_block, x - half_block] = best_offset * (255/max_disparity)

        return res

```

## Result and Discussion

Table 4: Disparity maps of Corridor image generated using different block sizes and maximum disparities.

<p>Block Size: 5, Max Disparity: 5</p>  <p>BMP: 96.71%, MSE: 68.0060, MRE: 1.4591 SZE: 1.18%, MAE: 7.6846</p>	<p>Block Size: 5, Max Disparity: 11</p>  <p>BMP: 79.49%, MSE: 17.2392, MRE: 0.4695 SZE: 4.93%, MAE: 2.9077</p>	<p>Block Size: 10, Max Disparity: 11</p>  <p>BMP: 75.70%, MSE: 14.7797, MRE: 0.4015 SZE: 6.10%, MAE: 2.5484</p>
<p>Block Size: 5, Max Disparity: 16</p>  <p>BMP: 62.13%, MSE: 16.5013, MRE: 0.3566 SZE: 17.36%, MAE: 2.6213</p>	<p>Block Size: 10, Max Disparity: 16</p>  <p>BMP: 59.29%, MSE: 14.2303, MRE: 0.2913 SZE: 18.79%, MAE: 2.2961</p>	<p>Block Size: 15, Max Disparity: 16</p>  <p>BMP: 58.90%, MSE: 13.9392, MRE: 0.2763 SZE: 19.45%, MAE: 2.2282</p>
<p>Block Size: 5, Max Disparity: 32</p>  <p>BMP: 92.16%, MSE: 30.6747, MRE: 0.6431 SZE: 2.50%, MAE: 4.7059</p>	<p>Block Size: 10, Max Disparity: 32</p>  <p>BMP: 92.92%, MSE: 29.7831, MRE: 0.6262 SZE: 2.56%, MAE: 4.6304</p>	<p>Block Size: 15, Max Disparity: 32</p>  <p>BMP: 93.02%, MSE: 29.8123, MRE: 0.6233 SZE: 2.78%, MAE: 4.6291</p>
<p>Block Size: 20, Max Disparity: 32</p>  <p>BMP: 90.77%, MSE: 25.8385, MRE: 0.5398 SZE: 5.03%, MAE: 4.1374</p>		

From visual inspection on the above image, we can see the effect of the two most important parameters:

**Max disparity (md):** Increasing the md causes the overall image to appear darker. This is because a higher max disparity allows the algorithm to match more distant objects, resulting in lower disparity values across the image.

**Block size (bs):** A smaller block size leads to more glitchy areas in the disparity map because smaller blocks don't capture enough of the object, resulting in frequent mismatches. On the other hand, if the block size is too large, the disparity map becomes smoother, but finer details of objects may be lost.

Since we have the ground truth value disparity of the image, we can now use some metrics to compare with our images to see which parameter generate the better image.

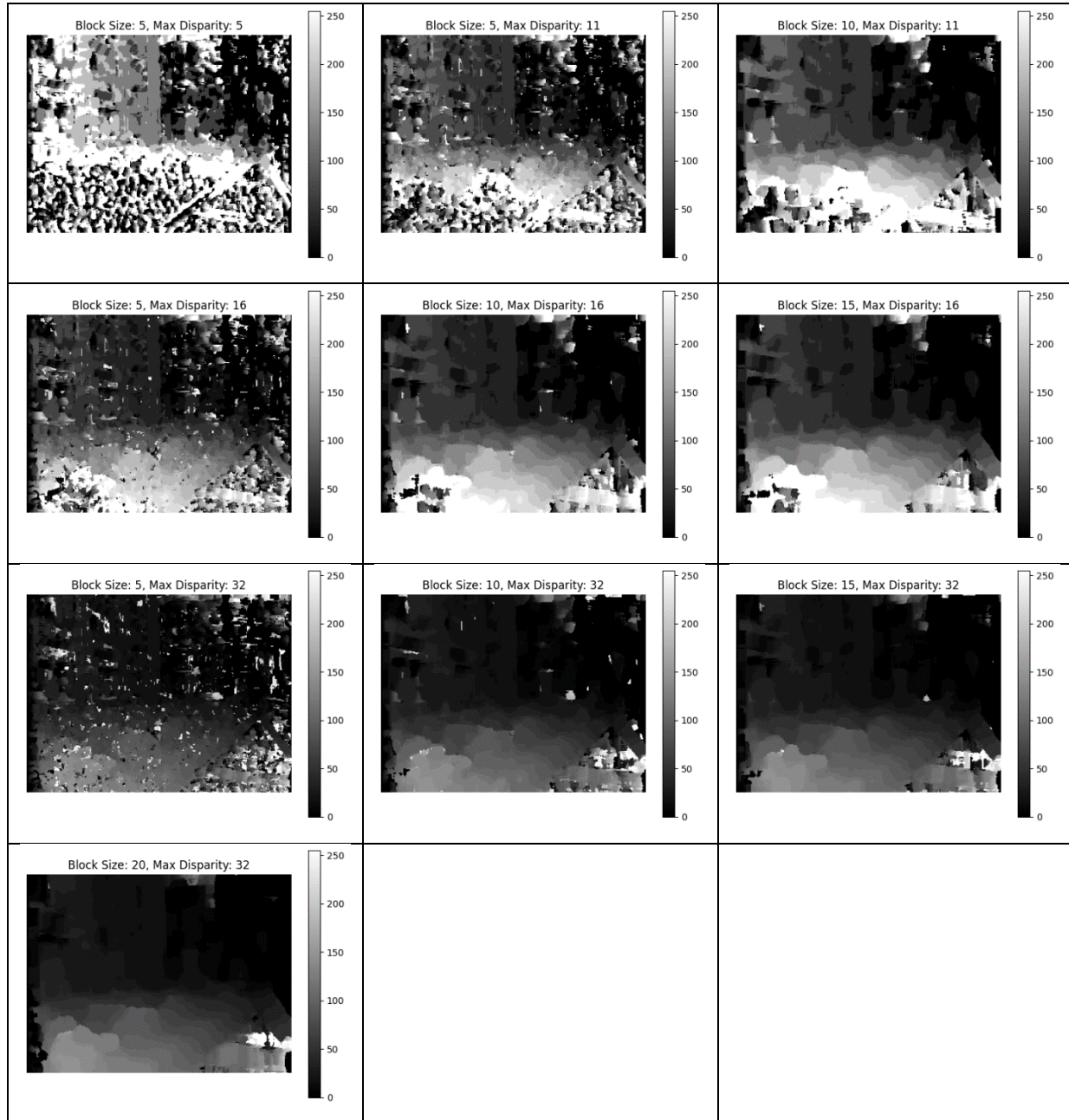
Table 5: Result comparison of disparity map for Corridor image from baseline Block Matching. The bolded values represent the best values for each metric.

(Block Size, Max Disparity)	MSE	BMP	MRE	SZE	MAE
(5, 5)	68.0060	96.71%	1.4591	1.18%	7.6846
(5, 11)	17.2392	79.49%	0.4695	4.93%	2.9077
(10, 11)	14.7797	75.70%	0.4015	6.10%	2.5484
(5, 16)	16.5013	62.13%	0.3566	17.36%	2.6213
(10, 16)	14.2303	59.29%	0.2913	18.79%	2.2961
<b>(15, 16)</b>	<b>13.9392</b>	<b>58.90%</b>	<b>0.2763</b>	<b>19.45%</b>	<b>2.2282</b>
(5, 32)	30.6747	92.16%	0.6431	2.50%	4.7059
(10, 32)	29.7831	92.92%	0.6262	2.56%	4.6304
(15, 32)	29.8123	93.02%	0.6233	2.78%	4.6291
(20, 32)	25.8385	90.77%	0.5398	5.03%	4.1373

Based on the metrics, the optimal parameters appear to be a block size of 15 and a max disparity of 16. Visually, we largely agree with this, although the image does show some issues, such as glitchy areas, object distortion, and a "grid-like" effect. Despite these imperfections, the disparity map still conveys a clear sense of depth for the objects in the image.



Table 6: Disparity maps of Triclops2 generated using different block sizes and maximum disparities, with Gaussian Smoothing pre-processing.



In the Triclops2 images, we observe similar trends with changes in block size and max disparity. The disparity map that appears to offer the best balance between detail and smoothness is likely the one with Block Size: 15, Max Disparity: 16. It captures sufficient details without being overly noisy and preserves depth structure effectively. The increasing depth of the grass, pavement, and the building on the left side is captured. Additionally, it clearly shows the boundary where the building on the left side ends. The smaller block size of 5 resulted in too much noise, while larger block sizes (15) began to lose fine details and become too smooth. Therefore, the map with Block Size: 15 and Max Disparity: 16 strikes a good balance for this specific scene.



As we see room for improvement to the disparity map generated by the baseline algorithm, in the next section of the assignment, we will discuss some advanced methods of Stereo Mapping.

## Semi-Global Matching (SGM)

Heiko Hirschmuller [4] proposed a Semi-Global Matching (SGM) method that performs pixelwise matching based on mutual information and the approximation of a global smoothness constraint. In addition, occlusion detection is incorporated into the algorithm and refine disparities estimation to sub-pixel accuracy. Hirschmuller was able to achieve results comparable to other global approaches like Graph Cuts [5] and Belief Propagation [6], while significantly outperforming them in terms of processing speed. In our study, we will be implementing the in-built function of SGM from OpenCV called StereoSGBM. However, the in-built function doesn't make use of mutual information cost function, instead it makes use of Birchfield and Tomasi [7] cost function which Hirschmuller has also experimented it in his study. The below paragraph highlights the three main steps that define the SGM algorithm.

### Algorithm steps for SGM

#### 1. Pixelwise Cost Calculation

In pixelwise cost calculation, the goal is to determine how similar or different the intensities of two corresponding pixels are by computing the matching cost. A lower cost indicates more similarity, while a higher cost indicates greater difference. The matching cost is computed by comparing the intensity  $I_{bp}$  of a pixel  $p$  in the base image with the intensity  $I_{mq}$  of a corresponding pixel  $q = e_{bm}(p, d)$  in the match image. The function  $e_{bm}(p, d)$  represents the epipolar line in the match image for the base image pixel where  $d$  is the disparity parameter. Below are the two pixelwise cost calculation:

##### *Birchfield and Tomasi Cost*

The BT cost  $C_{BT}(p, d)$  can be calculated as the absolute minimum difference of intensity between pixel  $p$  and pixel  $q = e_{bm}(p, d)$  in the range of half a pixel in both direction along the epipolar line.

##### *Mutual Information Cost*

The mutual information cost leverages on the concept of mutual information. Mutual information between two images measures the amount of information one image shares with the other image. In the context of Stereo Mapping, mutual information measures the statistical dependency between the two images based on their pixel intensity distributions. Mutual Information has been implemented in existing Stereo Mapping studies and has shown that it is more robust against complex intensity transformations and reflections. MI cost is defined as the equation below:

$$C_{MI}(p, d) = -mi_{I_b, f_D(I_m)}(I_{bp}, I_{mq}) \text{ with } q = e_{bm}(p, d)$$

## 2. Aggregation of Costs

After the pixelwise cost calculation has been completed, the next step is to aggregate the cost. This step addresses the ambiguity of pixelwise matching by enforcing smoothness constraints. During the formulation of the smoothness constraint, multiple issues were met however Hirschmuller proposed a new idea to address these issues. The idea is to aggregate matching costs along 1D paths from all directions to approximate the global 2D smoothness constraint. Figure 5 shows that the aggregated cost  $S(p, d)$  for a pixel  $p$  and disparity  $d$  is calculated by summing the costs along all minimum-cost paths that end in pixel  $p$  and disparity  $d$ . This idea helps to mitigate the streaking artifacts often associated with 1D optimization methods while avoiding the complexity of global 2D optimization.

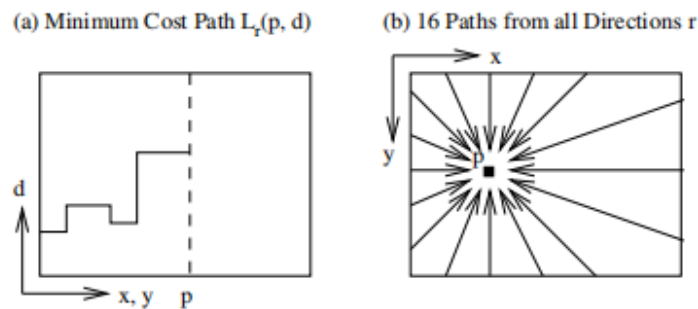


Fig 5: (a) The minimum cost path for a specific disparity and (b) 16 possible paths from all directions converging at point  $p$  in a disparity map calculation.

## 3. Disparity Computation

After aggregating the costs, the final disparity image is calculated through two steps.

### *Disparity selection*

The disparity images  $D_b$  and  $D_m$  can be calculated by selecting the disparity with the minimum aggregated cost  $S(p, d)$  for each pixel  $p$  and  $q$  respectively. For sub-pixel estimation, a quadratic curve is plotted around the neighboring costs such as the next higher or lower disparity and the position of the minimum is calculated.

### *Consistency Check*

With the disparity images  $D_b$  and  $D_m$  generated, a consistency check is performed between them to detect occlusions and false matches. Disparities that don't match between these maps are marked as invalid. This ensures that there is only a one-to-one mapping between corresponding pixels.

## Results and Discussion

In our analysis of the StereoSGBM implementation, we use a fixed value of 16 for the number of disparities parameter, which is the minimum allowable value. Since the disparity value must be divisible by 16, we experimented with the next valid value, 32. However, this larger disparity range resulted in poor performance. For the minimum disparity parameter, we kept it at the default value of 0. Regarding the mode, we selected STEREO\_SGBM\_MODE\_HH, as it performs a full-scale 16-path search similar

to our SGM algorithm which offers improved accuracy compared to the scaled-down alternatives. Although other modes prioritize speed over performance, we chose this mode to ensure high-quality results for our implementation. Lastly, for the block size, we plan to experiment within the optimal range of 3 to 11 to identify the most suitable value.

To determine the most effective pre-processing method, we generated disparity maps for each method using a default block size of 3. Among the techniques tested, Gaussian smoothing emerged as the best performer as shown in Table 6, achieving the lowest MSE (12.9789), MRE (0.2937), and MAE (2.2786). It also achieved the highest SZE (19.04%), indicating better subpixel precision, and the lowest BMP (59.76%), suggesting fewer mismatched pixels compared to other methods.

Table 7: Results of different pre-processing methods. Gaussian smoothing emerges as the winner. The bolded values represent the best values for each metric.

Block Size	Pre-processing methods	BMP	MSE	MRE	SZE	MAE
3	No pre-processing	61.84%	15.5953	0.3640	18.14%	2.6393
	Histogram Equalization only	63.56%	16.1603	0.3918	17.18%	2.7694
	Gaussian smoothing only	<b>59.76%</b>	<b>12.9789</b>	<b>0.2937</b>	<b>19.04%</b>	<b>2.2786</b>
	Both HE and GS	62.24%	15.2667	0.3549	17.69%	2.5962

## Hyperparameter Tuning

With the selected pre-processing method in place, we proceeded to tune the block size as part of our hyperparameter optimization process to determine the optimal block size for generating the most accurate disparity maps.

Table 8: Results of tuning the block size. Block size 11 emerges as the winner with 4 best metric values. The bolded values represent the best values for each metric.

Block Size	BMP	MSE	MRE	SZE	MAE
3	59.76%	<b>12.9789</b>	0.2937	19.04%	2.2786
5	59.37%	13.7773	0.2986	18.98%	2.3252
7	58.55%	13.9825	0.2924	19.37%	2.3071
9	57.74%	14.0962	0.2855	19.43%	2.2826
11	<b>57.06%</b>	14.3183	<b>0.2814</b>	<b>19.62%</b>	<b>2.2724</b>

Looking at the results above, block size 11 emerges as the winner with 4 best metric values. It achieved the lowest BMP (57.06%), MRE (0.2814), MAE (2.2724) and achieved the highest SZE (19.62%).

We generated our final results for the corridor and triclops2 images with the following model parameters:

- Maximum Disparity: 16
- Minimum Disparity: 0
- Block Size: 11
- Mode: STEREO\_SGBM\_MODE\_HH

Table 9: Comparing evaluation metrics of baseline and SGM

	BMP	MSE	MRE	SZE	MAE
Baseline	58.90%	13.9392	0.2763	19.45%	2.2282
SGM	57.06%	14.3183	0.2814	19.62%	2.2724

Table 10: Result comparison of disparity map for Corridor image against GT, Baseline and SGM

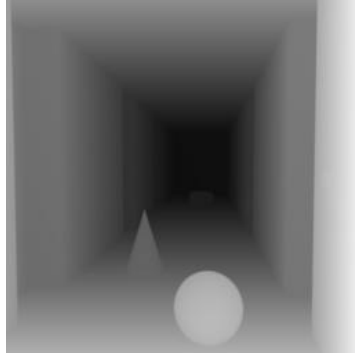
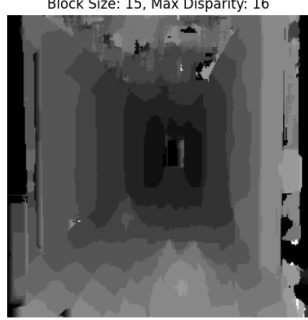
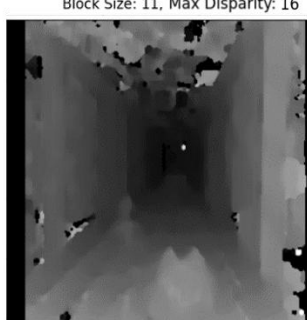
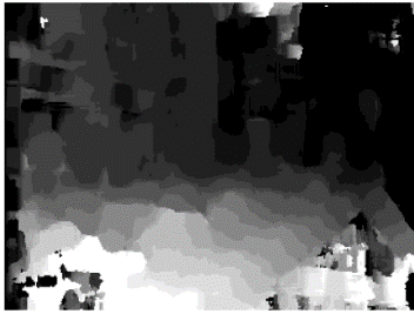

Ground Truth	Baseline Block Matching	SGM
	<p>Block Size: 15, Max Disparity: 16</p>  <p>BMP: 58.90%, MSE: 13.9392, MRE: 0.2763 SZE: 19.45%, MAE: 2.2282</p>	<p>Block Size: 11, Max Disparity: 16</p>  <p>BMP: 57.06%, MSE: 14.3183, MRE: 0.2814 SZE: 19.62%, MAE: 2.2724</p>

Table 11: Result comparison of disparity map for Triclopsi2 image against Baseline and SGM

Baseline Block Matching	SGM
<p>Block Size: 15, Max Disparity: 16</p> 	<p>Block Size: 11, Max Disparity: 16</p> 

The results show that the SGM method produces disparity maps that are visually smoother and less noisy compared to the Baseline method. However, the baseline Block Matching method produced more refined object boundaries, as clearly demonstrated by the cone object in the Corridor disparity map. This indicates that while SGM excels in overall smoothness and reducing noise, the Baseline Block Matching method may better preserve sharp edges and fine details, particularly at object boundaries.

However, there are some trade-offs between the two methods. While SGM produces more visually accurate maps, it introduces slight increases in certain error measures, suggesting that it might not always outperform the Baseline in all aspects. Nonetheless, the overall quality and smoothness of the SGM disparity map make it a favourable

choice for applications that prioritize visual consistency and precision in disparity estimation, especially in areas with significant depth variations.

## Graph Cut Stereo Mapping

Graph Cuts is a powerful Stereo Mapping algorithm proposed by Kolmogorov and Zabih in 2001 [4]. The algorithm employs a four-term energy function to compute the disparity map between two stereo images, where the disparity of each pixel corresponds to its relative depth in the scene. By optimizing this energy function, the algorithm finds the most consistent matches between the images while considering occlusions, smoothness, and uniqueness. Although Graph Cuts produces highly accurate results, especially in handling occlusions and disparity discontinuities, it requires significantly more computational time compared to traditional methods like block matching due to its global optimization approach.

### Explanation for energy function

The ultimate energy function is defined as:

$$E(f) = E_{data}(f) + E_{occlusion}(f) + E_{smoothness}(f) + E_{uniqueness}(f)$$

#### Data term

The best possible matches between corresponding pixels in the left and right images based on their color similarity. A smaller data term indicates a better match between pixels, minimizing the difference in intensity or color values between pixels at a particular disparity.

$$E_{data}(f) := \sum_{a, f(a)=1} D(a) = \sum_a D(a) \cdot \mathbb{1}(f(a) = 1),$$

#### Occlusion term

Pixels that do not have a match in the other image are considered occluded, and the algorithm penalizes these inactive assignments using a constant penalty  $K$ , thereby maximizing the number of successful matches.

$$E_{occlusion}(f) := \sum_{a, f(a)=0} K = \sum_a K \cdot \mathbb{1}(f(a) = 0) = K \times \#\mathcal{A} - \sum_a K \cdot \mathbb{1}(f(a) = 1).$$

#### Smoothness term

The neighbour pixels with similar colours are expected to have similar disparities. This term penalizes disparity jumps between neighbouring pixels. The penalty is lower if there is a significant contrast between the pixels, as this suggests a depth discontinuity.

Assigning different penalty weights depending on the intensity difference between the pixels. A smaller weight ( $\lambda$ ) is used when there's a significant contrast, while a larger weight ( $3\lambda$ ) is applied when the contrast is low. This approach allows the algorithm to

preserve sharp depth boundaries while smoothing out disparities within regions of similar intensity.

$$E_{\text{smoothness}}(f) := \sum_{a_1 \sim a_2} V_{a_1, a_2} \cdot \mathbb{1}(f(a_1) \neq f(a_2)),$$

where  $V_{a_1, a_2}$  is defined by

$$V_{a_1, a_2} := \begin{cases} \lambda_1 = 3\lambda & \text{if } \max(|I_1(p_1) - I_1(p_2)|, |I_2(q_1) - I_2(q_2)|) < 8 \\ \lambda_2 = \lambda & \text{otherwise.} \end{cases}$$

## Uniqueness term

The uniqueness term acts as a penalty, taking on an infinite value if the configuration violates the uniqueness constraint and zero otherwise. This means any configuration that attempts to match a single pixel to multiple pixels in the other image will be heavily penalized, making such configurations highly unfavourable in the energy minimization process.

$$E_{\text{uniqueness}}(f) := \sum_{\substack{a_1=(p, q_1) \\ a_2=(p, q_2) \\ q_1 \neq q_2}} \infty \cdot \mathbb{1}(f(a_1) = f(a_2) = 1) + \sum_{\substack{a_1=(p_1, q) \\ a_2=(p_2, q) \\ p_1 \neq p_2}} \infty \cdot \mathbb{1}(f(a_1) = f(a_2) = 1).$$

To evaluate the **Graph Cuts Stereo Mapping Algorithm** and optimize its performance, we obtained a source code implementation of the algorithm. This source code allows us to experiment with different parameter values, particularly the occlusion penalty  $K$  and the smoothness weight  $\lambda$ , which are crucial in balancing the trade-off between matching accuracy and the handling of occluded regions and smoothness in the disparity map.

## Result and discussion

Table 12: Disparity maps of Corridor image generated using different K and  $\lambda$  (Max Disparity = 32).

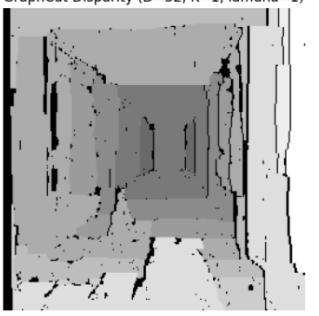
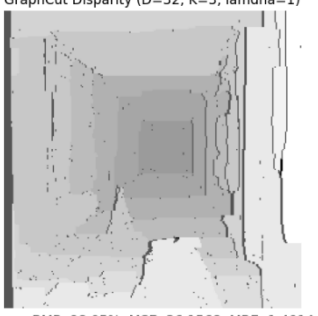
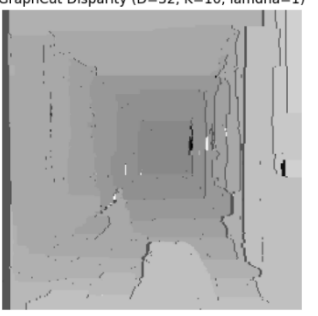
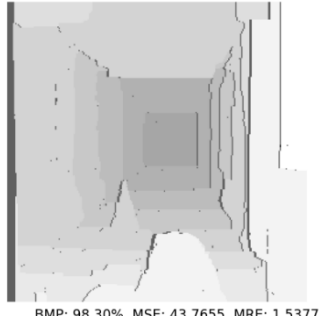
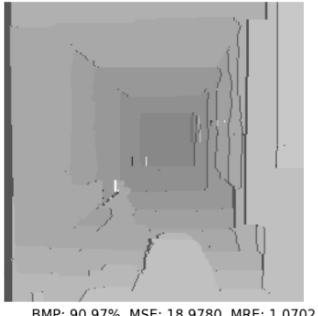
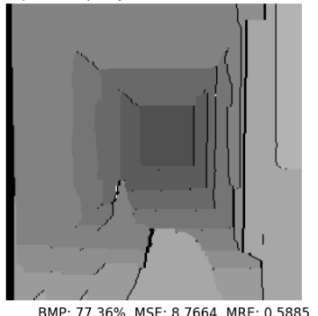
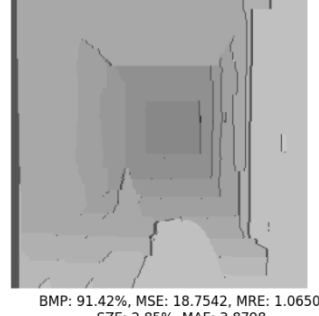
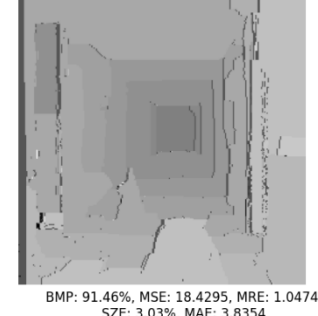
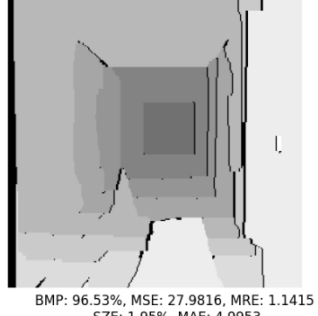
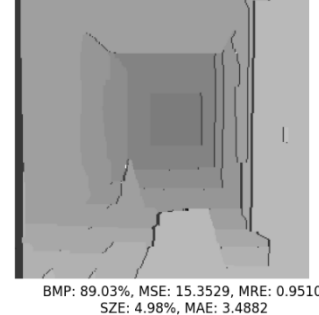
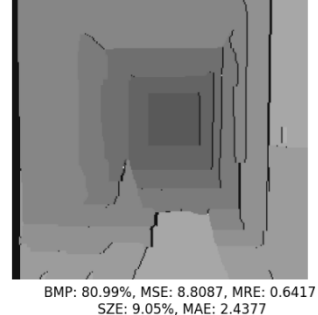
<p>GraphCut Disparity (D=32, K=1, lamdha=1)</p>  <p>BMP: 98.60%, MSE: 24.6493, MRE: 1.0501 SZE: 0.37%, MAE: 4.6294</p>	<p>GraphCut Disparity (D=32, K=3, lamdha=1)</p>  <p>BMP: 98.05%, MSE: 36.1568, MRE: 1.4014 SZE: 0.83%, MAE: 5.6843</p>	<p>GraphCut Disparity (D=32, K=10, lamdha=1)</p>  <p>BMP: 90.90%, MSE: 19.0567, MRE: 1.0652 SZE: 3.52%, MAE: 3.9075</p>
<p>GraphCut Disparity (D=32, K=10, lamdha=3)</p>  <p>BMP: 98.30%, MSE: 43.7655, MRE: 1.5377 SZE: 0.57%, MAE: 6.2609</p>	<p>GraphCut Disparity (D=32, K=30, lamdha=3)</p>  <p>BMP: 90.97%, MSE: 18.9780, MRE: 1.0702 SZE: 3.23%, MAE: 3.9026</p>	<p>GraphCut Disparity (D=32, K=30, lamdha=10)</p>  <p>BMP: 77.36%, MSE: 8.7664, MRE: 0.5885 SZE: 9.95%, MAE: 2.3120</p>
<p>GraphCut Disparity (D=32, K=50, lamdha=10)</p>  <p>BMP: 91.42%, MSE: 18.7542, MRE: 1.0650 SZE: 2.85%, MAE: 3.8798</p>	<p>GraphCut Disparity (D=32, K=50, lamdha=10) HE</p>  <p>BMP: 91.46%, MSE: 18.4295, MRE: 1.0474 SZE: 3.03%, MAE: 3.8354</p>	<p>GraphCut Disparity (D=32, K=50, lamdha=30)</p>  <p>BMP: 96.53%, MSE: 27.9816, MRE: 1.1415 SZE: 1.95%, MAE: 4.9953</p>
<p>GraphCut Disparity (D=32, K=143, lamdha=28)</p>  <p>BMP: 89.03%, MSE: 15.3529, MRE: 0.9510 SZE: 4.98%, MAE: 3.4882</p>	<p>GraphCut Disparity (D=32, K=250, lamdha=50)</p>  <p>BMP: 80.99%, MSE: 8.8087, MRE: 0.6417 SZE: 9.05%, MAE: 2.4377</p>	



Table 13: Result comparison of disparity map for Corridor image with different K and  $\lambda$ . The bolded values represent the best values for each metric.

HE	K	$\lambda$	BMP	MSE	MRE	SZE	MAE
No	1	1	98.60%	24.6493	1.0501	0.37%	4.6294
No	3	1	98.05%	36.1568	1.4014	0.83%	5.6843
No	10	1	90.90%	19.0567	1.0652	3.52%	3.9075
No	10	3	98.30%	43.7655	1.5377	0.57%	6.2609
No	30	3	90.97%	18.9780	1.0702	3.23%	3.9026
No	30	10	<b>77.36%</b>	<b>8.7664</b>	<b>0.5885</b>	<b>9.95%</b>	<b>2.3120</b>
No	50	10	91.42%	18.7542	1.0650	2.85%	3.8798
Yes	50	10	91.46%	18.4295	1.0474	3.03%	3.8354
No	50	30	96.53%	27.9816	1.1415	1.95%	4.9953
No	143	28	89.03%	15.3529	0.9510	4.98%	3.4882
No	250	50	80.99%	8.8087	0.6417	9.05%	2.4377

Table 14: Disparity maps of Triclops2 image generated using different K and  $\lambda$  (Max Disparity = 32).

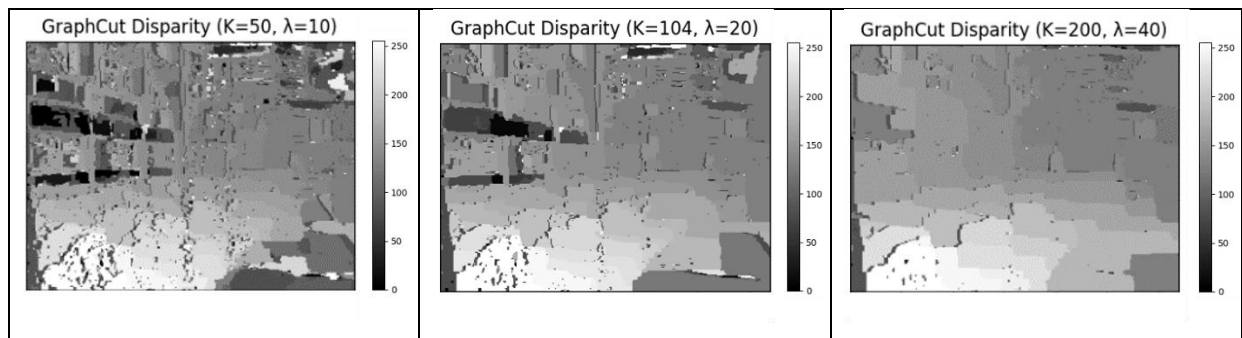
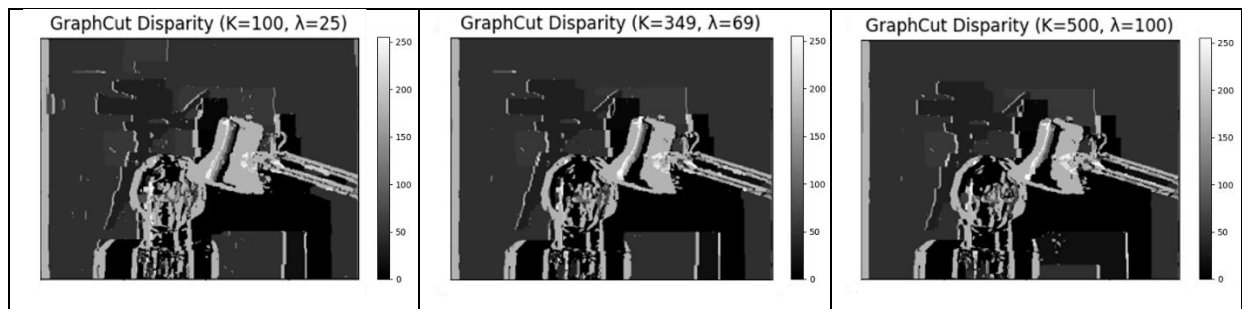


Table 15: Disparity maps of Tsukuba image generated using different K and  $\lambda$  (Max Disparity = 32).



Although the Tsukuba image is not one of the image pair given to us for analysis, a disparity map for the Tsukuba image is generated as per table 15 so as to provide more analysis for the Graph Cut Stereo Mapping algorithm as the Tsukuba image has good contrast and clarity, unlike the two image pairs provided to the team.

By comparing the result table and visually evaluating the output, we found that simply increasing the penalty values does not always lead to better results. Higher values of K, which penalize occluded pixels more heavily, and can reduce the number of occluded regions but often result in incorrect disparity assignments in complex areas,



particularly around object boundaries. Similarly, increasing  $\lambda$ , which enforces smoothness between neighbour pixels, tends to over-smooth the disparity map, causing a loss of detail and reducing the algorithm's ability to handle sharp depth discontinuities.

Furthermore, preprocessing with Histogram Equalization (HE) did not improve the performance of the graph-cut algorithm. This may be because HE distorts the image's intensity distribution, disrupting the intensity similarity information critical for calculating accurate penalties, ultimately leading to poorer disparity estimation.

In summary, it is evident that there is an optimal balance between  $K$  and  $\lambda$ ; excessively high penalties can degrade performance, resulting in inaccurate disparity maps.

Graph Cut Stereo Mapping offers highly accurate results by using global optimization, effectively handling occlusions, preserving sharp depth discontinuities, and producing smooth disparity maps, even in texture-less regions. However, it comes with significant computational and memory costs, making it slower and less suitable for real-time applications. Additionally, the algorithm is sensitive to parameter tuning, and improper values for the occlusion and smoothness terms can lead to over-smoothing or poor performance. While highly effective for applications requiring precision, it is more complex and resource-intensive compared to traditional methods like block matching.

Table 16: Comparing evaluation results of the Baseline and Graph Cut method

	BMP	MSE	MRE	SIZE	MAE
Baseline	58.90%	13.9392	0.2763	19.45%	2.2282
SGM	77.36%	8.7664	0.5885	9.95%	2.3120

Table 17: Result comparison of disparity map for Corridor image against GT, Baseline and Graph Cut

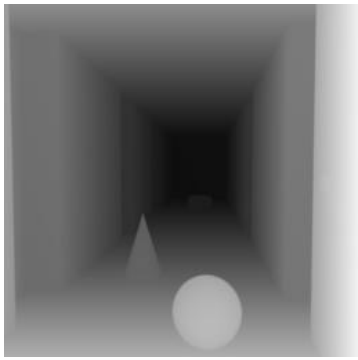
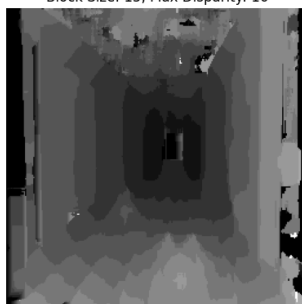
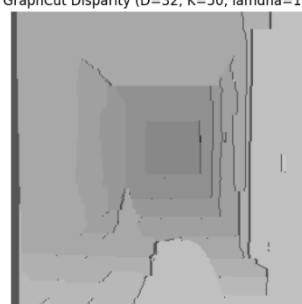
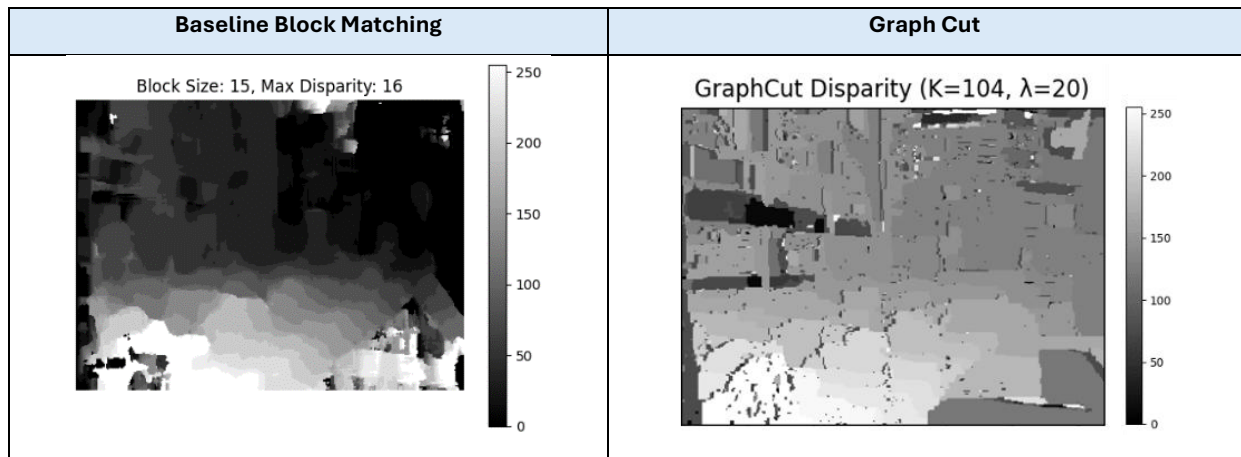
Ground Truth	Baseline Block Matching	Graph Cut
	<p>Block Size: 15, Max Disparity: 16</p>  <p>BMP: 58.90%, MSE: 13.9392, MRE: 0.2763 SIZE: 19.45%, MAE: 2.2282</p>	<p>GraphCut Disparity (D=32, K=50, lamdha=10)</p>  <p>BMP: 91.42%, MSE: 18.7542, MRE: 1.0650 SIZE: 2.85%, MAE: 3.8798</p>

Table 18. Result comparison of disparity map for Triclopsi2 image against Baseline and Graph Cut



From the tables above, comparing to the baseline, the disparity map Graph Cut Stereo Mapping method has less noise from that of the baseline block matching algorithm as seen from both corridor and Triclopsi2 images. This explains why the MSE value for Graph Cut Stereo Mapping method is smaller than that of the baseline as less pixels values are off by a large extent from the ground truth corridor disparity map.

Nevertheless, the Graph Cut Stereo Mapping method has underperformed in terms of BMP and SZE. This means that the disparity map produced from the Graph Cut Stereo Mapping method has more pixels with depth values not matching that of the ground truth than that of the Baseline Stereo Mapping method. From the corridor and Triclopsi2 images, we see that the disparity map from Graph Cut shows has similar shades of colours for both near and far objects. This meant that the disparity map from the Graph Cut algorithm is not sensitive to changes in depth. This can explain why the MAE and MRE value for the Graph Cut is higher than that of the baseline.

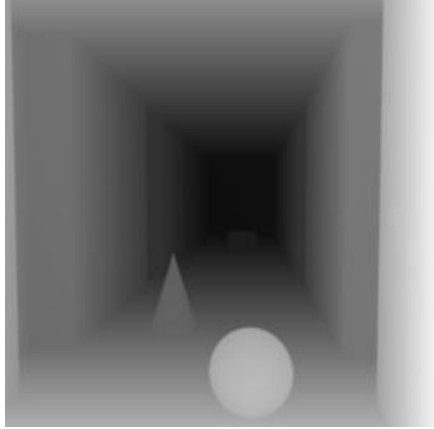
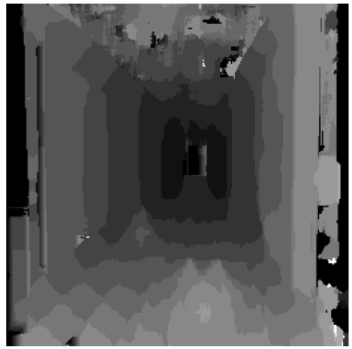
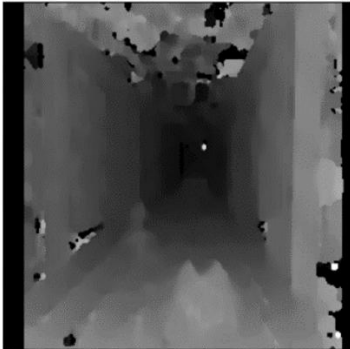
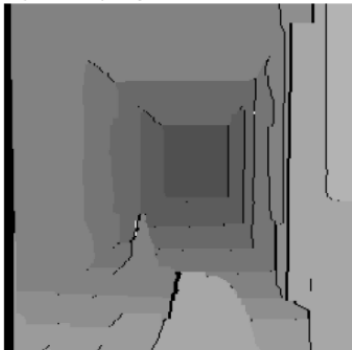
## Best Stereo Mapping model

### Best metrics of each Stereo Mapping model on Corridor image

Table 19: Evaluation results of all Stereo Mapping models, excluding visual analysis by team members. The bolded values represent the best values for each metric.

Model	BMP	MSE	MRE	SZE	MAE
<b>Baseline Block Matching</b> (Block size=15, Max Disparity=16, GS Pre-processed)	58.90%	13.9392	<b>0.2763</b>	19.45%	2.2282
<b>Semi-Global Matching</b> (Block size = 11)	<b>57.06%</b>	<b>14.3183</b>	0.2814	<b>19.62%</b>	<b>2.2724</b>
<b>Graph Cut Stereo Mapping</b> (Max Disparity=32, K=30, $\lambda=10$ )	<b>77.36%</b>	<b>8.7664</b>	<b>0.5885</b>	<b>9.95%</b>	<b>2.3120</b>

Table 20: Disparity maps of Corridor Image from all Stereo Mapping models selected based on best metrics.

Ground Truth	Baseline Block Matching
	<p>Block Size: 15, Max Disparity: 16</p>  <p>BMP: 58.90%, MSE: 13.9392, MRE: 0.2763          SZE: 19.45%, MAE: 2.2282</p>
Semi-Global Matching (SGM)	Graph Cut Stereo Mapping
<p>Block Size: 11, Max Disparity: 16</p>  <p>BMP: 57.06%, MSE: 14.3183, MRE: 0.2814          SZE: 19.62%, MAE: 2.2724</p>	<p>GraphCut Disparity (D=32, K=30, lamdha=10)</p>  <p>BMP: 77.36%, MSE: 8.7664, MRE: 0.5885          SZE: 9.95%, MAE: 2.3120</p>

The Ground Truth map presents a smooth and well-defined gradient where nearer objects are lighter in colour while further objects are darker in colour, showing clear outlines of objects such as the circular ball and cone on the ground, as well as the edges of the corridor. There are smooth transitions between different areas of depth along the length of the corridor. By observing the disparity maps of Corridor image, each model exhibits different levels of quality in capturing depth information.

The Baseline Block Matching method has the smallest MRE, suggesting that SGM provides a more accurate representation of both near and far objects compared to the other two algorithms. Nevertheless, there are some flaws with the disparity map. The colour transition from regions closer to the camera to the far end of the corridor are rough and inconsistent, making it difficult to interpret the depth details without smooth gradients. A faint outline of the cone's tip is visible, but the lower part of the cone is not discernible. In the area where the ball is located, the colour resembles that of the ground truth map, though the shape is not circular as per the ground truth disparity map. Moreover, there are noticeable noise around the left, right and top of the image, indicating that the Baseline Block Matching might be weak at depth estimation of areas closer to the camera. This could be because for areas closer to the camera, the search

space is reduced as there is no pixels to allow the algorithm to match outside the image boundary.

The Semi-Global Matching (SGM) algorithm demonstrated a better performance, producing a disparity map with the greatest number of pixels having its predicted depth being matched correctly with the ground truth disparity map as shown by its lowest BMP value and highest SZE value compared to the other 2 algorithms. Its lowest MAE value indicates that the object shape and edges of the corridor are the most accurately captured compared to the other algorithms. Unlike the Baseline Block Matching method, which shows a patchy increase in pixel darkness towards the end of the corridor, the SGM disparity map features a more gradual gradient of increasing blackness in that region. Despite these good results from the evaluation metrics, the disparity map still looks different from that of the ground truth image.

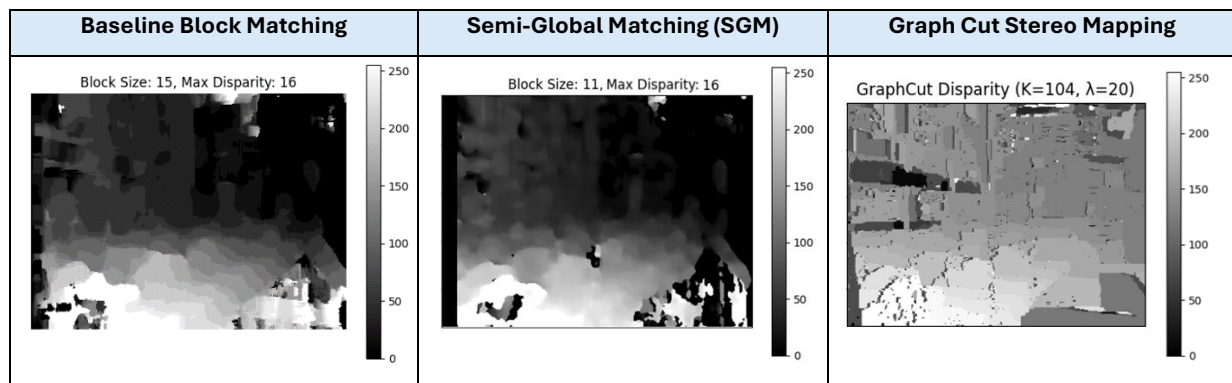
For SGM, there are many noticeable blocky artifacts and noise on the disparity map, especially in the upper regions and the left and right sides of the image. At both left and right side of the image, there is a black vertical line which is inconsistent with the ground truth image. This can explain why the MSE is the largest for the SGM method as the depth values of the pixels are off by a large extent due to the noises. The shape of the ball and cone are not captured with high clarity. Only some lighter colours at the region of where the cone and ball are, exists on the disparity map.

The Graph Cut Stereo Mapping produced the cleanest disparity map of all the models. This can explain why it has the lowest MSE as most of the depth values are not off by a large extent. Though the outline of the cone is captured more accurately than the other two algorithms, the outline of the round object in the foreground was not captured accurately. Similar to the Baseline Block Matching, as the depth of the corridor increases, the transition of the colour going from light to dark is not homogeneous and not in a smooth gradient.

Moreover, in the ground truth image, black pixels are allocated to the end of the corridor showing that that region is the furthest away from the camera. Nevertheless, that region in the Graph Cut disparity map is grey instead of black, meaning that the disparity values on the disparity map is incorrect. Black pixels are instead allocated to the edge and outline of objects in the disparity map, which is inconsistent with the ground truth. This can explain why the BMP and SZE is the lowest and MAE being the highest for Graph Cut as most of the pixels are mapped to incorrect depth values. Moreover, for the areas closer to the camera, the colours on the disparity map are different from that of the ground truth, explaining why Graph Cut has the highest MRE as it is not good as estimating region closer to the camera.

## Triclopsi2 image analysis

Table 21: Disparity maps of Triclopsi2 image from all Stereo Mapping models.



In the case of the Triclopsi2 image, both the disparity maps from the Baseline Block Matching and Semi-Global Matching (SGM) captures the general structure of the building nearer to the camera, the bushes and the pavement, and follows the intuition that regions further away have harder colours while regions nearer have lighter colours. Both the baseline and SGM showed noises at the lower left of the disparity maps corresponding to where the bushes are, and noise at the lower right of the disparity maps, corresponding to the pavement. Disparity map from SGM showed noticeably more noise than that of the baseline at the pavement area. The disparity map from the baseline showed a clearer outline of the buildings and bushes. While both algorithms capture the general structure of the scene, they continue to struggle with finer details, such as the bushes in the building further away from the camera.

The Graph Cut Stereo Mapping generated the disparity map with the least noise. Although the map captures the edge details, it captures the edges by outlining the objects with black colour instead of colouring each object with the same colour. The transitions between different depth levels are not obvious as the colours change is minimal, from white to light grey only. Moreover, the depth information is not accurately representing the depth gradient of the objects in the image. For example, the top right side of the image should be the furthest away from the camera while the pavement at the bottom right corner should be the nearest to the camera. The disparity map did not capture this and has given a darker colour to the pavement than the buildings further from the camera.

## Best Stereo Mapping model by Visual Analysis voting

Table 22: Visual Analysis votes by all team members for the best Stereo Mapping model for each image.

	Ming Yuan	Seik Man	Timothy	Lehan	Chen Fei	Winning algorithm
Corridor	SGM	SGM	SGM	SGM	SGM	SGM
Triclopsi2	Baseline	Graph Cut	SGM	SGM	SGM	SGM

The voting results shows that the Semi-Global Matching (SGM) model is the preferred choice for both the Corridor and Triclopsi2 images, with the most team members selecting it as the best stereo mapping model. Hence, we conclude that **SGM** is the best Stereo Mapping model.

## Conclusion

In this report, we compared the performance of the baseline Block Matching, Semi-Global Matching (SGM), and Graph Cut Stereo Mapping over 2 image pairs provided to the team, with possible pre-processing methods. The success indicators used are Visual Analysis by team members, Mean Square Error, Mean Absolute Error, Mean Relative Error, and Subpixel Zero Error. Based on the performance of the Stereo Mapping algorithms on two image pairs, the team deduced that SGM is the best model. Nevertheless, the best stereo mapping model might vary if different types of images were used instead of the Corridor and Triclops2 images. Therefore, to improve the accuracy in determining which stereo mapping algorithm is most suitable across a wider range of scenes, more diverse and complex image sets could be provided for evaluation. This would allow the team to better assess the strengths and weaknesses of each model, identifying which algorithm performs best for different types of environments and image characteristics.

## References

- [1] A. Phan and S. Chien, "Flight Test Validation of Collision Avoidance System for a Multicopter using Stereoscopic Vision," 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), Hanoi, Vietnam, pp. 51-56, Nov. 2019, doi: 10.1109/NICS48868.2019.8798023.
- [2] M. Aibin, "Disparity map in stereo vision," Baeldung on Computer Science, 26-May-2022. [Online]. Available: <https://www.baeldung.com/cs/disparity-map-stereo-vision>. [Accessed: 25-Oct-2024].
- [3] O. J. Woodford, P. H. S. Torr, I. Reid, and A. Fitzgibbon, "Global stereo reconstruction under second order smoothness priors," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.
- [4] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328-341, Feb. 2008. [Online]. Available: <https://elib.dlr.de/73119/1/180Hirschmueller.pdf>.
- [5] Kolmogorov, V., & Zabih, R. (2002). Computing visual correspondence with occlusions using graph cuts. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2, 508–515.
- [6] Sun, J., Shum, H.-Y., & Zheng, N.-N. (2002). Stereo matching using belief propagation. In *Lecture Notes in Computer Science* (pp. 510–524). Springer Berlin Heidelberg.
- [7] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," in *Proceedings of the Sixth IEEE International Conference on Computer Vision*, Mumbai, India, January 1998, pp. 1073–1080.