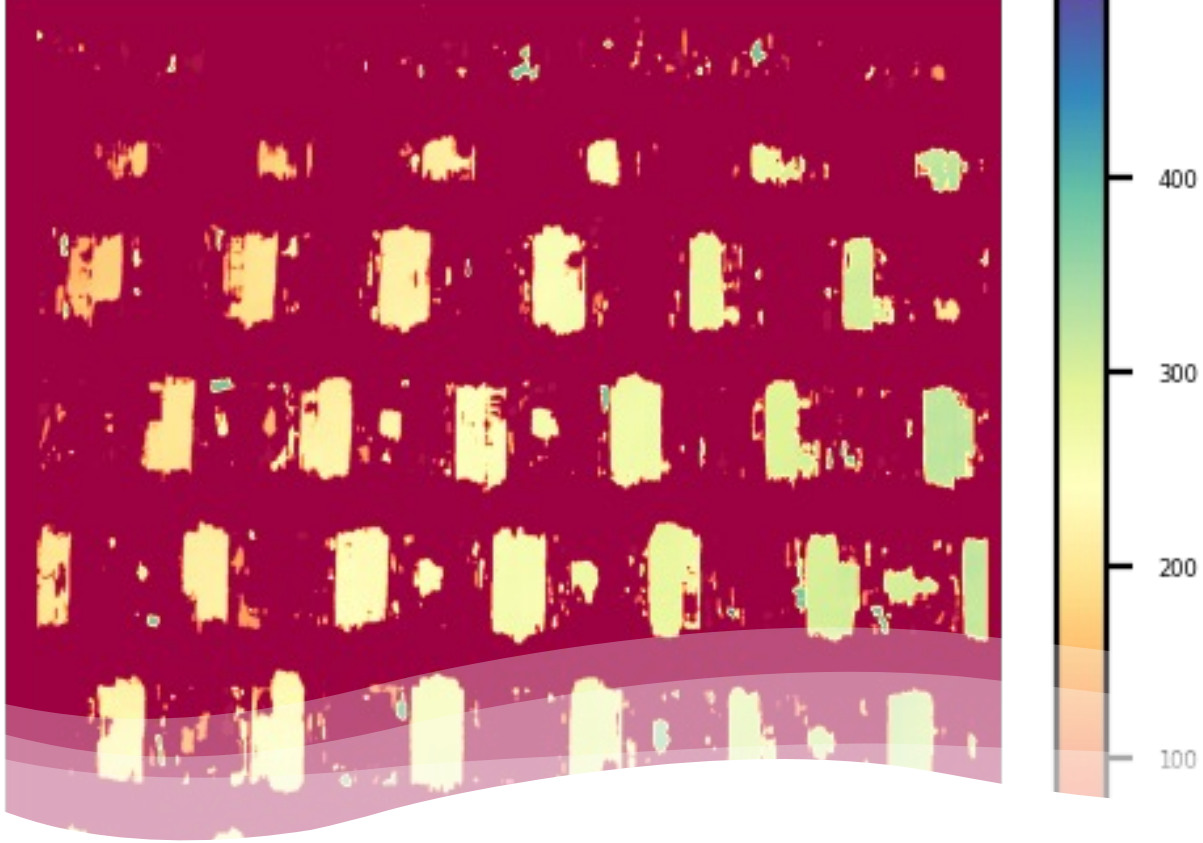
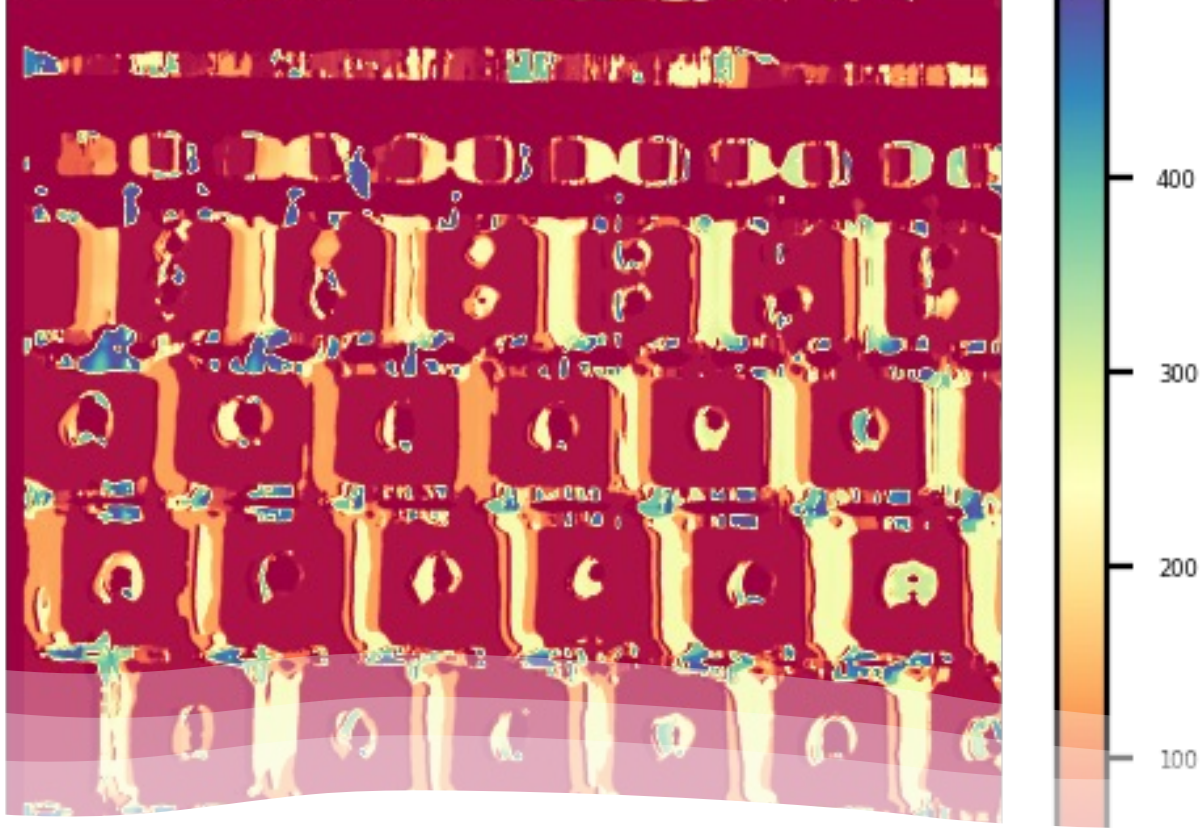




Activity 7:

Shape from Stereo

Physics 301

2nd Semester AY 2021-2022

Submitted by: Mark Jeremy G. Narag

Student Number: 2014-64423

Program: PhD Physics

Estimating depth of an object

Did you know that we can estimate the depth of an object from two images? Shown in figure 1 is a stereometry setup where a scene is captured by two identical cameras of focal length ***f***, spaced at a distance ***b*** slightly apart. Consider the red circle as a scene point. Let the vertical ***Z*** line be the x-axis and ***L*** be the distance of the scene point from the axis. Thus, the scene is seen to be at a distance x_1 for the left camera, and x_2 from the right camera.

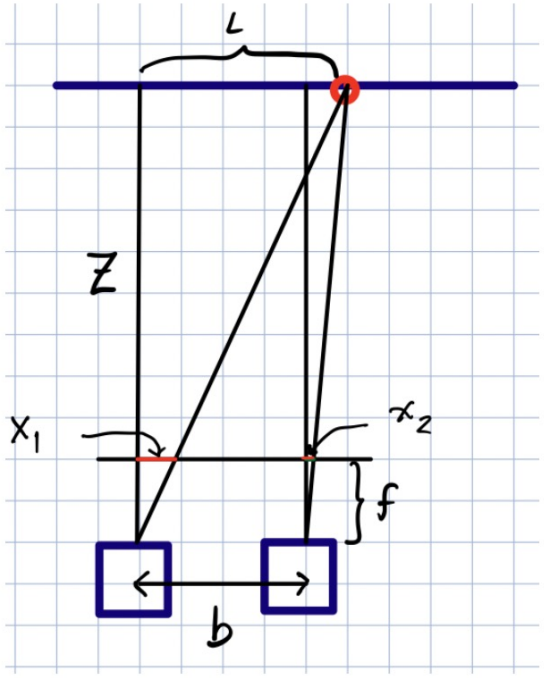


Figure 1. Stereometry setup

Doing the math:

$$\frac{L}{Z} = \frac{x_1}{f}$$

$$\frac{L-b}{Z} = \frac{x_2}{f}$$

Solving for Z, we have:

$$Z = \frac{bf}{x_1 - x_2}$$

The term (x_1-x_2) is known as disparity. The larger the disparity, the nearer the point, while the smaller the disparity the farther the point [1].

[1] M. Soriano, Physics 301 Shape from Stereo, 2022

In this activity, we will be using the Matlab code provided by Dr. Soriano. There are three disparity functions available in Matlab:

- **DisparityBM** – computes disparity map using block matching. The Block Matching block estimates motion between two images by comparing blocks of pixels. The block **matches the block of pixels between frames by moving the block of pixels over a search region.**
- **DisparitySGM** – computes disparity map using Semi-Global Matching. Semi-global matching **uses information from neighboring pixels in multiple directions to calculate the disparity of a pixel**.
- Disparity – this is not recommended by Mathworks so we will just focus on the two.

Furthermore, we will also do it in Python using the following functions from OpenCV:

- **cv2.StereoBM_create** – computes disparity map using block matching
- **cv2.StereoSGBM_create –** computes disparity map using Semi-Global Matching

Python Code:

```
[1]  1 from google.colab import drive
     2 drive.mount("/content/drive")

     Mounted at /content/drive
```

```
[2]  1 !ls drive/My\ Drive
     2 file_path = "/content/drive/My Drive/Physics 301 2S AY2122/Activity 7 Stereometry/"
```

```
[3]  1 import cv2
     2 import numpy as np
     3 import matplotlib.pyplot as plt
     4 from mpl_toolkits.mplot3d import Axes3D
     5 from mpl_toolkits.axes_grid1 import make_axes_locatable
```

```
[4]  1 imgL = cv2.imread(file_path + "board1.jpg", cv2.IMREAD_GRAYSCALE)
     2 imgR = cv2.imread(file_path + "board2.jpg", cv2.IMREAD_GRAYSCALE)
     3
     4 stereoBM = cv2.StereoBM_create(numDisparities=32, blockSize=51)
     5 disparityBM = stereoBM.compute(imgL, imgR)
     6 stereoSGM = cv2.StereoSGBM_create(numDisparities=32, blockSize=51)
     7 disparitySGM = stereoSGM.compute(imgL, imgR)
```

Sections 1-3 are just the usual imports and preliminary steps.

Section 4 computes the disparity from two images. Here are the parameters to take note:

numDisparity determines the resolution of your stereo map. So if you set it to 16, the algorithm will attempt to define 16 different levels of depth for the map. It must be divisible by 16.

BlockSize is the size of the block of pixels the algorithm will compare for each stereo pair. Must be odd number.

The parameters that I set here are my own parameters and is open for exploration. These numbers were chosen from based on my quick parameter exploration in slide 7.

Python Code: Displaying the plots in 2D and 3D

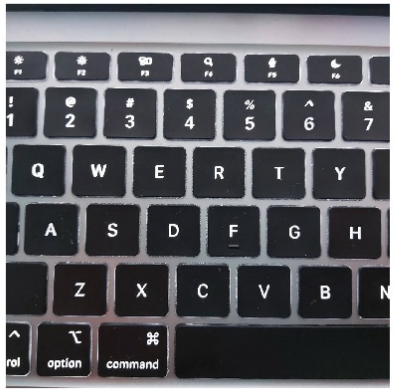
```
[5]   1 plt.figure(figsize = (8,4))
      2 plt.subplot(121)
      3 plt.imshow(imgL, cmap = 'gray')
      4 plt.title("Left")
      5 plt.subplot(122)
      6 plt.imshow(imgR, cmap = 'gray')
      7 plt.title("Right")
```

```
[6]   1 plt.figure(figsize = (5,4), dpi=200)
      2 plt.subplot(121)
      3 plt.imshow(disparityBM, cmap = 'Spectral')
      4 plt.title("Disparity Map BM", fontsize=6)
      5 plt.axis("off")
      6 cbar = plt.colorbar(fraction=0.045)
      7 cbar.ax.tick_params(labelsize=3)
      8 plt.tight_layout()
      9 plt.subplot(122)
     10 plt.imshow(disparitySGM, cmap = 'Spectral')
     11 plt.title("Disparity Map SGM",fontsize=6)
     12 plt.axis("off")
     13 cbar = plt.colorbar(fraction=0.045)
     14 cbar.ax.tick_params(labelsize=3)
     15 plt.tight_layout()
```

```
1 plt.figure(figsize = (6,4))
2 ax = plt.axes(projection='3d')
3 X, Y = np.meshgrid(range(len(disparityBM)), range(len(disparitySGM)))
4 p = ax.plot_surface(X, Y, disparityBM, cmap = 'Spectral')
5 ax.view_init(elev = 50,azim=150)
6 plt.show()
```

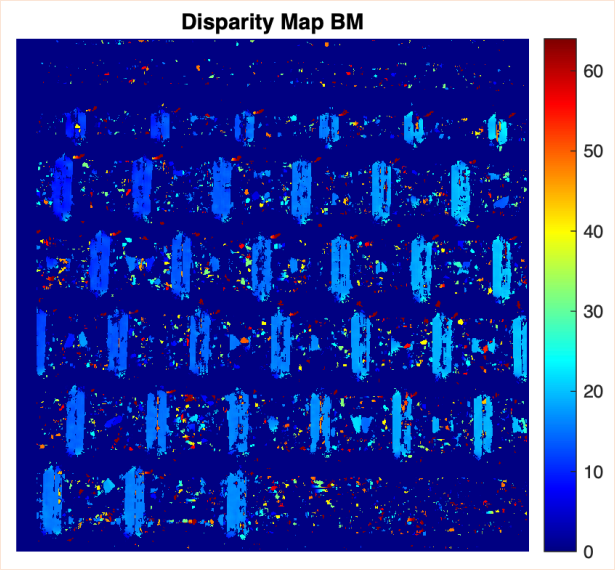
Image of interest: **Laptop keyboard**

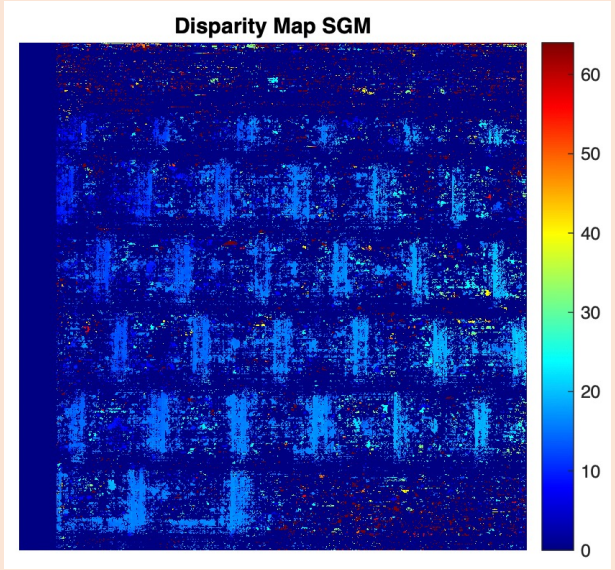
left



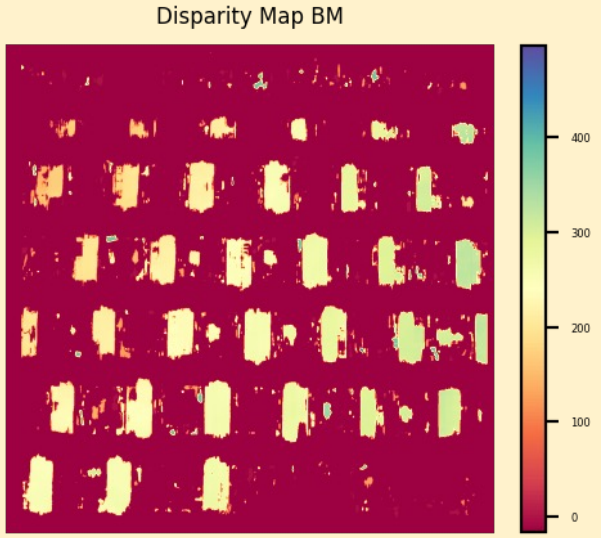
right

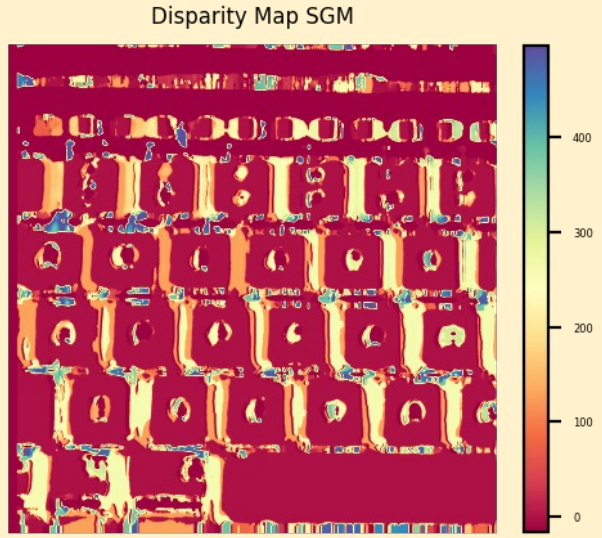
Matlab code:

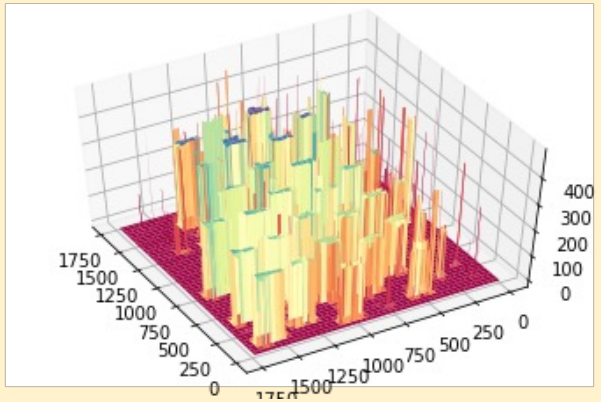


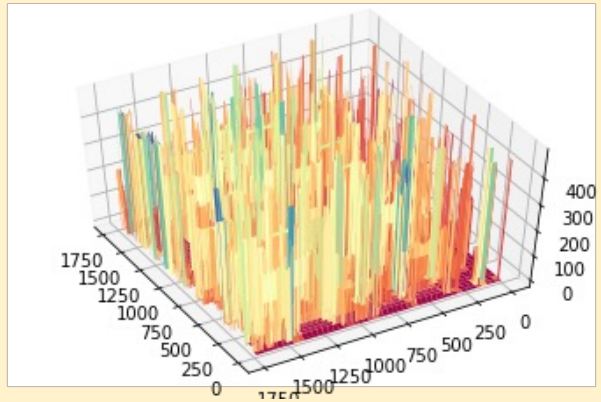


From python code (top: 2D, bottom: 3D):









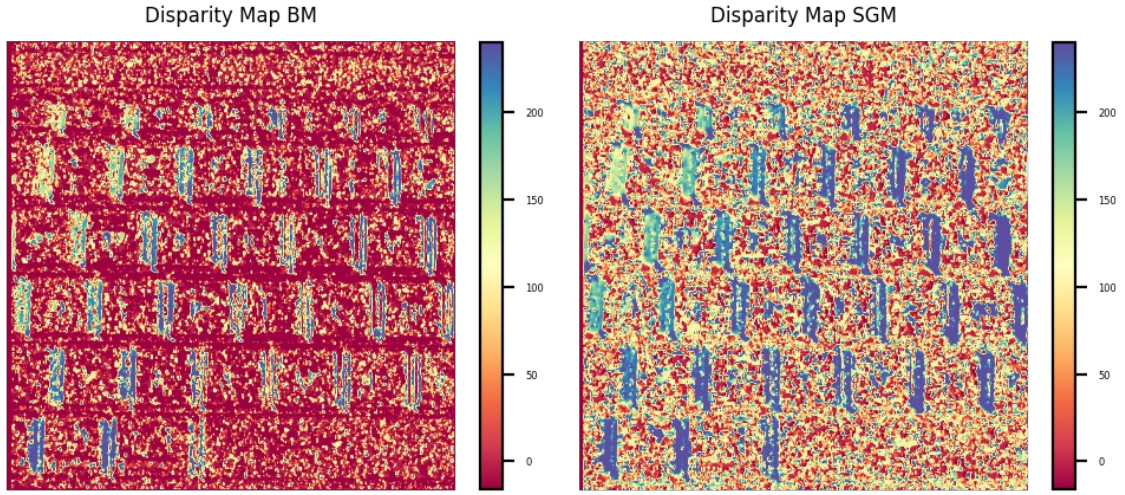
The problem with SGM in this case is it made a mistake in estimating the depth of areas around the keys. Thus in 3D plot, it was those areas that protruded, not the keys itself.

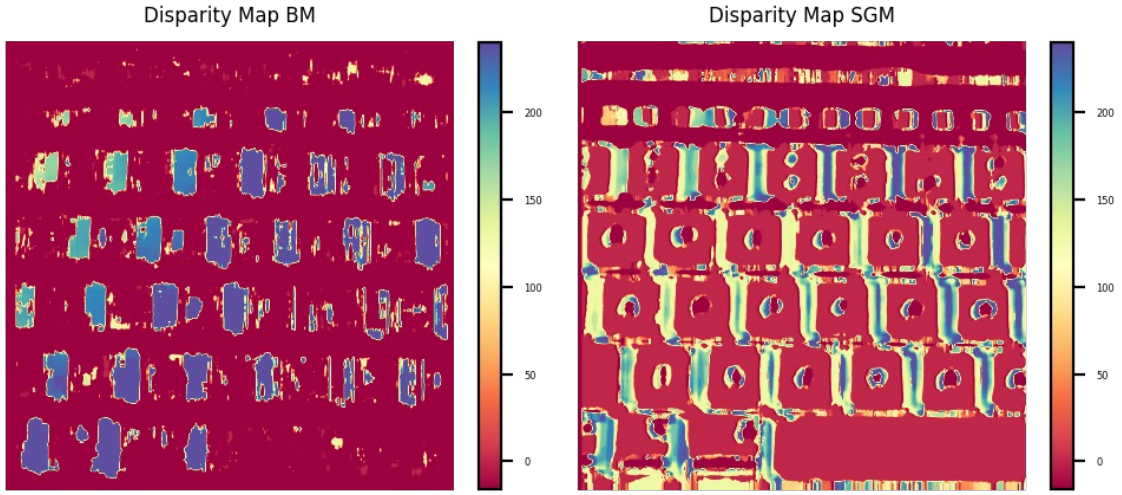
Exploring the parameters of the stereo-disparity function in OpenCV

Blocksize = **11**

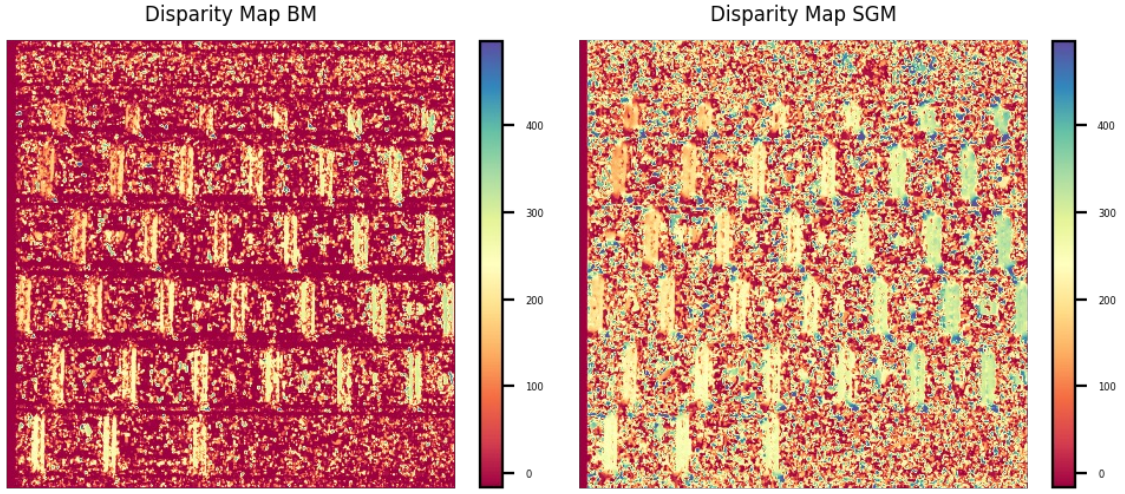
BlockSize = **51**

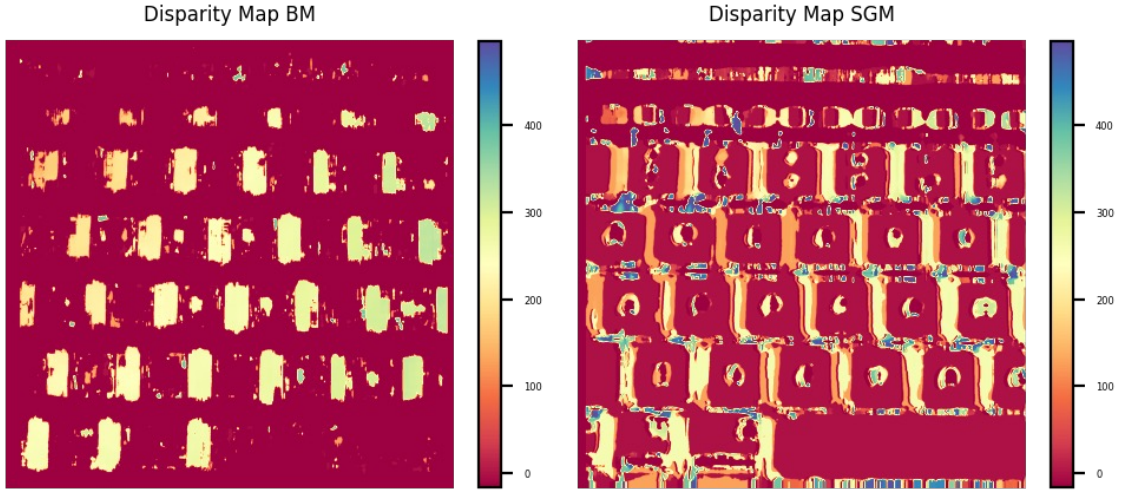
numDisparities = **16**





numDisparities = **32**





Lower blocksize means grainy map. So far, we can see that numDisparties=32 and BlockSize = 51 gives the best output.

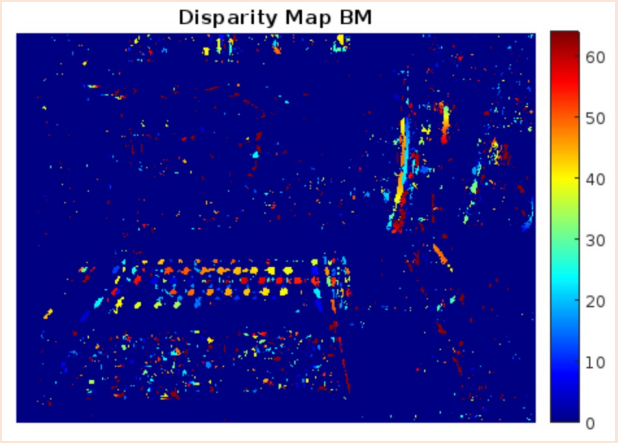
Image of interest: **Laptop with objects** [FAIL]

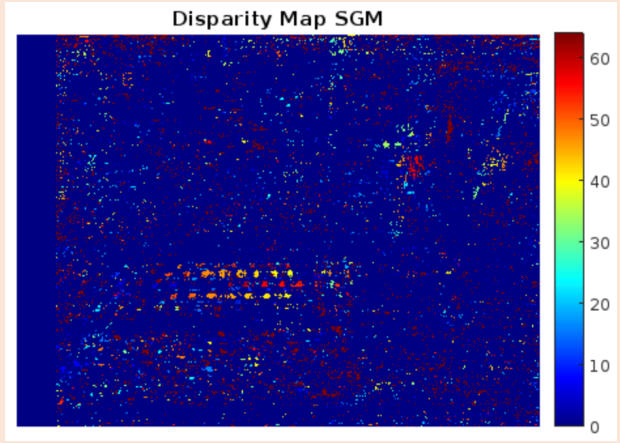
left

right

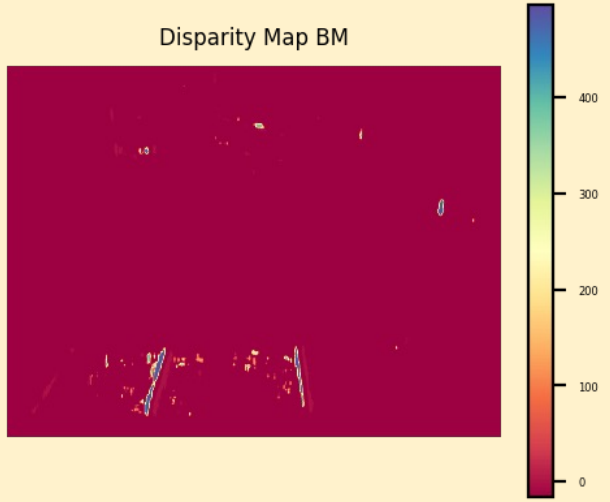
Here is an example of where I failed to get the disparity map of the object/s. One big mistake here is the distance that I took between two images. The distance between x1 and x2 was just too large for both Matlab and python functions to render. Matlab was able to have a better map tho compared to OpenCV function in Matlab. It was able to render the keyboard area of the laptop. The python output can still be improved by varying the parameters.

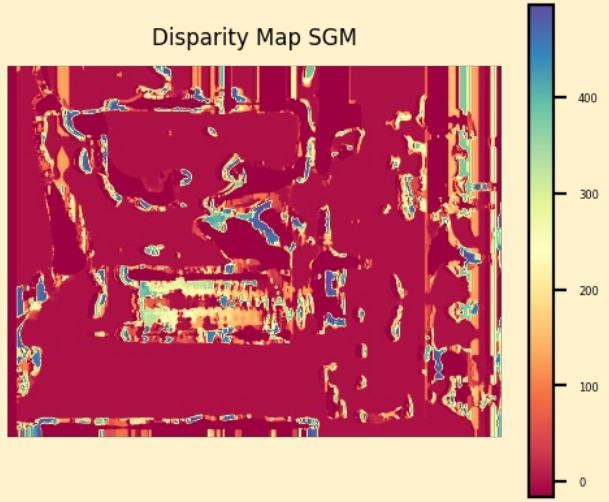
Matlab code:



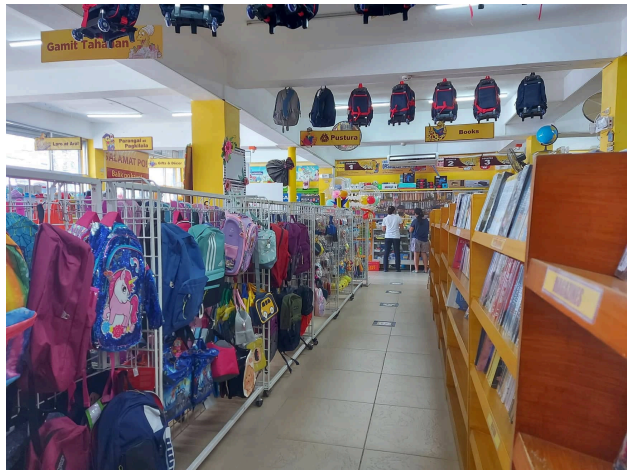


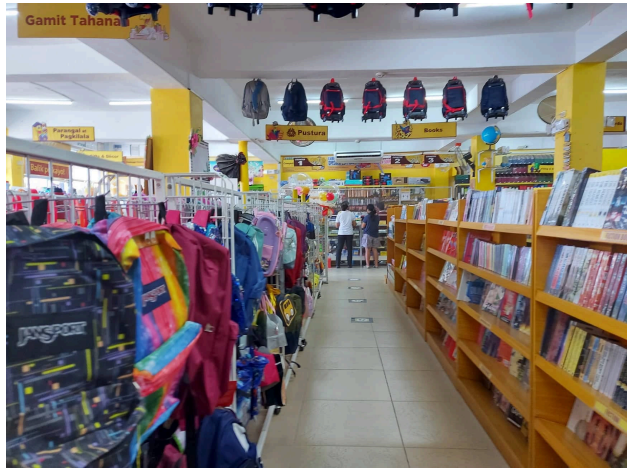
From python code





Reflection: I honestly took a lot of images for this activity. I even went outside (Pandayan bookshop, June 21) but when I tried getting the disparity map for all the images when I got home, none of them worked successfully. It took me some time to realize that x1 and x2 should be very close to each other! Like really really close! Luckily, I was able to get one for the keyboard on my laptop in which I still even had a hard time capturing it because the distance should really be very small.





CRITERIA	QUALIFICATIONS	SCORE
Technical correctness	☑ Met all objectives. ☑ Results are complete. Results are verifiably correct. ☑ Understood the lesson.	30
Quality of presentation.	☑ All text and images are of good quality. ☑ A picture or diagram of the setup is shown (if the activity has one). ☑ Code has sufficient comments and guides. ☑ All plots are properly labeled and are visually understandable. ☑ The report is clear.	30
Reflection	☑ Explained the validity of results. ☑ Discussed what went right or wrong in the activity. ☑ Justified the self score. ☑ Acknowledged sources (e.g. persons consulted, references, etc.)	30
Ownership	❑ Applied the technique on other data ❑ Discovered limitations in the technique ❑ Introduced improvements to the technique (I just did it on python and tried exploring the parameters but that isn't enough.)	0

90

Although I was able to do the bare minimum for this activity. I, too, am frustrated with my output. I wasn't able to explore much this time as compared to the rest of my outputs in this course.