Computer Vision

Assignment 3

Implementing Stereo Vision

Prepared By:

Eman Mohamed Salah El-Din	6072
Nael Mostafa Mohamed	6099
Ahmad Abdelrahman Marouf	6543

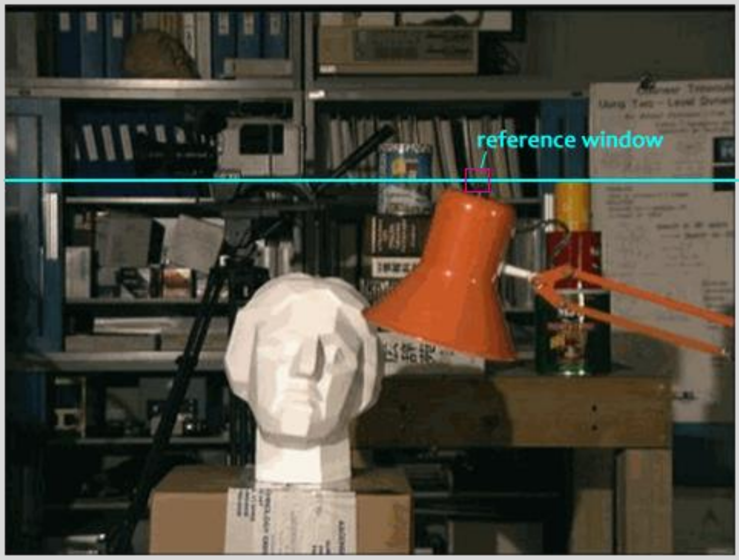
Contents

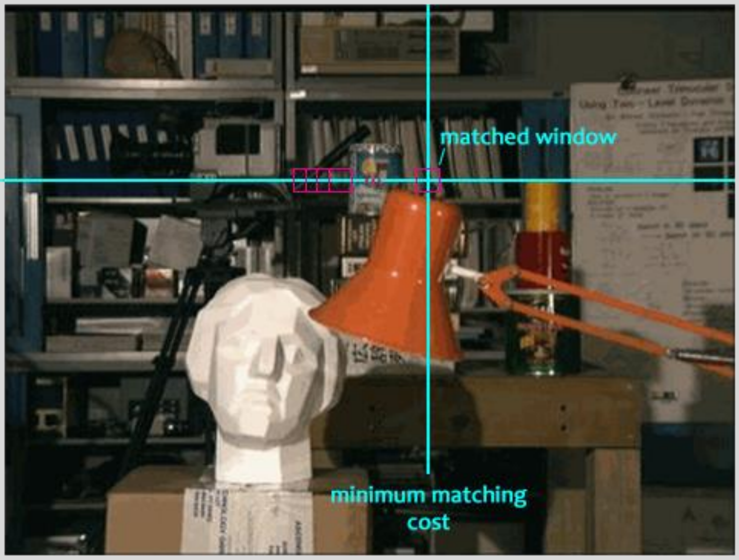
1. First Method: Block Matching 3
   1.1 Process Description 3
   1.2 Notes 3
2. Second Method: Dynamic Programming 4
   2.1 Process Description 4
3. Comparing Results 6
   3.1 Image 1: 6
   3.2 Image 2: 7
   3.3 Image 3: 8

1. First Method: Block Matching

1.1 Process Description

- In this method we find the disparity in the images by matching every pixel in the left image to a certain pixel in the right image.
- First, we define a reference window in the left image, then we slide a similar window across the epipolar line in the right image and compare the contents with the reference window using a certain metric. This process is repeated for every pixel in each scanline.





- We compare the results of window sizes 1 (pixel to pixel comparison), 5, and 9.
- We use 2 metrics to estimate matching cost:
  1. **<u>Sum of Absolute Differences (SAD)</u>**
     Computed by calculating the absolute difference of the reference window and matched window's pixels element by element then adding them up.
  2. **<u>Sum of Squared Differences (SSD)</u>**
     Computed by calculating the squared difference of the reference window and matched window's pixels element by element then adding them up.

1.2 Notes

- We use a maximum disparity value to limit our window search within a reasonable range to save some computations

2. Second Method: Dynamic Programming

2.1 Process Description

- This method uses a dynamic programming algorithm to calculate the minimum cost for matching a whole row in the images.
- For every pixel in the image there are 3 possible conditions:
    1. The pixel is visible in the left and right images (it is matched).
    2. The pixel is visible in the left image but occluded in the right image.
    3. The pixel is visible in the right image but occluded in the left image.

- In the case that a pixel is visible in both images, the following formula is used to estimate the cost of matching a pixel from the left image scanline $I_l(i)$ to a pixel in the right image scanline $I_r(j)$

$$d_{ij} = \frac{(I_l(i) - I_r(j))^2}{\sigma^2} \quad \sigma: some\ measure\ for\ pixel\ noise\ (we\ use\ \sigma = 2)$$

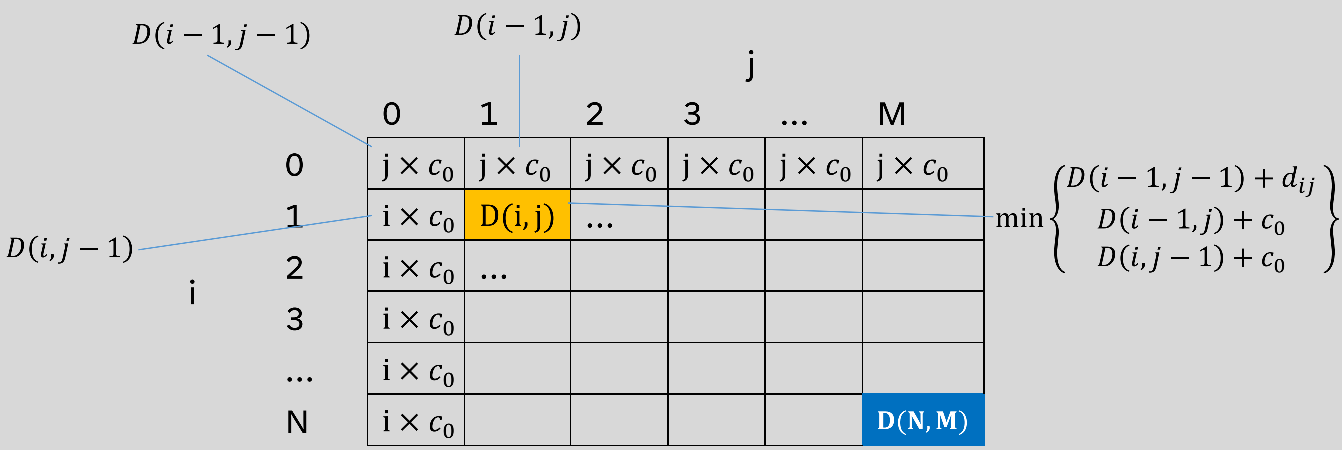
- In the case that a pixel is visible in one image and occluded in the other, we use a constant $c_0 = 1$ as the cost of skipping a pixel.

- We define the cost matrix D with dimensions (N x M) where N and M are the number of columns in the left and right images respectively (N x N if images have matching dimensions). We compute D as follows:

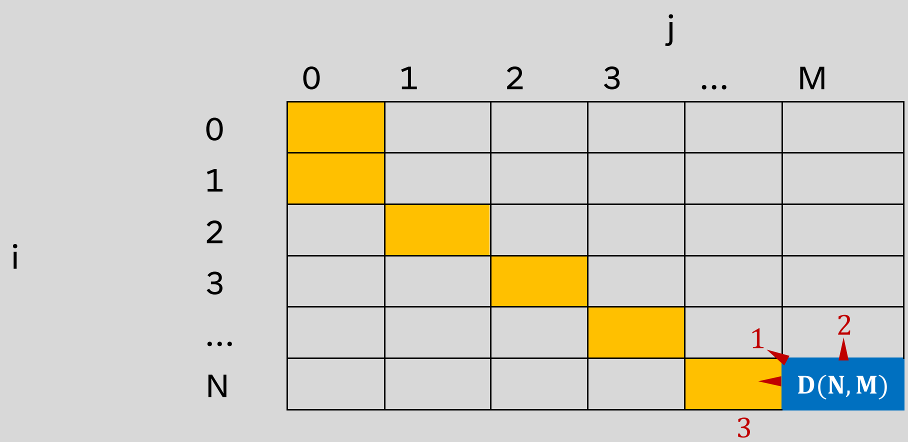
$D(i,0) = i \times c_0 \qquad for\ 0 \le i \le n$ (Initialize first column)

$D(0,j) = j \times c_0 \qquad for\ 0 \le j \le m$ (Initialize first row)

$$D(i,j) = \min\{[D(i-1,j-1) + d_{ij}], [D(i-1,j) + c_0], [D(i,j-1) + c_0]\}$$
$$for\ \ 1 \le i \le n\, , 1 \le j \le m$$

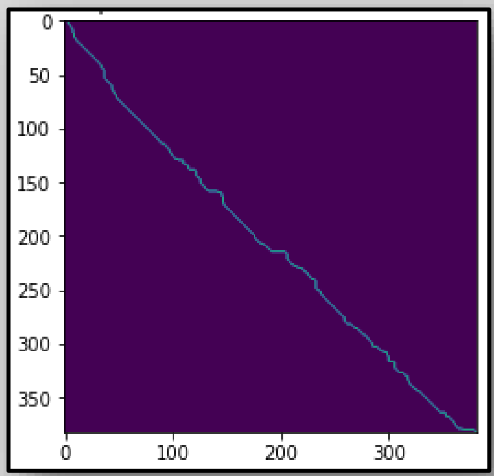


- After D is fully computed, the total cost of matching two scanlines can be found in the element $D(N, M)$. We now perform a backwards pass from $D(N, M)$ along the optimal path (minimum cost) to get the optimal alignment for the 2 images and calculate the disparity of each pixel in the corresponding pixel maps.



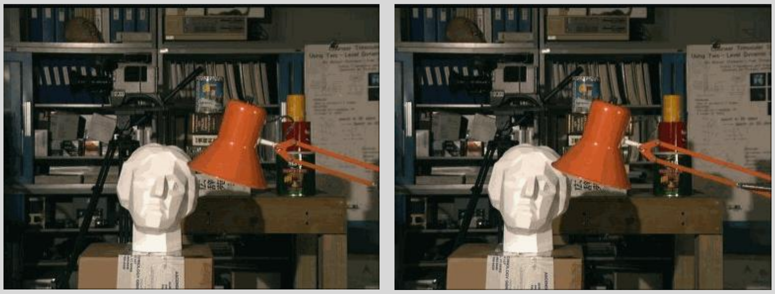
To get optimal path:
- Start from $D(N, M)$
- Find minimum value: $\min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\}$
  1. $D(i-1, j-1)$ – Minimum step is diagonal
     Pixels $I_l[i]$ and $I_r[j]$ were matched.
     $Disparity\ left[i] = Disparity\ right[j] = abs(i-j)$
  2. $D(i-1, j)$ – Minimum step is vertical
     Pixel $I_l[i]$ is occluded in I_r $\rightarrow$ $Disparity\ left[i] = 0$
  3. $D(i, j-1)$ – Minimum step is horizontal
     Pixel $I_r[j]$ is occluded in I_l $\rightarrow$ $Disparity\ right[j] = 0$

- Finally, we plot the optimal path taken while computing the backwards pass for visualization.

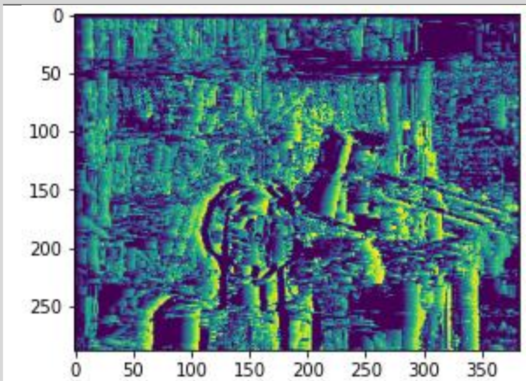
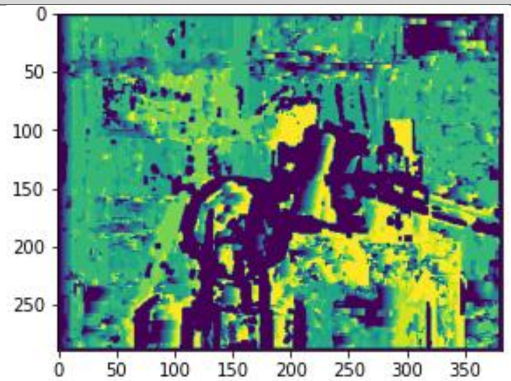
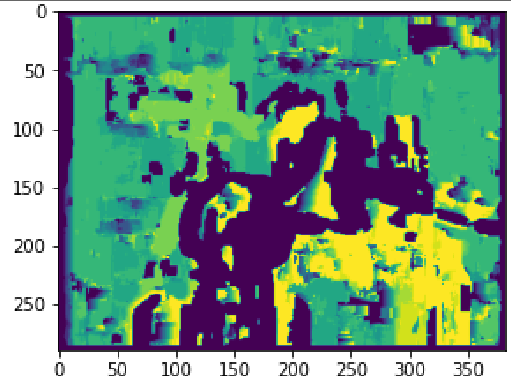


3. Comparing Results

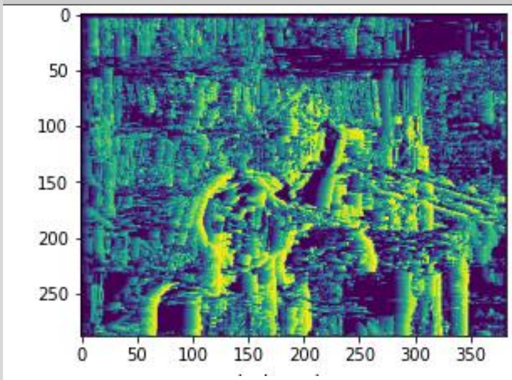
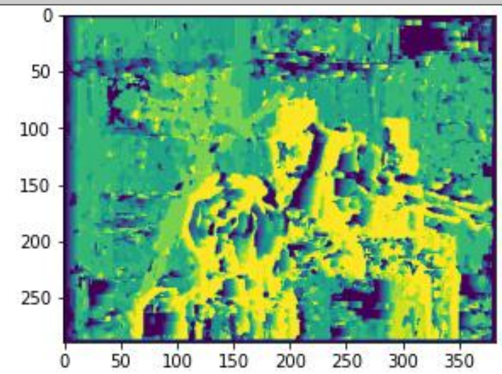
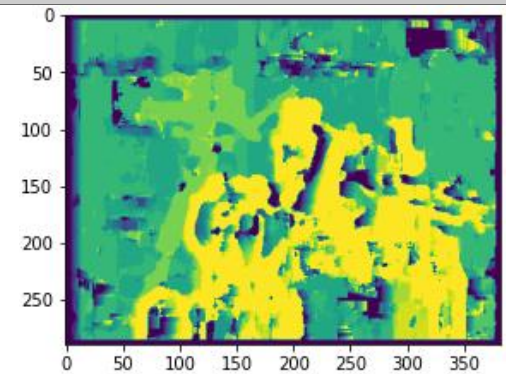
3.1 Image 1:



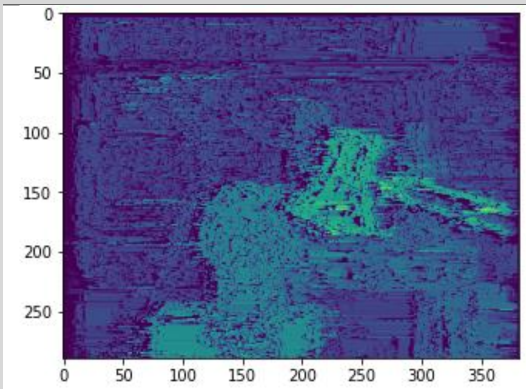
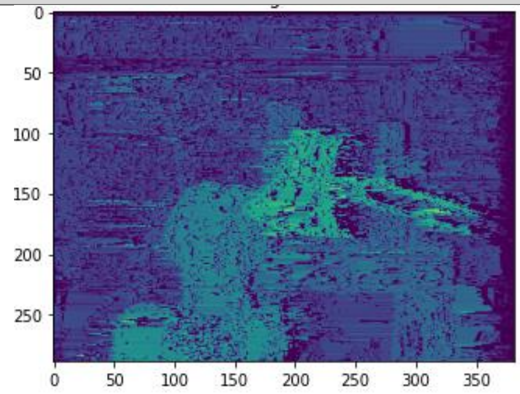
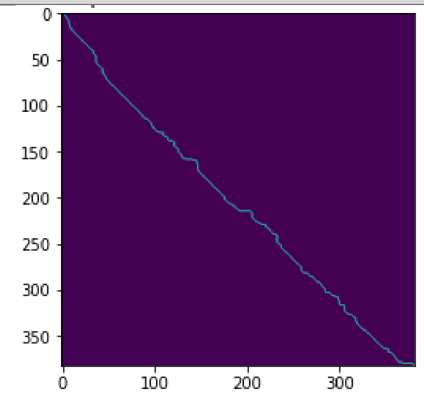
Block Matching: SSD

Window size = 1	Window size = 5	Window size = 9
		

Block Matching: SAD

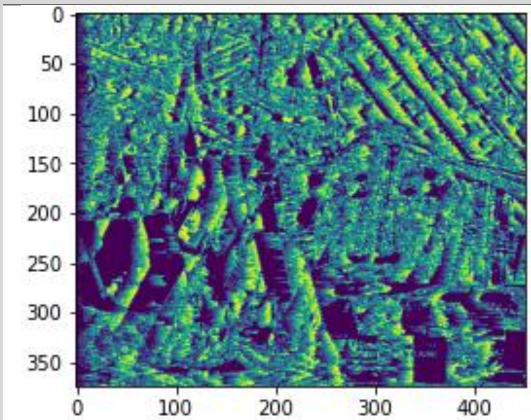
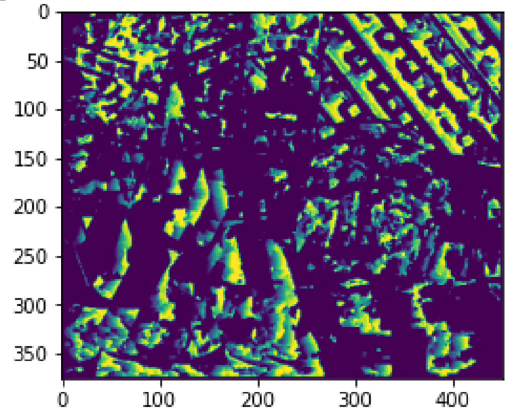
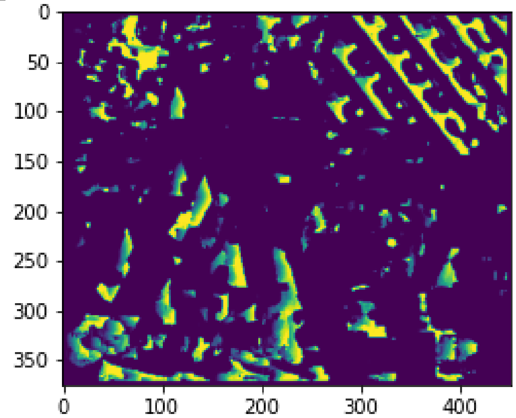
Window size = 1	Window size = 5	Window size = 9
		

Dynamic Programming

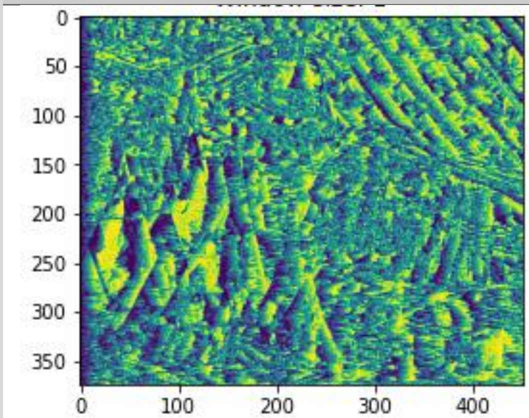
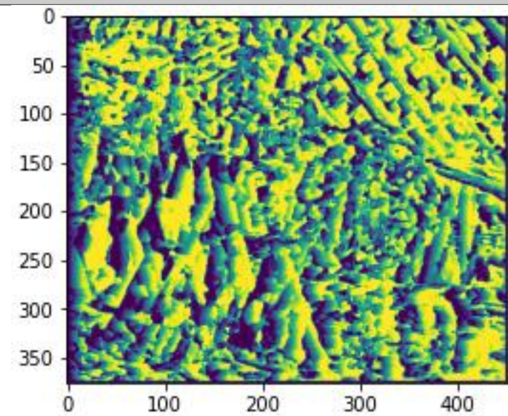
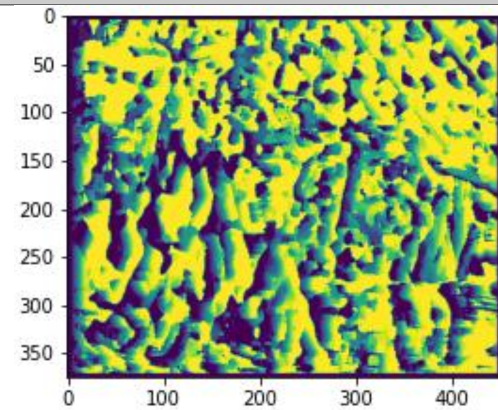
Left Disparity Map	Right Disparity Map	Optimal path for randomly selected scanline
		

3.2 Image 2:

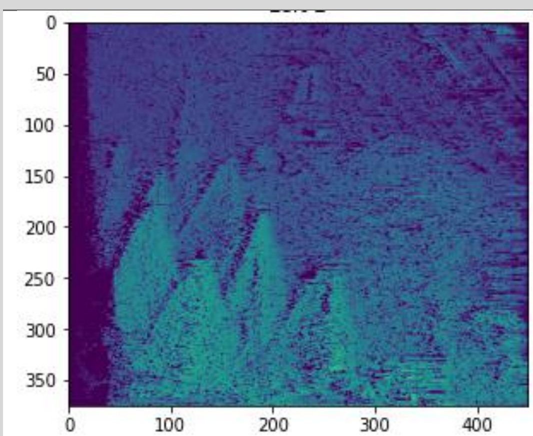
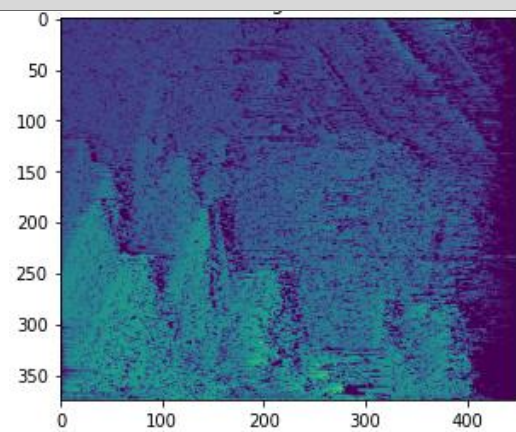
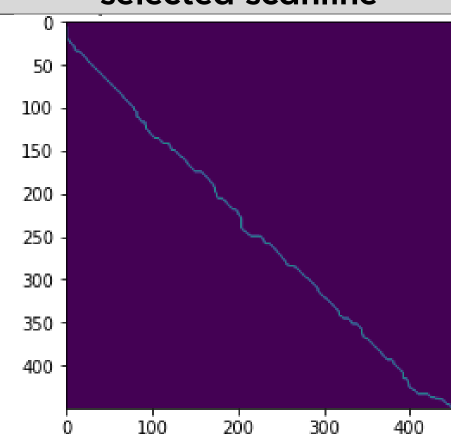
Block Matching: SSD

Window size = 1	Window size = 5	Window size = 9
		

Block Matching: SAD

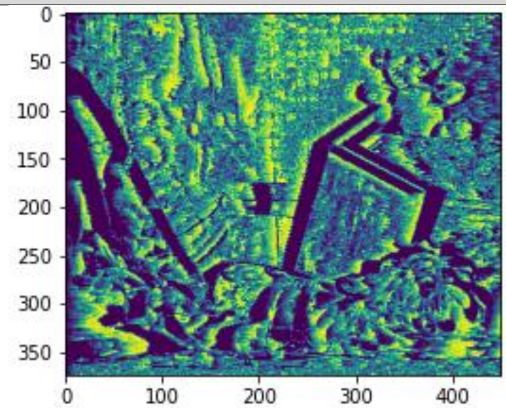
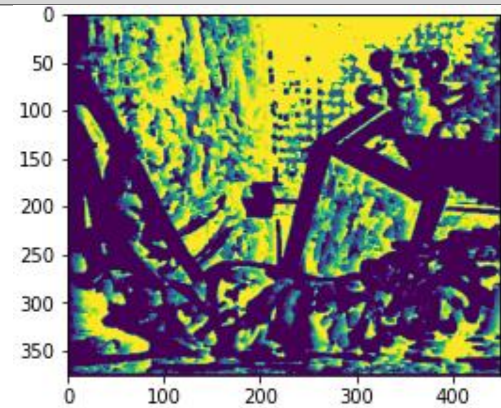
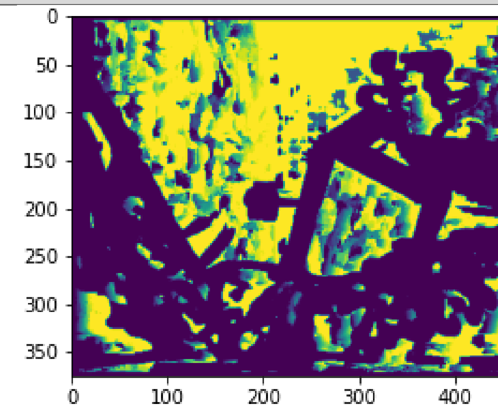
Window size = 1	Window size = 5	Window size = 9
		

Dynamic Programming

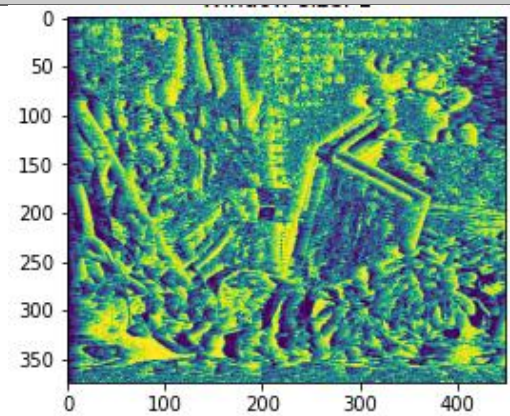
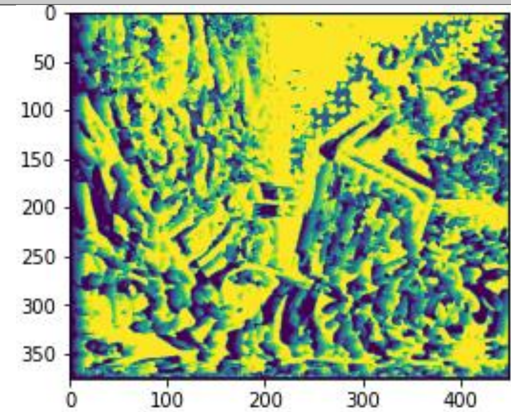
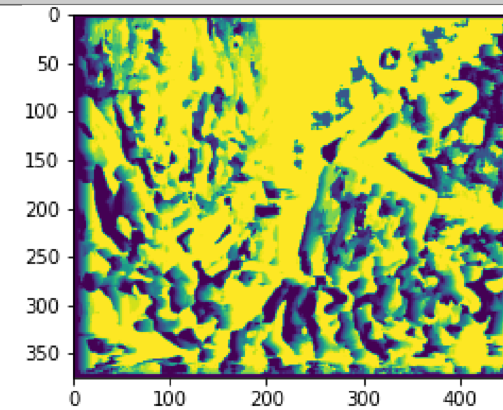
Left Disparity Map	Right Disparity Map	Optimal path for randomly selected scanline
		

3.3 Image 3:

Block Matching: SSD

Window size = 1	Window size = 5	Window size = 9
		

Block Matching: SAD

Window size = 1	Window size = 5	Window size = 9
		

Dynamic Programming

Left Disparity Map	Right Disparity Map	Optimal path for randomly selected scanline