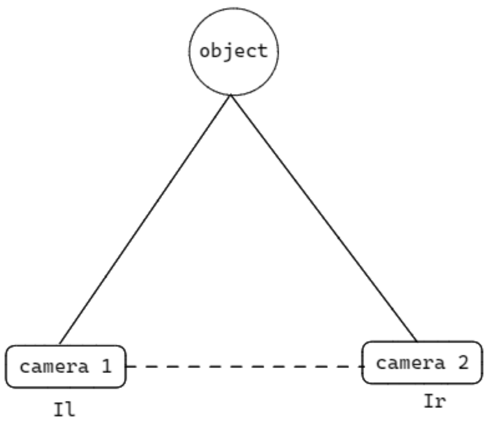
Computer Vision

Assignment 3

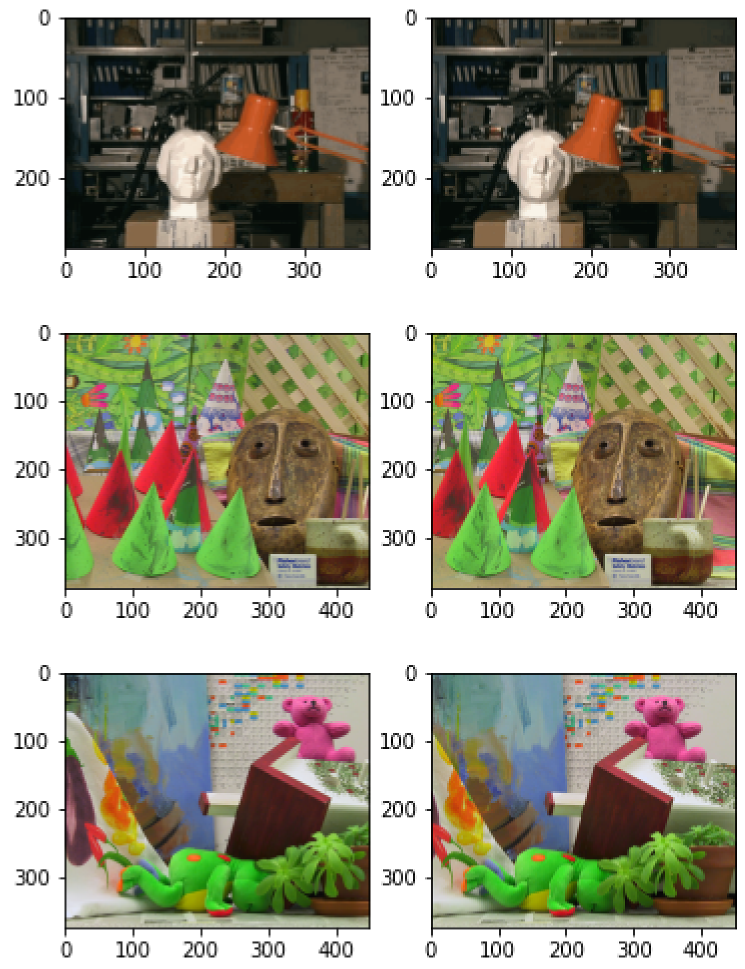
Nouran Ehab	6216
Salma Abulfetoh	6126
Heba Elwazzan	6521

[Link to Colab Notebook]

In this assignment, we are calculating the disparity between two images, taken by two cameras where the transformation between them is a horizontal translation along the x-axis, as shown in the fig.



Test Images



We have an image pair for each scene, taken by two cameras positioned side by side (or one camera that was shifted horizontally along the scene).

1 Block Matching

The first method is to match each pixel in the left image to a pixel in the right image using 2 different measures, Sum of Absolute Differences (SAD) and Sum of Squared Differences (SSD). We vary the size of the windows and use windows of sizes 1, 5, and 9.

SAD:

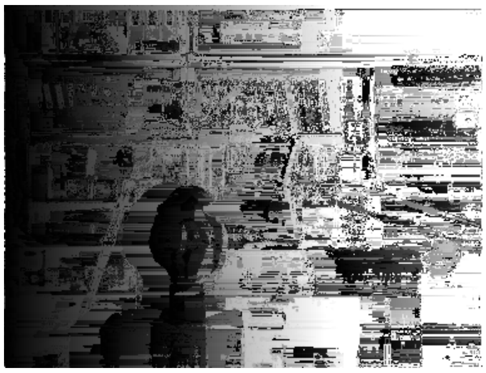
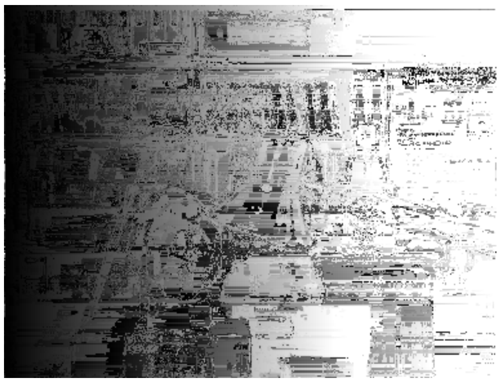
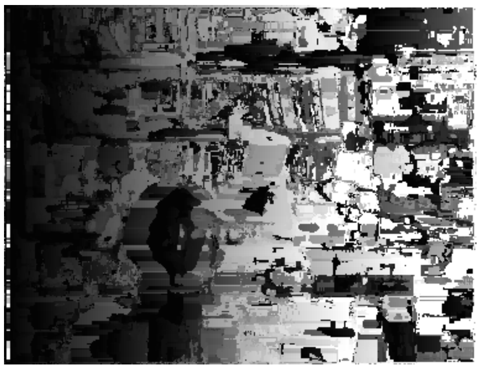
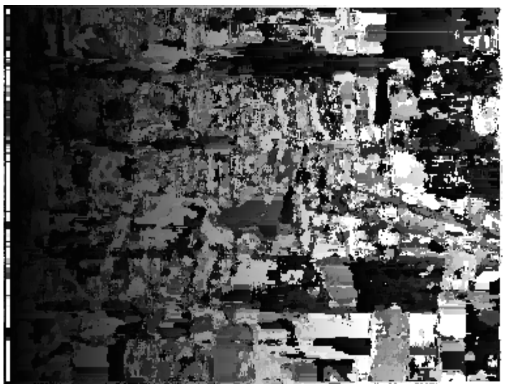
$$\sum |a_i - a_i|$$

SSD:

$$\sum (a_i - a_i)^2$$

We match each window pair in the first image with a sliding window in the second image, and calculate our measure. The window with the lowest disparity is then matched.

Image pair 1 results

W	SAD	SSD
1		
5		

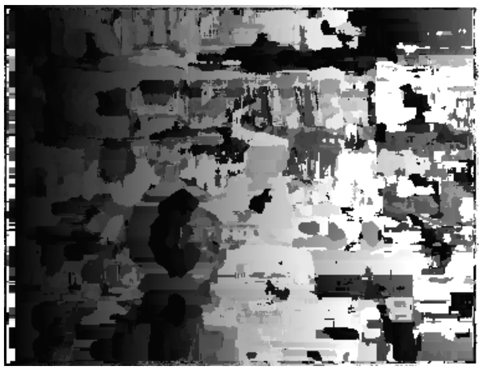
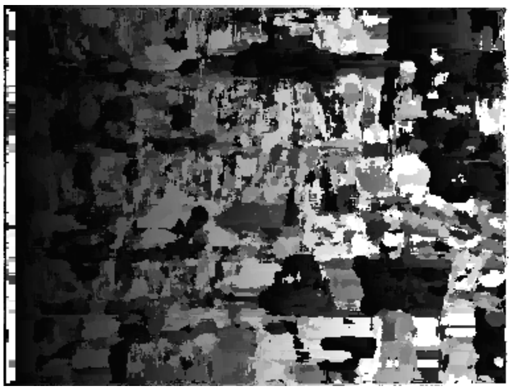
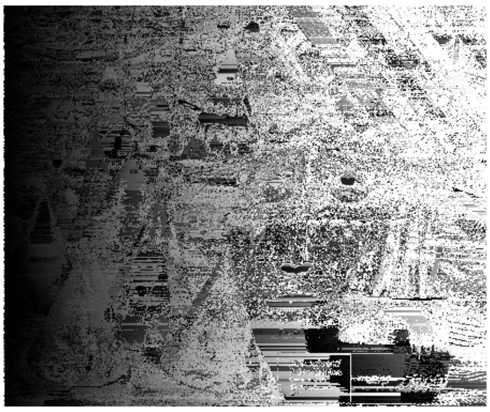
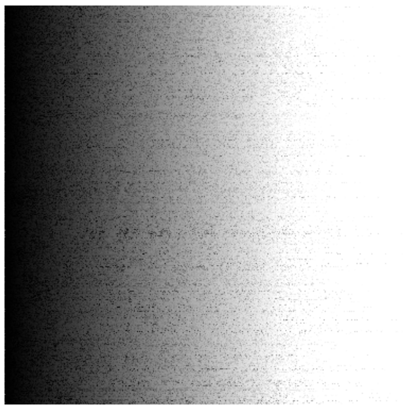
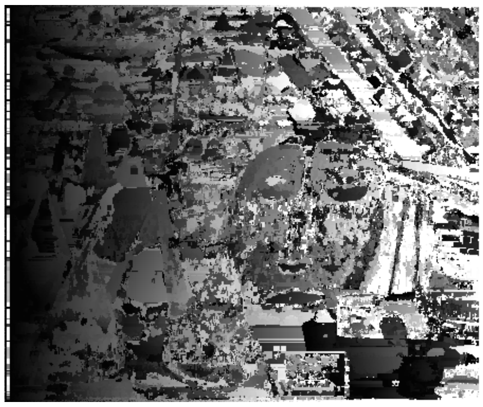
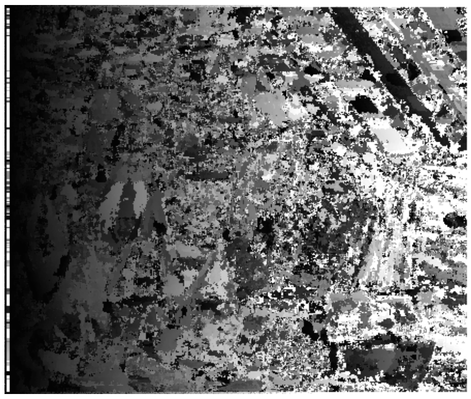
9		

Image pair 2 results

W	SAD	SSD
1		
5		

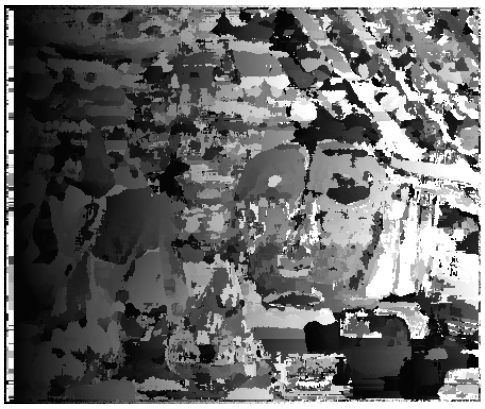
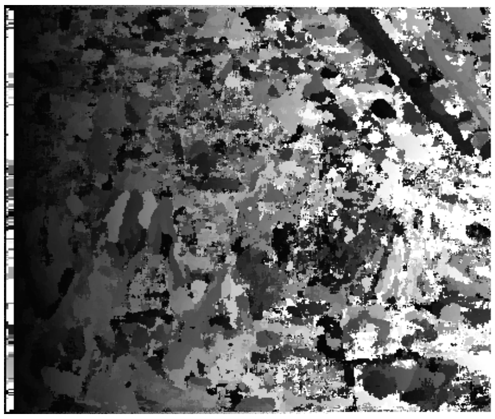
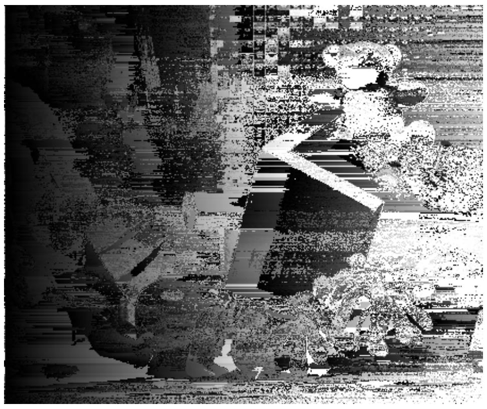
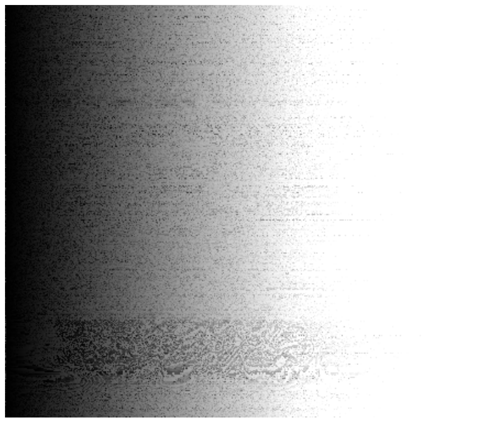
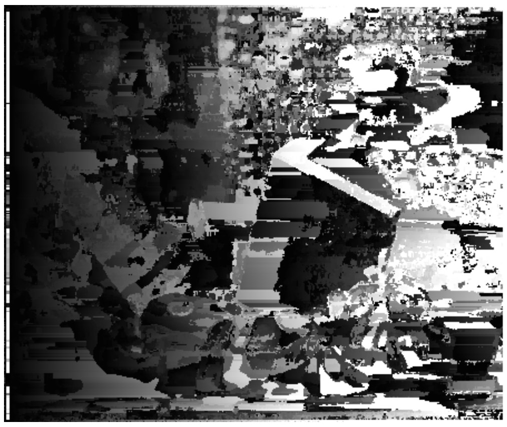
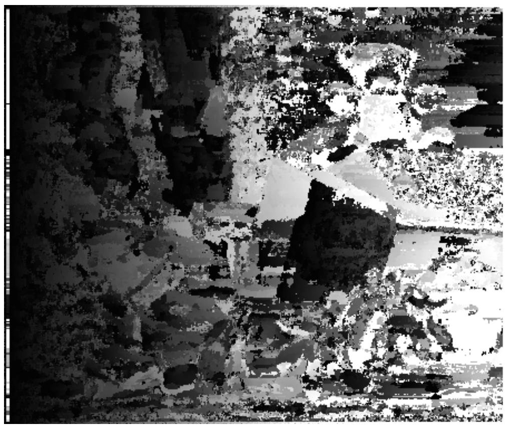
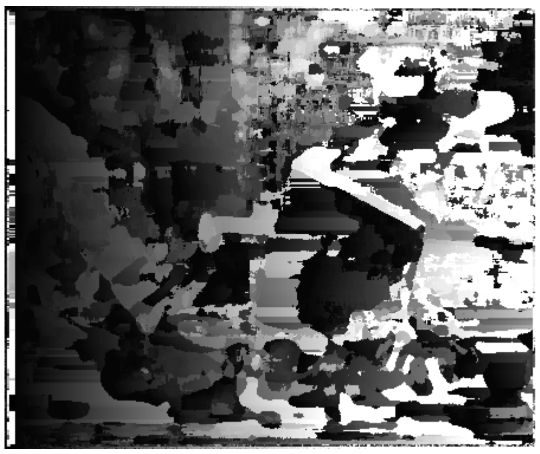
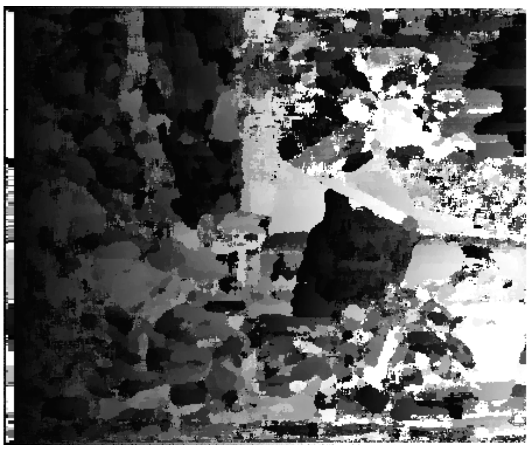
9		

Image pair 3 results

W	SAD	SSD
1		
5		

9		

The results are not very good as these winner-takes-all approaches fail the principle of having no two pixels matching to the same picture.
A better approach would be to consider all the possibilities all at once using our second method, dynamic programming.

2 Dynamic Programming

We will construct an NxN matrix for each scanline (row) where N=number of columns. The i dimension represents the pixels in the left image for this particular row, and the j dimension represents the pixels in the right image. We will work our way down from D(1,1). This matrix contains the cost associated with matching a particular pixel i in the left image to a pixel j in the right image. The cost is calculated using this equation:

$$D_{ij} = min(D(i - 1, j - 1) + d_{ij}, D(i - 1, j) + c_0, D(i, j - 1) + c_0)$$

where

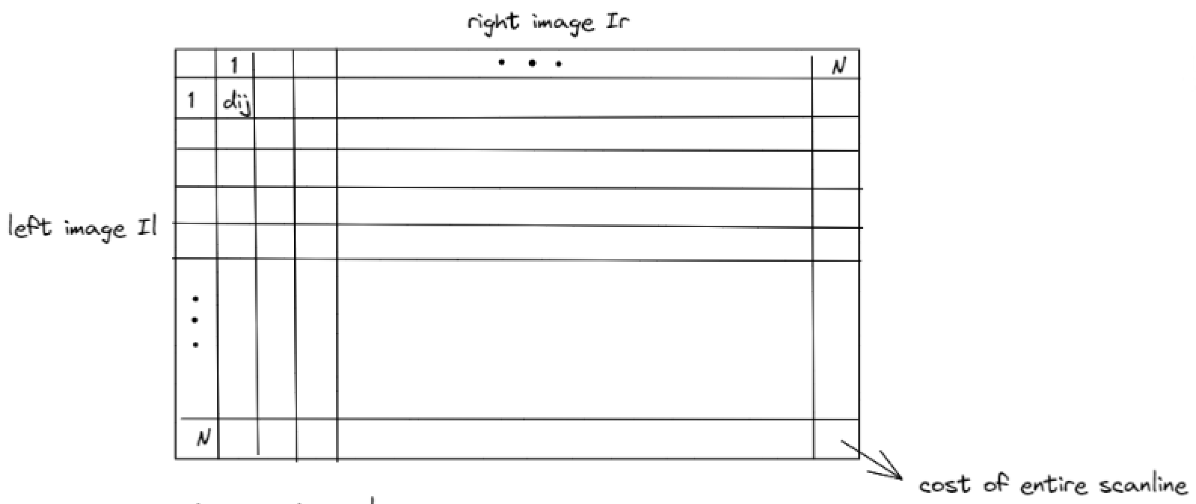
$$d_{ij} = \frac{(I_l(i) - I_r(j))^2}{\sigma^2}$$

and the initial $D(1, 1) = d_{11}$

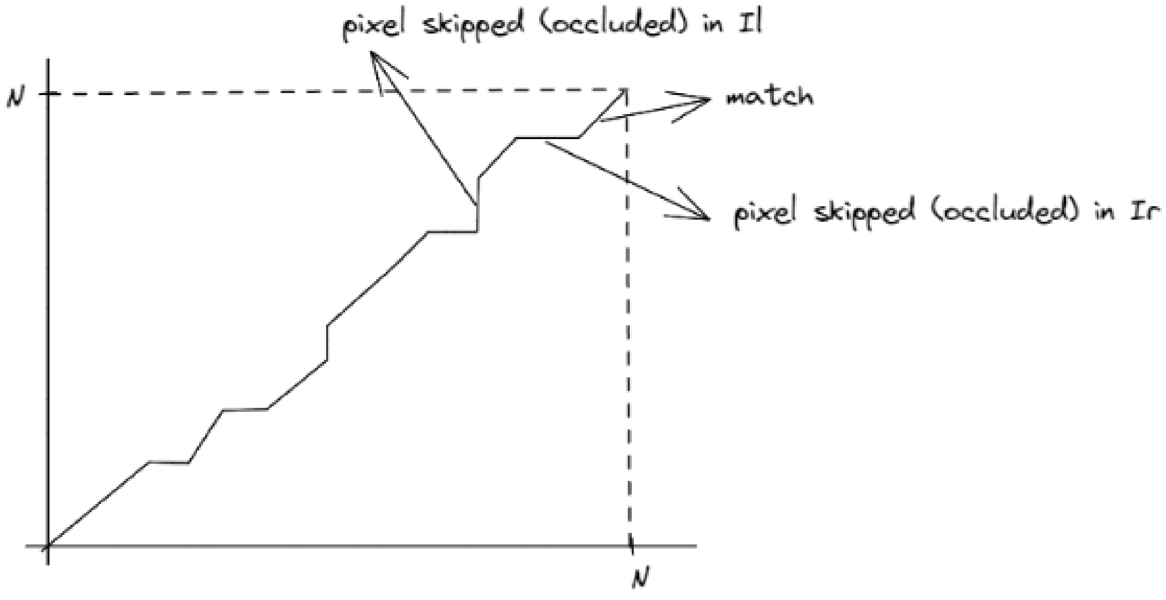
Along with the NxN matrix m that contains the optimal costs for matching each pixel, we can also construct another NxN matrix b that contains one of 3 values, 0, 1, 2. Where:

0 : denotes that the minimum cost was in the direction of moving (i-1, j-1) and indicates a match
1 : denotes that the minimum cost was in the direction of moving (i-1, j) and indicates that a pixel was skipped (occluded) in the right image
2 : denotes that minimum cost was in the direction of moving (i, j-1) and indicates that a pixel was skipped in the left image

So from b we can construct 2 disparity maps, one from the right image's perspective and one from the left's.



i = row, j = column


pixel skipped (occluded) in Il
N
match
pixel skipped (occluded) in Ir
N

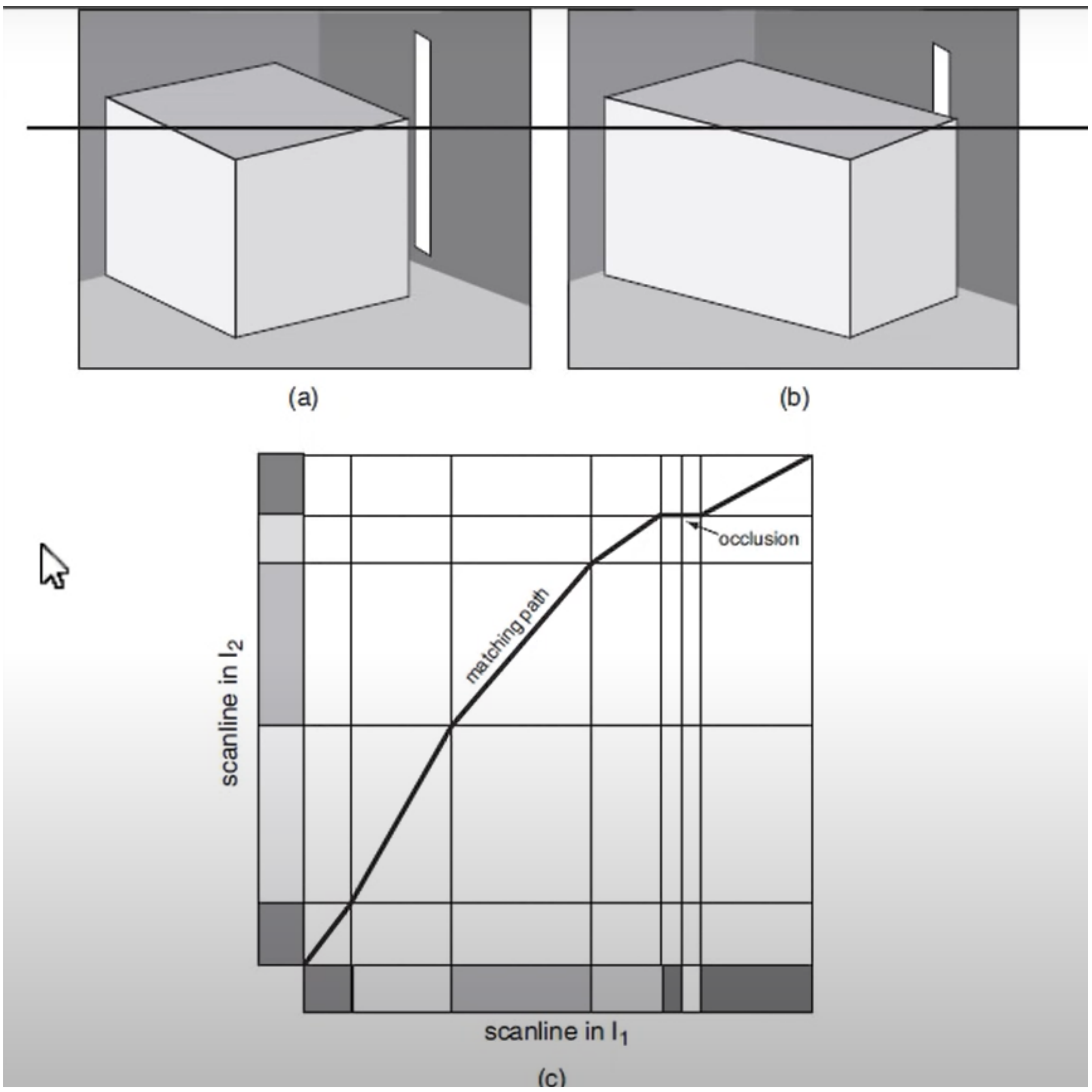


Image source: ECSE-6969 Computer Vision for Visual Effects, Rich Radke, Rensselaer Polytechnic Institute (video)

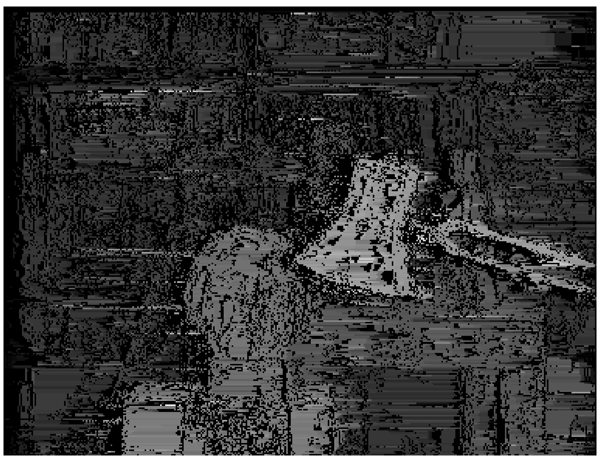
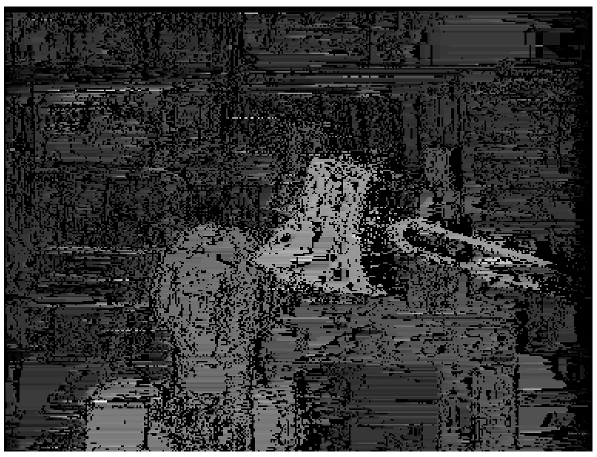
Code

The code is divided into several steps:
For each scanline (row of the image, since we're assuming the epipoles meet at infinity and so the scanlines match, ie. we will find a match in the other image if we just follow the scanline):

- Construct the cost matrix D for the scanline, along with a helper matrix b that represents if a match occurred, or if the pixel is occluded in either image.
- Moving from D(N,N), we compute the disparity map. If there is a match, we subtract the values of the indices of the pixels at which there was a match. Otherwise, the value is zero.

If we concatenate each disparity map we get from all the scanlines, we get the disparity map.

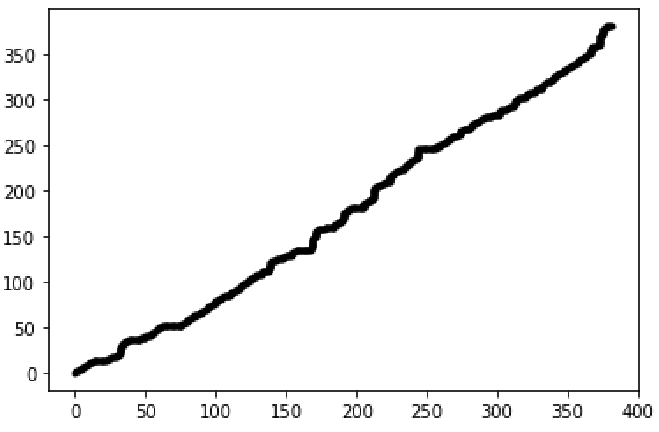
Results

Two disparity maps are shown, one from each camera’s perspective. As we can see, parts that were not matched appear as black, particularly at the edges of the image, where the other image has no corresponding pixels as it was out of view.

Bonus

If we plot the path taken for one of the rows (the last row was used here), denoting a match with a diagonal, an occluded pixel in the left image with a horizontal line, and an occluded pixel in the right image with a vertical line, we can visualize our dynamic programming algorithm results as such:


0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350
400