CORRELATION MATCHING AND DISPARITY MAP FOR STEREO VISION

DECEMBER 14, 2021

TA's: Muhammed Zemzemoğlu and Mehmet Emin Mumcuoğlu

Name of Student: Moses Chuka Ebere
Student Number: 31491
Course: Computer Vision (EE 417) Lab

TABLE OF CONTENTS

INTRODUCTION

Stereo vision basically entails the ability to infer information on the 3D structure and distance of a scene from two or more images taken from different viewpoints.

As much as stereo vision helps us recover depth of points or objects in a scene, it introduces another issue of disparity. Given that the two (or more images) were taken from different viewpoints, corresponding points in the mage planes will appear displaced. This notion is called disparity. Although it appears problematic, it is key in depth-recovery.

In this lab, the focus is on using a correlation-based algorithm to obtain the disparity map of a stereo pair of images.

EXPLANATION OF METHODS

CORRELATION-BASED ALGORITHM

This algorithm is also termed template matching in that it requires us to move a template (or window) in the reference image of a stereo pair over a search window in the second image. While doing this, one of several similarity measures is computed and maximized/minimized (depending on the similarity measure selected). The advantage with this algorithm is that it produces a dense set of correspondences.

In this lab, the Sum of Squared difference is applied as the similarity measure.

$$C(d) = \sum_{k=-W}^{k=W} \sum_{l=-W}^{l=W} \Psi\Big(f(i+k, j+l),\ g(i+k-d_1, j+l-d_2) \Big) \tag{1}$$

where Ψ is the similarity measure such as SSD which can be calculated as follows:

$$SSD = \sum_{k=-W}^{k=W} \sum_{l=-W}^{l=W} [f(i+k, j+l) - g(i+k, j+l)]^2 \tag{2}$$

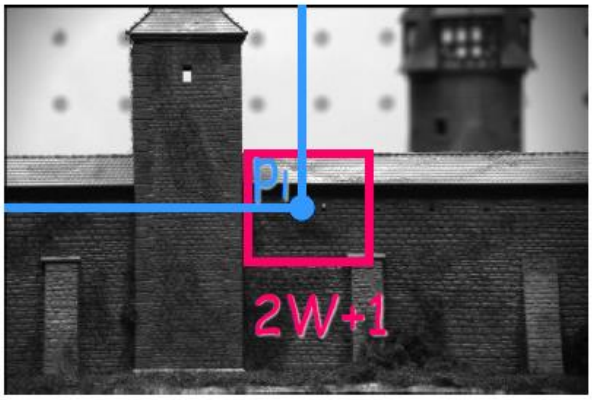
To compute the above, the following need to be defined:

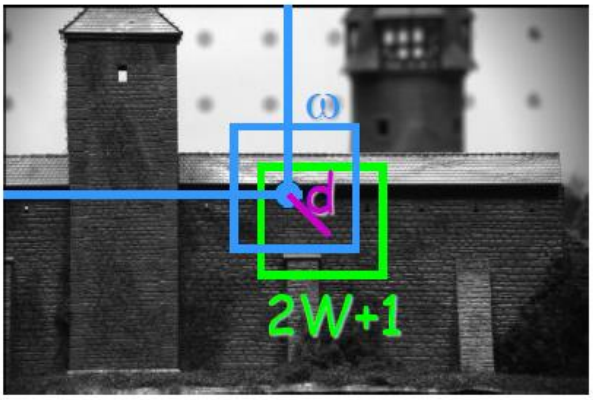
- Input images: left and right images of the same scene.
- Correlation window size – 2W+1
- Search window size – ω
- Similarity measure, Ψ

The steps for the algorithm are as follow:

- Find a point in the left image, p_l and
- Pass the corresponding window over the search window in the right image and compute SSD for each pixel.
- Minimize the SSD.
- The minimum value is recorded as the corresponding point in the right image.

- Calculate the displacement vector of the corresponding point from the original point and record the value. This is for the disparity map
- Repeat the above process for every pixel in the left image.
- Display the disparity map.
- calculate the apparent motion of brightness patterns, local deformation models like translational model, affine model, transformation of intensity values and occlusions, etc., could be used.

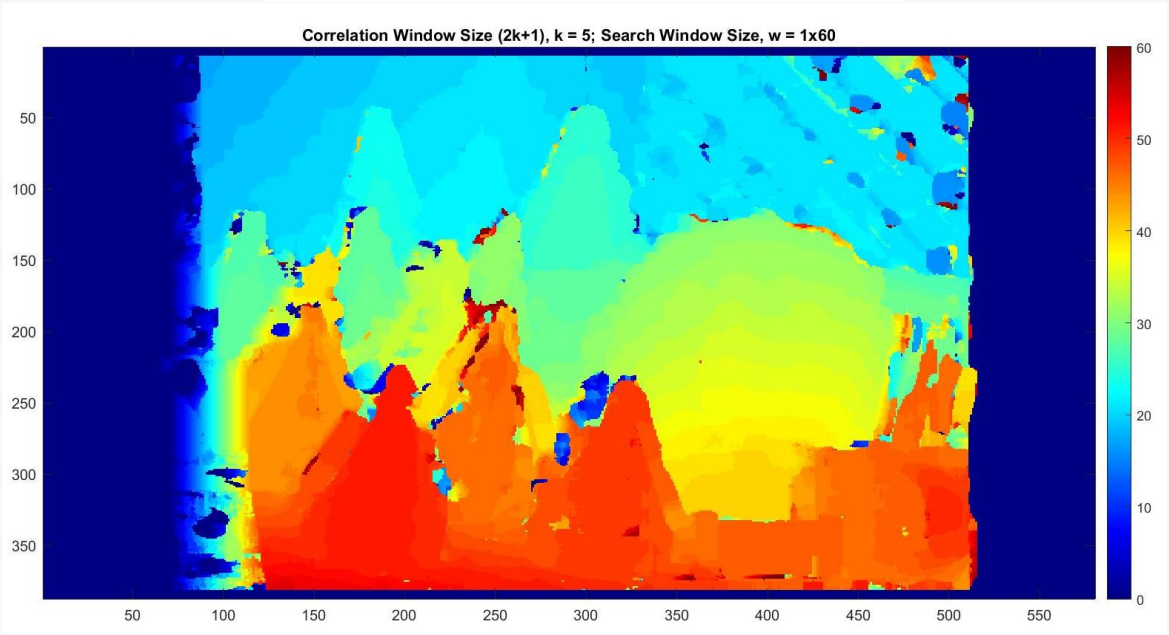




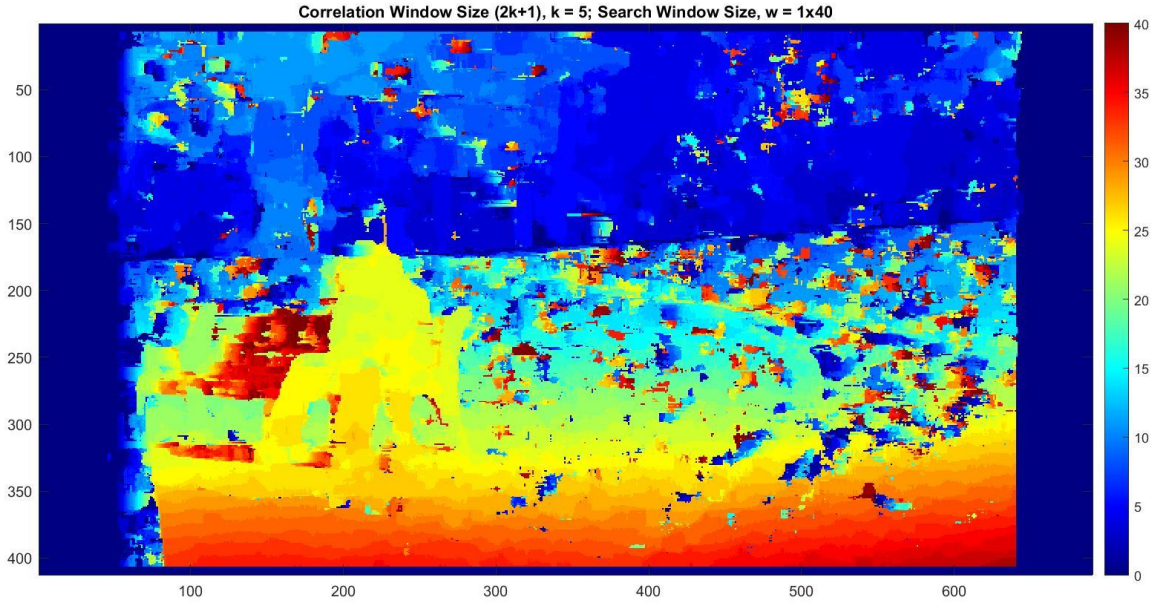
IN-LAB RESULTS

S00


Fisherbrand
Safety Matches

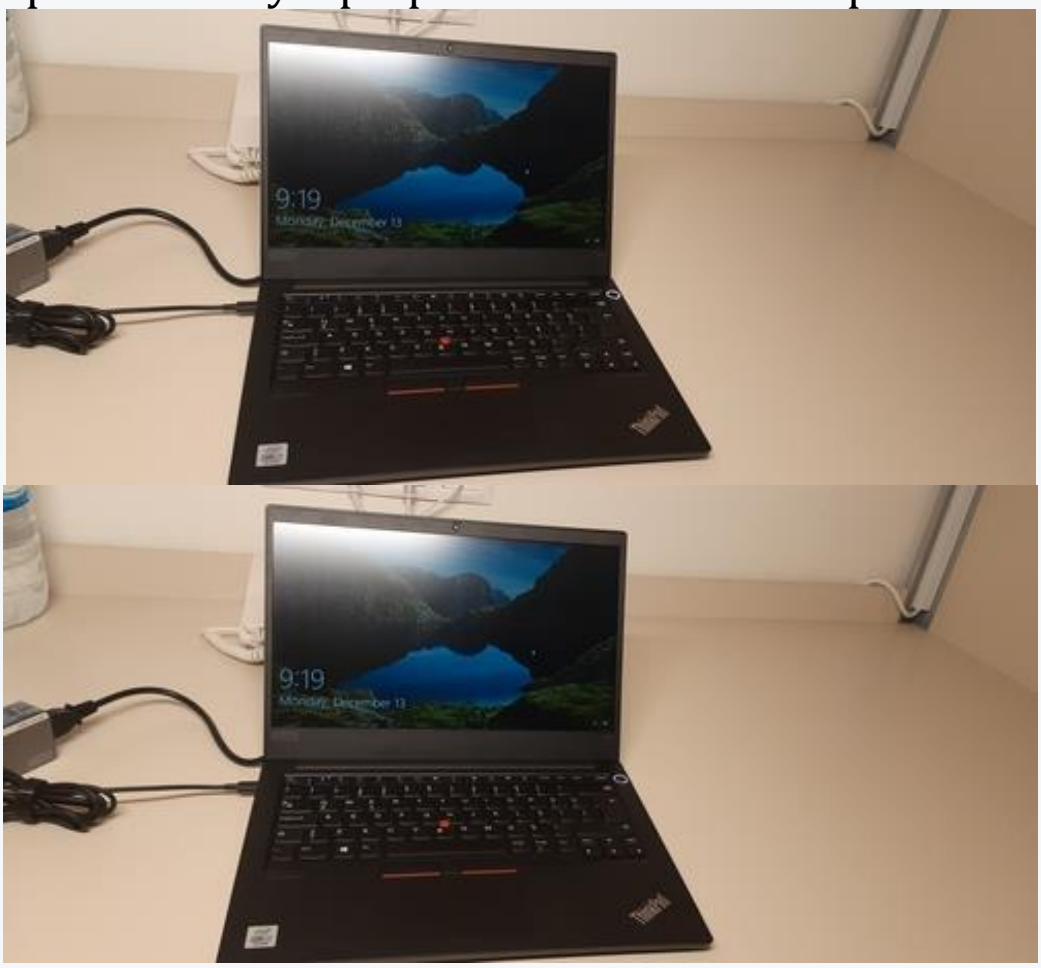

Correlation Window Size (2k+1), k = 5; Search Window Size, w = 1x60
50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
450
500
550
60
50
40
30
20
10
0

S01

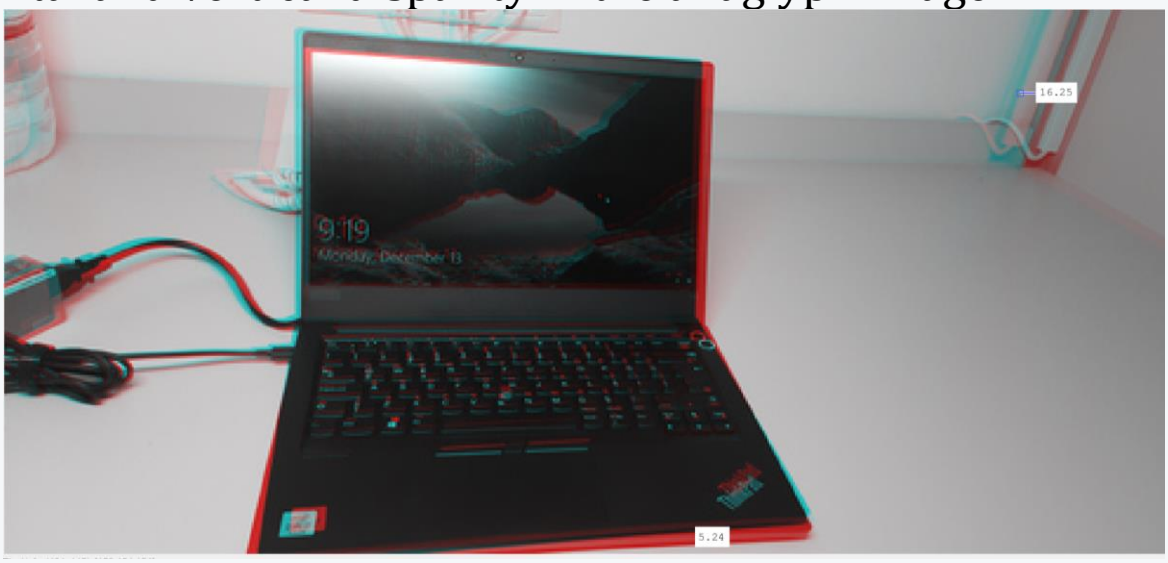

Correlation Window Size (2k+1), k = 5; Search Window Size, w = 1x40
50
100
150
200
250
300
350
400
100
200
300
400
500
600
40
35
30
25
20
15
10
5
0

MULTIPLE RESULTS

I took two photos of my laptop to serve as a stereo pair:

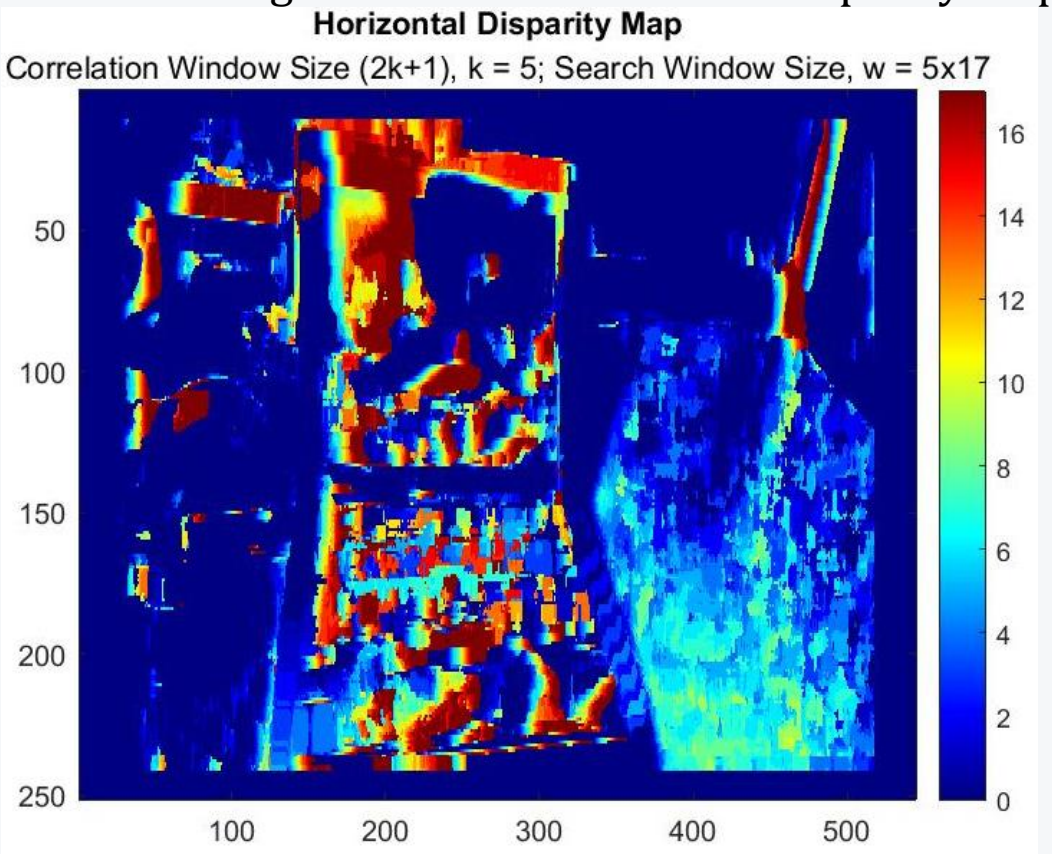


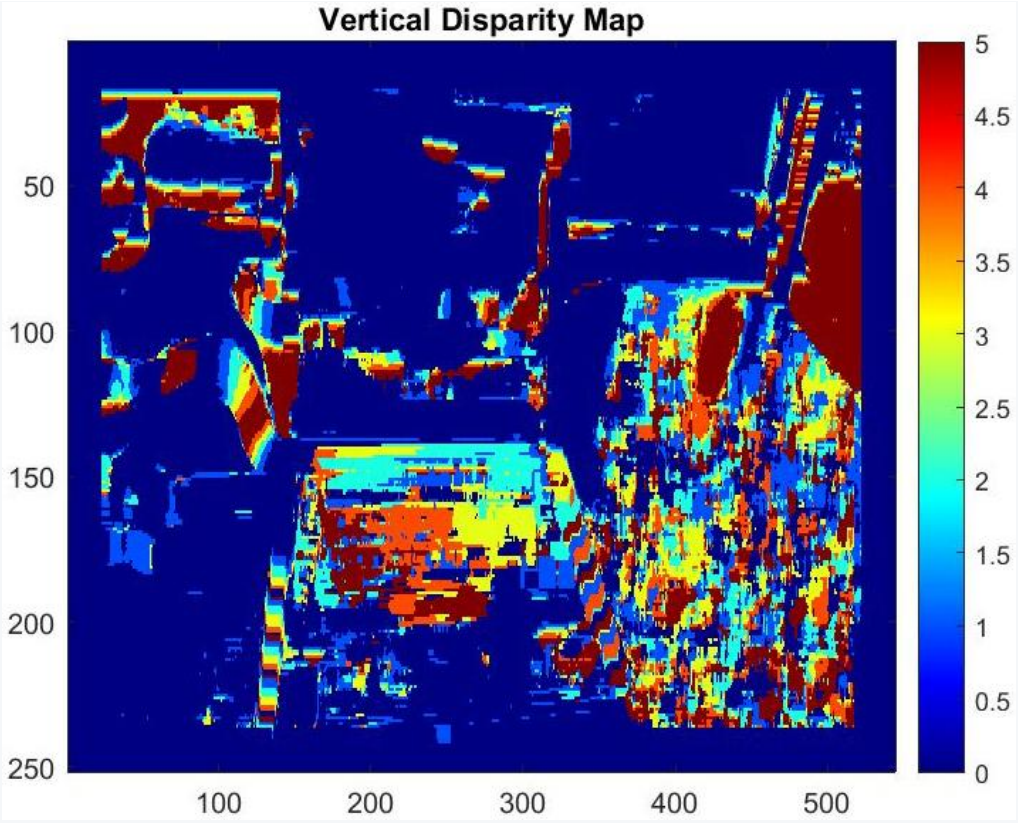
Using the distance tool in the Image Viewer App, I estimated the largest horizontal and vertical disparity in the anaglyph image:



Horizontal ~ 17; Vertical ~ 5. I used this as the size of my search window.

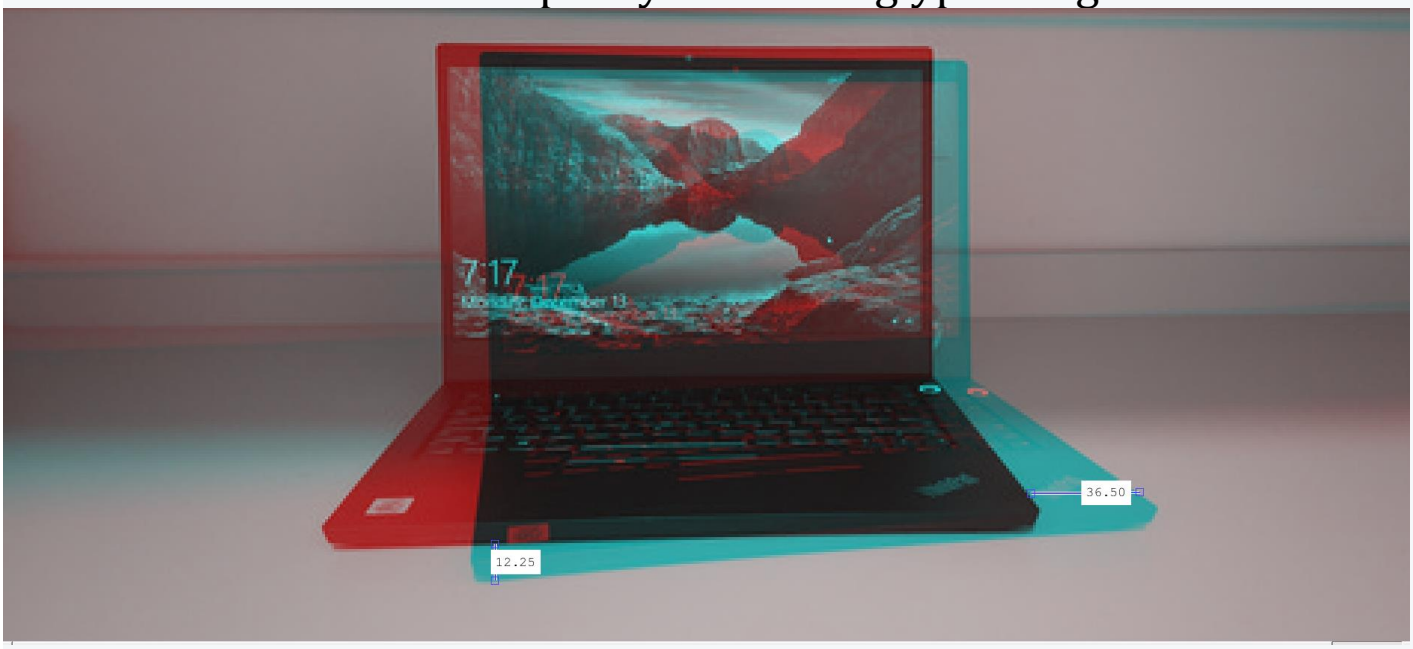
I obtained the following horizontal and vertical disparity map:



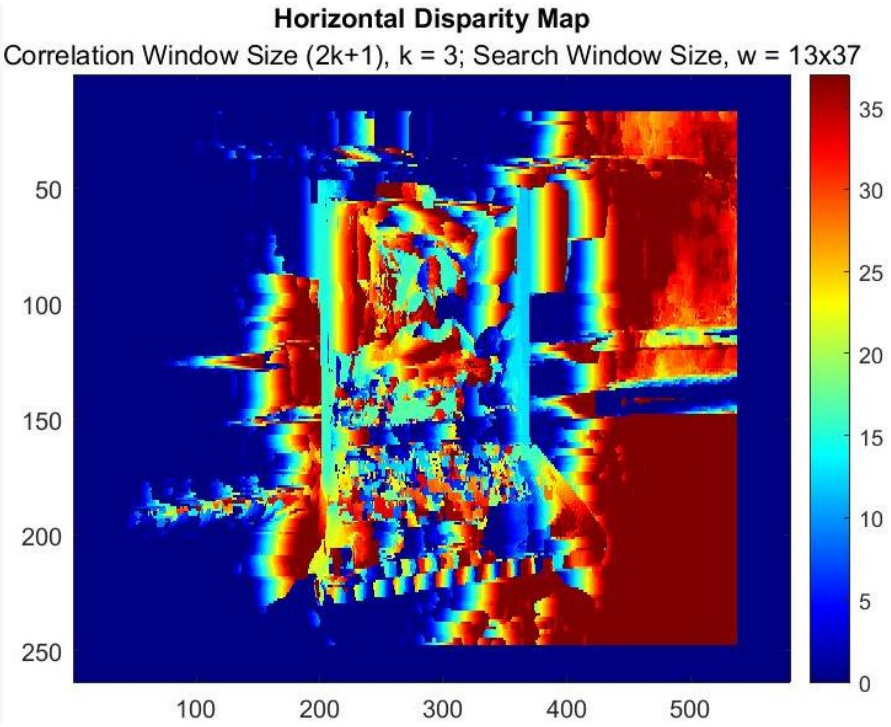


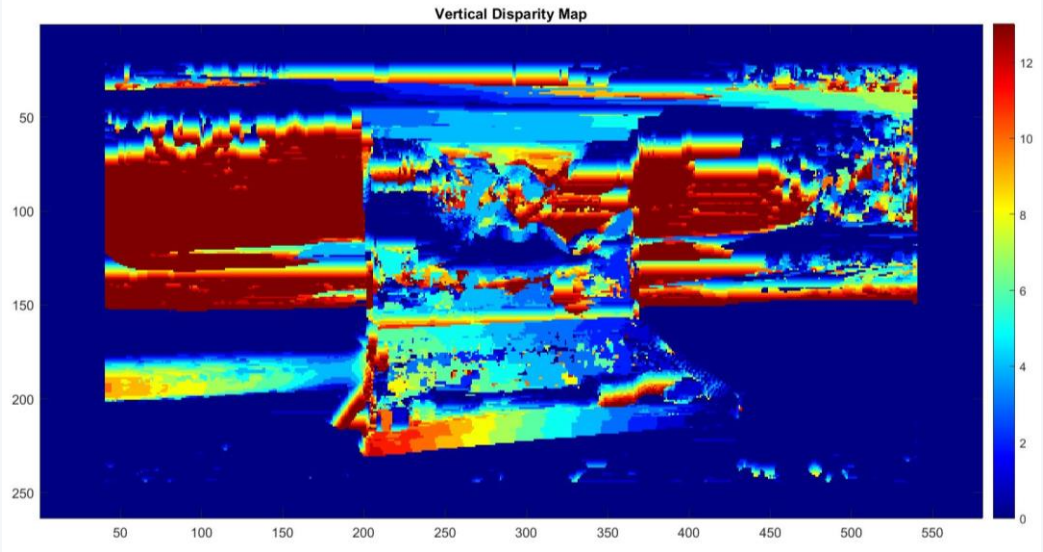
I took two more photos of my laptop to serve as a stereo pair:

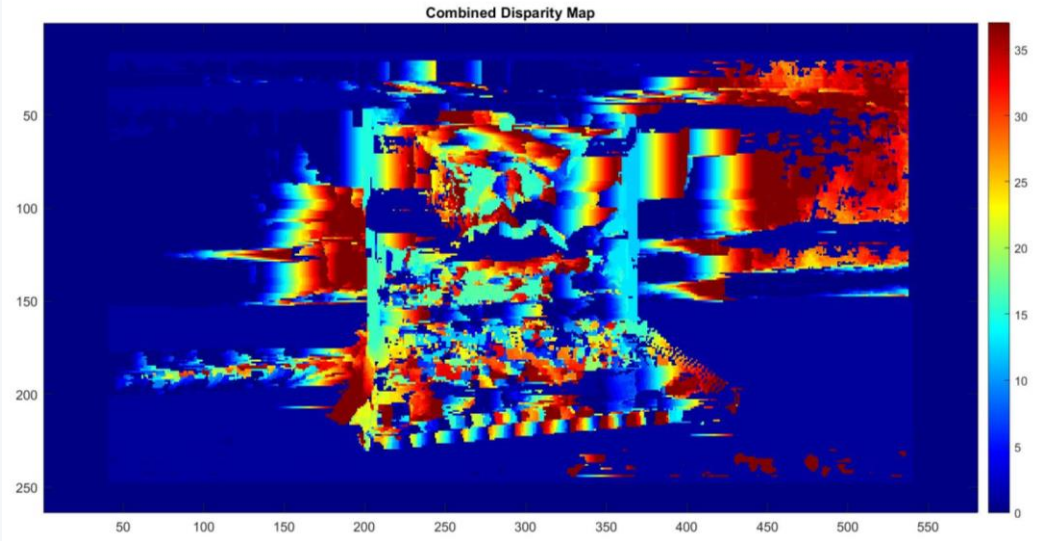
Using the distance tool in the Image Viewer App, I estimated the largest horizontal and vertical disparity in the anaglyph image:



Horizontal ~ 37; Vertical ~ 13. I used this as the size of my search window.


Horizontal Disparity Map
Correlation Window Size (2k+1), k = 3; Search Window Size, w = 13x37
50
100
150
200
250
100
200
300
400
500
35
30
25
20
15
10
5
0

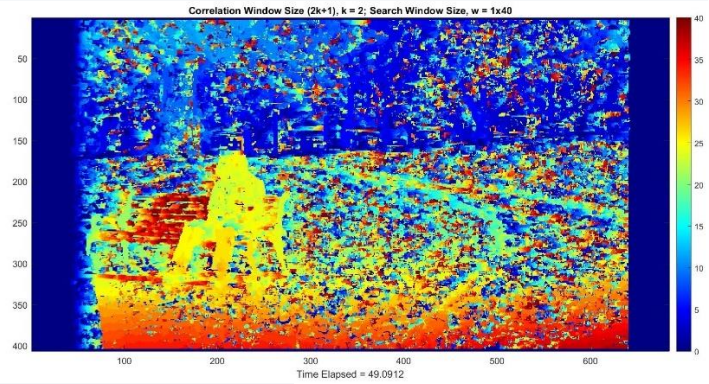

Vertical Disparity Map
50
100
150
200
250
50
100
150
200
250
300
350
400
450
500
550
12
10
8
6
4
2
0

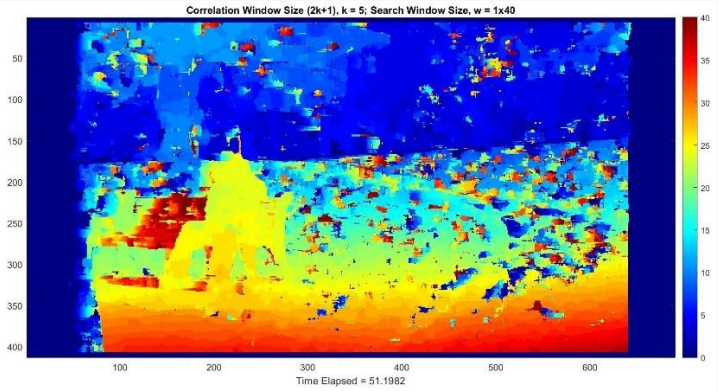

Combined Disparity Map
50
100
150
200
250
50
100
150
200
250
300
350
400
450
500
550
35
30
25
20
15
10
5
0

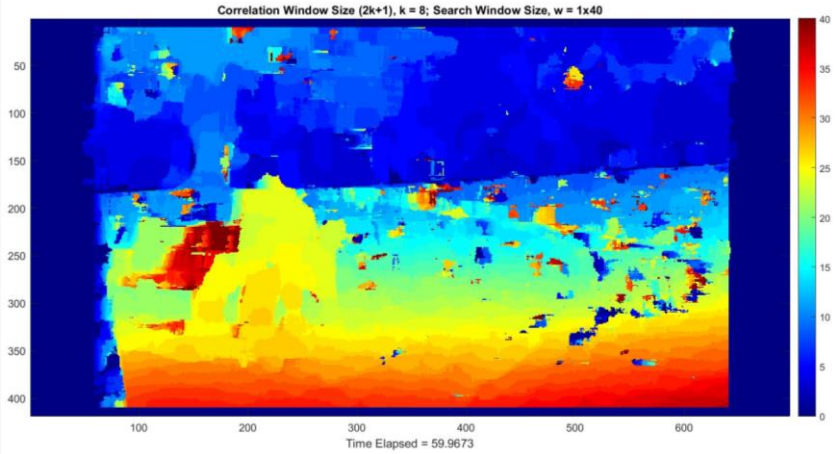
POST-LAB QUESTIONS

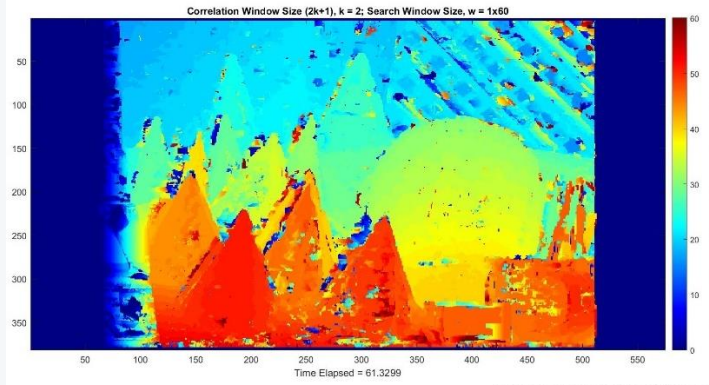
Results with Different Window Sizes and Search Areas

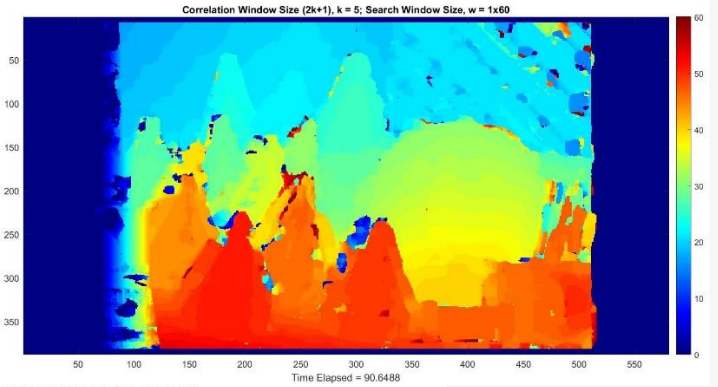
Changing the Correlation Window size with Constant Search Area

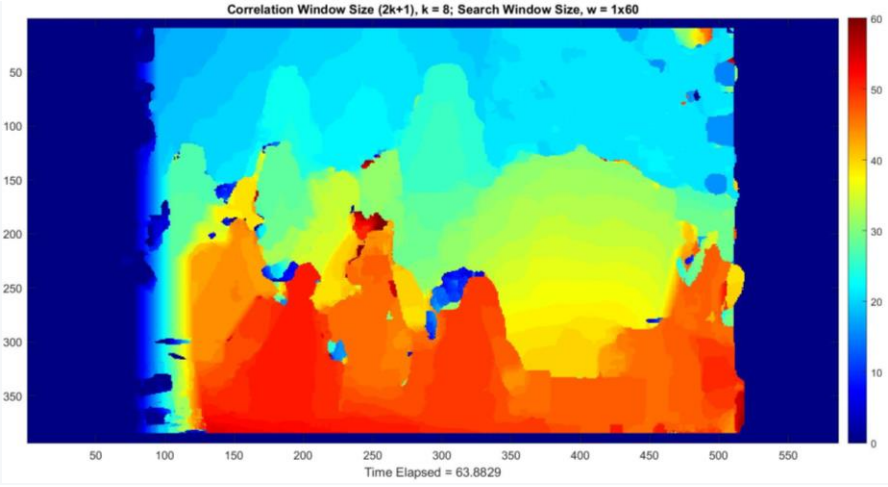






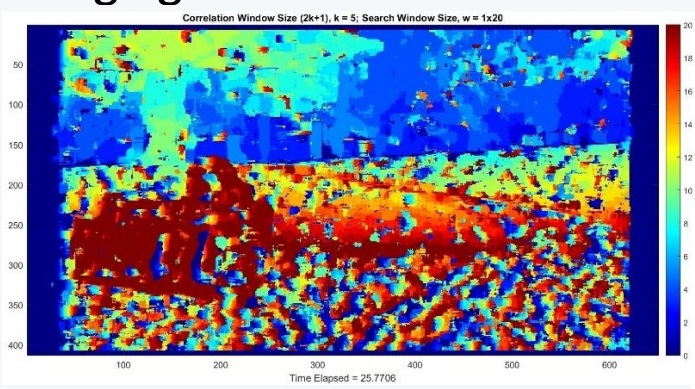


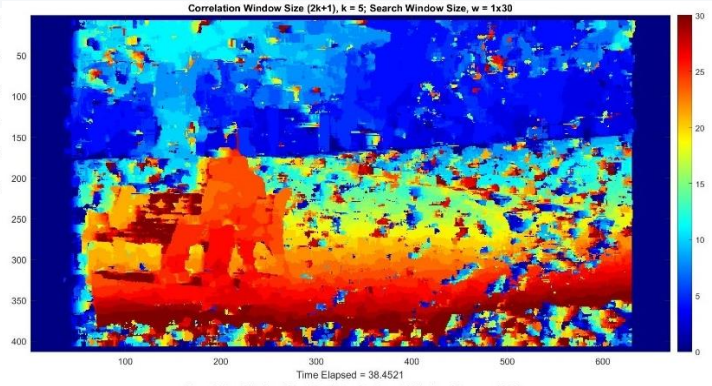


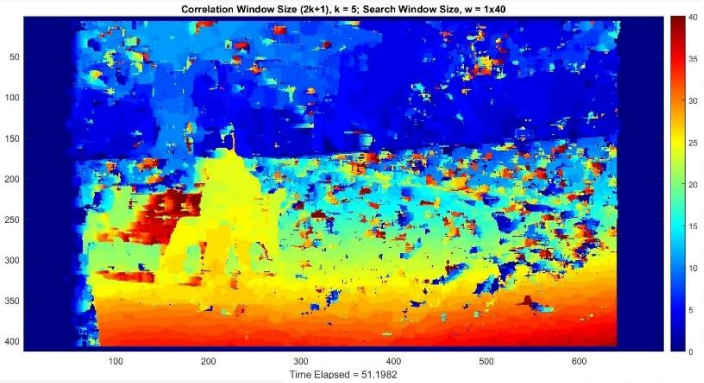


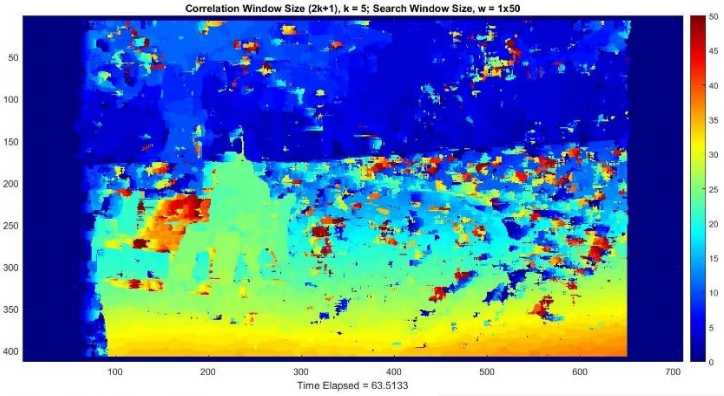
- The first observation is that the time taken to compute the result increases as we increase the correlation window size, k. (I used the tic toc function to compute the time).
- We observe that increasing the correlation window size sort reduces the number of noisy disparities or disparities that are affected by projective distortion.
- It's fair to say that having prior knowledge of the positioning of the stereo cameras or the baseline would be helpful in determining the optimal correlation window size.

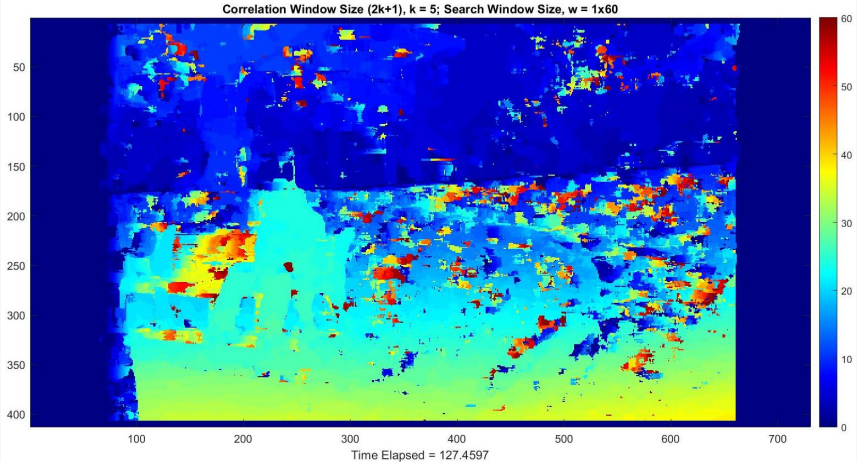
Changing the Search Area with Constant Correlation Window Size

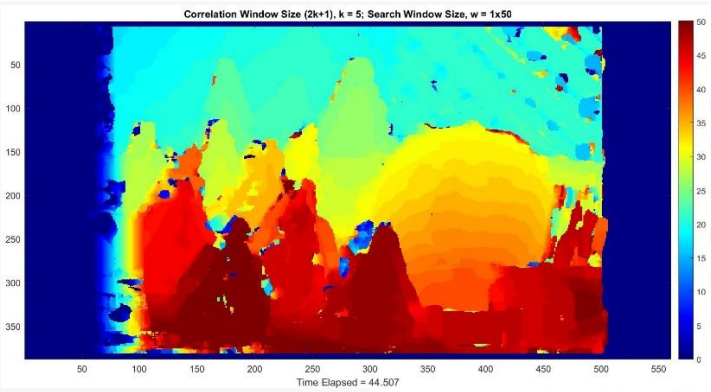


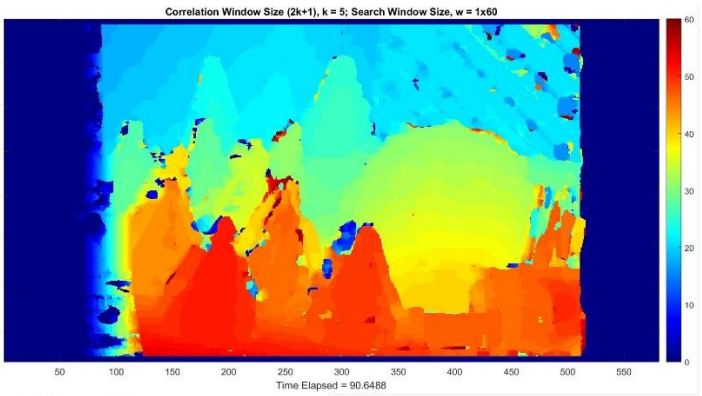


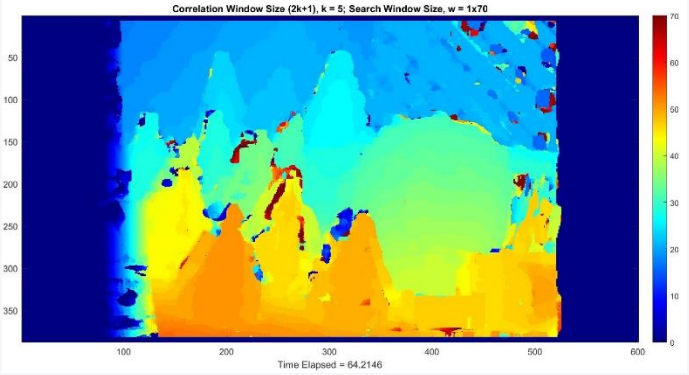






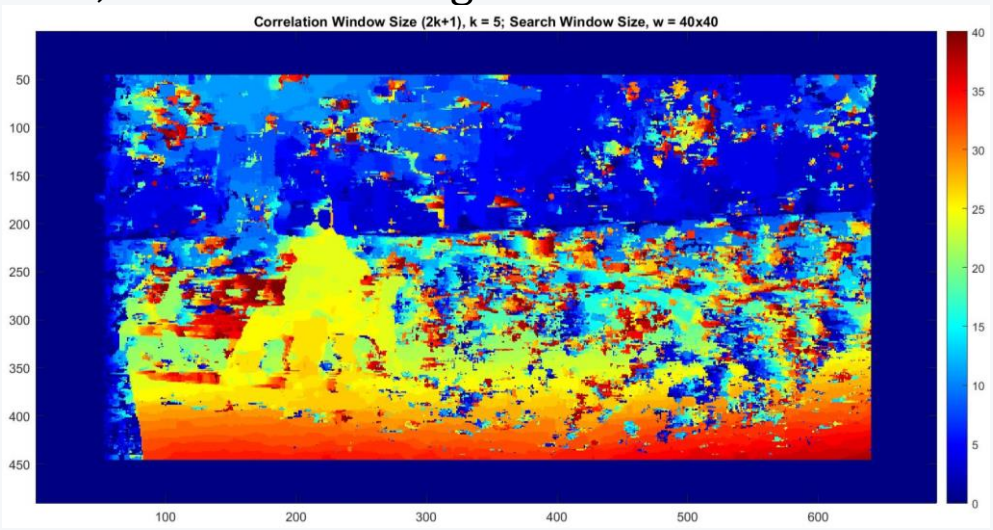






- Similar to the previous case, the computation time increases as we increase the size of the search area.
- One strange observation is that the disparities obtained generally decrease as we increase the size of the search area. Notice how the image has more regions with blue patches as w increases.

I decided to try a large square search area to see what the outcome would look like, and the following was obtained:



- As one would guess, this took a really long time to compute (about half-an-hour to be precise.

How to Choose the Sizes of Sub-images and Search Windows

Procedure I follow

- A relatively easy method would be to display the stereo anaglyph of the input images in the **Image Viewer App** and use the Distance tool to measure distances between pairs of corresponding points. This goes a long way in giving us relevant info for the window sizes.

Code: imtool(stereoAnaglyph(ImLeft,ImRight));

In the above case, I used the distance tool to manually measure the disparity. I immediately observed that the disparity is higher for objects closer to the camera (as expected), so my choice of a search window would be based on the higher distances. I can decide to choose a search window of size = 50 along each row (for the image on the left).

Furthermore, I found the following quote from a 1994 paper by T. Kanade:

"A central problem in stereo matching by computing correlation or sum of squared differences (SSD) lies in selecting an appropriate window size. The window size must be large enough to include enough intensity variation for reliable matching, but small enough to avoid the effects of projective distortion. If the window is too small and does not cover enough intensity variation, it gives a poor

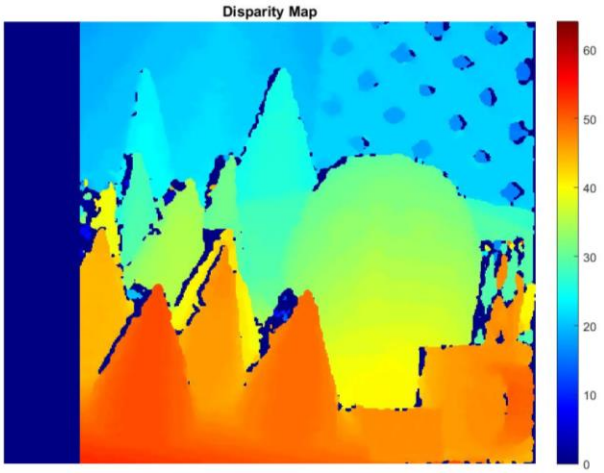
disparity estimate, because the signal (intensity variation) to noise ratio is low. If, on the other hand, the window is too large and covers a region in which the depth of scene points (i.e., disparity) varies, then the position of maximum correlation or minimum SSD may not represent correct matching due to different projective distortions in the left and right images. For this reason, a window size must be selected adaptively depending on local variations of intensity and disparity."

Due to the constraints of this lab, the size of sub-images and search windows are chosen by **trial-and-error**. By simply observing the resulting image, it is not so difficult to identify whether or not the result was affected by noise, projective distortion, etc.
For example, in some of the above outputs, notice how there are excessive random patches of alternating colors in small regions. This signals that our choice of window size is not optimal.

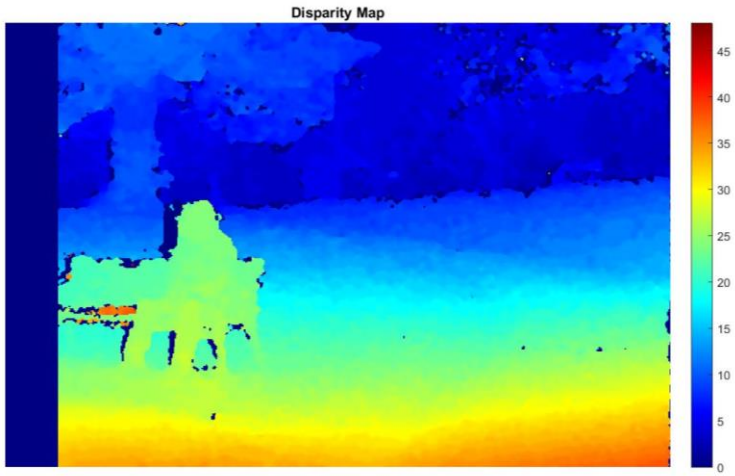
Other Pieces of Information that Could Help in Selecting the Search Area

- *The search area depends on the distance between the two cameras and the distance between the cameras and the object of interest. Increase the search area when the cameras are far apart, or the objects are close to the cameras.*
- *Also, if the camera used to take left image was to the right of the camera used to take right image, we would be better off selecting a search window that moves towards the left.*

Results with MATLAB's In-built Disparity Function



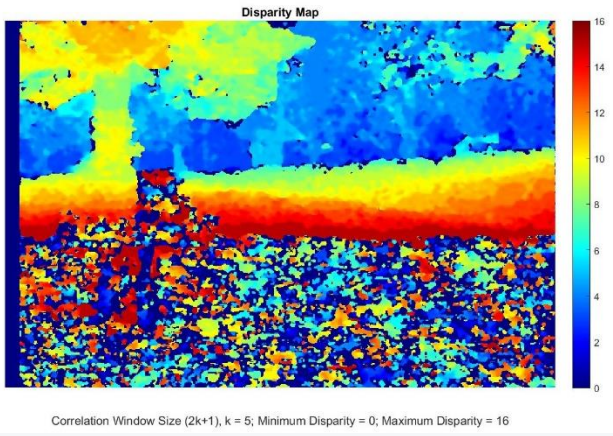
Correlation Window Size (2k+1), k = 5; Minimum Disparity = 0; Maximum Disparity = 64



Correlation Window Size (2k+1), k = 5; Minimum Disparity = 0; Maximum Disparity = 48

Comparison

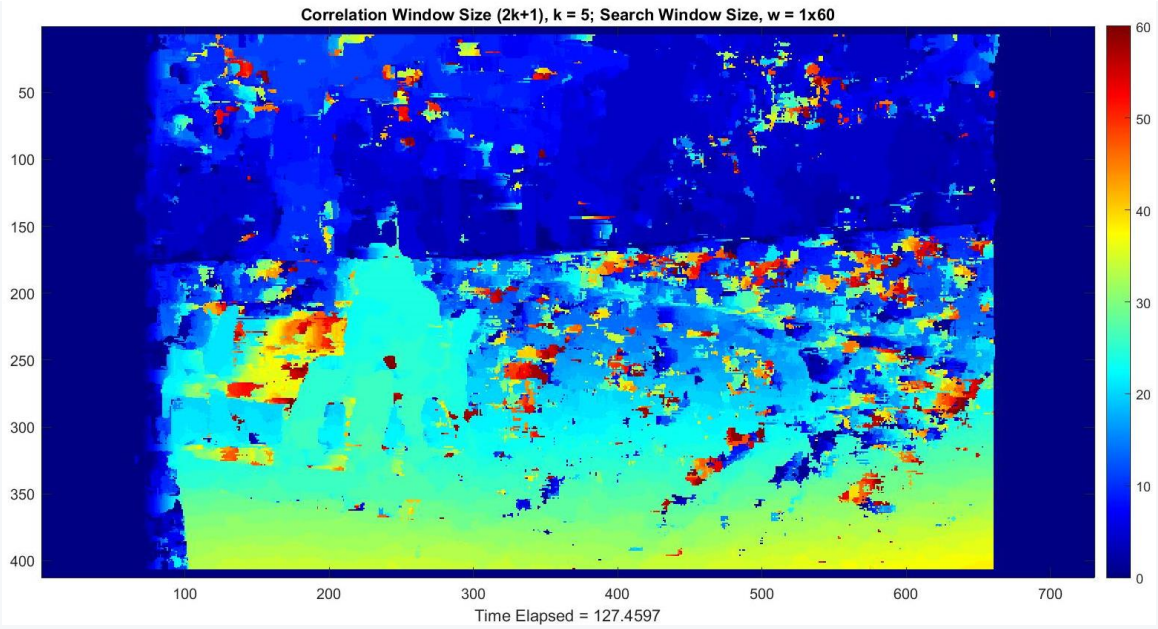
- The in-built function produces smoother disparity maps than the previous method used.
- With this function, silhouettes of obvious objects are preserved (like the man seating on the bench). In this regard, this method performs significantly better than the previous method.
- There's hardly any noisy region with this function (if the window size and search areas are properly selected; however, with the previous method, it is difficult to obtain noiseless results.



Correlation Window Size (2k+1), k = 5; Minimum Disparity = 0; Maximum Disparity = 16

- When the window size and search areas are not properly selected, both methods perform poorly:
  In this image using the function, a search area of size = 16 was used and a very bad result was obtained.

- In the minor case of preserving not-so-obvious contrast, the previous method seems to have the upper hand. For example, the outline of the pathway is still relative visible in the following image:


Correlation Window Size (2k+1), k = 5; Search Window Size, w = 1x60
50
100
150
200
250
300
350
400
100
200
300
400
500
600
700
60
50
40
30
20
10
0
Time Elapsed = 127.4597

DISCUSSION

- With the correlation-algorithm, we are able to obtain a dense list of correspondences. This kind of increases the accuracy of whatever subsequent operation is performed.
- A major disadvantage to the above is that this method is computationally demanding. For example, I tried processing 4k images with this, and the code ran non-stop. Why? Every pixel had to be checked.
- The choice of window size and search areas determines the type of results we obtain. This also sort of determines the accuracy of the results.
- Priori information (like baseline, distance from camera, etc.) goes a long way in selecting the above parameters optimally.
- Displaying the stereo anaglyph of the input images in the **Image Viewer App** and using the Distance tool to measure distances between pairs of corresponding points is also a simple trick to determine the window size and search area.
- The window size must be large enough to include enough intensity variation for reliable matching, but small enough to avoid the effects of projective distortion. If the window is too small and does not cover enough intensity variation, it gives a poor disparity estimate, because the signal (intensity variation) to noise ratio is low. If, on the other hand, the window is too large and covers a region in which the depth of scene points (i.e., disparity) varies, then the position of maximum correlation or minimum SSD may not represent correct matching due to different projective distortions in the left and right images.

REFERENCES

Reinhard Klette. 2014. Concise Computer Vision: An Introduction into Theory and Algorithms. Springer Publishing Company, Incorporated.

T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: theory and experiment," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 9, pp. 920-932, Sept. 1994, doi: 10.1109/34.310690.

https://www.mathworks.com/help/matlab/ref/imagesc.html

https://www.mathworks.com/help/vision/ref/disparity.html

APPENDIX

DISPARITY MAP

In-Lab Code

Lab8.m

```
clear all; close all; clc;

% Read the image to be preprocessed
ImLeft = imread('S00L.tif');
ImRight = imread('S00R.tif');

% ImLeft = imread('S01L.png');
% ImRight = imread('S01R.png');

% This is added in case the image introduced is an RGB
image.
% It functions to convert it to a gray-scale image.
if (size(ImLeft,3) == 3)
    ImLeft = rgb2gray(ImLeft);
    ImRight = rgb2gray(ImRight);
end

%Convert the image to double before performing any
mathematical operation
IL = double(ImLeft);
IR = double(ImRight);

% Define the left window size
k = 5;

% Define the search window size
ww1 = 40; ww2 = 1;

offset1 = ww1 + k; % Column offset
offset2 = ww2 + k; % Row offset

paddedImL = padarray(IL,[offset2 offset1],'both');
paddedImR = padarray(IR,[offset2 offset1],'both');

% The numbers of rows and columns of the padded image are
obtained
[r, c, ch] = size(paddedImL);
```

```
disparity1 = zeros(r,c);

for i =(offset2+1):r-offset2
    for j =(offset1+1):c-offset1
        % Reset the data storage matrix before any cycle.
        dist = [];

        % The window will go from -k to +k
        wL = paddedImL(i-k:i+k, j-k:j+k);

        for ii = 0:ww2
            for jj = 0:ww1
                wR = paddedImR(i-k-ii:i+k-ii, j-k-jj:j+k-
jj);
                ssd = sum(sum((wL - wR).^2));
                dist = [dist; i-ii j-jj ssd];
            end
        end
        ind = find(dist(:,3) == min(dist(:,3)));
        d1 = j - dist(ind(1),2);
        d2 = i - dist(ind(1),1);
        disparity1(i,j) = d1;
        disparity2(i,j) = d2;
    end
end

dispar1 = uint8(disparity1);

% Show stereo pair in a red-cyan anaglyph
imshow(stereoAnaglyph(ImLeft,ImRight));
% Show disparity map with colorbar
figure; imagesc(dispar1); colormap jet; colorbar
title(['Correlation Window Size (2k+1), k = ', num2str(k),
'; Search Window Size, w = ', num2str(ww2), 'x',
num2str(ww1)])
```

Post-Lab Codes

lab8_post_lab.m

```
clear all; close all; clc;
```

```
% Read the image to be preprocessed
% ImLeft = imread('S00L.tif');
% ImRight = imread('S00R.tif');

ImLeft = imread('S01L.png');
ImRight = imread('S01R.png');

% This is added in case the image introduced is an RGB
image.
% It functions to convert it to a gray-scale image.
if (size(ImLeft,3) == 3)
    ImLeft = rgb2gray(ImLeft);
    ImRight = rgb2gray(ImRight);
end

%Convert the image to double before performing any
mathematical operation
IL = double(ImLeft);
IR = double(ImRight);

% Define the left window size
k = 8;

% Define the search window size
ww1 = 60; ww2 = 1;

offset1 = ww1 + k; % Column offset
offset2 = ww2 + k; % Row offset

paddedImL = padarray(IL,[offset2 offset1],'both');
paddedImR = padarray(IR,[offset2 offset1],'both');

% The numbers of rows and columns of the padded image are
obtained
[r, c, ch] = size(paddedImL);

disparity1 = zeros(r,c);
disparity2 = zeros(r,c);

tic
for i =(offset2+1):r-offset2
    for j =(offset1+1):c-offset1
        % Reset the data storage matrix before any cycle.
```

```
        dist = [];

        % The window will go from -k to +k
        wL = paddedImL(i-k:i+k, j-k:j+k);

        for ii = 0:ww2
            for jj = 0:ww1
                wR = paddedImR(i-k-ii:i+k-ii, j-k-jj:j+k-
jj);
                ssd = sum(sum((wL - wR).^2));
                dist = [dist; i-ii j-jj ssd];
            end
        end
        ind = find(dist(:,3) == min(dist(:,3)));
        d1 = j - dist(ind(1),2);
        d2 = i - dist(ind(1),1);
        disparity1(i,j) = d1;
        disparity2(i,j) = d2;
    end
end
toc

dispar1 = uint8(disparity1);
% dispar2 = uint8(disparity2);

% Show stereo pair in a red-cyan anaglyph
imshow(stereoAnaglyph(ImLeft,ImRight));
% Show disparity map with colorbar
figure; imagesc(dispar1); colormap jet; colorbar
title(['Correlation Window Size (2k+1), k = ', num2str(k),
'; Search Window Size, w = ', num2str(ww2), 'x',
num2str(ww1)])
xlabel(['Time Elapsed = ', num2str(toc)]);
% figure; imagesc(dispar2); colormap jet; colorbar
```

lab8_post_lab_using_disparity_function.m

```
clear all; close all; clc;

% Read the image to be preprocessed
ImLeft = imread('S00L.tif');
ImRight = imread('S00R.tif');
```

```
% ImLeft = imread('S01L.png');
% ImRight = imread('S01R.png');

% It functions to convert it to a gray-scale image.
if (size(ImLeft,3) == 3)
    ImLeft = rgb2gray(ImLeft);
    ImRight = rgb2gray(ImRight);
end

min_disp = 0; max_disp = 16; % Range for the search area
win_size = 5; % correlation window size
disparityRange = [min_disp max_disp];
disparityMap = disparity(ImLeft, ImRight,
'Method','SemiGlobal', 'BlockSize',
win_size,'DisparityRange',disparityRange);

figure
imshow(disparityMap,disparityRange);
title('Disparity Map');
xlabel(['Correlation Window Size (2k+1), k = ',
num2str(win_size),...
    '; Minimum Disparity = ', num2str(min_disp), '; Maximum
Disparity = ', num2str(max_disp)])
colormap(gca,jet)
colorbar
```

lab8_post_lab_multiple.m

```
clear all; close all; clc;

% Read the image to be preprocessed
ImLeft = imread('LL.jpg');
ImRight = imread('RR.jpg');

% This is added in case the image introduced is an RGB
image.
% It functions to convert it to a gray-scale image.
if (size(ImLeft,3) == 3)
    ImLeft = rgb2gray(ImLeft);
    ImRight = rgb2gray(ImRight);
end
```

```
%Convert the image to double before performing any
mathematical operation
IL = double(ImLeft);
IR = double(ImRight);

% Define the left window size
k = 5;

% Define the search window size
ww1 = 17; ww2 = 5;

offset1 = ww1 + k; % Column offset
offset2 = ww2 + k; % Row offset

paddedImL = padarray(IL,[offset2 offset1],'both');
paddedImR = padarray(IR,[offset2 offset1],'both');

% The numbers of rows and columns of the padded image are
obtained
[r, c, ch] = size(paddedImL);

disparity1 = zeros(r,c);
disparity2 = zeros(r,c);

for i =(offset2+1):r-offset2
    for j =(offset1+1):c-offset1
        % Reset the data storage matrix before any cycle.
        dist = [];

        % The window will go from -k to +k
        wL = paddedImL(i-k:i+k, j-k:j+k);

        for ii = 0:ww2
            for jj = 0:ww1
                wR = paddedImR(i-k-ii:i+k-ii, j-k-jj:j+k-
jj);
                ssd = sum(sum((wL - wR).^2));
                dist = [dist; i-ii j-jj ssd];
            end
        end
        ind = find(dist(:,3) == min(dist(:,3)));
        d1 = j - dist(ind(1),2);
        d2 = i - dist(ind(1),1);
```

```
        disparity1(i,j) = d1;
        disparity2(i,j) = d2;
    end
end

dispar1 = uint8(disparity1);
dispar2 = uint8(disparity2);

% Show stereo pair in a red-cyan anaglyph
imshow(stereoAnaglyph(ImLeft,ImRight));
% Show disparity map with colorbar
figure; imagesc(dispar1); colormap jet; colorbar
title(['Horizontal Disparity Map'], ['Correlation Window
Size (2k+1), k = ', num2str(k), '; Search Window Size, w =
', num2str(ww2), 'x', num2str(ww1)])
figure; imagesc(dispar2); colormap jet; colorbar
title('Vertical Disparity Map')
```