LAB 7
POSTLAB
REPORT

NAME:NIDANUR GUNAY

ID:24231

Correlation Matching for finding Correspondences:

- We will use correlation matching to solve the correspondence problem in stereo vision. We will search for the best *subR* (right sub-image) similar to *subL* (left sub-image) starting from the same pixel location and along the vicinity of that location (in window $R = \omega \times \omega$). In order to achieve this, we need to calculate the similarity between the sub-images for each displacement $d = [d_1, d_2]$ in R as follows:

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

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

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

- We will store the displacements (d_1 and d_2) and the similarity values in each row of a matrix called `dist`. We can then use `find` command to retrieve the row index of the minimum similarity value in `dist` matrix as follows:

```
ind = find(dist(:,3) == min(dist(:,3)));
```

- Note that the correspondence points will be found for every pixel and the disparity map will be calculated from these corresponding pixels of stereo images, i.e., $d = x_{left} - x_{right}$.
- Save your codes as "lab7.m".

Fig 1.1

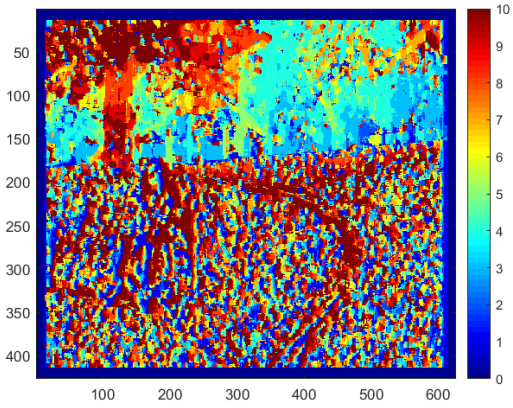
Human binocular vision perceives depth by using Stereo disparity which refers to the difference in image location of an object seen by the left and right eyes, resulting from the eyes' horizontal separation. Similarly, stereo cameras use the same technique and there are 2 cameras for the triangulation. We have 2 image that is taken from left and right camera. We sustain stereo vision by finding the corresponding window that is taken from right image in the image that is taken from left. For the matching finding we use SSD that is described in the Fig 1.1. In the firs step we found the correspondence image window in the left image. In order the find the corresponding image window, we find the image window that is located at the same location as right image window. Then we shift the image 1 pixel to the right and thanks to SSD function, we can measure the difference between 2 window. We move the window as the offset amount and find the window that has the minimum SSD and it gives us the corresponding image. And the distance between these 2 window gives us the depth.

W=3 w=10

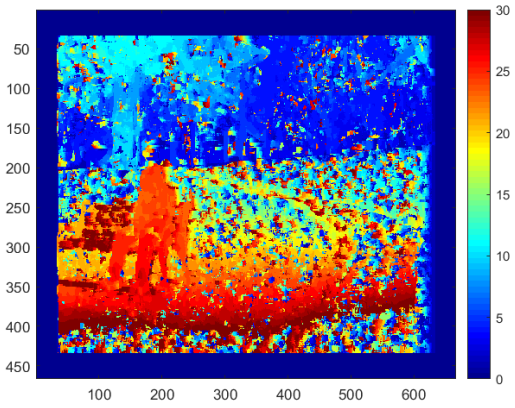
W=3 w=30

W=3 w=60

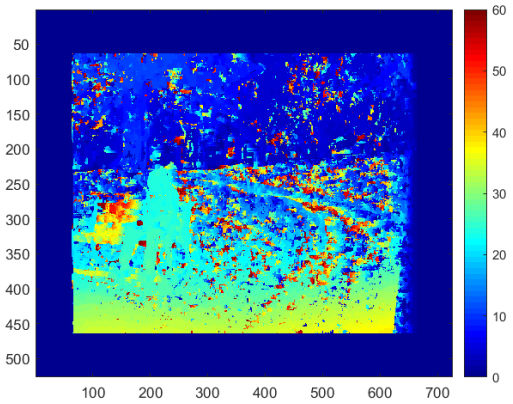
W=3 w=100



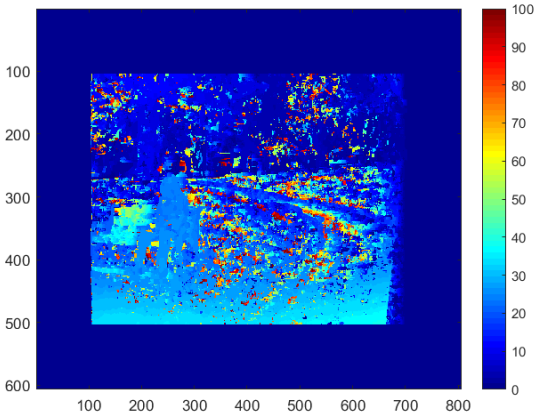
W=3 w=10



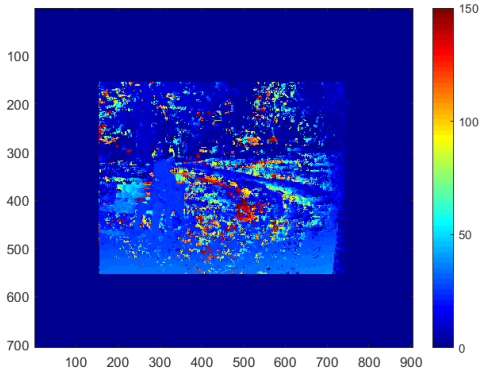
W=3 w=30



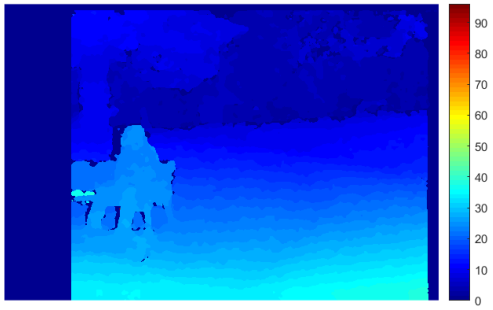
W=3 w=60



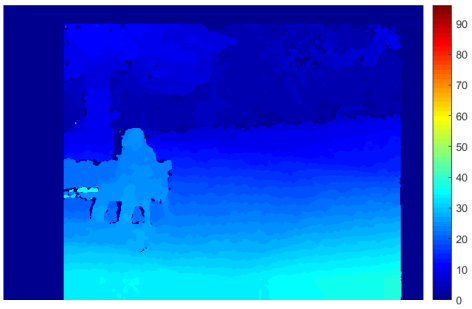
W=3 w=100



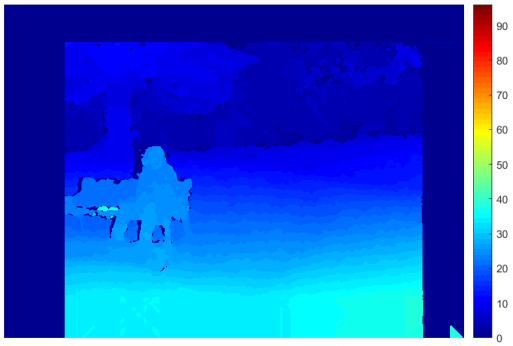
W=3 w=150



W=3 w=10

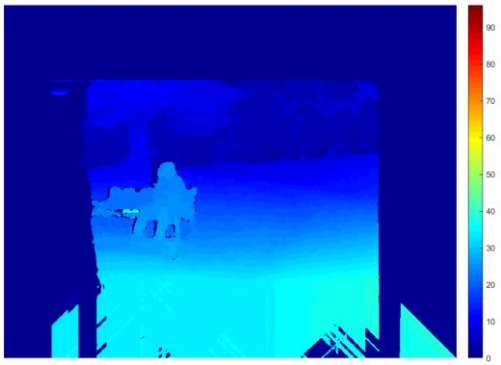


W=3 w=30



W=3 w=60

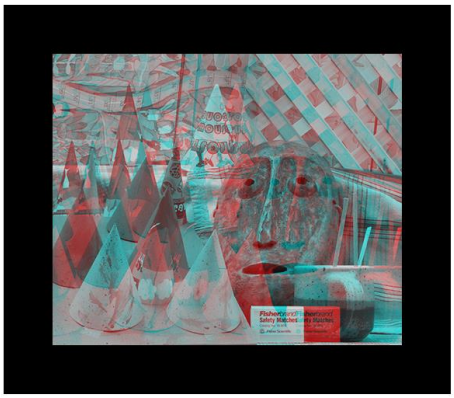
W=3 w=100



W=3 w=150

Discussion: As the w value change, the amount of offset would be changed. In the first image group there is no visible change however since the offset amount change much more the image size gets smaller and we can see that from the thickness on the borders. In the second image that displays the disparity, we can observe significant changes. w value also controls how many pixels we search through right in the left image in order to find the corresponding value and we do that to decrease the computational complexity. When we set it so small like 10, we wouldn't get the concrete results since the searched pixels are so limited and the maximum disparity would be 10. When we set it too large as 150, we don't observe the different colors and it shows us we lose the depth perception. It results that the corresponding windows disparity range would decrease and their color closer to each other.

In the third image group there is no significant change between the images however object size gets smaller. It is resulted form the total offset amount gets larger and object becomes smaller due to the intersection between left and right image becomes smaller.

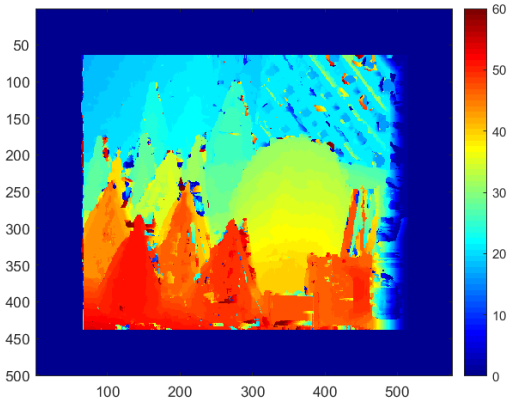


W=3 w=60

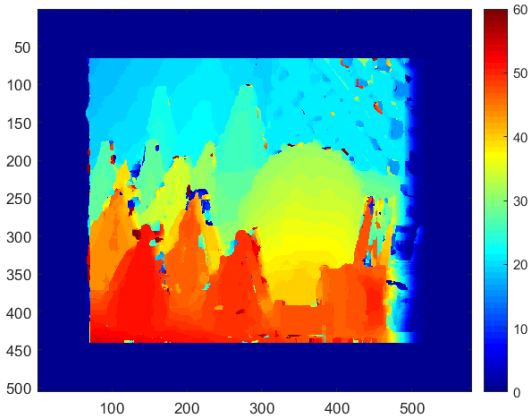
W=5   w=60

W=13 w=60

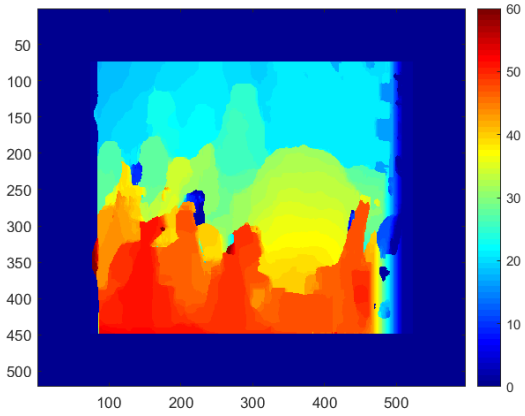
W=21 w=60



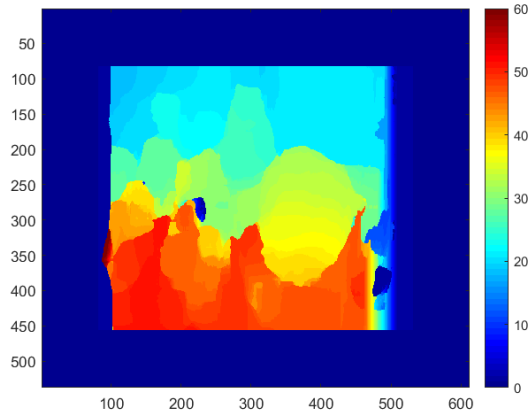
W=3 w=60



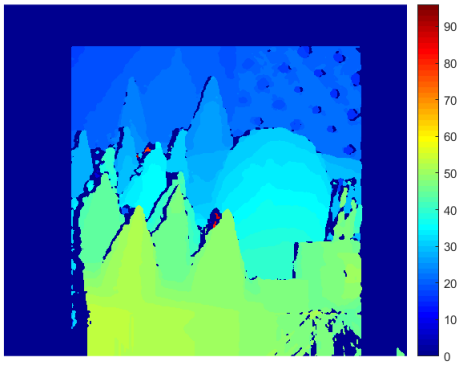
W=5 w=60

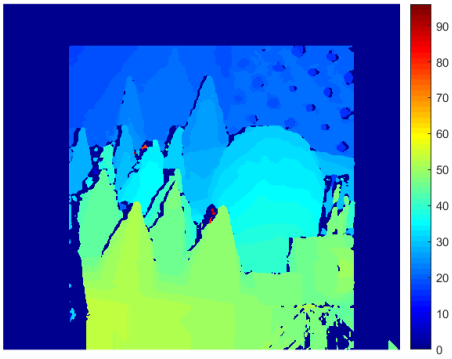


W=13 w=60



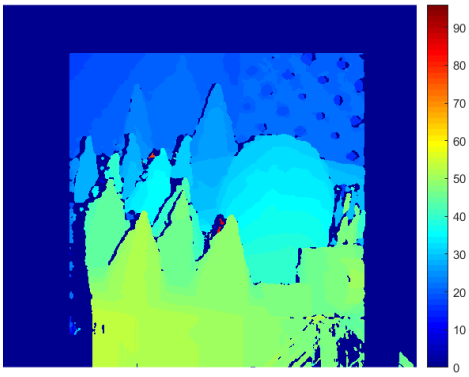
W=21 w=60

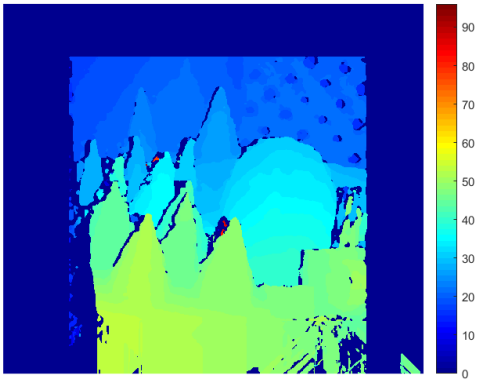




W=3 w=60

W=5 w=60





W=13 w=60

W=21 w=60

Discussion: W symbolizes the window size and as we increase the W value, window size gets bigger. In the first image pairs that shows the padded images are the same. And the third image pairs show the disparity range doesn't differ significantly. However, in the second image pair that shows the disparity, we can observe that when the window size increases, details become disappear and unlike the w change, colors haven't change. Because, bigger window contains more pixel, we would traverse less image detail.

APPENDIX: mainnida.m

```
Left = imread('S01L.png');
Right = imread('S01R.png');


[rl, cl, ch] = size(Left);
W=3;
w = 60;

Left = padarray(Left,[W+w W+w],'both');
Right = padarray(Right,[W+w W+w],'both');

if (ch==3)
        Left = rgb2gray(Left);
        Right = rgb2gray(Right);
end

imshow(stereoAnaglyph(Left,Right));

Left = double(Left);
Right = double(Right);

dispar = zeros(size(Left));

for x=W+w+1:1:rl+W+w
    for y=W+w+1:1:cl+W+w
        distance = zeros(w,3);
        for i = w:-1:1
                SSD = sum(sum((Left(x-W:x+W, y+i-W:y+i+W)-Right(x-W:x+W, y-
W:y+W)).^2));
                comparisonmatrix = [0, i, SSD];
                distance(i,:) = comparisonmatrix(:);
        end
            ind = find(distance(:,3) == min(distance(:,3)));
            dispar(x,y) = distance(ind(1), 2);
    end
end
figure; imagesc(dispar); colormap jet; colorbar
Left = uint8(Left);
Right = uint8(Right);

disparityRange = [0, 96];disparityMap = disparity(Left, Right,'BlockSize',
7,'DisparityRange',disparityRange);
hold on;

figure; imshow(disparityMap,disparityRange); colormap jet; colorbar
```