Aynur Süne - 21041

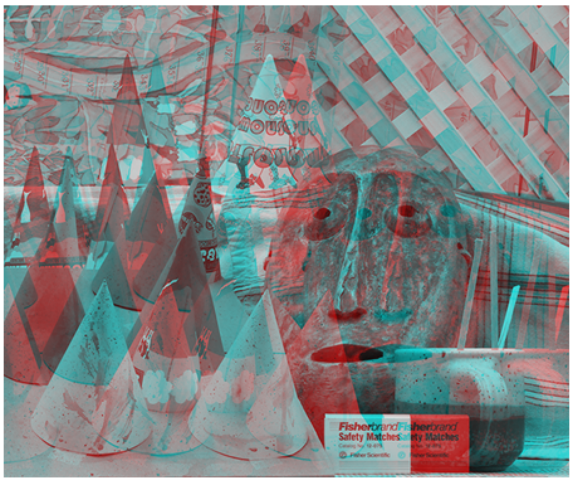
EE417 Post-Lab #7

```
imgR = imread('S01R.png');
imgL = imread('S01L.png');

[r,c,color] = size(imgR);
if(color==3)
    imgR = rgb2gray(imgR);
end

[r,c,color] = size(imgL);
if(color==3)
    imgL = rgb2gray(imgL);
end

k=3;
omega = 60;
offset = 60;

imshow(stereoAnaglyph(imgL,imgR));

paddedImL = padarray(imgL,[offset offset],'both');
paddedImR = padarray(imgR,[offset offset],'both');

paddedImR = double(paddedImR);
paddedImL = double(paddedImL);

dispar = zeros(size(paddedImR));

for i=offset+k+1:1:r-k+offset-1
    for j= offset+k+1:1:c-k+offset-1

        subimgL = paddedImL(i-k:i+k,j-k:j+k);
        dist =[];

        for j2= j-omega:1:j
             subimgR = paddedImR(i-k:i+k,j2-k:j2+k);

            %SSD
            ssd = sum(sum((subimgR-subimgL).^2));
            dist = [dist; i j2 ssd];

        end

       ind = find(dist(:,3) == min (dist(:,3))) ;
       ind = ind(1);
       dispar(i,j) = j - dist(ind(1),2);

    end
end

figure; imagesc(dispar); colormap jet; colorbar
```

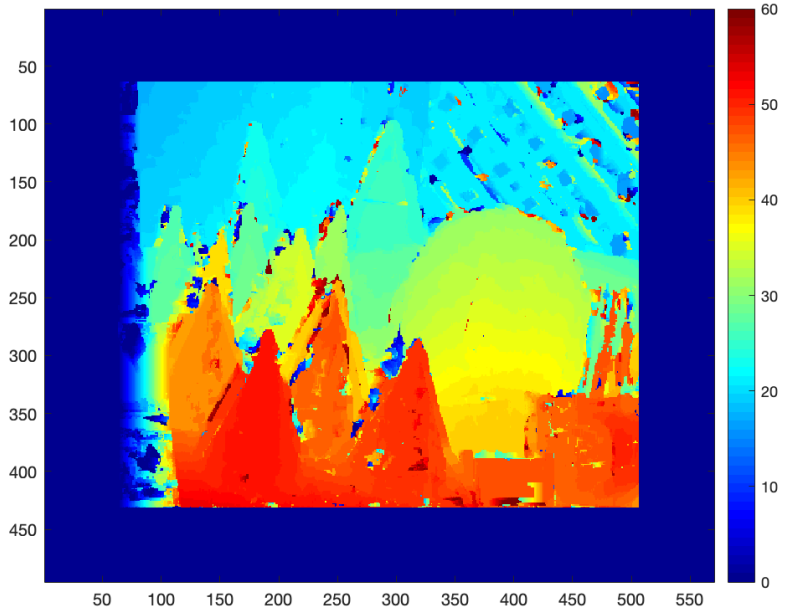
First, we take two images of the same scene captured from different angles and call them as imageLeft (imgL) and imageRight (imgR). We convert them to gray-scale and define our parameters, k as 3, omega as 60 and offset as 60. k is defined for our 2k+1 window size, omega is for the length of line that we will search similarity between left and right image.

Then we show stereo pair in a red-cyan anaglyph. We apply padding by offset amount to prevent windowing operation exceeding the image. Later, we convert padded images to double and initialize our disparity matrix. Since we padded our image size offset from all sides, during windowing operation we need to start after offset amount of pixels and finish before offset. Offset part is different than our usual double for-loop. Inside the double for-loop, we take the current window of the left image and search for similarities with right image along a line sized omega. We create a distance vector called dist. To be able to search along that line, we operate in another for-loop. In this loop, we take same sized windows of the right image along the omega sized line and similarity is calculated via SSD (sum of square differences). We take every pixel value in both windows and calculate difference between corresponding pixels, take the square of each difference and sum all values. Resulting value gives the similarity of the windows in terms of error. Then we enter each SSD of the windows into dist array. After calculation for each window in the inner for-loop is done, we take the smallest SSD value since SSD is in terms of error and we need the minimum error in order to get the most similarity. When we find the most similar window in the right image, we add the difference of j values ($x_{left} - x_{right)}$ to dispar matrix of the same size with our images.

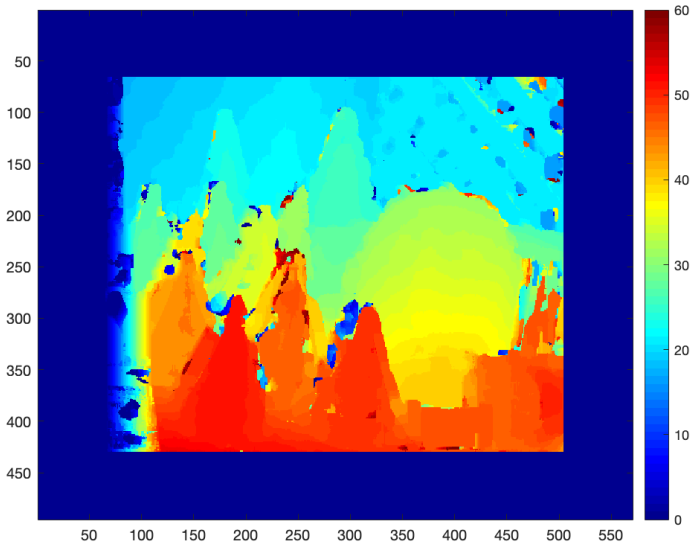
Red-Cyan Anaglyph



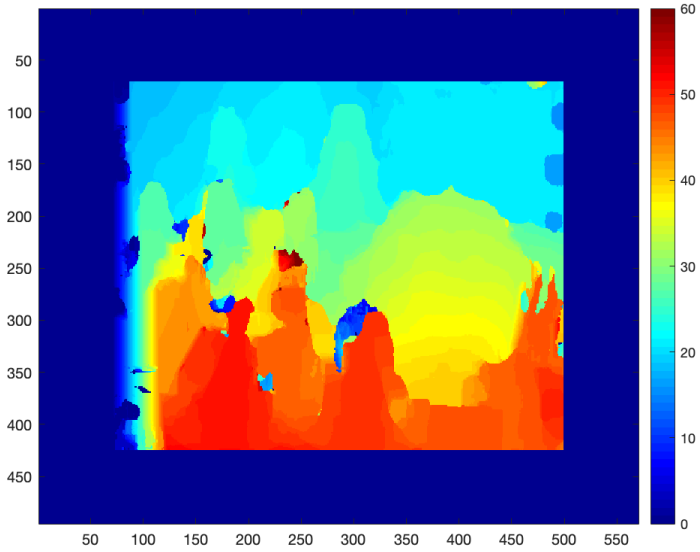
k=3 & omega = 60 & offset = 60



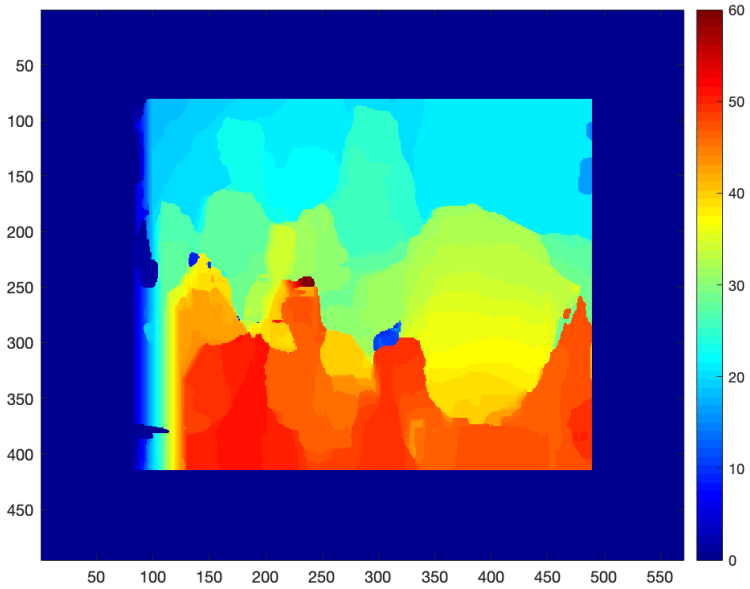
k=5 & omega = 60 & offset = 60



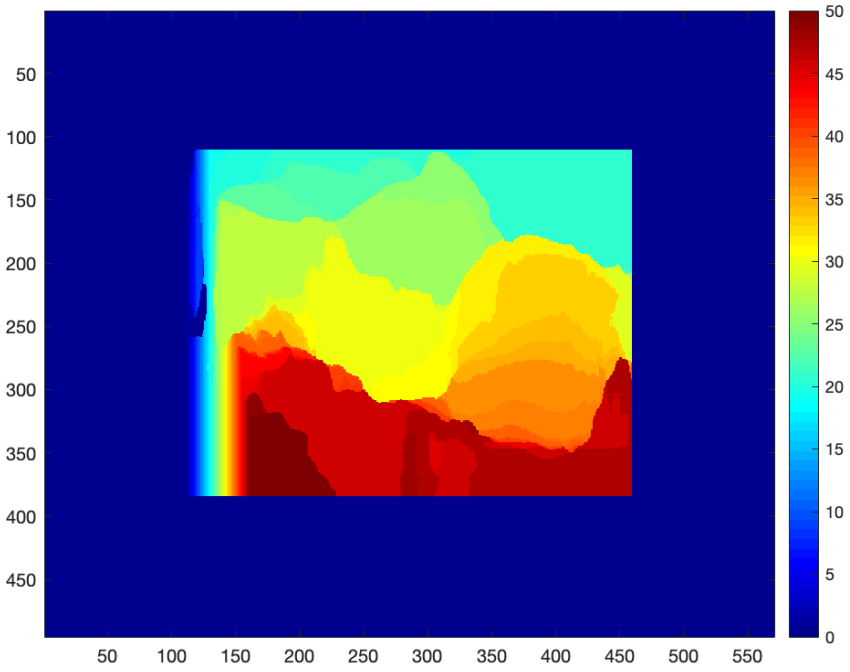
k=10 & omega = 60 & offset = 60



k=20 & omega = 60 & offset = 60

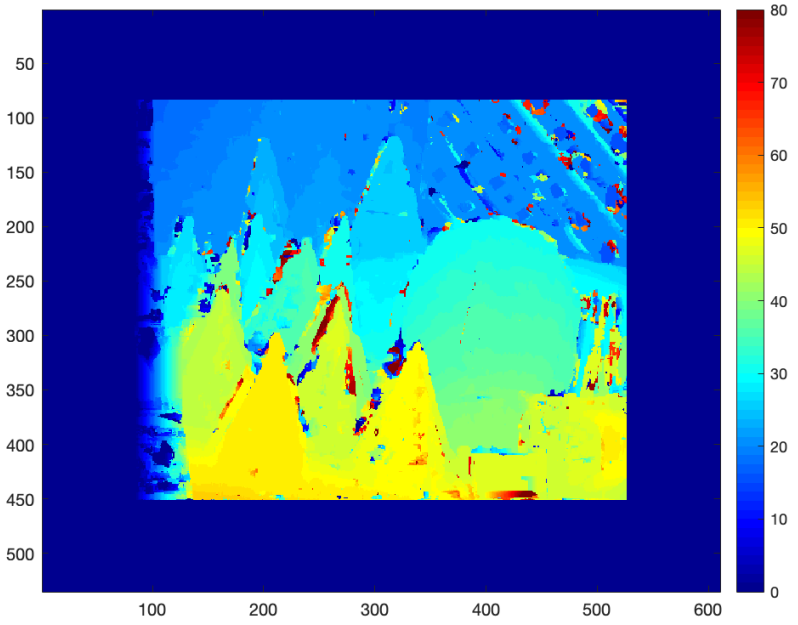


k=50 & omega = 60 & offset = 60

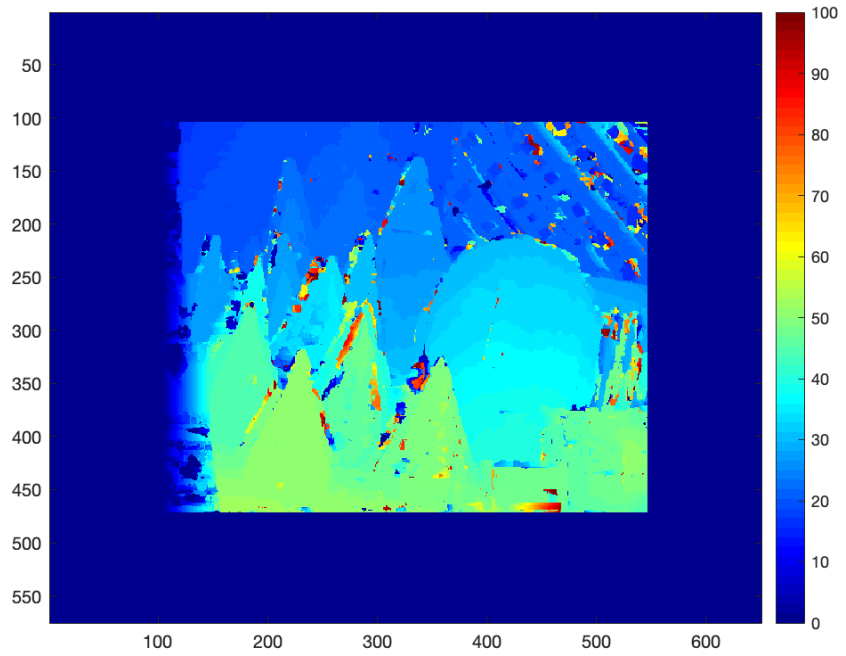


Notice that as k increases, details are lost, resulting disparity map becomes more general. Image is more smoothed because we consider more pixels in the window. Outlier details are lost. Search area remains same except window size that operates on it, so SSD values calculated are similar. That's why we don't see exceptional change in terms of colors.

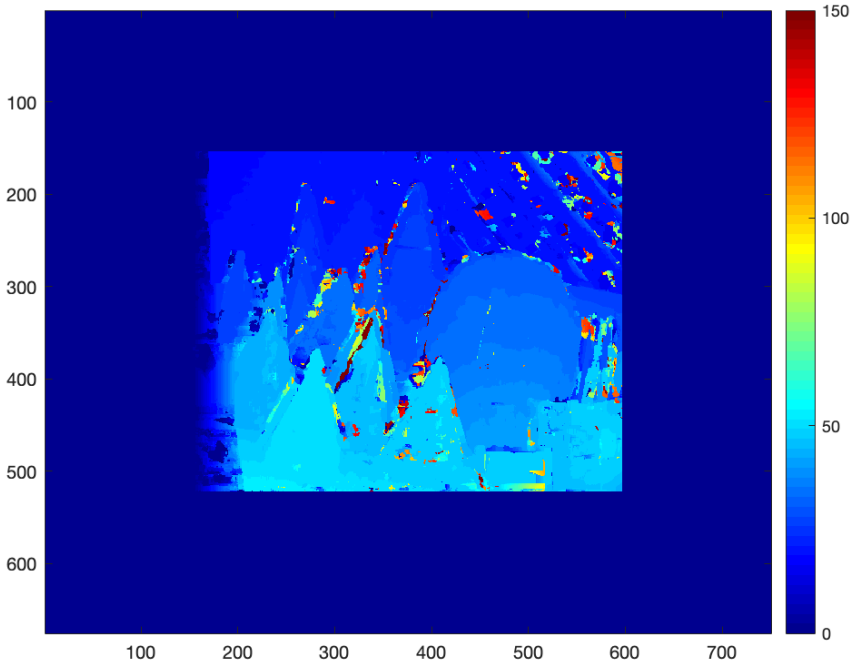
k=3 & omega = 80 & offset = 80



k=3 & omega = 100 & offset = 100



k=3 & omega = 150 & offset = 150

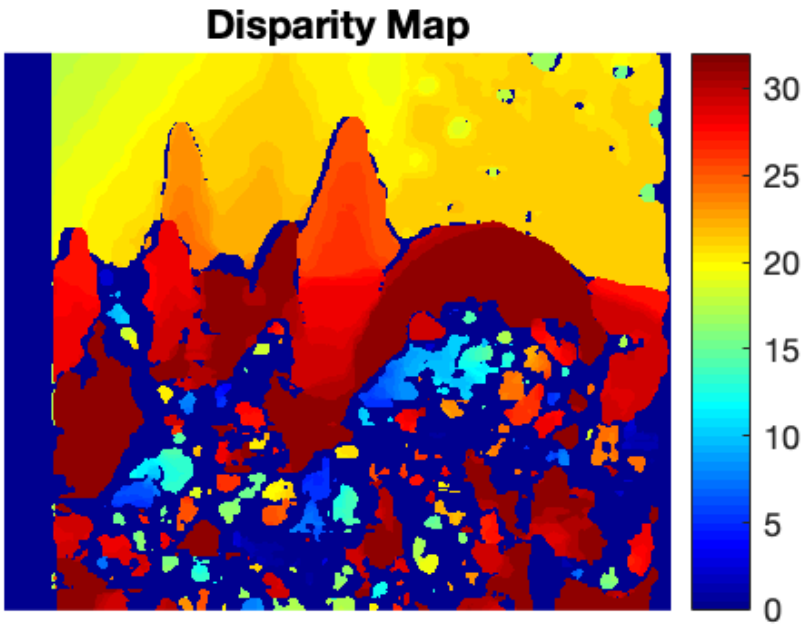


When omega is increased, search area gets bigger. During similarity search, padding pixels are more and more included and they tend to be chosen because their SSD value is small. That's why we see disparity map getting bluer like the color of the offset around image. But k value is small, so details are preserved.

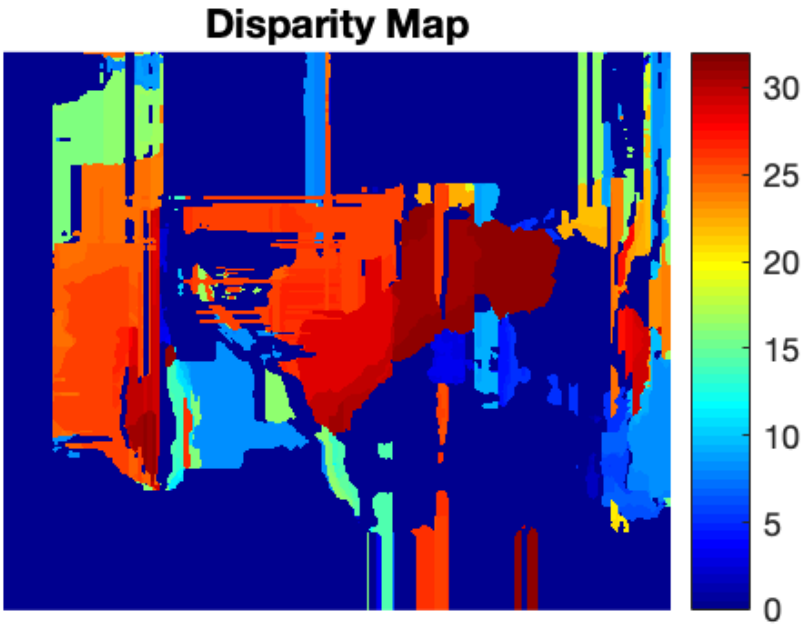
Built-in Disparity Function

```
range = [0,32];
map = disparity(imgL,imgR,'BlockSize',15,'DisparityRange',range);
figure;
imshow(map,range);
title('Disparity Map');
colormap(gca,jet)
colorbar
```

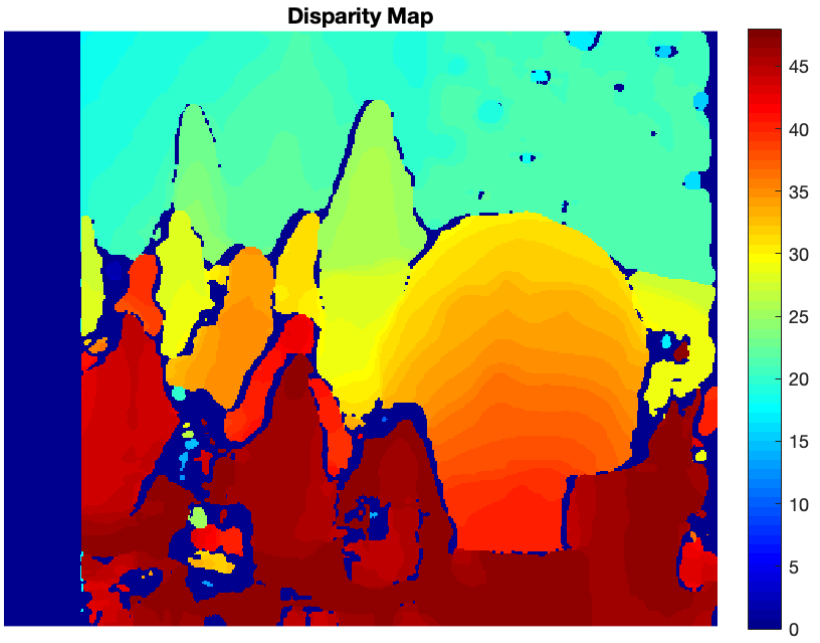
range = [0 32] & BlockSize = 15



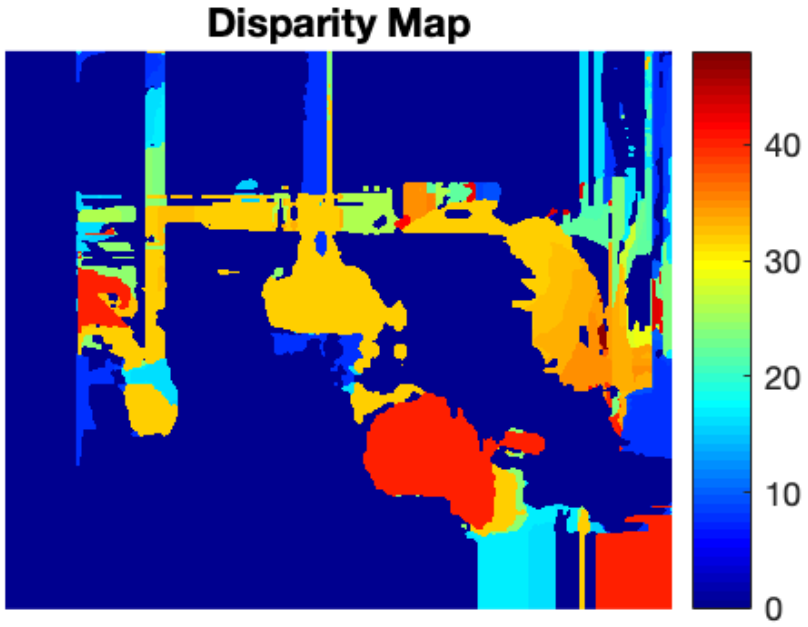
range = [0 32] & BlockSize = 101



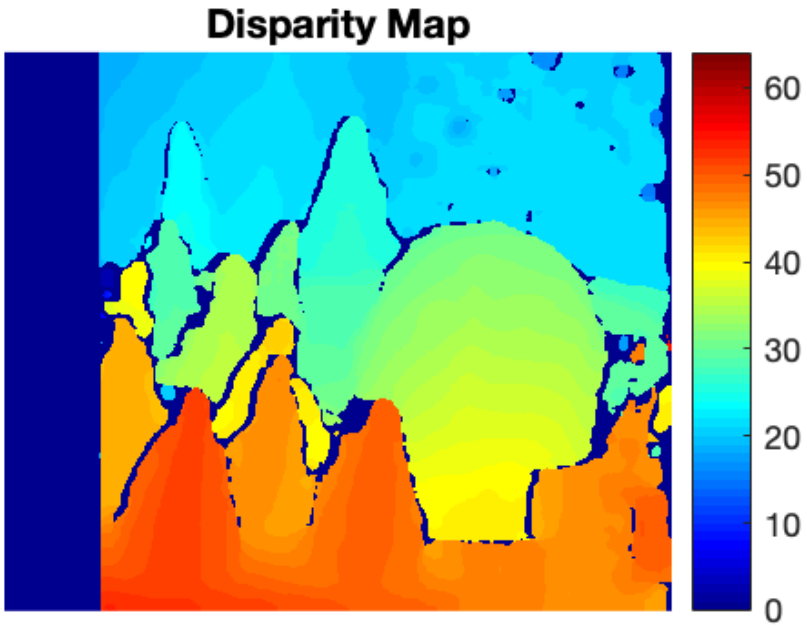
range = [0 48] & BlockSize = 15



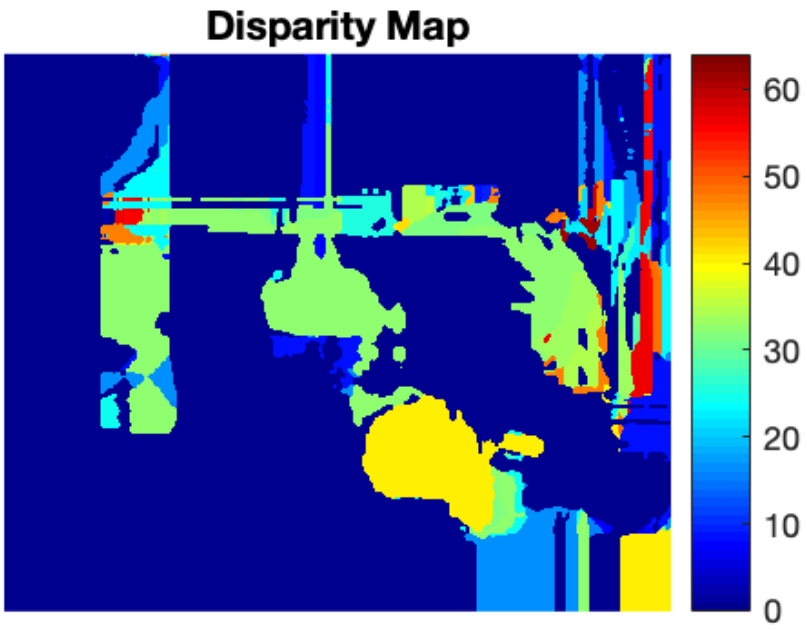
range = [0 48] & BlockSize = 101



range = [0 64] & BlockSize = 15



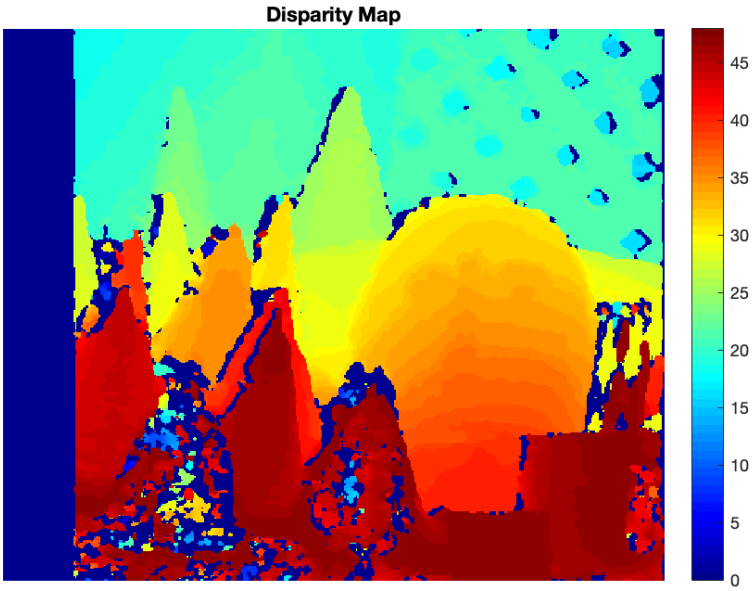
range = [0 64] & BlockSize = 101



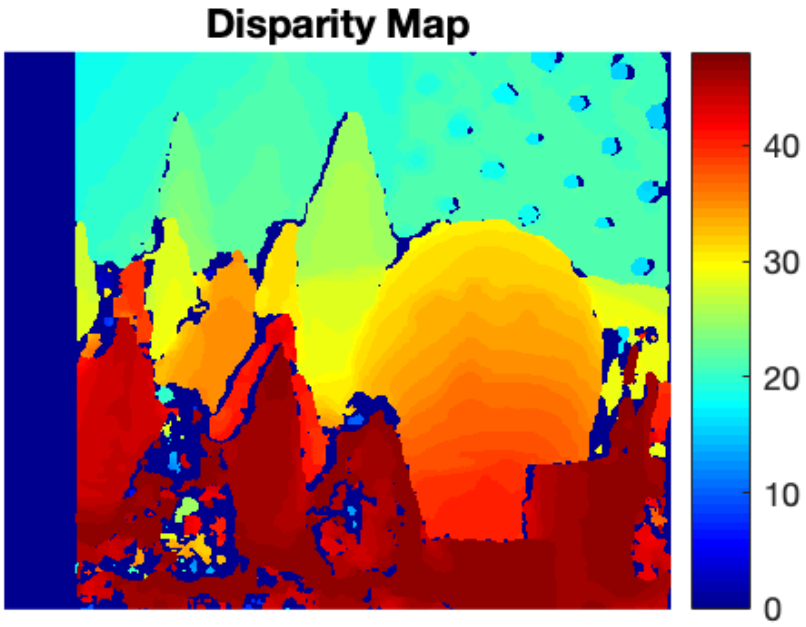
Range parameter should be divisible by 16, so 3 different range values that fit the definition are evaluated. Also block size should be odd, one low and one high block size value is also evaluated. Best result among trials were when block size is 15. Among result of outputs with block size 15 is the one with range [0 48]. Lower range values were resulting in meaningless outputs.

After finding range [0 48] was better than other ranges I've tried, I decided to decrease BlockSize parameter even more.

range = [0 48] & BlockSize =15



range = [0 48] & BlockSize =9



I think optimal results are obtained with even lower BlockSize parameter values. With built-in function, outlines and colors are sharper than all. But I believe my implementation did pretty good job as well.