EE 417 Computer Vision

Berkant Deniz Aktaş

19515

Post-Laboratory Report

December 10, 2018

Sabancı University

Faculty of Engineering and Natural Sciences

Computer Science and Engineering

Introduction

In this lab, we implemented a correlation matching algorithm in order to solve correspondence problem in stereo vision. There are various methods for solving this problem but we used correlation matching with minimizing the SSD.

In lab

First we computed related sub images with parameter K and compared a sub image with shifting the other sub image in the second image. The search window range area is determined with omega. During each iteration of a window, we stored x values of first image, x values of the second image and SSD of the sub images in a dist matrix. For each window, we computed sub window pairs which has minimum SSD, which gives us a clue about similar objects in a window. After finding the minimum distances we computed the difference of the x values and stored in a new image matrix.

Comparison

We used padding on 4 sides of the image for fixing size issues as its guided in the lab. The padding was computed with Omega + Window size for consistency reasons. Omega is computed based on the difference in the images as its showed in Figure 1.

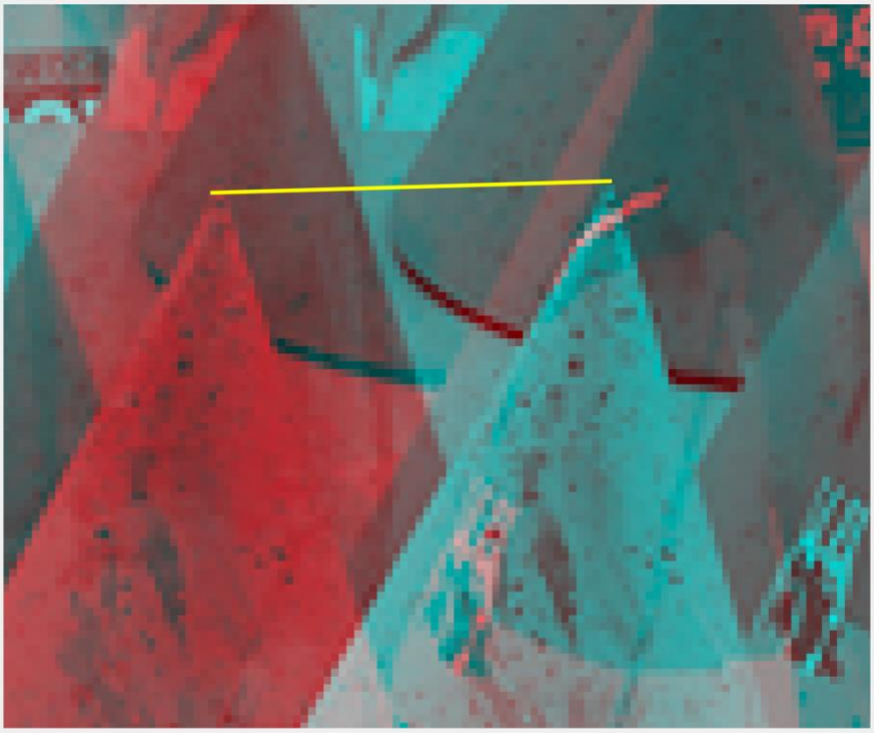

Figure 1. Distance in pixels

Based on the distance I approximated 50 as omega and achieved good results.

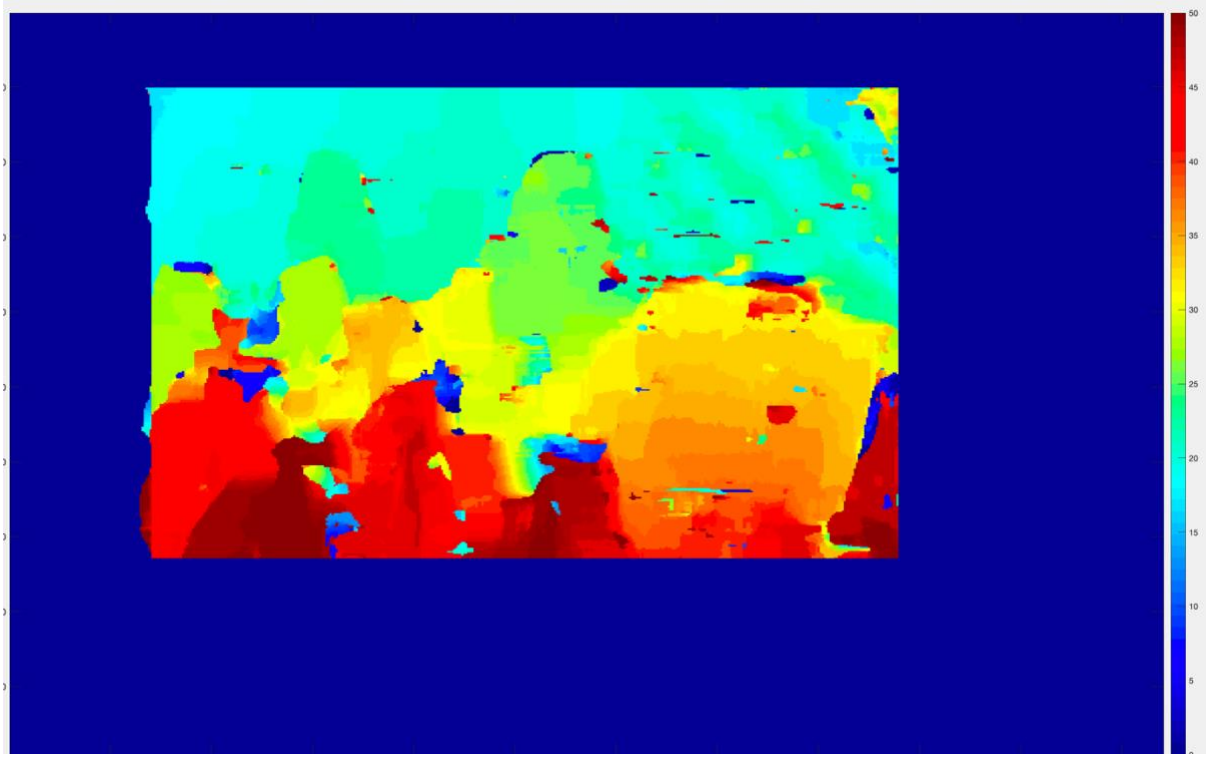

Figure 2. K = 10, Omega = 50

According to tests, a smaller Omage would give better results but not too small like 20.

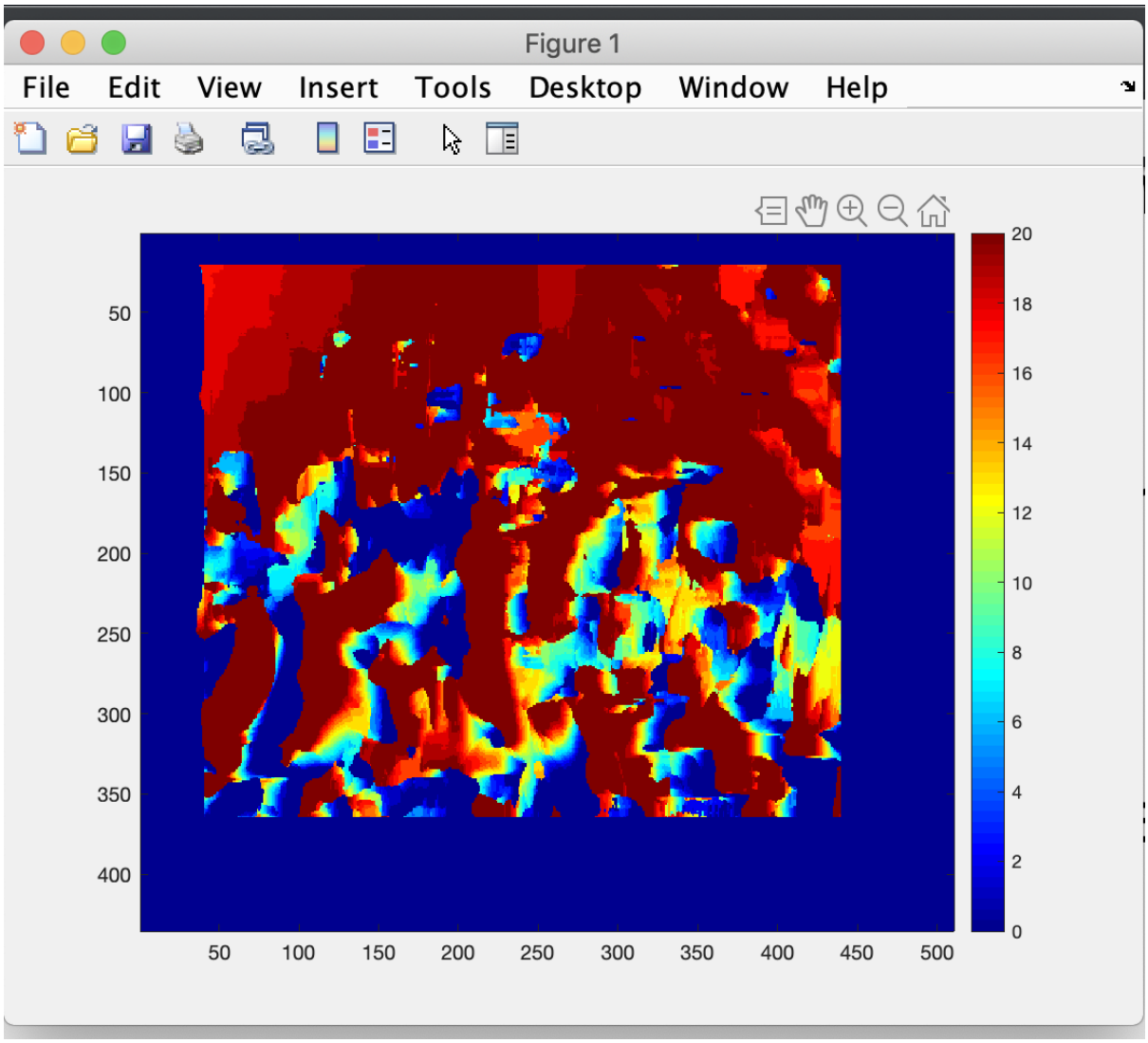

Figure 3. K = 10, Omega = 20

As its shown on the figure 3, decreasing omega too much caused irrelevant results. I guess having 40-50 is good enough. Also increasing it too much is not that good.

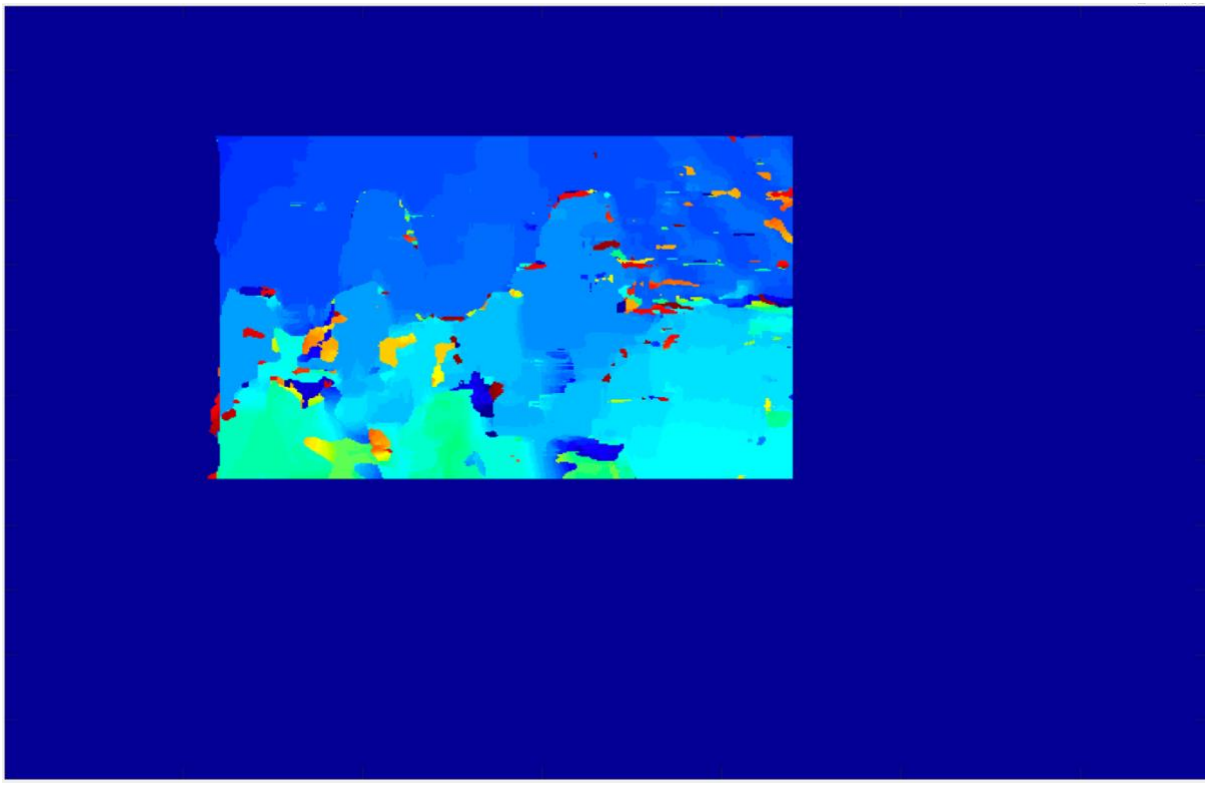

Figure 4. K = 10, Omega = 100

Having too small Omega like 20 or having it too big like 100 resulted irrelevant results.

Of course Omega parameter is strongly related with the camera's baseline. If baseline is small, one should use small omega. Also we should test different K values, window sizes.

If we increase K, there will be more details but also there will be irrelevant areas.

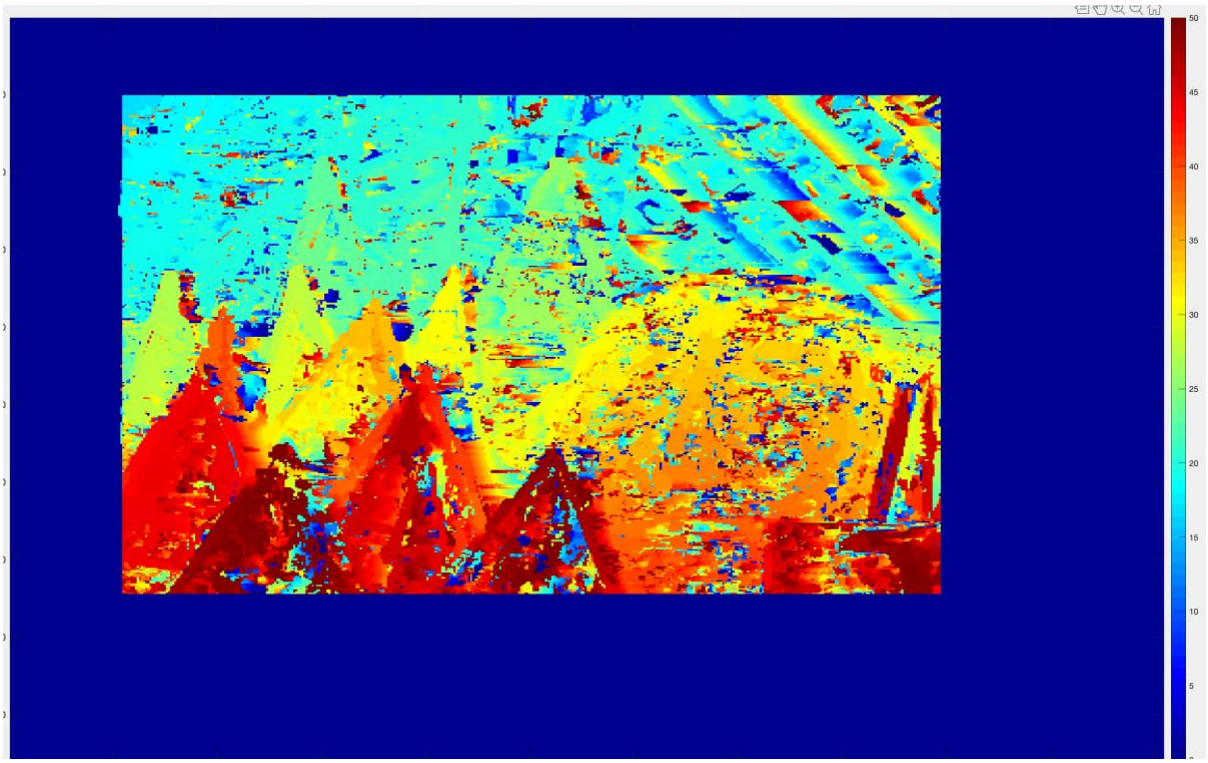

Figure 5. K = 2, Omega = 50

We can compare Figure 2 and Figure 5, we can detect objects better in Figure 5 but there lots of sketchy pixels in the image. So choosing between 2 – 10 is nice for this image. Again, K is a parameter which depends on the image size and camera parameters. If we increase K too much, we will lose details.

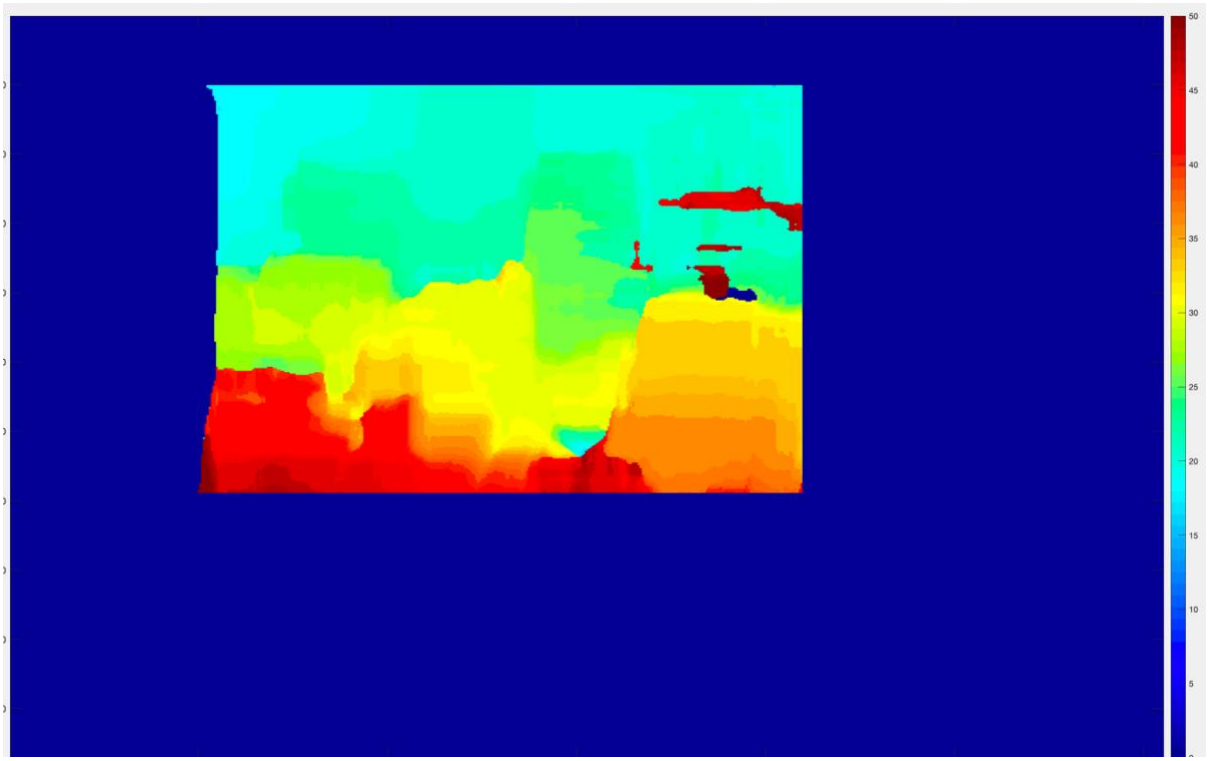

Figure 6. K = 30 Omega = 50.

Discussion

There are different methods for stereo matching and we used min SSD method to compute disparity map. I checked matlab disparity function and they are also using similar block matching method but I guess they are computing intensity difference with SAD. However, results were similar but MATLAB's figure is more fancy and cool.

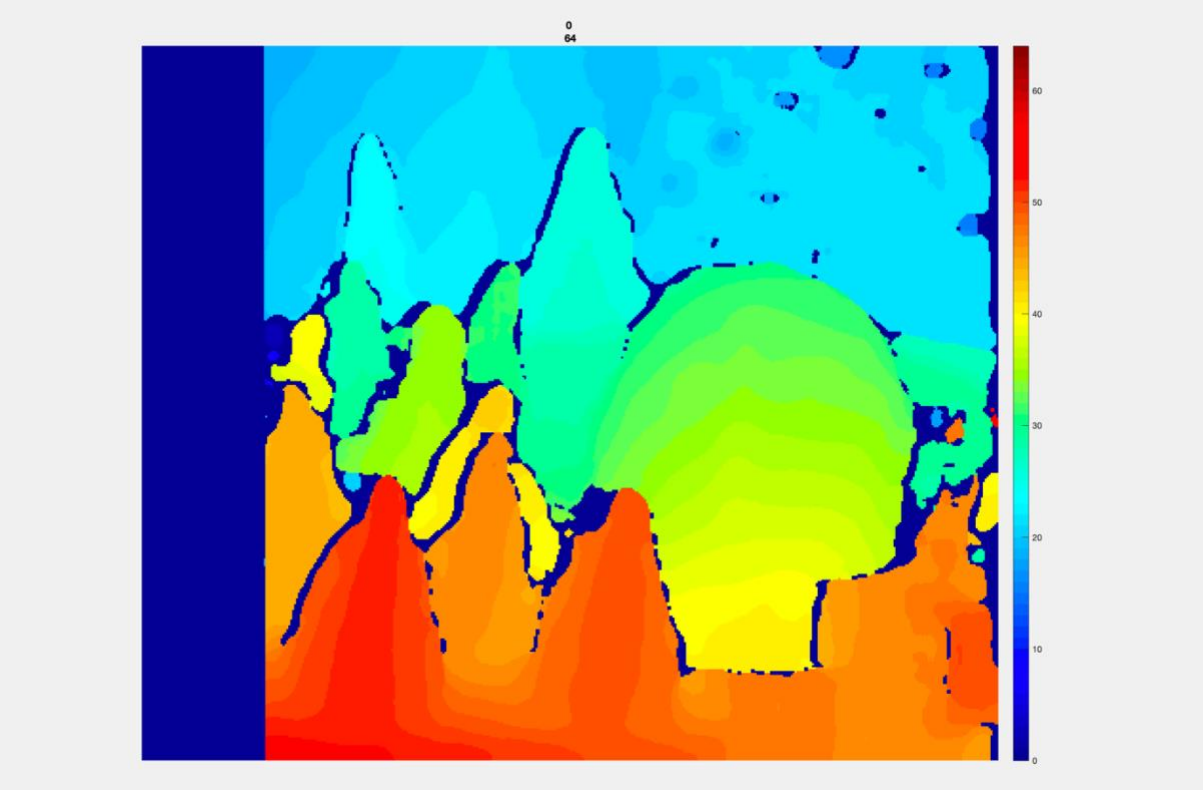

Figure 7. MATLAB's disparity map

Furthermore, we can check different block-matching operations and compare all of them. It will be very good exercise in the future. This lecture slides are really good for understanding topic and improving the knowledge about it .

http://www.diegm.uniud.it/fusiello/teaching/mvg/stereo.pdf

Codes

```
ImLeft = imread('S01R.png'); % ignore this
ImRight = imread('S01L.png'); % ignore left right mismatch



[row, col, ch] = size(ImLeft);
```

```
ImLeft = rgb2gray(ImLeft);
ImRight = rgb2gray(ImRight);

K=10; % window size
Omega = 70;

 imshow(stereoAnaglyph(ImLeft,ImRight));
ImLeft = double(ImLeft);
ImRight = double(ImRight);

ImLeft =   padarray(ImLeft,[K+Omega K+Omega],'both');
ImRight =   padarray(ImRight,[K+Omega K+Omega],'both');




% for i=K+1:1:row-K-1
%     for j=K+1:1:col-K-1
dispar = zeros(size(ImLeft));

for i=K+1:1:row-K-1
    for j=K+1:1:col-K-1

        dist=[];
        subL = ImLeft(i-K:i+K,j-K:j+K);

       for t = 0:1:Omega
           subR = ImRight(i-K:i+K,j-K+t:j+K+t);

           SSD = sum(sum(subL-subR).^2);
           dist = [dist; j j+t SSD];
       end

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

    end
end

dispar = uint8(dispar);
figure;
title(K);
imagesc(dispar); colormap jet; colorbar
```