EMİR ALAATTİN YILMAZ

EE417

POST-LAB #7 REPORT

Correlation Matching for finding Correspondences

Finding correspondences by correlation matching is a “search” problem we need to find a given element in the left image and search for the corresponding element in the right image. We need to determine geometric constraints to reduce the size of the search space such as choosing the elements to match and a similarity measure to compare elements.

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 = ω × ω). In order to achieve this, we need to calculate the similarity between the sub-images for each displacement d = [d1, d2] in R as follows:

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

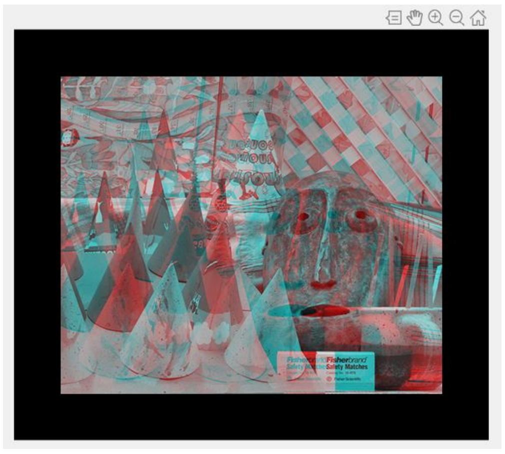
where Ψ is the similarity measure such as SSD which can be calculated as follow:

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

Procedure:

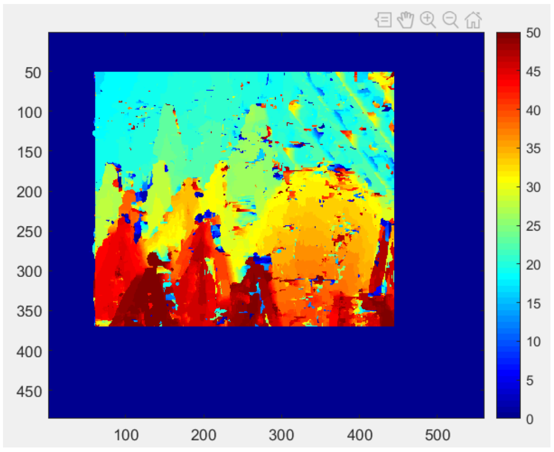
1. We take a ω × ω window from left image and right image
2. We slide the right window one by one (pixel) to the right while holding the left image as a reference (no slide)
3. For each slide of the right window, we calculate the SSD of the each pixels in these two windows (left and sliding right window) and record the sliding amount for this SSD measurement
4. Minimum SSD measurement shows that similarity of these windows is high
5. We take the sliding amount of the minimum SSD measurement and determine this amount as disparity for that pixel that will be located in the disparity map

We need to determine the size of sub-images and the size of search window.

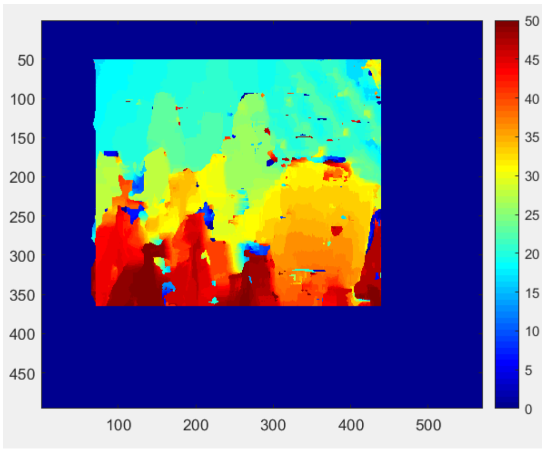


The size of sub-image is important. If it is too small, it will be inefficient and also we may not be able to extract the feature properly because the frame is so narrow.

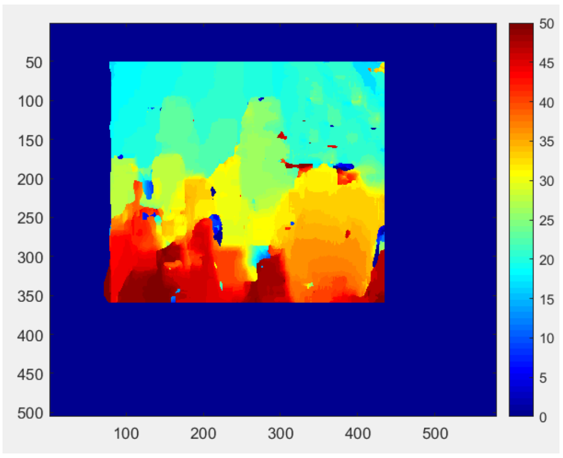
k = 5, omega = 50



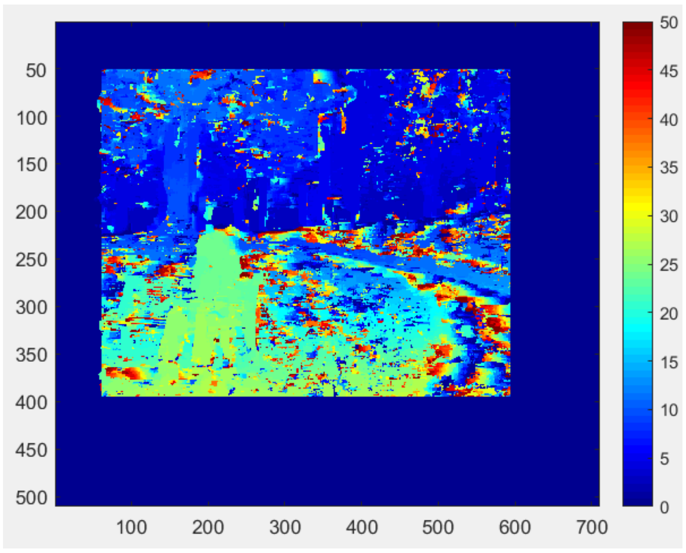
k=10, omega=50



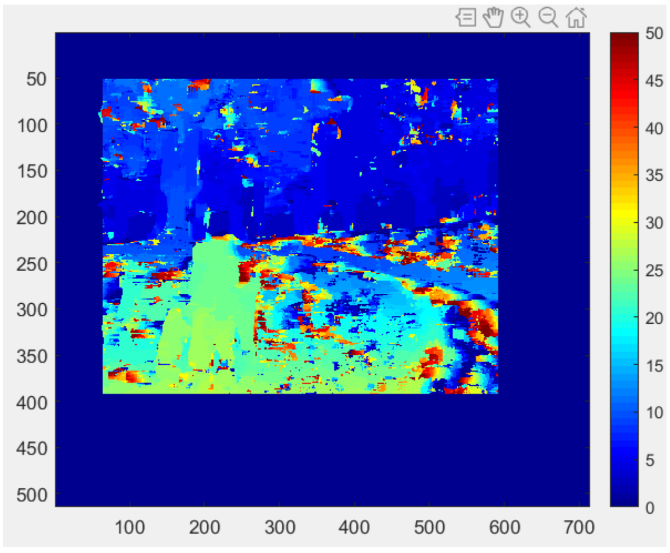
k = 15, omega = 50



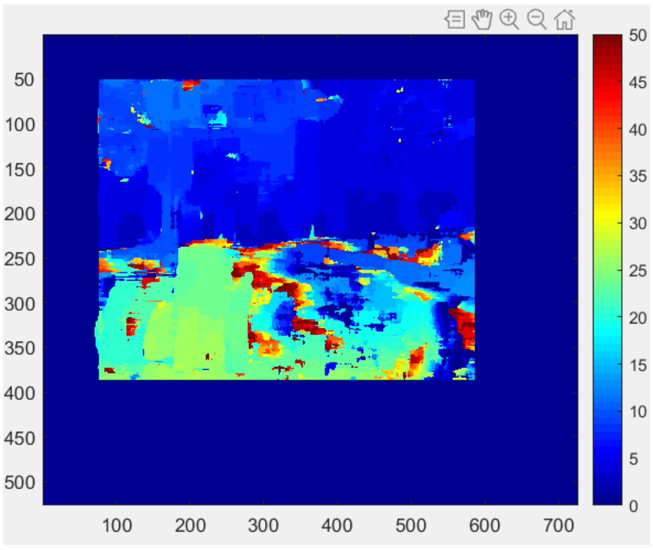
k=5, omega = 50



k=7, omega=50

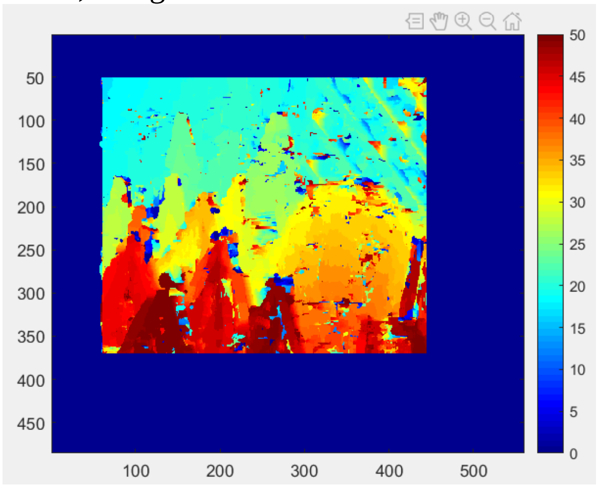


k=13, omega=50



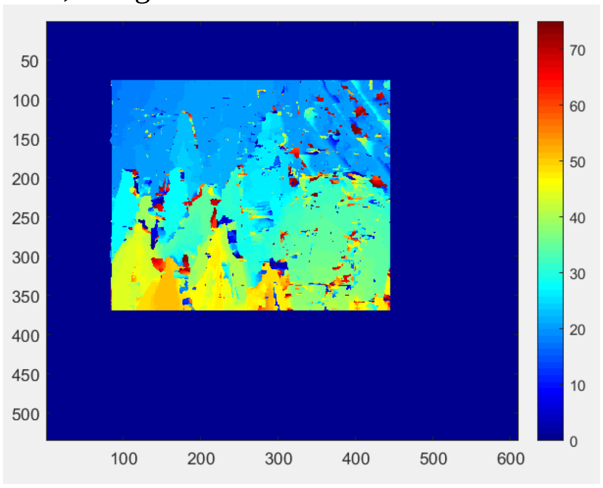
As can be seen above, increasing k will cause to reduce the details in disparity map. Because it behaves to more number of pixels in the same way as they are having the same disparity. Therefore, disparity map will be in less detail.

k = 5, omega = 50



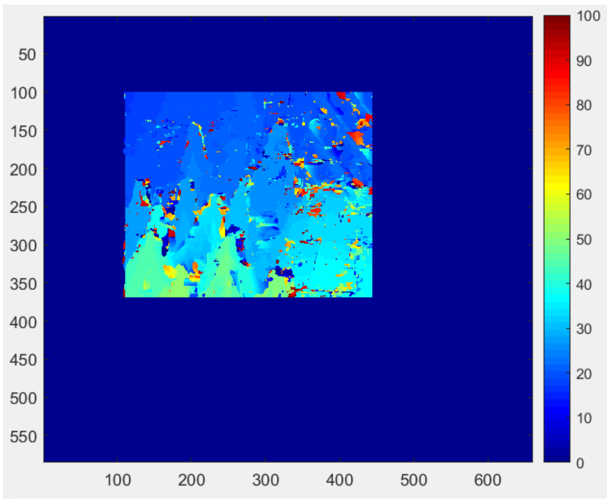
Elapsed time is 25.561268 seconds.

k=5, omega = 75



Elapsed time is 38.793810 seconds.

k=5, omega = 100



Elapsed time is 46.626053 seconds.

Size of the search window is another important factor because as can be seen above, while it is increasing, the computational cost is also increasing which can be understood by the elapsed times. Also, increasing omega corresponds to sliding more through the row which means searching in higher area. In terms of disparity, it shows little changes in the higher area (omega=100). Therefore, it did not detect small changes properly due to vastness of the search space. We can conclude that 50 pixel is enough for our case, this size can be understand by looking at the images and estimating the change between images which may give more robust results.

CODES

lab7.m

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

[row,col,ch] = size(imgR);
imgR = rgb2gray(imgR);
imgL = rgb2gray(imgL);

imgR = double(imgR);
imgL = double(imgL);

omega = 100;
k = 5;

imgL = padarray(imgL,[omega+k omega+k],'both');
imgR = padarray(imgR,[omega+k omega+k],'both');

dispar = zeros(size(imgL));

tic
    for i = k+1:1:row-k-1
        for j = k+1:1:col-k-1
            dist = [];
            subImL = imgL(i-k:i+k,j-k:j+k);
            for p = 0:1:omega
                subImR = imgR(i-k:i+k,j-k+p:j+k+p);
                SSD = sum(sum(subImL-subImR).^2);
                dist = [dist; j j+p SSD];
            end
            ind = find(dist(:,3) == min(dist(:,3)));
            ind = ind(1);
            d = dist(ind, 2) - dist(ind, 1);
            dispar(i,j) = d;
        end
    end
toc

imgL = uint8(imgL);
imgR = uint8(imgR);
imshow(stereoAnaglyph(imgL,imgR));
figure; imagesc(dispar); colormap jet; colorbar
```