EE417 - Assignment 7 - Post-Lab Report

Baris Sevilmis
Lab Date: 04/12/2018
Due Date: 10/12/2018

Overview

Seventh lab assignment of EE417 focuses on first half of the Stereo Vision problem. As known, Stereo Vision indicates reconstruction of the 3D world given multiple images of the same instance from different rotations and translations. This reconstruction phase consists of two steps that are the following:

- Correspondence Problem/ Feature Matching
- Reconstruction of the 3D world

Therefore, main focus of this lab assignment will be on solving the correspondence problem.

Stereo Vision: Correspondence Problem

Correspondence problem, or in advanced terms feature matching is an essential part of Stereo Vision problem. Correspondence problem refers to the corresponding pixel values in multiple images given the images have same subset of features, in which same feature appears with a different rotation and translation. As already known, image correspondence is due to the epipolar constraint. Figure 1 depicts the visualization of epipolar constraint. Given feature X, it is impossible to detect its depth with only a single camera. With multiple cameras lost dimension of depth can be recovered in addition.

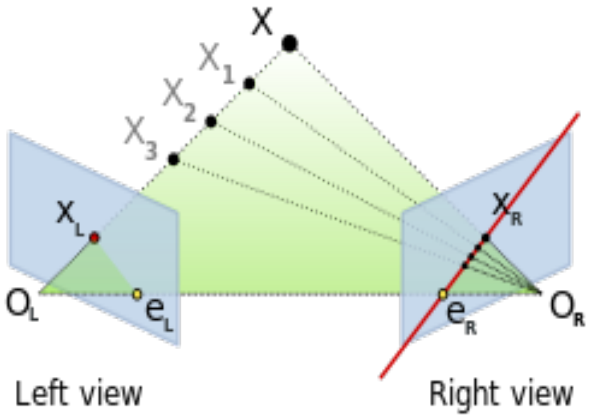

Figure 1: Epipolar constraint

In our problem, main purpose is to find the X point, which is matching in both O_R & O_L. Depth recovery belongs to the reconstruction of 3D world, therefore won't be the crucial focus of this assignment. In order to find corresponding feature in both of the images, we must check the epipolar lines. Epipolar line implies the line between

intersections of baseline with image planes/projections of the other camera center and intersection of feature point with image planes. These lines can be seen in Figure 1 as lines between $x_l - e_l$ & $x_r - e_r$. By the epipolar constraint, it is clear that if x_l is detected point by left camera, right camera should detect x_r on the given epipolar line no matter the depth of X. $x_1, x_2, x_3, .., x_n$ will be all on the epipolar line.

Therefore, once the features are established on first image, epipolar line on the second image will provide the corresponding feature coordinate. In the lab assignment, correlation based matching method will be used to match these coordinates. Search will be along the right image, in which same coordinates will be chosen as start points. A window with size $\omega X \omega$ will be used to slide across the right image in order to calculate similarity between two image windows. Similarity refers to a distance measure between two chosen windows, where maximum similarity will indicate the true correspondence. Similarity will be computed for each possible displacement in right image as following:

$$d = [d1, d2]$$

$$C(d) = \sum_{k=-\omega}^{k=\omega} \sum_{l=-\omega}^{l=\omega} \psi(f(i+k, j+l), g(i+k-d_1, j+l-d_2))$$

ψ stands for the similarity measure that can be calculated as Sum of Squared Differences(SSD) as the following:

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

These displacements are kept in a matrix, and when the computation for the specific feature ends, displacements with minimum SSD's are chosen as the corresponding coordinates. This technique will be used to find every corresponding pixel. As a consequence, these correspondences will be used to create the disparity map, in which following formula is used for calculation:

$$d = x_{left} - x_{right}$$

As for the results of the correlation based matching technique, two different image sets are used both with the lab implementation and built-in version providing enough results to discuss. Listing 1 provides MatLab Code for correlation based matching including plot functions and built in disparity function. Before discussing the results, there are a few points to be cleared out. Anaglyph depicts the stereo pair on top of each other visualizing their rotational & translational differences. Padding is used for the search through displacements as the $\omega X \omega$ window may overflow out of the image given no padding. Last but not least, it can be seen that similarity measurement is made between $(2k+1)X(2k+1)$ windows, as all image is traversed with these windows. Instead of searching displacement of whole image at once, windowing ensures decomposition of the problem into subproblems, in which window based matching returns more reliable and accurate results in addition to the computational efficiency. In other words, we proceed to use our regular $(2k+1)X(2k+1)$ window besides the displacement window because of the exact same reason as previous algorithms and implementations.

Listing 1: MatLab Code for Correlation Based Matching

```
function lab7(img_left,img_right, k, w, offset)
    close all;
    [row, col, ch] = size(img_left);
    % k = 3, w =50, offset = 50 for S01L.png and S01R.png
    % k = 5, w =40, offset = 40 for S00L.tif and S00R.tif
    % Show stereo pair in a red-cyan anaglyph
    figure(1);
    subplot(2,2,1); imshow(img_left);title('Left Image');
    subplot(2,2,2); imshow(img_right); title('Right Image');
    subplot(2,2,[3 4]); imshow(stereoAnaglyph(img_left,img_right
       ));title('Stereo pair in a red-cyan anaglyph');
    if (ch == 3)
        img_left = rgb2gray(img_left);
        img_right = rgb2gray(img_right);
    end
    % Pad the image by offset amount
    paddedImL = padarray(img_left,[offset offset],'both');
    paddedImR = padarray(img_right,[offset offset],'both');
    paddedImL = double(paddedImL);
    paddedImR = double(paddedImR);
    dispar = zeros(size(paddedImL));
    for i = k+1+offset: 1 : row - k - 1 +offset
        for j = k+1+offset: 1 : col - k - 1 +offset
            dist = [];
            subL = paddedImL(i-k: i+k, j-k: j+k) ;
            for m = j:-1:j-w
                subR = paddedImR(i-k: i+k, m-k: m+k) ;
                SSD = sum(sum((subR - subL).^2));
                dist = [dist; j m SSD];
            end
            ind = find(dist(:,3) == min(dist(:,3)));
            ind = ind(1);
            dispar(i,j) = (dist(ind,1) - dist(ind,2));
        end
    end
    % Show disparity map with colorbar
    figure(2); imagesc(dispar); colormap jet; colorbar;
    %Built-In method
    disparityRange = [0 48];
    disparityMap = disparity(img_left, img_right, '
       DisparityRange', disparityRange);
    figure(3); imshow(disparityMap, disparityRange); title('
       Built-In Disparity Map');colormap jet; colorbar;
end
```

First set of images to be processed are demonstrated in Figure 2. Left image, right image and their anaglyph can be seen in the Figure 2. As for the disparity map, Figure 3 depicts our lab implementation version of disparity map. Parameters that are used for the mapping can be seen on the caption of Figure 3 such as k = 5.

Figure 2: Image Set-I, Man in the Park

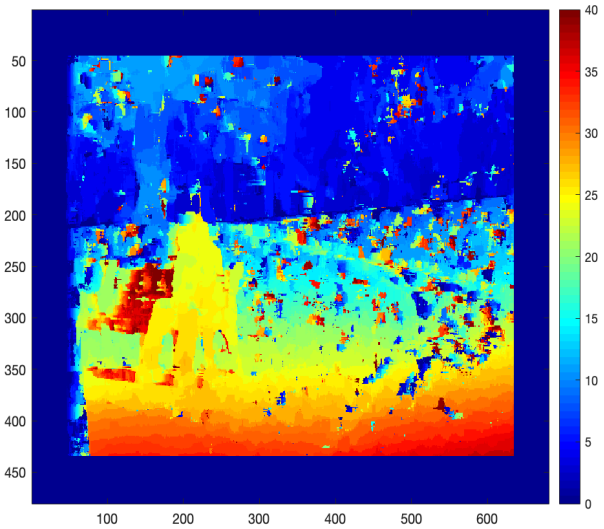

Figure 3: Disparity Map with k = 5, $\omega = 40$, padding = 40

To see the alterations due to the parameter changes, Figure 4 depicts same same correlation based matching with a smaller window size(k = 3). As it is clearly shown, smaller window causes more detail to be taken in consideration and increased noise. On the other hand, Figure 5 depicts same method but with larger window size(k = 9). Larger window results in smoother disparity maps, and is much less prone to noise.

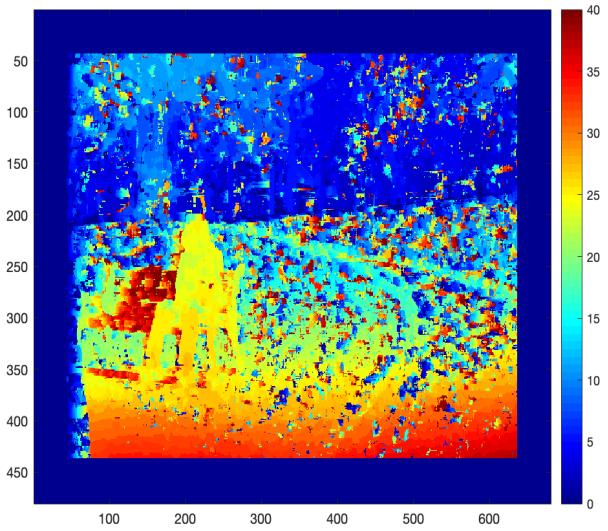

Figure 4: Disparity Map with k = 3, $\omega = 40$, padding = 40

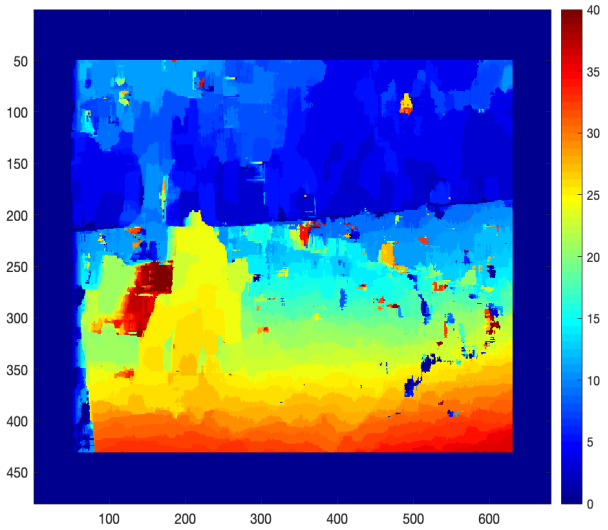

Figure 5: Disparity Map with k = 9, $\omega = 40$, padding = 40

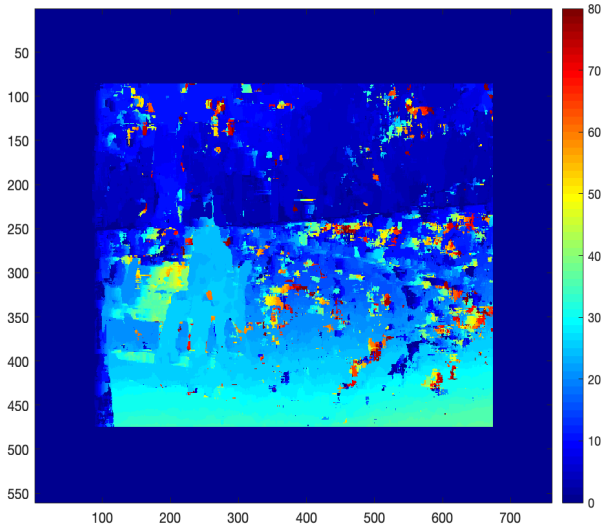

Figure 6: Disparity Map with k = 5, $\omega = 80$, padding = 80

There exists also two more parameters, that are padding and ω. These parameters are interconnected to each other such that increase in ω requires increase in padding, otherwise $\omega X \omega$ window can get out of boundary error. For convenience, Figure 6 demonstrates same operation with padding and ω equal to 80. Figure 6 can be compared with Figure 3 in terms of padding and ω. It is clearly represented that adding more padding and increasing ω results in smaller image and decrease in image colors. Greater the padding and ω, more padding area included with lowest values and therefore, image values also decrease as $\omega X \omega$ window traverse through the image.

Lastly, Figure 7 represents built-in disparity function instead of lab implementation. Color range is same as previous images. It becomes clear that built-in disparity implementation is optimized such that provided disparity map is much more smooth than the lab implementation, because of the default window size. Some details are lost in the procedure, as we know greater windows ensures smoothing but loses some details. On the other hand, Figure 8 depicts built-in disparity function, with block size of $11X11$ as in Figure 3. Therefore, performance of built-in disparity with lab implementation can be compared as the window size are same, by simply comparing Figure 3 and Figure 8. Difference in details can be seen between Figure 7 and Figure 8. Smaller window size results in better detail detection, but larger window provides better smoothing.

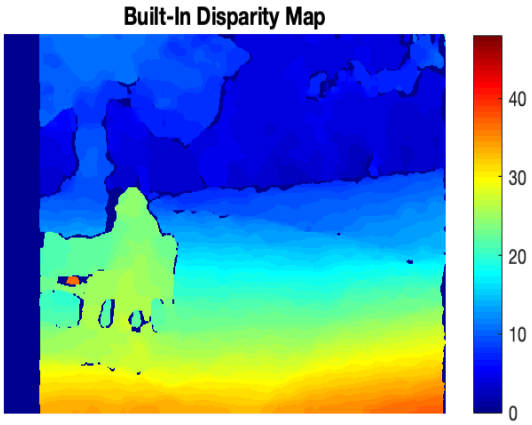

Figure 7: Built-in Disparity Map with Block Size = 15X15

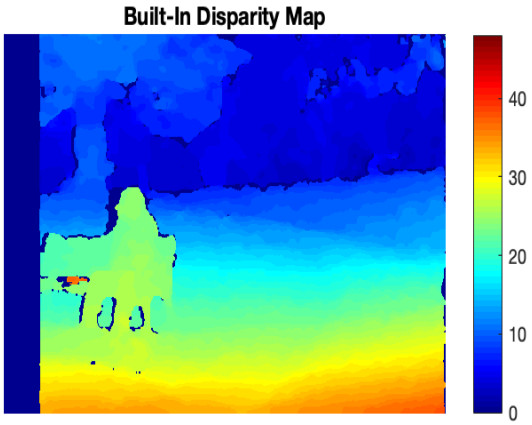

Figure 8: Built-in Disparity Map with Block Size = 11X11

For the sake of clarity, same procedure is applied to the mask stereo-pair, for which same set of operations are executed in the same order. Figure 9 demonstrates the left and right images, and anaglyph of the stereo pair. Figure 10 represents the lab implementation disparity map. In order to obtain same results as in the Lab Assignment Report, parameters were tuned such that k=3, $\omega = 50$ and padding = 50. In Figure 11, windows size is decreased to 3X3, in other words k is equal to 1. It is clear that, detail detection increased as the noise increased. On the other hand, Figure 11 represents disparity map with windows size of 19X19, to be more specific k is equal to 9. Result is much more smoother with less details detected.

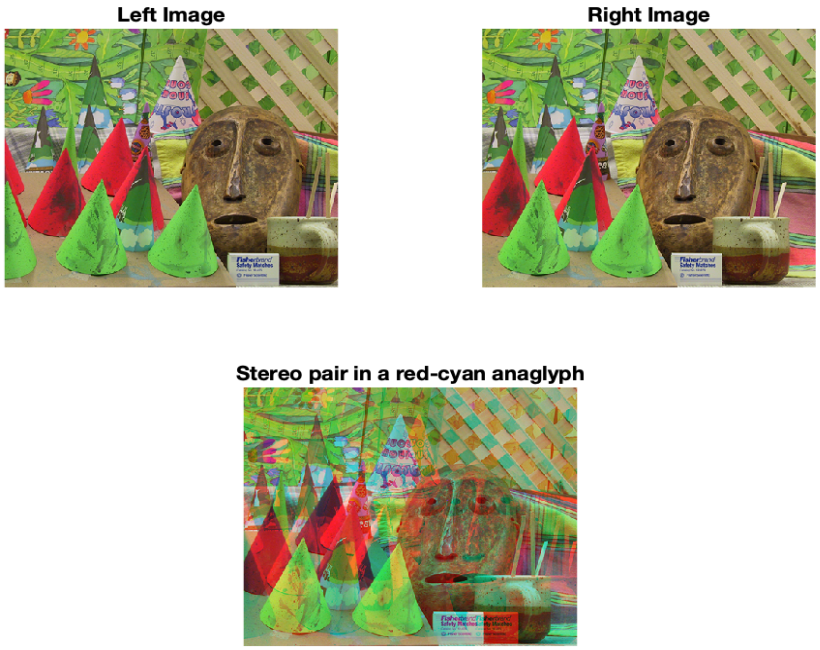

Figure 9: Image Set-I, Man in the Park

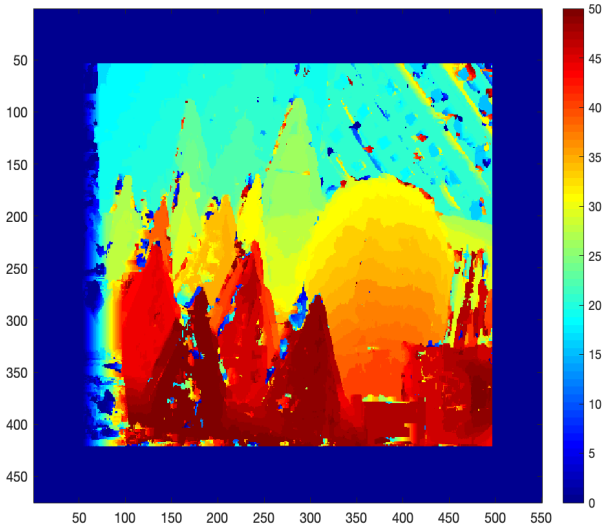

Figure 10: Disparity Map with k = 3, $\omega = 50$, padding = 50

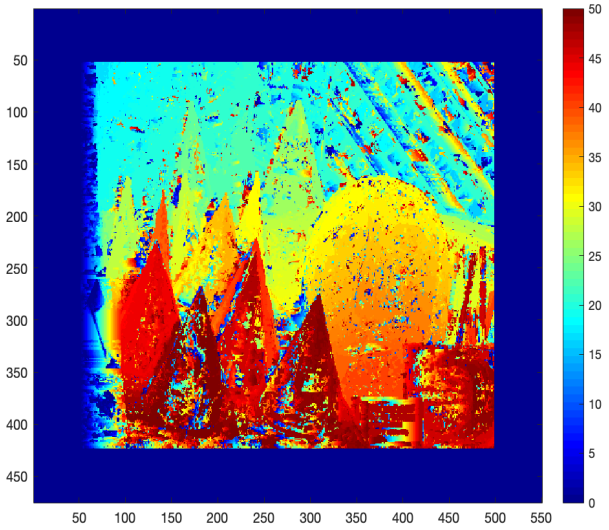

Figure 11: Disparity Map with k = 1, $\omega = 50$, padding = 50

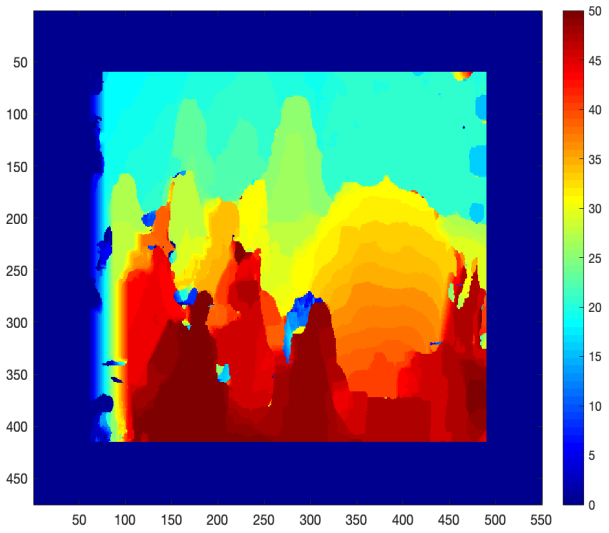

Figure 12: Disparity Map with k = 9, $\omega = 50$, padding = 50

In Figure 13, ω and padding are increased to 80 with k = 3 such that Figure 13 can be compared with Figure 10. Same as the last time, color intensities dropped because greater padding and window with larger ω causes more padding are to be involved in windowing process.

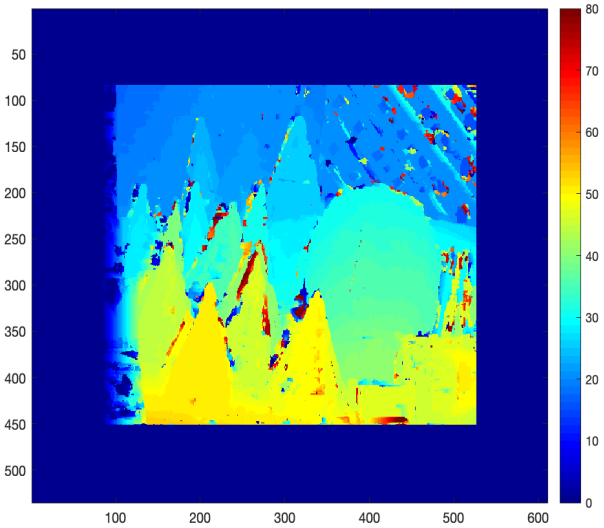

Figure 13: Disparity Map with k = 3, $\omega = 80$, padding = 80

Lastlty, Figure 14 provides the built-in disparity result with block size of 15X15. For comparison purposes, Figure 15 ensures the block size of 7X7, namely k = 3. For this reason, Figure 15 can be compared with Figure 10. Results are more clear in built-in function as expected. In other words, it seems to be built-in function provides better smoothing with better feature detection, though window size used for lab implementation and built-in disparity function are same.

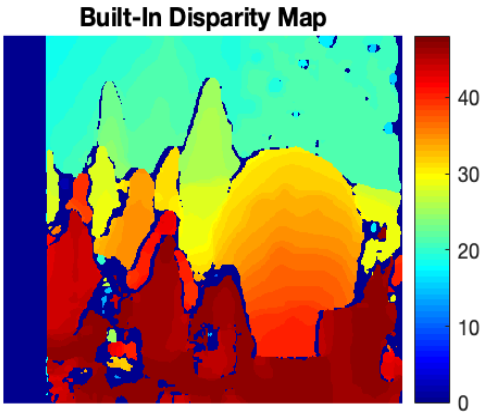

Figure 14: Built-in Disparity Map with Block Size = 15X15

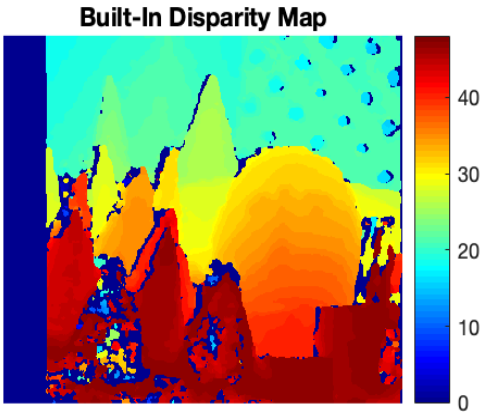

Figure 15: Built-in Disparity Map with Block Size = 7X7

Conclusion

Correspondence problem with correlation based matching were provided, as effects of parameter change were depicted. For instance, increasing k improved smoothing but reduced detail detection. On the other hand, decreasing k caused vice versa situation. Lastly, built-in disparity function was used in terms of measuring accuracy and efficiency of lab implementation. Built-in parameters were arranged such that they were comparable with lab implementations. In addition, block size parameter was also altered and changes were compared. For detailed analysis, upper section should be read, as all the necessary details were provided.