Computer Vision LAB#6

Report Submitted By Ahmed, Muhammad Farhan , EMARO

Introduction

Given two or more images of the same 3D scene, taken from different points of view, the correspondence problem refers to the task of finding a set of points in one image, which can be identified as the same points in another image. To do this, points or features in one image are matched with the corresponding points or features in another image. The images can be taken from a different point of view, at different times, or with objects in the scene in general motion relative to the camera(s).

A small window is passed over a number of positions in one image. Each position is checked to see how well it compares with the same location in the other image. Also Several nearby locations are also compared for the objects in one image may not be at exactly the same image-location in the other image. It is possible that there is no fit that is good enough. This may mean that the feature is not present in both images, it has moved farther than your search accounted for; it has changed too much, or is being hidden by other parts of the image.

LAB Assignment

Disparity maps

1. write a function `compute disparity` that takes:

- two rectified images, I1 and I2
- the size of pixel neighborhoods W
- a vector containing the minimum and maximum disparity possible from image I1 to I2 [dmin, dmax]

and produces (returns in output) the disparity map D (of I1 with respect to I2) where for each pixel we store its disparity value.

2. The function will visit all points p=(i,j) of image I1 (with the exception of the ones in the external frame of thickness W/2) and find the "best" corresponding points in I2 (looking at the search range [j+dmin, j+dmax]).

The similarity between image patches should be evaluated with SSD (we suggest you use an ad hoc function `my_ssd` that takes two neighbourhoods as inputs and returns an integer similarity value as an output)

Left-Right Consistency

1. Write a function `left_right_consistency` that takes two disparity maps (from I1 to I2, and from I2 to I1) and then checks the consistency of the results, assigning a "special" value to points without correspondence: it could be a large value. An alternative is to use 0 .

Main

You are encouraged to provide a main file that reads two images and includes the following

```
W=...
dmin=...
dmax=...

D12=compute_disparity(I1,I2,W,[dmin,dmax]);
D21=compute_disparity(I2,I1,W,[-dmax,-dmin]);
D=left_right_consistency(D12,D21);
```

visualize all three disparity maps noticing that they are integers with a sign and (only for visualization purposes) need to be shifted on the appropriate range of values (if needed) and possibly stretched to improve visibility

Procedure/steps in main program

The goal of this lab was to find correspondences between rectified images. Some pairs of rectified images were given. The procedure adopted was follows

LAB_6.m

1. Read both images as I1 and I2.
2. Select window size e. g 2
3. Select maximum and minimum disparity allowed dmin and dmax.
4. If image is color or 3D covert it to grayscale using Rgb2gray.
5. User can also convert image pixel values to double precision floating values for example noise suppression in large images.
6. Call the function compute disp and pass I1,I2,w,dmin,dmax.
7. The value returned by this function is disparity map from left to write.
8. Display image by calling imagesc and converting it to uint8 (for displaying) only.
9. Compute right to left disparity by passing compute-disp(I2,I1,w,-dmax,-dmin)in similar manner.
10. Since some values will be negative we should use the abs () function for displaying.
11. Call the function left_right_con and pass DLR and DRL as arguments.
12. The function simply adds the two matrices together and gives disparity map corresponding to both images.

Compute_disp.m

1. Takes I1,I2,w,dmin,dmax and returns corresponding disparity matrix.
2. Form I1 starting for w/2 + some scaling factor we compute sum to square of differences for (j+dmin,j+dmax) neighborhoods of corresponding I2
   Using the formula

$$\phi_{SSD}(N1, N2) = -\sum_{k,l=-\frac{W}{2}}^{\frac{W}{2}} (N1(k,l) - N2(k,l))^2$$

3. We stack each result in a dmax x 1 column vactor and compute Max of it.
4. This Max values is or maximum disparity value. We save it and increment or row count (for I1) and continue as before.
5. At the end we get complete disparity(left to right or right to left) map for our to images with exception of W/2 + some scaling factor at edges.

$$c(d) = \phi(N1(i,j), N2(i,j+d))$$

$$\bar{d} = \text{argmax}_{d \in [d_{min}, d_{max}]}\{c(d)\}$$

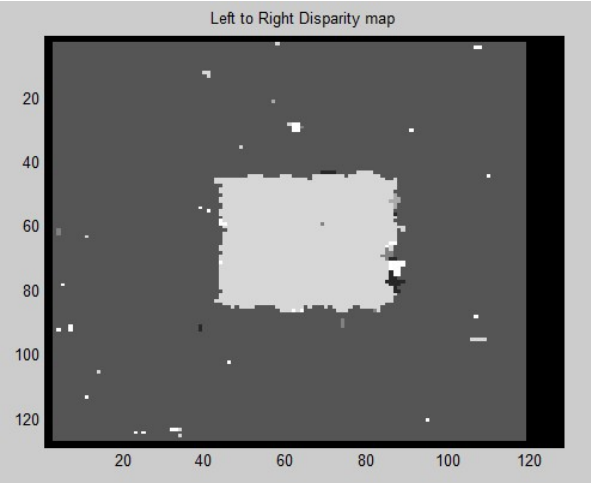
left_right_con.m

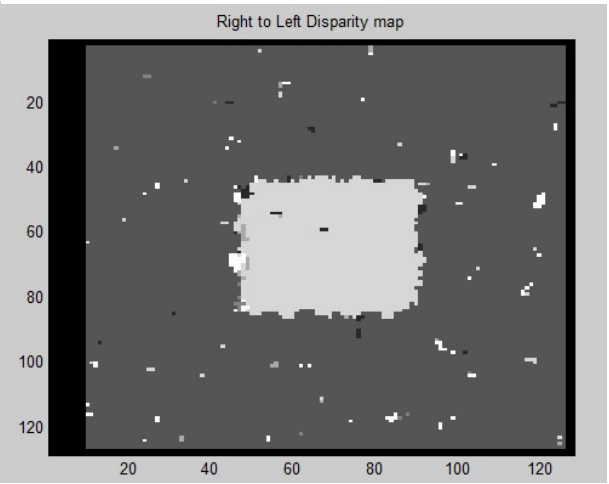
1. This function takes the two disparity matrices and combiles them together to obtain full disparity(left to right and right to left) for both the images.

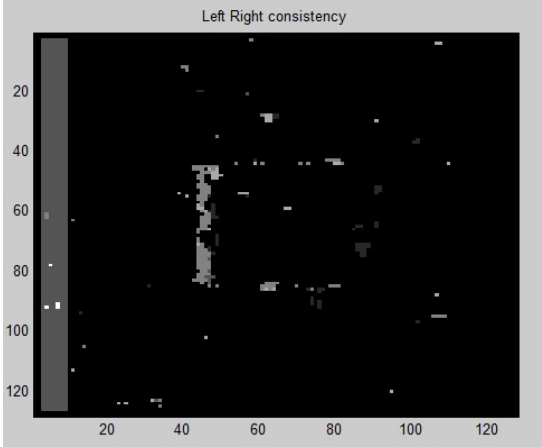
RESULTS

Niegnborhood = 4

Dmin = 1 Dmax = 6

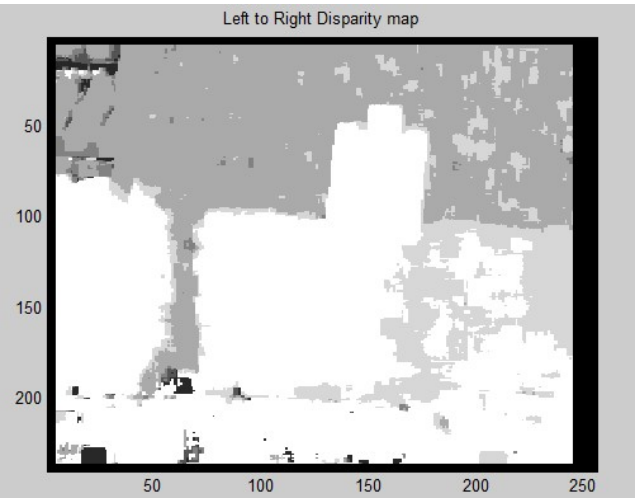


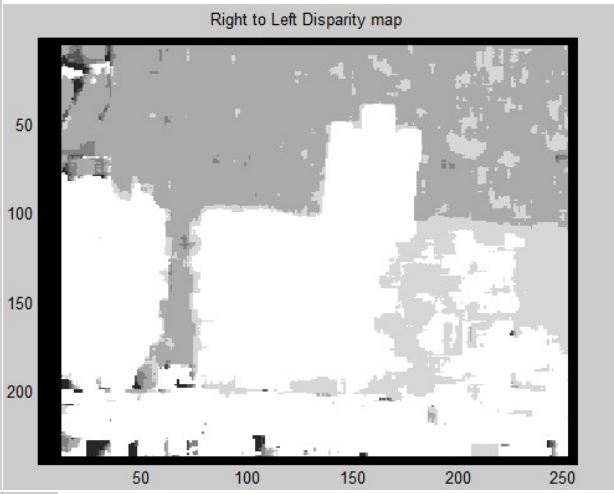


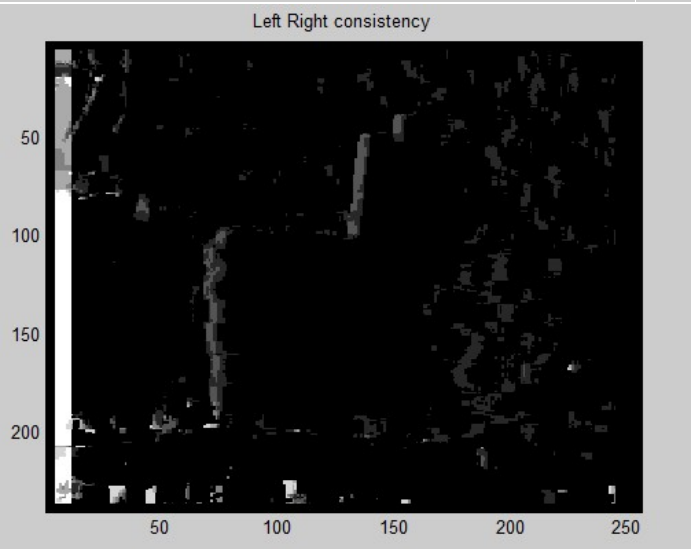


Niegnborhood = 8

Dmin = 1 Dmax = 6







OBSERVATIONS/Findings

As size of Neighborhood increases Localization increases.