CSE473/573 Computer Vision and Image Processing

SUMMER 2016 PROGRAMMING ASSIGNMENT #2

Harsha Sudarshan
harshasu@buffalo.edu

1. Estimation of disparity using Block matching technique

The objective is to get estimate of Disparity Map using Block matching technique on given stereo images. The window sizes to be used are 3x3 and 9x9.

Because we are using stereo-corrected images, the comparison between image blocks can be restricted to one direction (x-axis). The measure used for finding the matching blocks is by using SSD – Sum of Square Difference.

The formula for SSD calculation is:

$$E(x, y; d) = \sum_{(x',y')\in N(x,y)} [I_L(x'+d, y') - I_R(x', y')]^2$$

d is the *disparity* (horizontal motion)

For a particular block on reference image, the block on second image with the least SSD gives the best-match pixel. Using this methodology, the disparity matrices were calculated using the given images.

Given below are the images, ground-truth and their estimated disparity maps:

Fig1: View1 and its ground truth

Fig2: view5 and its ground truth

a. With a block-size or window-size of 3x3

As can be expected, the smaller block-size gives a sharper disparity matrix but because of this it is distinctly not smooth.

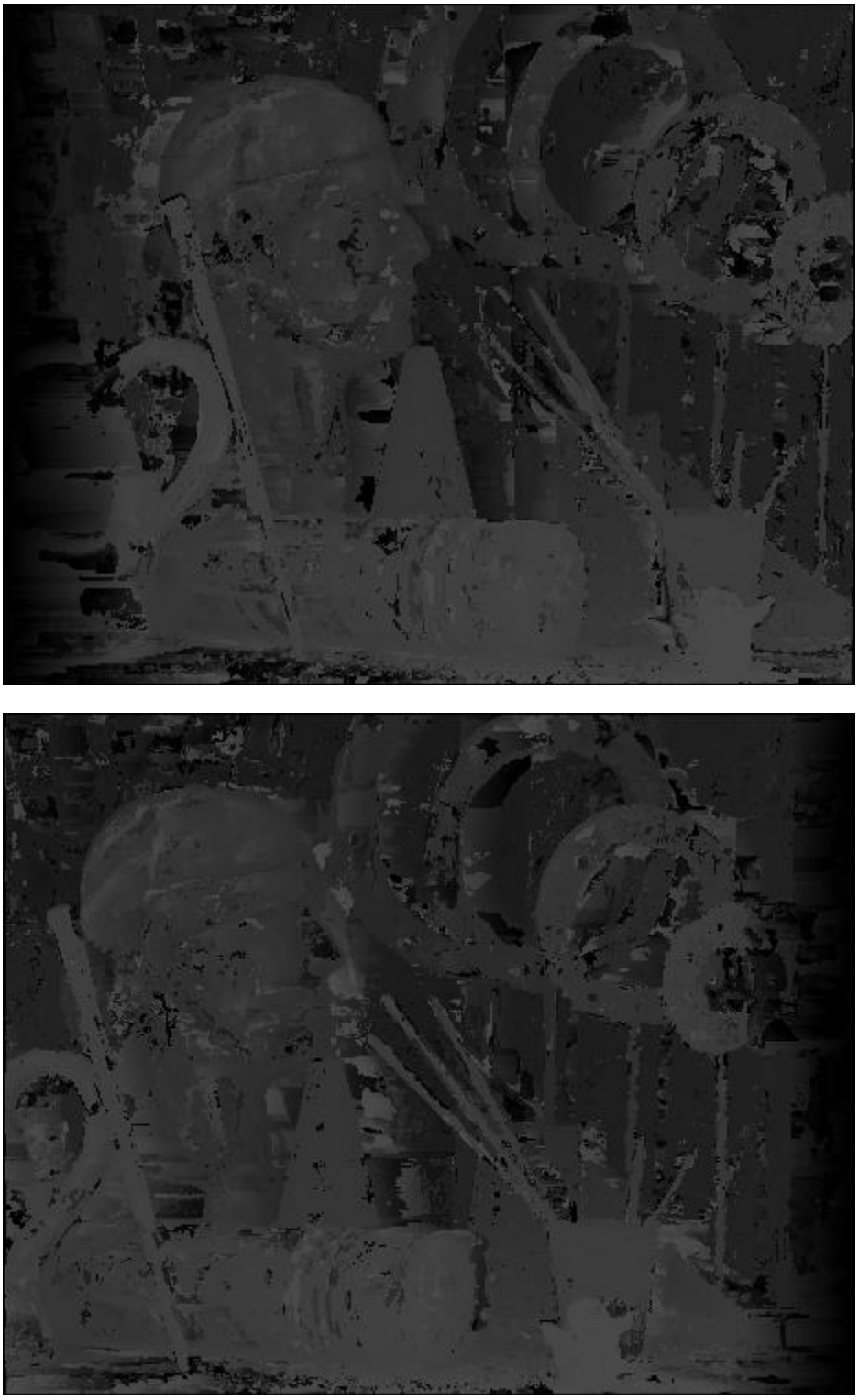

Fig3: Disparity Maps of the images using 3x3 block

b. With a block-size or window-size of 9x9
As can be expected, the bigger block-size gives a smoother disparity matrix.
Given below are their estimated disparity maps:

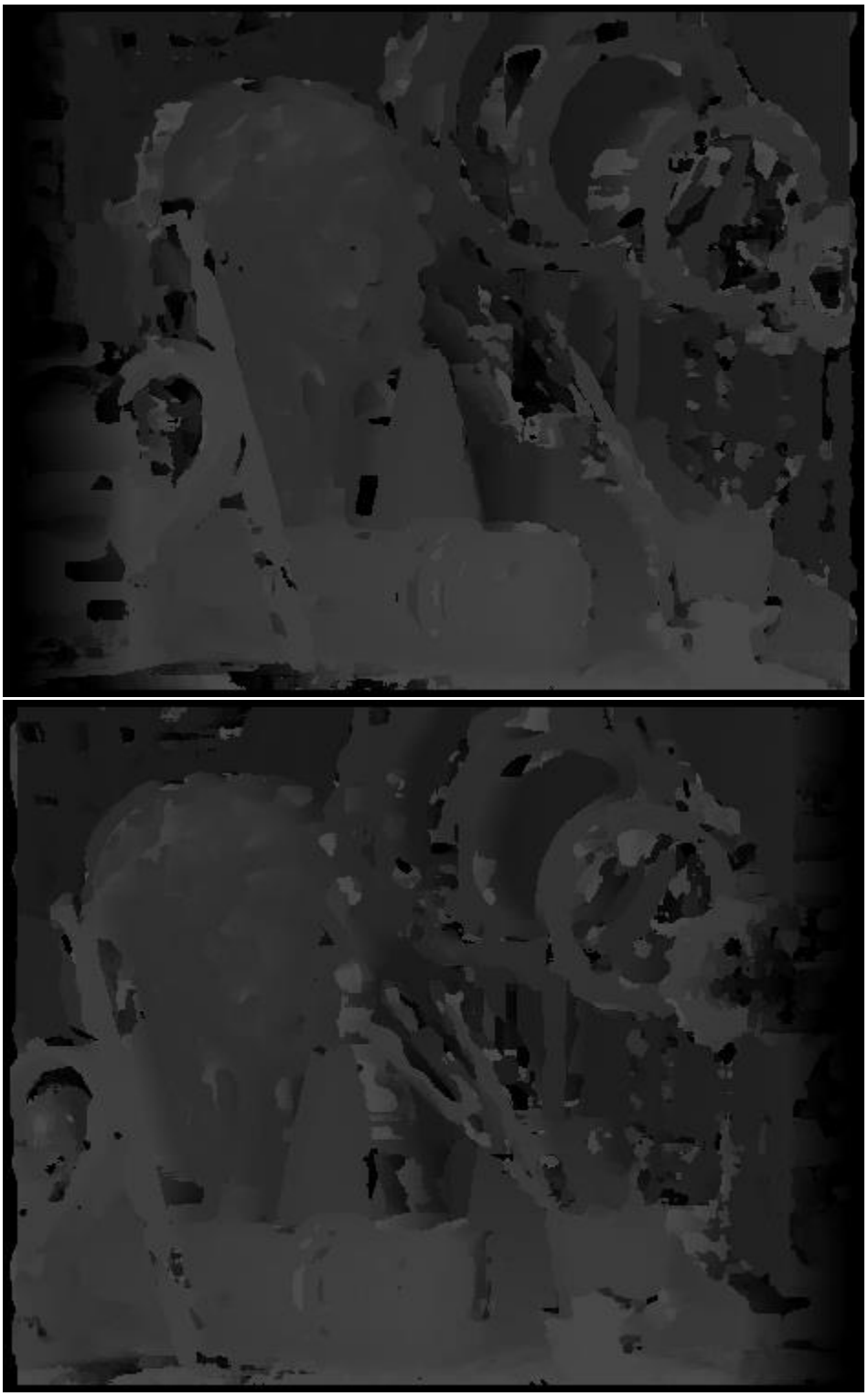

Fig4: Disparity Maps using 9x9 block size

c. Use of back-propagation method to refine the estimated disparity maps – Consistency Check

The estimates obtained can be further refined by using the back-propagation method and using both the estimated matrices.

In this method, the pixel value in one estimated matrix is compared against the estimated matrix of the corresponding stereo-pair. The disparity value is used to traverse the second matrix and find the corresponding position of a pixel. If the two values match, the pixel value is retained, else it is set to zero(0).

To check the error of the estimates, MSE(Mean Square Error) is used:

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

The MSE values for the images, window-sizes and refinement are as follows:

- For view 1:
  - Window-size:3x3
    - Without consistency check
      - MSE = 295
    - With consistency check
      - MSE = 39
  - Window-size:9x9
    - Without consistency check
      - MSE = 255
    - With consistency check
      - MSE = 28
- For view 2:
  - Window-size:3x3
    - Without consistency check
      - MSE = 211
    - With consistency check
      - MSE = 40
  - Window-size:9x9
    - Without consistency check
      - MSE = 182
    - With consistency check
      - MSE = 30

Given below are their estimated disparity maps:

Fig5: Disparity Maps using 3x3 block size and performing a consistency check

Fig5: Disparity Maps using 3x3 block size and performing a consistency check

2. Use of Dynamic Programming to estimate disparity

Another approach to estimating the disparity values between a stereo image pair is by the use of dynamic programming to find the matching pixels.

This method is based on dynamic programming paradigm of finding the longest common subsequence (LCS) between two strings or arrays. The LCS is applied to the Direction Matrix, which is obtained after calculating the cost.

Taking one row at a time, from each of the images, first the values for the Cost matrix is calculated and then the Direction Matrix. After all the rows are processed, the Direction Matrix is complete. This is used to get the disparity.

The pseudo-code for finding the direction matrix is as given below. For our implementation, the 'Occlusion' value was been fixed at 20.

```
Occlusion = [ln(P_D/(1-P_D) * φ/|(2π)^d S_s^-1|^(1/2))]
for (i=1;i≤ N;i++){ C(i,0) = i*Occlusion }
for (i=1;i≤ M;i++) { C(0,i) = i*Occlusion}
for(i=1;i≤ N;i++){
   for(j=1;j≤ M;j++){
      min1 = C(i-1,j-1)+c(z_1,i , z_2,j);
      min2 = C(i-1,j)+Occlusion;
      min3 = C(i,j-1)+Occlusion;
      C(i,j) = cmin = min(min1,min2,min3);
      if(min1==cmin) M(i,j) = 1;
      if(min2==cmin) M(i,j) = 2;
      if(min3==cmin) M(i,j) = 3;
   }}
```

Fig6: Psuedo-code to create CostMatrix and DirectionMatrix

```
p=N;
q=M;
while(p!=0 && q!=0){
   switch(M(p,q)){
      case 1:
         p matches q
         p--;q--;
         break;
      case 2:
         p is unmatched
         p--;
         break;
      case 3:
         q is unmatched
         q--;
         break;
   }}
```

Fig7: Psuedo-code to find the matching pixels

Fig7: Disparity Maps generated using the above methodology

3. Synthesizing image or creating virtual image

The objective is to create a virtual view or image by using the stereo images and the ground truth maps.

Using a pair of corrected stereo images, a synthetic view can be created by using the ground truth maps as accurate disparity indicators and hence, moving the pixels of either image by half the depth/disparity value.

The below composite image was formed by using the color images and the ground truth maps.

Fig8: Virtual Image created

Fig9: Left image shifted by half the ground truth value

Fig10: Right image shifted by half the ground truth value

4. References:

a. https://en.wikipedia.org/wiki/Block-matching_algorithm
b. http://www.cs.cornell.edu/courses/cs664/2008sp/handouts/cs664-10-stereo.pdf
c. http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/
d. http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-16-Stereo.pdf

5. Code:

The filters were implemented in python using functions available in numpy and opencv.

The code has been submitted in the form on iPython notebook for readability.