Jason Hatfield – 29163434
CSE 473 Programming Assignment #2
Due: 7/12/17

1.1 Disparity estimation using block matching:

The goal of this exercise is to generate disparity map using 3 x 3 and 9 x 9 block matching. Algorithmically speaking, we begin by iterating over each column of each row and compare the minimum sum of square differences between the left image (view1) and the right image (view5).

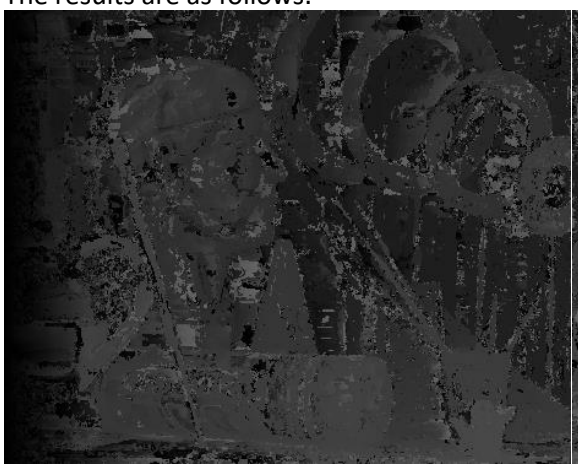

View1


View5

This value will be entered for the selected pixel as the smallest sum of squared differences between the corresponding images. Once the disparity maps have been generated, we compare them with the ground truth disparity maps and calculate the mean squared error using the following equation:

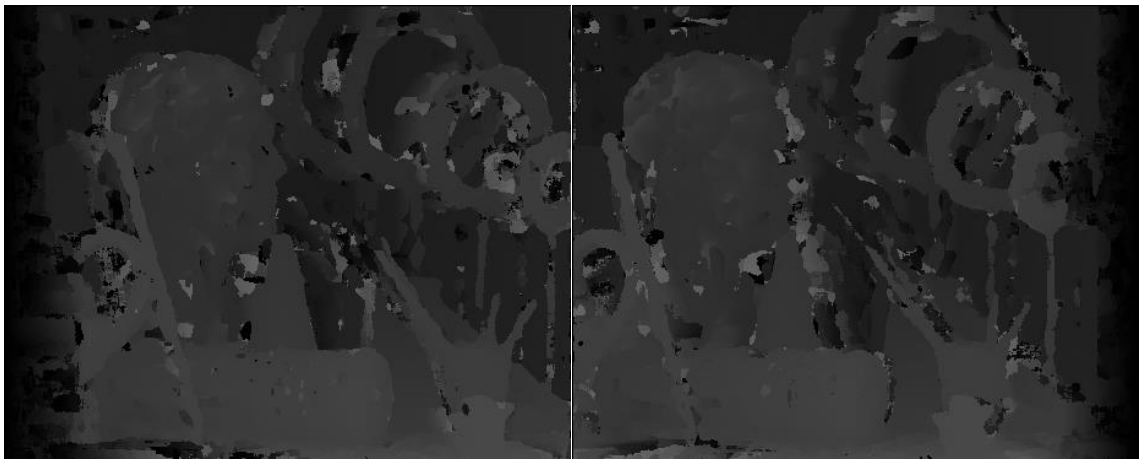
$$MSE = \frac{\sqrt{(Ground\_truth - our\_disp^2}}{\#pixels}$$

The results are as follows:



Left 3 x 3

Right 3 x 3



Left 9 x 9 Right 9 x 9

$$MSE\ for\ left\ 3\ x\ 3\ disparity\ map = 489.108024049$$
$$MSE\ for\ right\ 3\ x\ 3\ disparity\ map = 388.08581917$$
$$MSE\ for\ left\ 9\ x\ 9\ disparity\ map = 348.726036015$$
$$MSE\ for\ right\ 9\ x\ 9\ disparity\ map = 234.438417641$$

We can see that by using the 9 x 9 block matching technique we obtain a smoother disparity map with fewer errant areas of disparity and an overall smaller mean squared error. So overall the larger the block-matching matrix, the more accurate the disparity map will be at the cost of the increased number of calculations needed.

1.3 Disparity Estimation using Dynamic Programming

The goal is to implement disparity estimation using dynamic programming as based on the provided pseudo code:

```
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;
   }}
```

```
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;
   }}
```

The results are as follows:

Left Disparity Right Disparity

We can see that the results are generally smoother than when the block matching algorithm was applied.

1.4 View Synthesis

We are able to generate an image based off the left and right images that would appear to have been taken between the two cameras. By using the given images and ground truth disparity maps we can generate an left and right shifted image and combine them (leaving occluded pixels set to zero) to achieve this task. My results are as follows:


Desired view3


Generated view3

We can see the results are very close to that desired with the exception of a few areas of incomplete information due to areas occluded from either camera.

2 Image Segmentation

Implementing a very basic version of mean shift segmentation allows us to achieve a rudimentary edge detection with minimal complexity. The basic algorithm involves converting the image to a set of 5 – value vectors, randomly selecting a pixel, calculating it's neighborhood within a particular mean tolerance, removing the neighborhood from the set of vectors after adding to the result image, and continuing until all pixels have been removed. Results are as follows:

Original Image

Mean shift – Iter = 10, h = 30

Mean shift – Iter = 10, h = 60

Mean shift – Iter = 10, h = 90

We can see from the results that as the variable h increases, the range that we use to calculate the neighborhood increases and thus our clusters become larger or more tolerant.