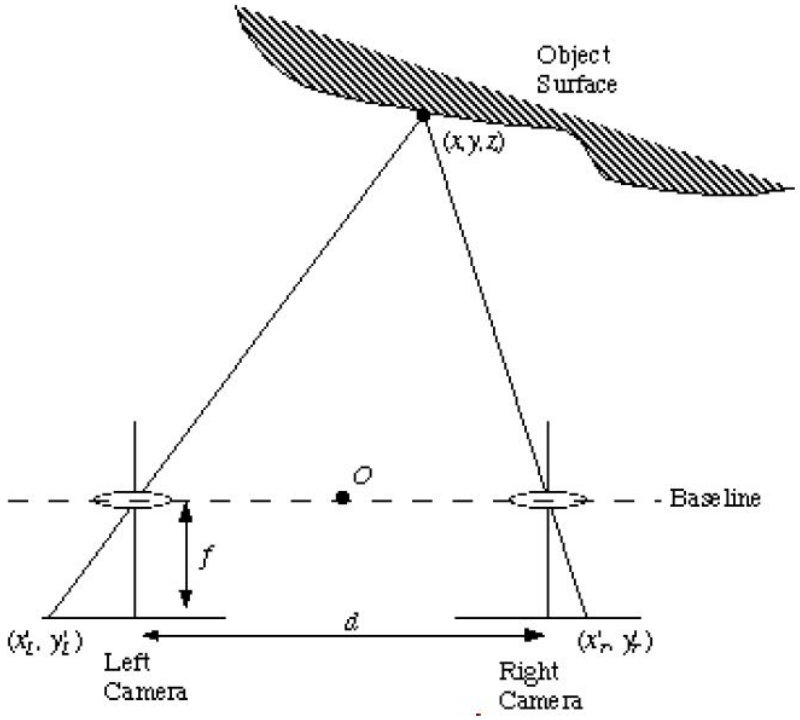
Disparity for Stereo Vision – Block Matching and Dynamic Programming

Prashanth Chandrashekar
50207911
Rakshit Muthappa Padetira
50206908

Date of submission : December 16th 2016

Literature Review:

The calculation of disparity between the two images is necessary to estimate the depth of the object in the image. Current 3D technologies are still based on 3D viewing of stereo pairs. It involves presenting a different image to the left and the right eye. The images are slightly translated to the right or to the left. The two images are obtained for the same object using two cameras separated by a certain distance.



Formula to calculate depth(given by 'z') if we know the distance between the cameras 'd', the focal length of the two camera lenses 'f' and the disparity "$x'_l - x'_r$" is given by:

$$Z = \frac{df}{x'_l - x'_r}$$

The quality is 3D viewing can be calculated by estimating the disparity between the 2 images. The disparity of the objects that are closer to the camera is high while the disparity between the objects that is away from the camera is low. Therefore, nearby objects are accurately constructed in 3D.

Correspondence problem:

The main challenge in calculating the disparity is finding the correspondence in between 2 images i.e, we have to match a pixel in 1 image to the corresponding pixel in the other image. Having found this correspondence, we can estimate the shift of the pixel relative to the other, which is nothing but the disparity.

The first approach is the basic stereo matching algorithm where, for a pixel on a scanline on the left image, we slide a window on the corresponding scan line on the right image and then calculate the SSD (Sum of Squared Differences). We pick the pixel with the least SSD as the matching pixel and then calculate the disparity. We perform this operations under the following assumptions:

- The baseline(d), i.e, the distance between the cameras taking the pictures is small
- Most of the points on the objects are visible on both images

Drawback of the brute force approach as mentioned above is that, if certain parts of the object are occluded i.e, some regions of the object are not seen in an image, owing to the difference in the position of the camera that took the pictures.
To overcome this problem, the dynamic programming approach as implemented in A Maximum Likelihood Stereo Algorithm[1] which associates a fixed cost if an object is found to be occluded. It's a dynamic programming method which essentially finds the longest common subsequence of each scan lines in the 2 images.
The pixels which form the longest common subsequence satisfy the order and uniqueness constraints. The pixels not part of the subsequence represent the occlusions.

Introduction:

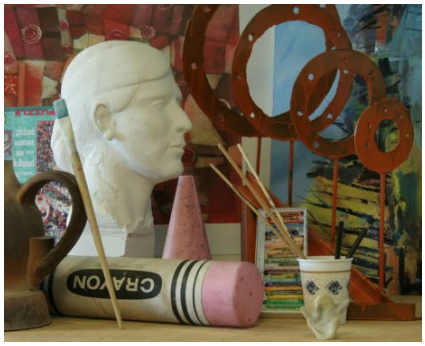
The calculation of disparity between the two images is necessary to estimate the depth of the object in the image. The main challenge in calculating the disparity is finding the correspondence in between 2 images. The current 3D technologies use polarized glasses to present 2 different images to either eyes resulting in 3D perception. 3D TVs use active shutter glasses which are synced with the 3D TV such that, at any instance either the left eye or the right eye is shown the appropriate image. The quality of 3D viewing can be measured using disparity. We use 2 technologies to address the correspondence problem; one is SSD and the other is Dynamic programming.

Our Approach:

We have used 2 technologies to address the correspondence problem:
SSD: For every window in one image find the window in the 2^{nd} image with the least Sum of Squared differences. Having found the pixels calculate the disparity. We analyze the effect of multiple window sizes on disparity.
Dynamic Programming: Consider a block of pixels in a scanline centered around the pixel under consideration. Find the matching pixel on the 2nd image by finding the longest subsequence that matches the considered block of pixels.
Dynamic Programming is a method of solving complex problems by breaking it down into smaller overlapping subproblems and we also store the results of the subproblems so that we need not compute them multiple times.
We use mean square error(MSE) between the ground truth disparity image and the generated disparity image to measure the accuracy of the algorithms.

Outcomes and Deviations:

Test Image: View 1



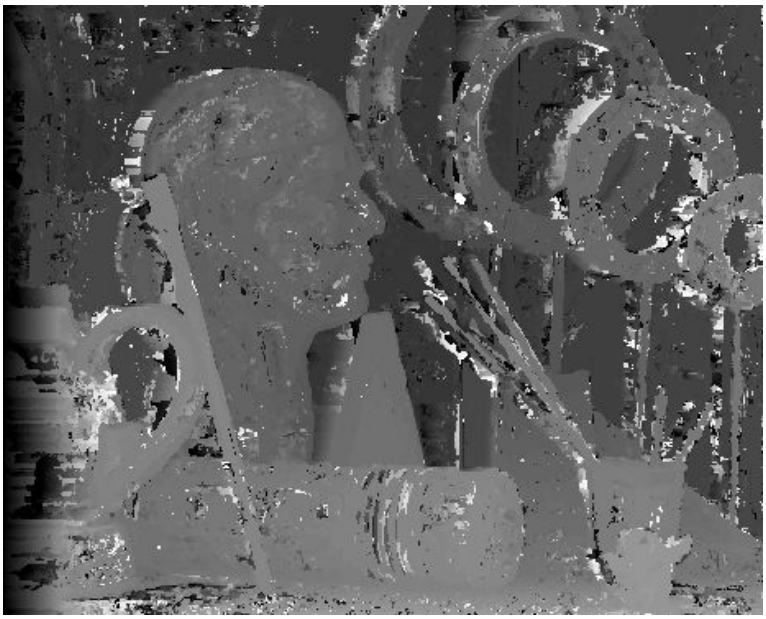
Test Image: View 3

Ground Truth Disparity Map

SSD:

The SSD method was tested with block sizes of 3x3 and 9x9 which produced the below shown disparity maps respectively.

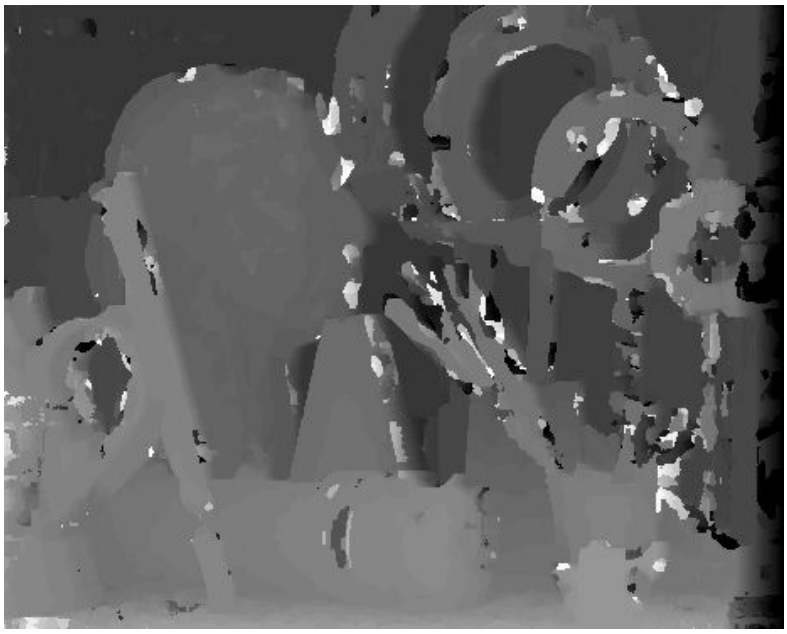
3x3 Block Matching right disparity image



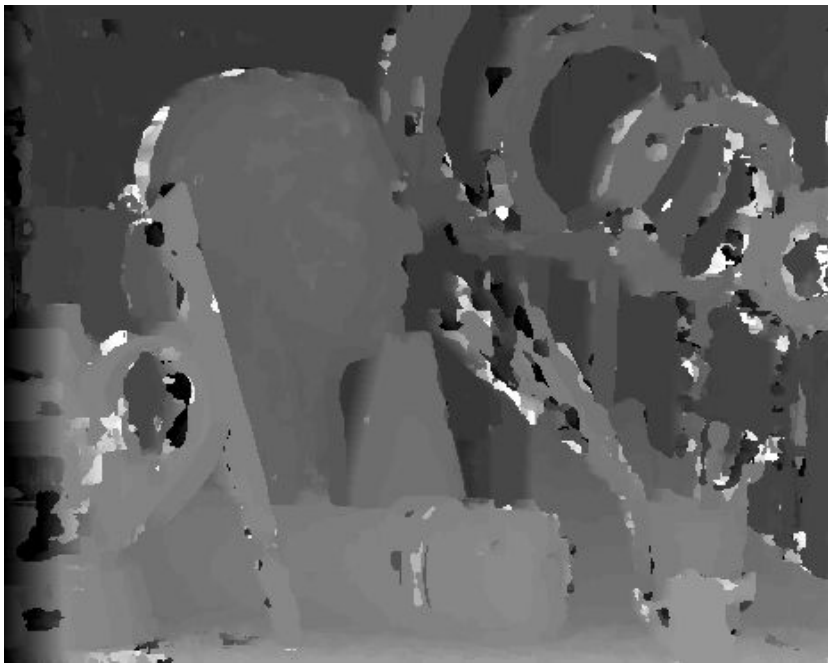
3x3 Block Matching left disparity image

3x3 Block Matching results

Using the 3x3 blocks to find correspondence resulted in a disparity map as shown above. The MSE for the left disparity image was found to be 635.202



9x9 Block Matching right disparity image



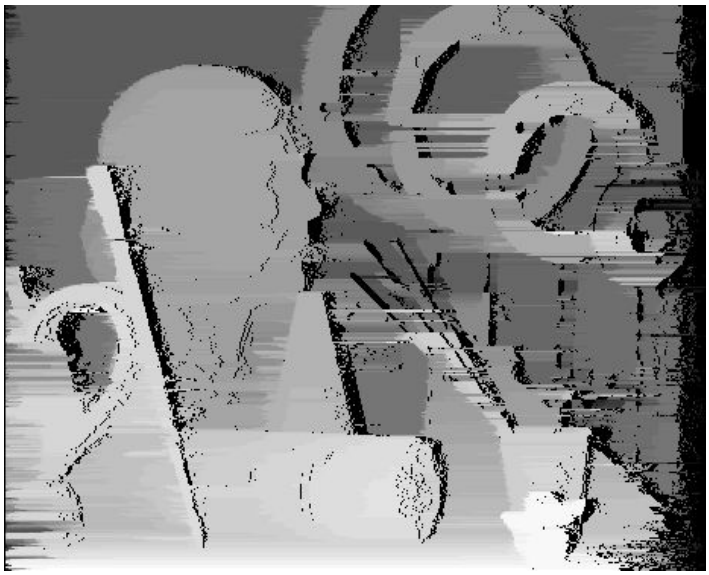
9x9 Block Matching left disparity image

9x9 Block Matching results

We increased the block size to 9x9 to find correspondence and we got the above disparity map. This had an MSE of 607.273 when compared to the ground truth version.

Dynamic Programming:

Using the dynamic programming approach we got the below result with the test images as show. The MSE of this Disparity map against the ground truth was calculated to be 757.525.



Output Image: Right Disparity Map

Output Image: Left Disparity Map

Algorithm Development lessons learned:

In SSD, we can see that the 9x9 block matching will have a lesser MSE, this is because we are finding a better correspondence as we are taking more number of neighbouring pixels into consideration to determine the corresponding pixel in the second image. We also looked at SAD approach, ie, Sum of Absolute Differences, but we could see that this was not that sensitive to changes in the neighbourhood. The SSD approach is more sensitive as we are squaring the difference and hence any minute change/deviation is scaled up considerably.

In case of dynamic programming approach initially we tried to find the correspondence between pixels of the 2 images by considering a sequence of pixels centered around the pixel on left image and searched for a sequence of pixel in the 2nd image in the same scanline. This approach did not factor in the occlusions and was similar to the SSD method. In this case instead of minimizing SSD we tried to maximize Longest common subsequence. This approach did not yield good results for small sequences but it improved as the sequence length was increased. But this algorithm was inefficient and time complex. The dynamic programming approach illustrated in A Maximum Likelihood Stereo Algorithm[1] overcame the above mentioned drawback by factoring in occlusions.

Summary and Discussions:

In this project we have discussed 2 techniques to fine the disparity between 2 images. The 2 techniques discussed are SSD(Sum of Squared Differences) and Dynamic programming. We have noticed that the neighbourhood window size in case of SSD affects the disparity calculation. The bigger the size of the matching block size, the better will be the correspondence between the two blocks from the two images and hence we get a better disparity map. SSD is a brute force method which doesn't consider occlusions in the images, The dynamic programming method is efficient in finding the correspondence between pixels while factoring in the occlusions in the images. The mean square errors we got in all these cases are as follows:

- **3x3 SSD:** 635.202
- **9x9 SSD:** 607.273
- **Dynamic programming approach:** 757.525

Lessons learned:

- Imaging process, stereoscopic vision
- Digitization of images using sampling and quantization
- Representation of images in frequency domain using Fourier Transform
- Extraction of basic features from images like edges, segments, shapes and disparity maps
- Linear and Nonlinear preprocessing techniques: convolution for linear preprocessing and mathematical morphology for nonlinear methods
- Image understanding techniques like ASM, AAM, PDM etc
- Implementation of these techniques in python using relevant libraries like opencv, scipy, numpy etc

Reference list:

[1] A Maximum Likelihood Stereo Algorithm
[2] A Dense Stereo Matching Using Two-Pass Dynamic Programming with Generalized Ground Control Points
[3] http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/