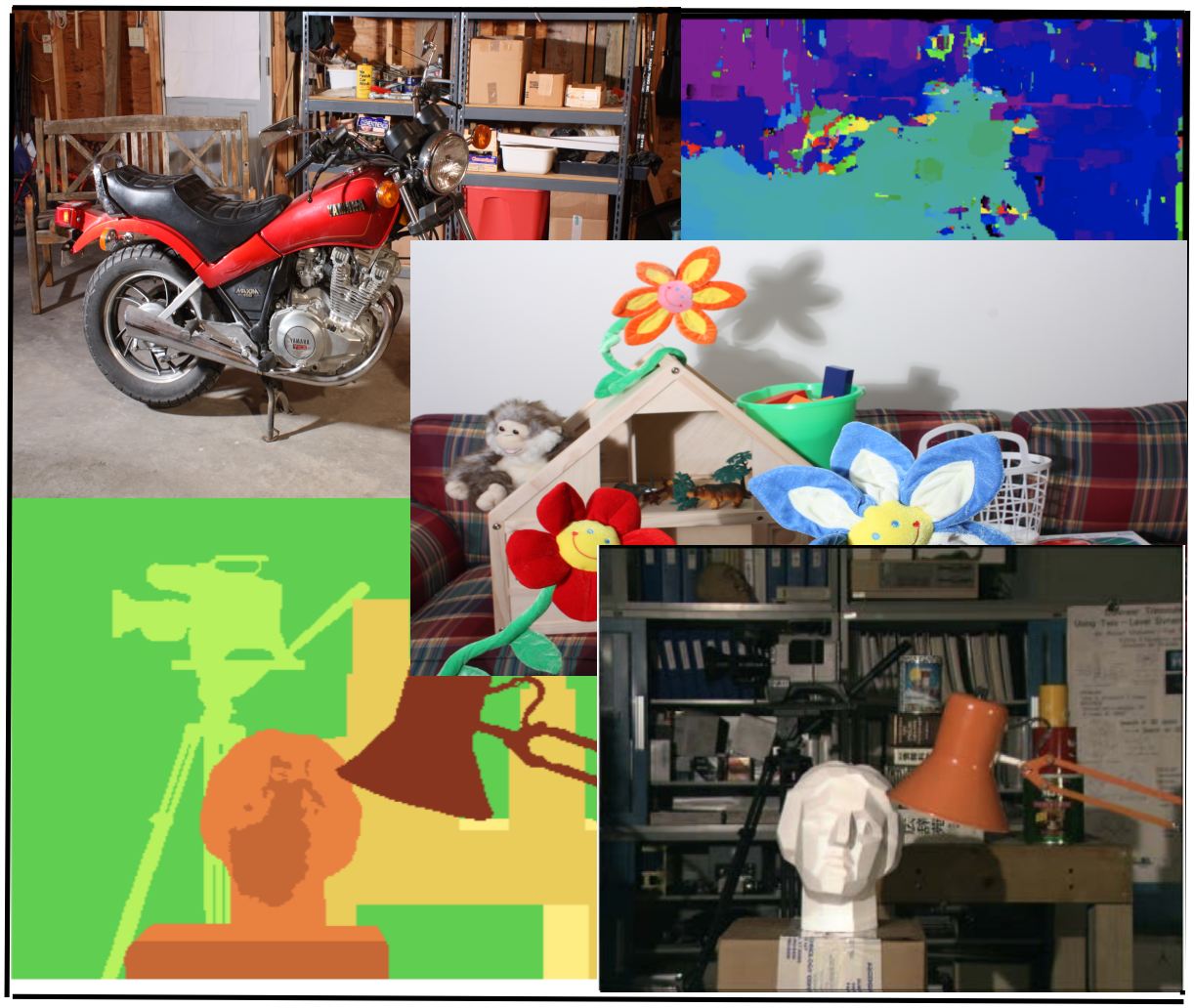
CS-GY6643: Computer Vision and Scene Analysis

Professor Guido Gerig

FINAL PROJECT REPORT

STEREOPSIS

Achyut Tibrewalla (at3617), Binal Modi (bjm470)



Contents

Introduction

Recent developments in multimedia technology is supported by 3D imaging systems to provide a vivid viewing experience. Depth perception introduced by stereoscopic systems, offers a greater realism and has many applications.

In this project, we are going to discuss stereo vision with the help of images captured by translating the camera. This will allow us to discuss depth determination which is not possible by using one image, because due to the perspective projection equations it is impossible to reconstruct depth from a single picture. Here, we are going to show that it becomes possible if we use images taken from different positions which is the basis of human vision system.

In this report, we will first discuss the theoretical aspects of the depth perception by using the two images captured by translating the camera and then we will present their implementation.

The implementation is done using Python and OpenCV. We plotted the disparity, depth map and a 3D view of the depth map using Matplotlib and Plotly.

Approach

We consider a pair of stereoscopic images provided by Middlebury vision and search for correspondence along the epipolar lines. Since we have horizontal stereo image pairs, the epipolar lines will be the scanlines. We define a neighborhood region in the left image and search for the same in the right image and we do so for every pixel in the image. By comparing region of the same size on the left and right scanline, we find the difference in the location of the same pixel. This difference in the position of the same neighborhood in both the images is the disparity, which is inversely proportional to the depth.

The images can be acquired by the translating the camera in the horizontal direction. The setup will be as shown in Fig. 1.

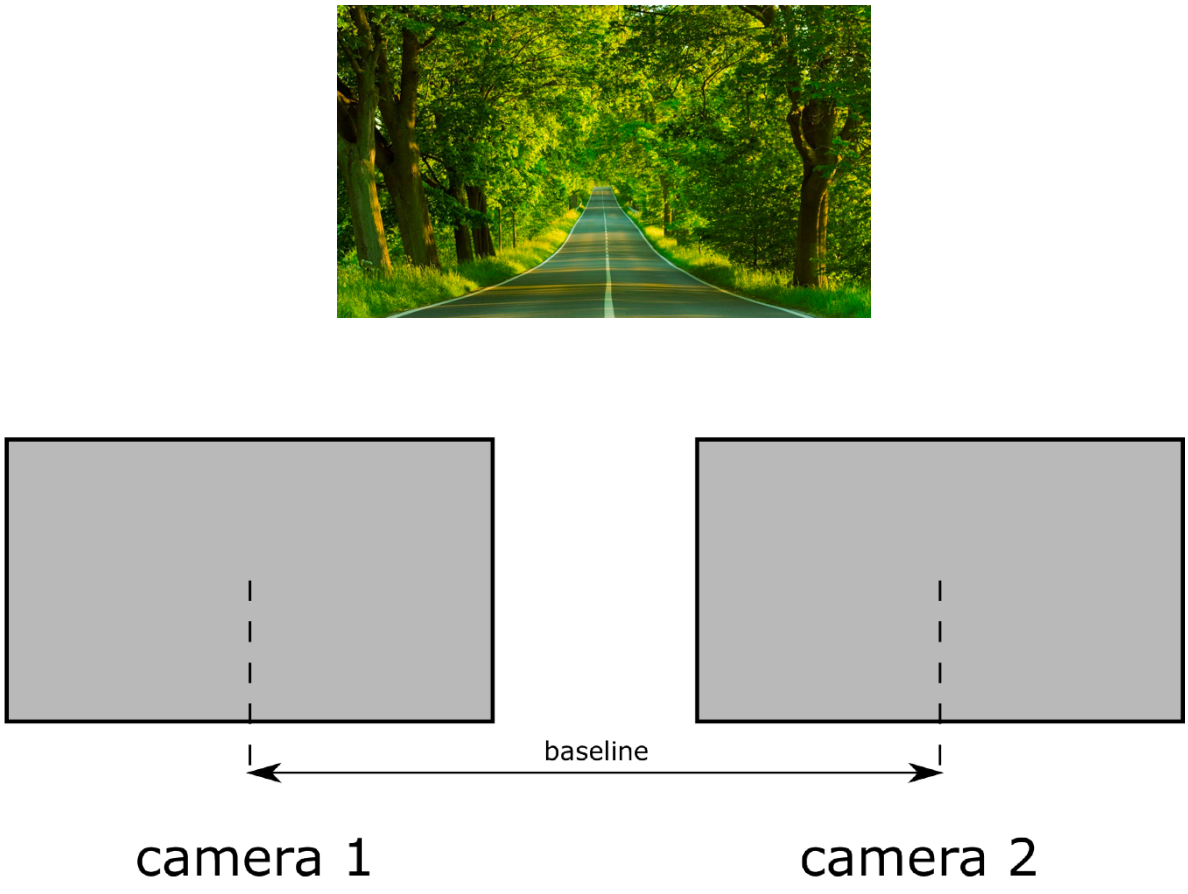


Figure 1: Horizontal stereographic acquisition

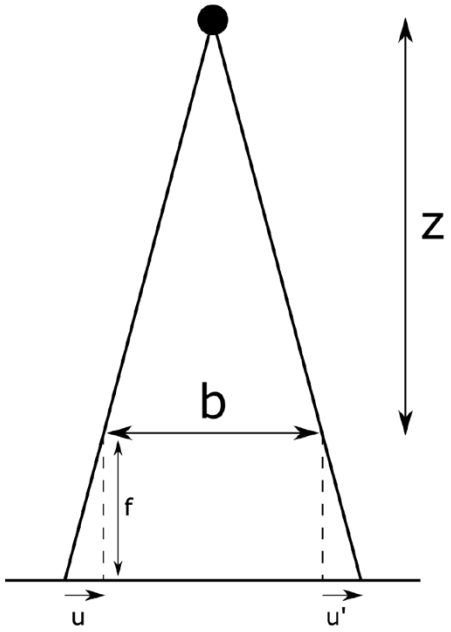


Figure 2: Geometric Model

Based on the two images and the fact that we know the baseline and the focal length, we are going to measure the disparity between the two images and reconstruct the depth map. With this setup, the disparity is given by:

Disparity (d) = | u' - u |

From the similar triangles we can write,

$$\frac{B}{z} = \frac{d}{f}$$

And finally, we get the formula for obtaining the depth:

$$z = f\frac{B}{d}$$

In this project, we make use of template matching to find the pixel correspondences, which gives us the disparity and hence the depth map. Cross correlation is a measurement used to quantify how close the two shapes are from each other. This technique is widely used in image processing to perform template matching and determine how close is the content of an image to a given template.

To perform template matching, we use Zero Mean Normalized Cross Correlation technique, and check for the maximum value of the correlation along the scanline. The next section explains the methods used and the implementation details, followed by the results and its discussion.

As mentioned previously, we use template matching to find the correspondence. Our approach involves defining a neighborhood region in the left image for each pixel, which works as a template and finding a corresponding match on the right image along the epipolar lines, shown in yellow color in Fig. 3.

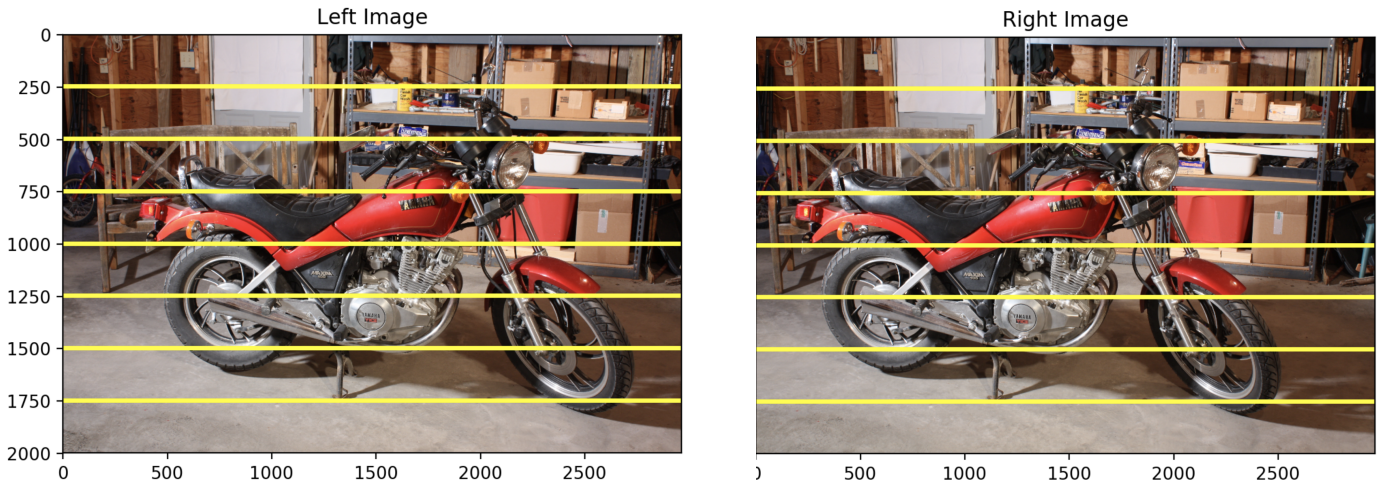


Figure 3: Rectified Images

Challenges

1. Although this method seems promising, where we perform NCC between the neighborhood region of a pixel in the left image with the neighborhood region for each pixel along the epipolar line in the right image, it is still computationally expensive.

2. Since we are checking for correspondence along the complete epipolar line, sometimes we get multiple maxima and therefore get incorrect values of correspondence in the right image. It can be observed from the figure 4 below.

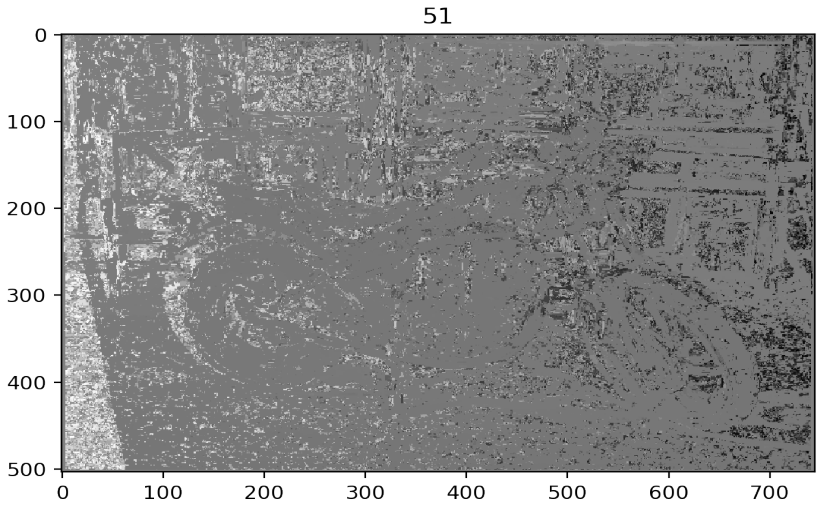


Figure 4

Workarounds

We know the baseline and therefore for a pixel in the left image, we can limit our search in a pre-defined range along the epipolar line in the right image, since the maximum displacement of the pixel won't be greater than the baseline. This method helped us reduce the computational time drastically. We were also able to solve our problem of getting the incorrect corresponding location of pixels of the left image in the right image.

Implementation Details

Resize Image

The image resolution was very high and therefore it would have taken a very long time to compute disparity for each pixel. Therefore, we resized our image with the help of OpenCV 'resize' function.

Padding

We will be performing normalized cross correlation, and in order to not skip the border pixels, we add a padding i.e., zeros around the image, which is half the size of the window, in order to start from the first pixel. The padding was added using the Numpy 'pad' function, which automatically adds the padding value around the 2D array, in our case which is the image, since OpenCV natively reads the image as Numpy array.

Image Correspondence

- **Normalization**
  Since the template is not normalized and may accumulate high values in bright regions and smaller ones in dark regions, we will have to apply the correlation between a zero-mean template and a normalized image region. To perform this, we select a neighborhood region around a pixel in the left image, which acts as our template. We calculate the mean intensity of this template region and deduct the mean from each template pixel. This results in a zero-mean template with positive and negative pixel values. We use this template and move this template in the right image along the epipolar line to obtain the correlation value. For each move of the template in the right image, we calculate the mean intensity within the respective neighborhood in the right image and subtract this mean from each pixel value while calculating the correlation. We divide the calculated correlation value by the norms of the two zero-mean regions, which guarantees correlation value between -1 and 1.

- **Template Matching / Cross Correlation**
  The next step would be, actually performing image correspondence. For this step, we implemented NCC, or we could see it as a template matching problem. We notice, that since the camera is well calibrated and the pictures are taken by only moving the camera parallel without any rotation, it is worth taking into account, that the shift would only be present in one dimension and not both.

This simplifies our template matching problem, now we just calculate the maximum correlation value for the neighborhood region of a pixel in the left image and for the neighborhood region of a pixel in the right image along the scanline.

The template matching would give us the location of the maximum correlation value in right image for every pixel in the left image. We take the absolute value of the difference of the position, which is the disparity.

Depth Calculation

From the disparity map, we can calculate the depth of each pixel. We know that disparity is inversely proportional to the depth. The objects which are deeper in the image will have less disparity than the objects which are near in the image. The depth is calculated using the formula:

$$z = f\frac{B}{d}$$

where,

z = depth,

f= focal length,

B= baseline,

d = disparity

Results and Discussions

We present our results which we obtained on the stereo pair of images. We show the original stereo pair of images, the corresponding disparity and depth map and effect of the kernel size used to compute the correlation.

The first stereo pair of image we used is presented here –

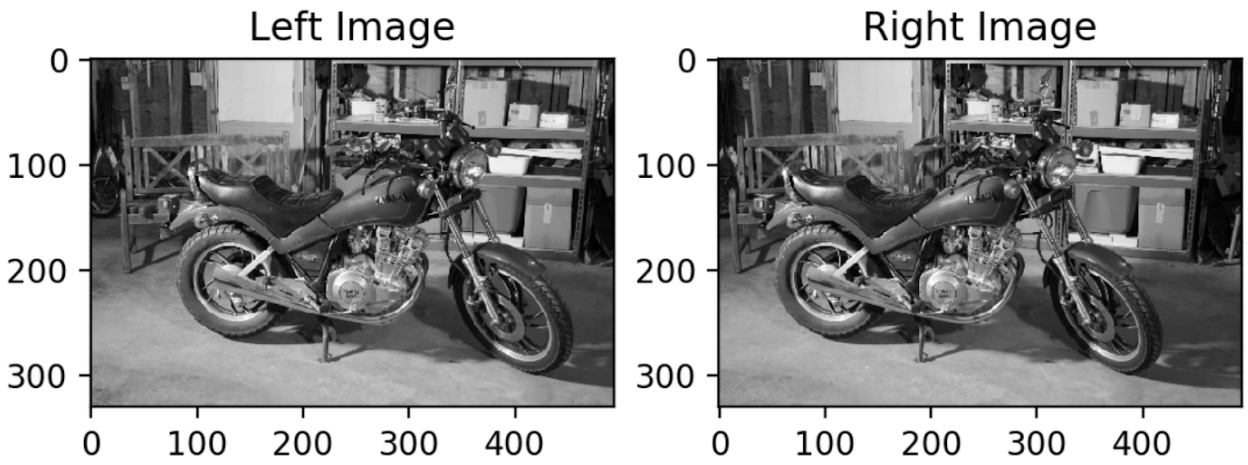


Figure 5: Left and right image of the first stereo pair of image

Here are the results of the disparity and the depth with a 5, 11 and a 21 kernel:

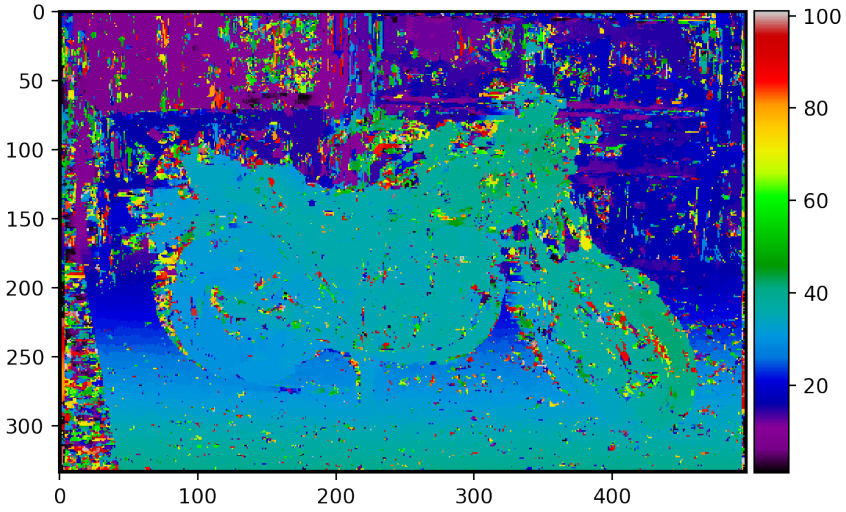


Figure 6: Disparity map with 5 by 5 kernel

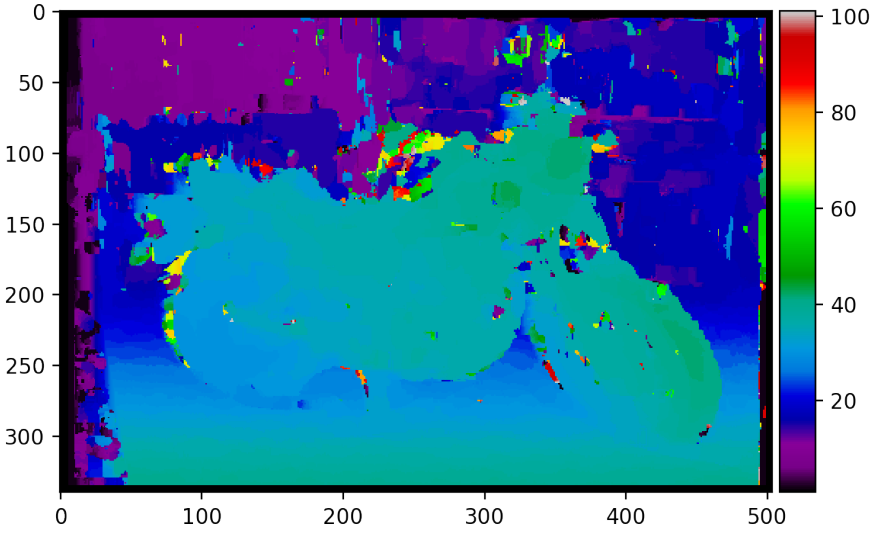


Figure 7: Disparity map with 11 by 11 kernel

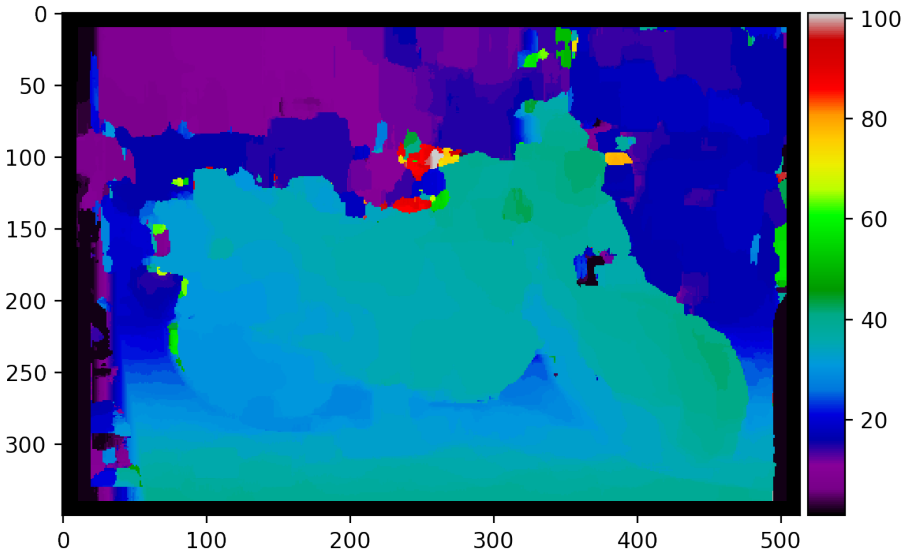


Figure 8: Disparity map with 21 by 21 kernel

Discussion

From the above disparity maps, we can see the effect of the kernel size. The size of the kernel influences the "sharpness" of the disparity map. The first disparity image with kernel size 5 seems messier while the second and the third disparity image with kernel size 11 and 21 seems more homogeneous.
With a bigger kernel size, it is easier to match structures with more accuracy since the intensity variation pattern is bigger and it allows a better match. The small kernel allows us to have more details on the structure but it contains lot of noise too. In this project, we are interested in locating the position of the object in both the images so as to calculate the disparity and which in turn will provide us with the depth map.

From the above disparity maps, we can see that the objects which are at a more depth has a lesser disparity than the object which are near. We are able to clearly locate the bike, then the chair which is located behind the bike, and then we can see the wall on the left and the shelf on the right with a bit more complication. If we look at the original image and then the depth map it will be a bit easier to locate the objects with loss of precision in the disparity map.

Now we calculate the depth with the help of the equation presented before. We use the formula presented in the approach part of the report. The depth is inversely proportional to the depth, we will observe this in the below plots. We plot the depth in 2D, which will be able to give an illustration of depth of the various objects by varying color.

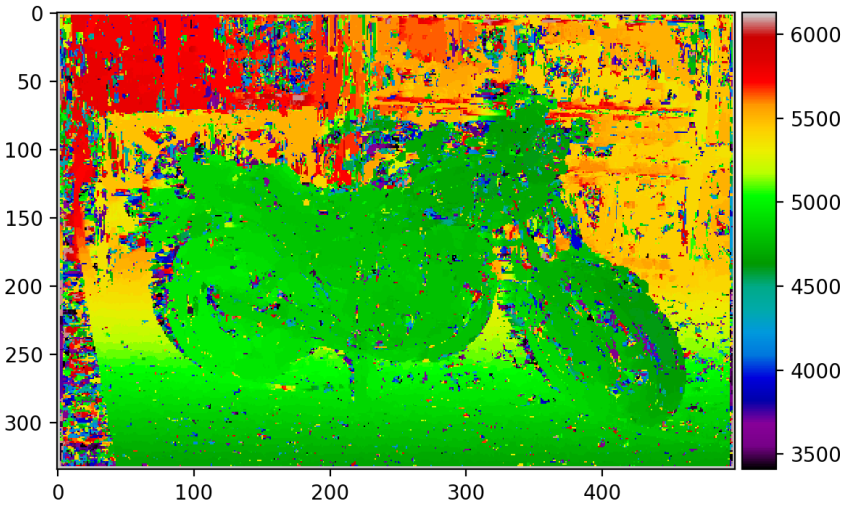


Figure 9: Depth image with 5 by 5 kernel

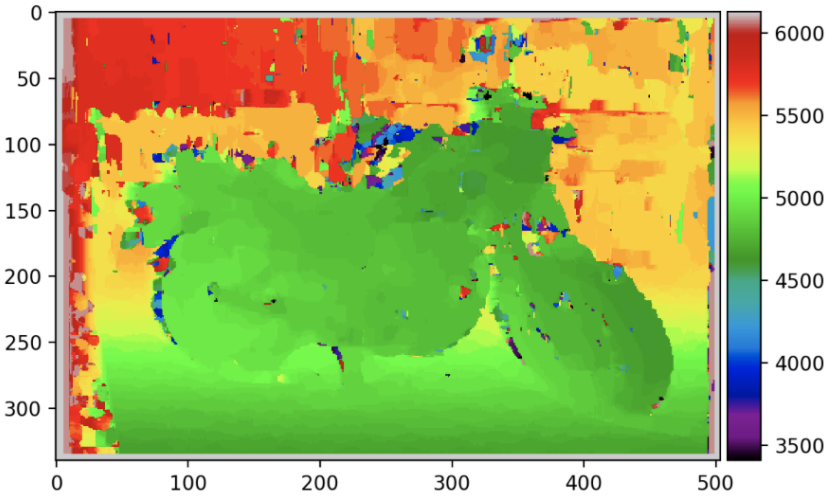


Figure 10: Depth image with 11 by 11 kernel

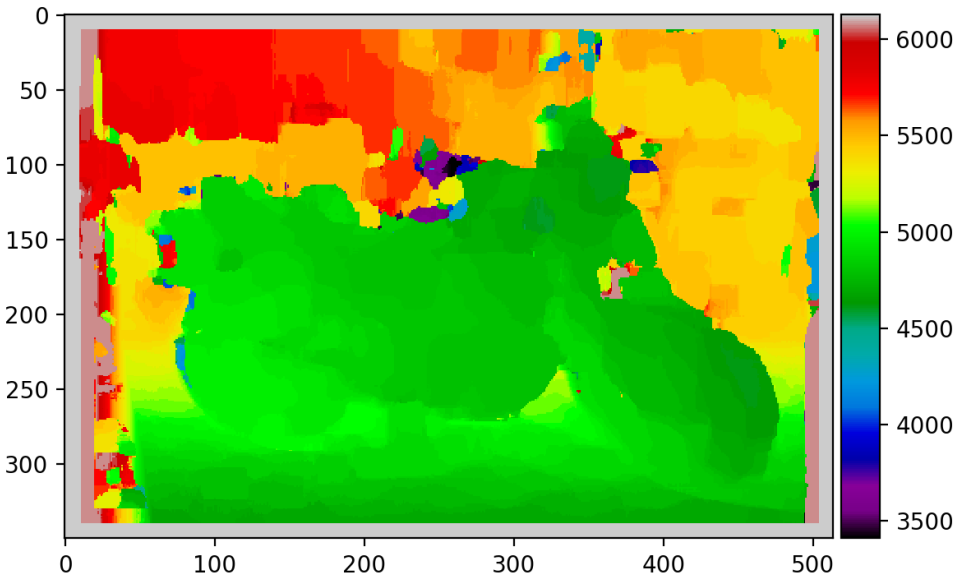


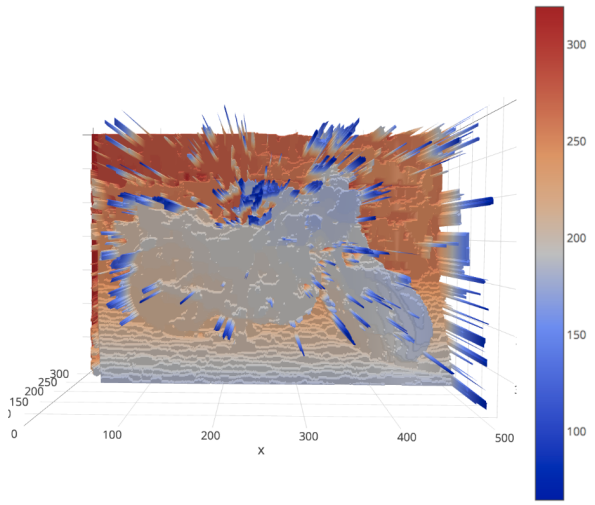
Figure 11: Depth image with 21 by 21 kernel

Discussion

The use of the color scale allows us to have an estimated depth of the various objects. The objects which are near in the image should have a less depth than the object which are far, and the color map shows us the same. The motorcycle which is near in the image is shown in the green color which has a range of around 5000 and the chair which is located behind the image is shown in orange color which is in the range of 5500, thereby adhering to the original image.

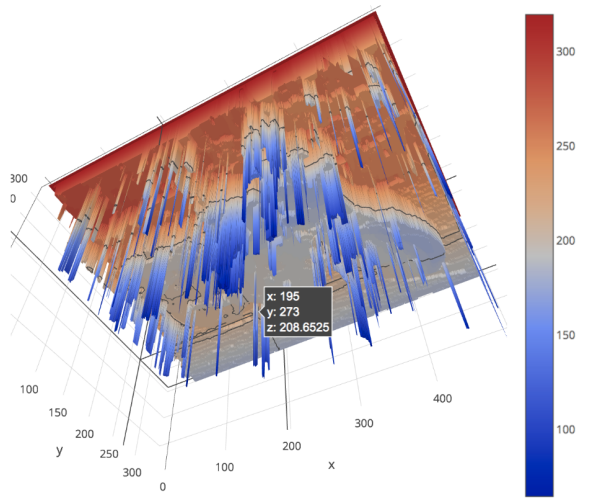
We now plot the depth in 3D, which was realized using the Plotly API. The plot gives us the estimate of the depth. If we see the plot from different angles, we will see holes, since we don't have complete rotational image plane of the scene.

Interactive Depth Map

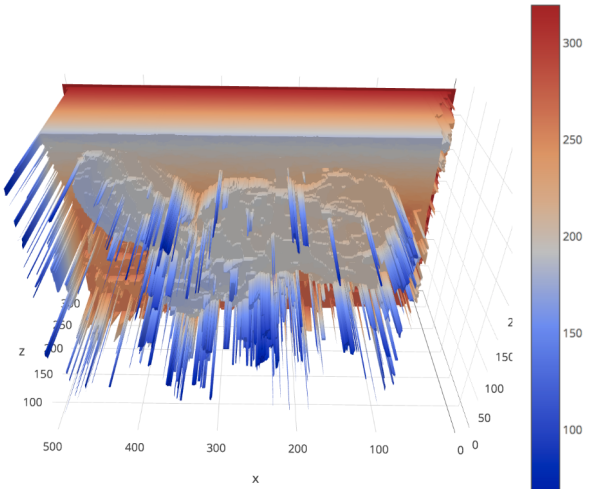


EDIT CHART

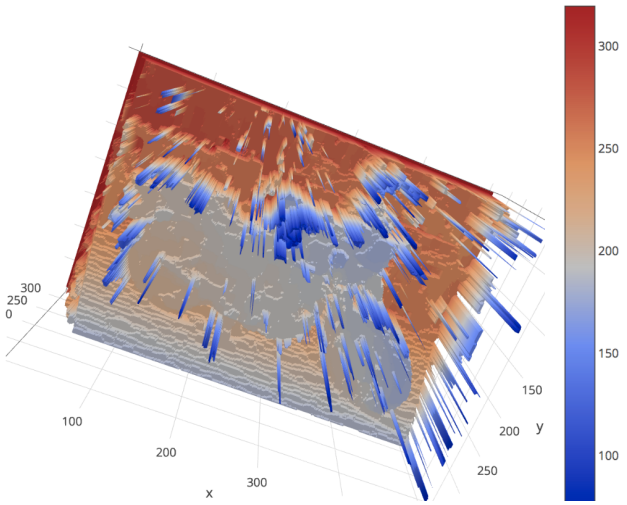
Interactive Depth Map



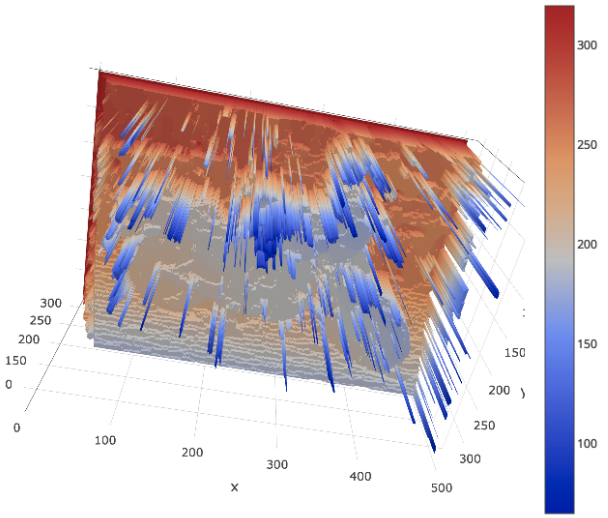
EDIT CHART



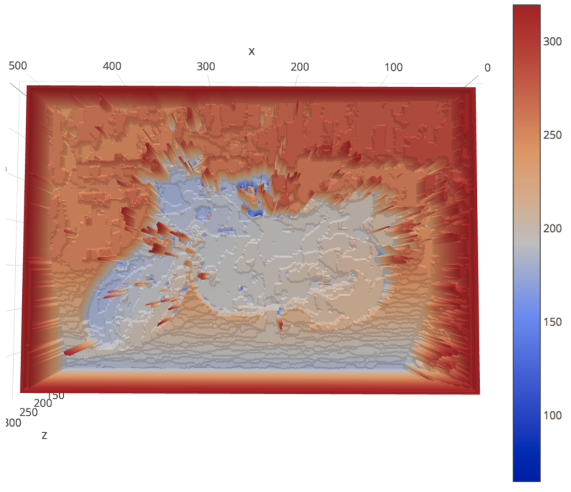
Interactive Depth Map



Interactive Depth Map



EDIT CHART



EDIT CHART

Let’s now test the second stereo pair of images and see whether the result confirms to our previous discussion of the effect of the kernel size.

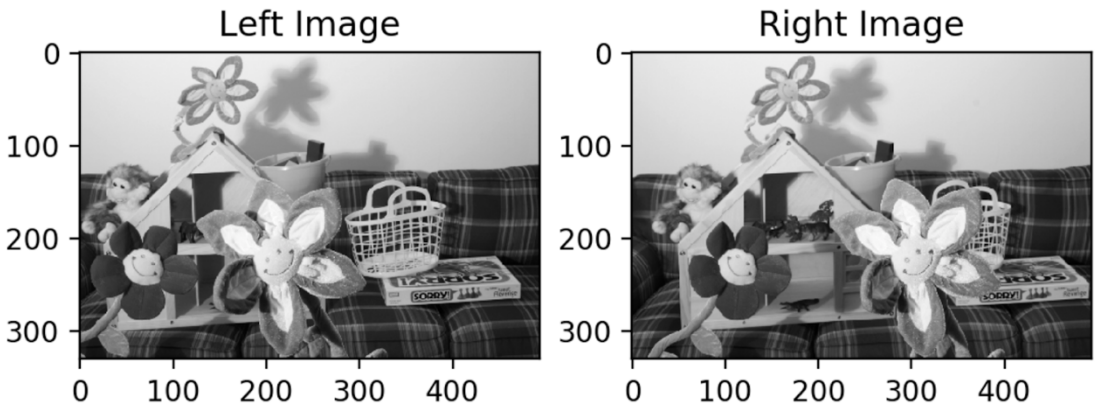


Figure 12: Left and right image of the second stereo pair of image

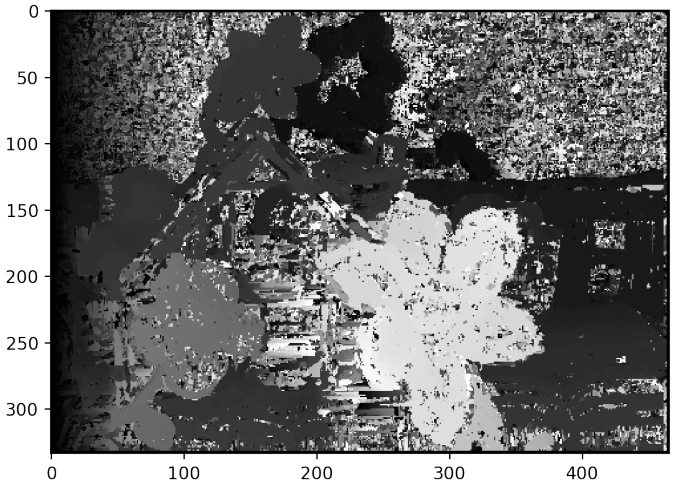


Figure 13: Disparity map with 5 by 5 kernel

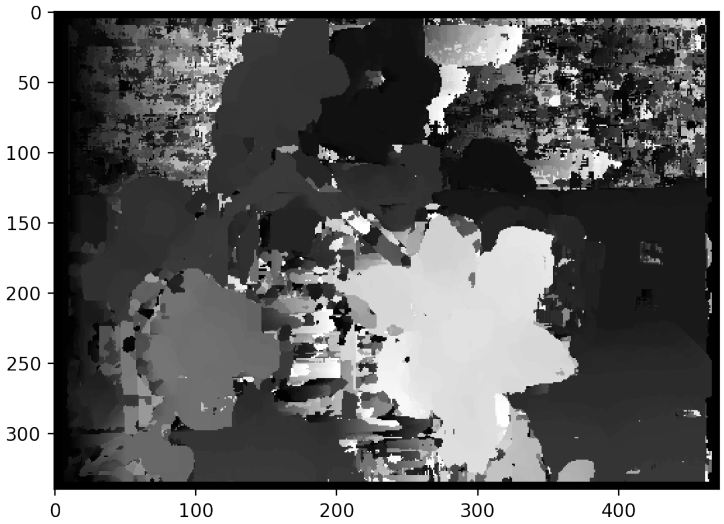


Figure 14: Disparity map with 11 by 11 kernel

Discussion: From the above disparity images, we again see that with a small kernel we get a blurry image and when we increased the kernel size from 5 to 11, we have reduced noise and more visible structures. We can distinguish structures which are a bit more complicated. For example, the basket behind the nearest flower, if we have prior knowledge of the image. Now we calculate the corresponding depth maps.

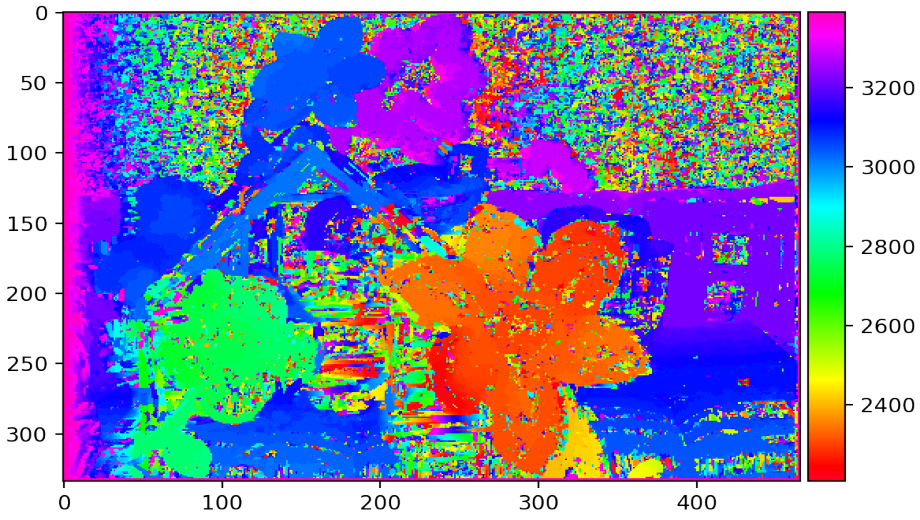


Figure 15: Depth image with 5 by 5 kernel

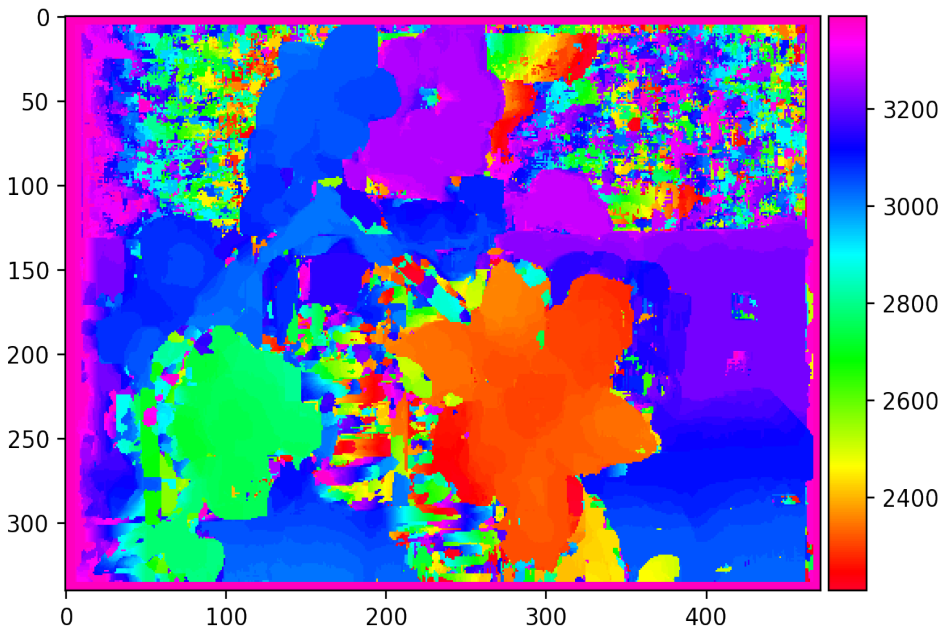


Figure 16: Depth image with 11 by 11 kernel

As with our first set of images, the use of the color scale provides us with the illustration of depth 2D for this set of images too. We can clearly see the flower in red which is near followed by the flower in green and blue and then at the maximum depth we can see the shadow of the flower in pink, which was on the wall in the original image. We also observed some noise in the region where we didn't have any kind of texture in some parts of the image. In our case, since the wall was completely white, we observe a messier disparity map. This might be due to the fact that for a specific pixel we can't find its exact location in the right image, since the region doesn't have any texture and we will find maximum correlation for many pixels in the right image.

Conclusion and Future Work

In this project, we had the opportunity to work and study some important aspects regarding a very important technique which is stereoscopy and which allows us to visualize depth using two images. We saw the geometry behind the depth calculation of the pixels in the image.

We dealt with the depth reconstruction from a stereo pair of images. We used the Zero Mean Normalized Cross Correlation method to implement the template matching. In the process, we figured out various aspects of the matching which also helped us in reducing our computational expense by limiting our search range along the epipolar lines.

We realized the above method on two pair of stereoscopic images and plotted the depth in 2D with varying color and in 3D. We also saw the effect of the different kernel size while implementing the template matching using Normalized Cross Correlation.

If we had more time, we could have worked on other method than template matching i.e., Sum of Square Difference (SSD), or we can generate disparity using edge constraints and Mean Shift Segmentation of the disparity map. We have started reading on the above methods and would like to implement it and compare our disparity map with the new disparity map.