Robot Eyes Report

David Kenny, Jamie Kerr
BSc (Hons) Applied Computing

March 2016

Abstract

The team was given the task of creating a disparity map from a pair of stereo images using techniques that were talked about in class.

1 Introduction

The assignment required the team to acquire new knowledge of computer vision by producing their own disparity map from scratch using Matlab. We were encouraged to experiment with variables and publish the results of the findings. This report details the experiments that the team performed and the notable results that were acquired from them.

2 Background

A disparity map [5] is used to show the difference between a pair of stereo images. The disparity is used to calculate depth information from a combination of the images.

The disparity value is the value of the shift required to get the minimum sum of squared differences [1] for that pixel or section of the image.

Stereo pairs are images typically taken with only a slight difference in the placement of the cameras. Preferably the camera settings should be the same and the hardware should be the same as well to ensure that the images are as similar as possible.

The information gained from the disparity map will be useful to robots and machinery to provide extra depth information to base movements or actions on for example google car [7].

3 Description of Algorithm

The team used a brute force method to gain their results using this method meant that the run time was extremely high due the fact of a disparity value having to be calculated for every pixel of the image with no shortcuts.

This was a problem when the team were testing their program. Also since we were learning at the same time it didn't help as a simple error could take in excess of 10 minutes to produce a result even on a 100 x 100 image which lowered the teams productivity.

The team discovered an algorithm called ENCC [4] which produces an estimate of disparity with sub pixel accuracy. This could have been used in place of the brute force method to create a disparity map but may not have had the same accuracy.

4 Assumptions Made

The team implemented a rectification option that assumes that the images entered have already been rectified. The program will work with non rectified images as well if the tick box isn't checked.

5 Description of Images Used

The team used the stereo images provided on the VLE to create their program. The pair that was mainly used was the bookcase image shown in figure 1. It was selected as there are clear levels of depth between the different books and the team thought that they would produce a noteworthy result.

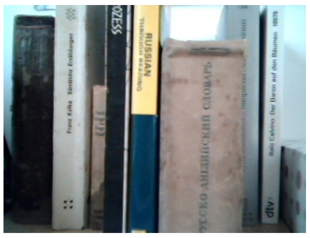

Figure 1: Bookcase

The other pair that the team used was the ”Scene” pair which we had seen used a lot in the computer vision community [6] [3] for various benchmark tests including disparity maps. We believe that this is due to there being an obvious deviation of layers in the image. The image also has a small resolution too which means that it won’t take long to produce a result.

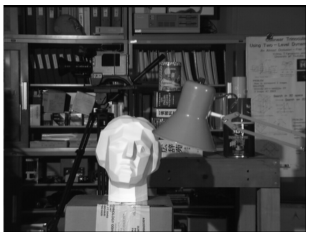

Figure 2: Scene

We used our own image which was taken with an iPhone 6S. It has an iPad box in the background and a bottle of coke in front of it to indicate clear layers of depth in the stereo pair.

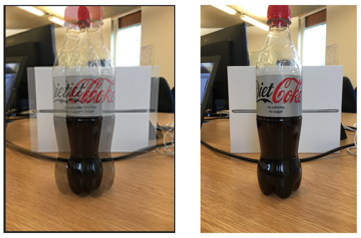

Figure 3: Our Image

The image was scaled down from a resolution of 3024 x 4032 to 150 x 200 using photoshop for testing purposes as it would take significantly less time to compute. On the left of figure 3 is the left and the right images overlaid. When the images were taken the team used a table to place the camera on to ensure that the position of the camera remained constant from taking the left and right image. We were aware that rotation of the camera wasn’t fixed which can result in a sub optimal stereo pair.

6 Results

The team created a program that worked out the disparity and outputted the SSD(Sum of square differences) and disparity for one specific pixel. First of all we started using a search window of 15 x 15 and a support window of 3 x 3. After this was done we then started trying to create a disparity map using the SSD values for each pixel.

The team found that the SSD values were not correctly working since it was either producing a value of 0 or 255 as the final output this was confirmed when the disparity maps pixels were either black or white.

We found that the displacement values had to be converted into an 8 bit unsigned integer to be correctly displayed as a greyscale image. This then resulted in figure 4 being produced which was a step in the right direction.

It was noted that the black band shown in figure 4 is due to the padding that was added to image.

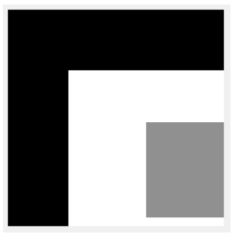

Figure 4: Greyscale Output

After figure 4 was produced the team then ran a test using a 200 x 200 image. This test took over two hours to run on a modern laptop and the end result wasn't what the team were hoping to produce. This turned out to be a huge waste of productivity.

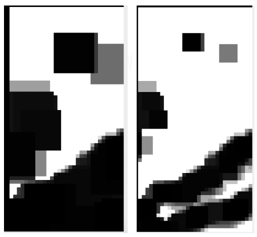

Figure 5: Search Window

Since the team couldn't find out much information from the corner of the books we decided to try using the scene image (figure 2) and focus on the section of the image where the lamp is in front of the head. The result that was produced from this was the image on the left of figure 5. The team were pleased with this result as there was a larger variation in values.

The result seemed pixelated the team believed that this was because of the search window size since we knew that decreasing the size of the window would increase the resolution and provide more clarity in the disparity map how ever the trade off is that there will be more noise present in the map.

The window size was reduced from 15 to 9 and the image on the right of figure 5 was produced. This seemed to be hopeful.

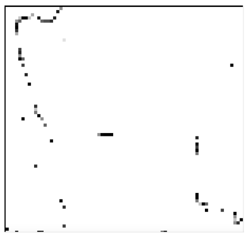

Figure 6: Coke Result

After the result in figure 6 was produced on our full sized image (figure 3). The team realised that the disparity map wasn't functioning correctly. Assistance was then provided by Andrew Mcneil and it was pointed out that we were using the SSD values to create the map and not the disparity values which was an oversight.

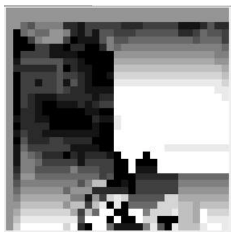

Figure 7: Disparity vectors result

After making alterations to our program to use the disparity values we managed to produce figure 7. This was produced by mapping the disparity values to a relevant greyscale value for each pixel.

We knew that this still wasn't what we were expecting and since the structure of our program was increasingly complex it became difficult to debug. At this point the team decided to go back to the drawing board by getting rid of two weeks worth of work to when we simply had the disparity vector of one pixel being computed since the code was a lot more manageable at this point.

We then modified the code to calculate every pixel similar to what it was doing before hand and started to get some noteworthy results instead of hardcoding values we set up variables for search window size, support window size, map height, map width and the type of algorithm.

A GUI was created during the run time of generating the maps to help run future tests this was built to increase our productivity during run time and also assist with testing.

One issue that we had created previously was that we were looping through the whole right image and not just the search window which was the reason why our program was taking so long. The padding was also removed as it was causing major problems when creating the maps. A combination of these two actions majorly reduced the running time of the program.

The SAD matching algorithm was added into our program as well so we could run tests to compare SSD to SAD.

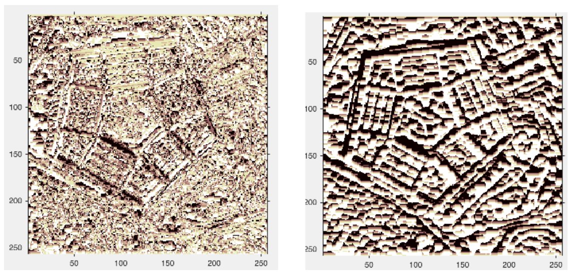

Figure 8: SSD vs SAD

Figure 8 shows the difference between using SSD and SAD on the pentagon image. The disparity maps colours were mapped using the pink colormap function as well to produce a clearer visualisation. In this situation we believe that SAD produced a more accurate representation of depth in the image.

The team then ran the SAD algorithm on the Coke bottle image and produced our final output of figure 9. Which clearly shows the bottle in front of the iPad box.

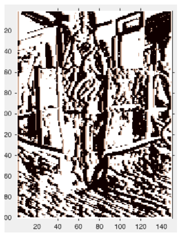

Figure 9: Coke Bottle SAD

7 Conclusion

From the results gained, Even though they may not be exactly spot on the team felt they were on the right lines.

The team had never worked with Matlab before and were pleased with what they were able to achieve within this short space of time.

The GUI that was produced helped a lot with testing and the team were pleased to have created it with no prior experience.

The full source code for the teams work can be found in a git hub repository [2].

8 Future Work

If the team had more time available we would like to have been able to perform research on 3D reconstruction techniques and also work on ways of optimising the matching algorithm to minimise computation.

Acknowledgements

The team would like to thank Andrew McNeil for his support throughout this module.

References

[1] Alessandro Verri Emanuele Trucco. 1998. *Introductory techniques for 3D - Computer Vision*. Prentice Hall.

[2] Jamie Kerr. 2016. Matlabrobot. (2016). `https://github.com/jkerr123/matlabrobot`

[3] Shawn Lankton Online. 2007. 3D Vision with Stereo Disparity. (2007). `http://www.shawnlankton.com/2007/12/3d-vision-with-stereo-disparity/`

[4] Emmanouil Z. Psarakis and Georgios D. Evangelidis. 2005. *An Enhanced Correlation-Based Method for Stereo Correspondence with Sub-Pixel Accuracy*. Technical Report. Department of Computer Engineering and Informatics University of Patras, 26500 Patras, Greece. `http://xanthippi.ceid.upatras.gr/people/evangelidis/george_files/ICCV_2005.pdf`

[5] Jay Rambhia. 2013. Disparity Map. (2013). `http://www.jayrambhia.com/blog/disparity-maps/`

[6] Stanford. 2016. Correlation-Based stereo. (2016). `http://ai.stanford.edu/~mitul/cs223b/dp.html`

[7] Ryan Whitwam. 2014. How Google's self-driving cars detect and avoid obstacles. (2014). `http://www.extremetech.com/extreme/189486-how-googles-self-driving-cars-detect-and-avoid-obstacles`