Stereo Pi: Portable Digital Stereo Camera

Aniq Masood
aniq.masood@gmail.com

Abstract

Stereoscopy, a technique for creating the illusion of depth, has existed as early as the 1950s. With the ubiquity of smartphones, digital camera technologies has greatly advanced over the past decade. Although stereo cameras have existed for many years using film, there are few digital stereo cameras available. This project aims to build a portable, digital stereo camera using Raspberry Pis.

1. Introduction

Depth imaging has become an increasingly popular field in research and consumer electronics over the past decade. Consumer products such as the Microsoft Kinect and Oculus Rift have made the capture and display of three dimensional information available for applications such as robotics, gaming, entertainment, and surveillance.

With the advent of open source electronics platforms such as the Arduino microcontroller and Raspberry Pi computer, hobbyists and researchers have greater access to hardware capabilities than ever before. This has made prototyping hardware and software projects faster and more affordable. This advantage can be leveraged by the computer vision community to create synergistic hardware and software solutions for depth imaging. This project aims to create depth imaging device to be used for 3D reconstruction applications.

2. Hardware Components

2.1. Processor

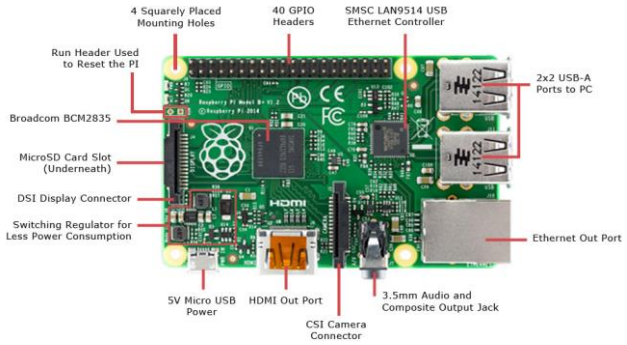


Figure 1: Raspberry Pi 2 Model B Diagram

The Raspberry Pi is a credit card sized computer that was developed in 2006 to teach children how computers work. Although its primary function is to be an educational platform to teach programming to children, hobbyists, researchers, and engineers have adopted it to build small electronics projects due to its functionality and price. The Raspberry Pi 2 Model B, released in 2015, has a 900 MHz quad-core ARM Cortex-A7 CPU running Debian Linux. It has 1 GB onboard RAM, 40 GPIO pins, Camera Serial Interface (CSI) connector, 4 USB ports, and an Ethernet port.

2.2. Camera

Figure 2: Arducam Raspberry Pi Camera Module

The Arducam Raspberry Pi camera module was used for this project. The camera module uses the OmniVision OV5647, a 5 megapixel sensor with a pixel size of 1.4 microns. It has an onboard lens with a fixed focus, and a maximum picture resolution of 2592 x 1944 pixels. It is capable of taking 1080p video at 30 frames per second. It is connected to the Raspberry Pi via the CSI interface. For this project, a picture and video resolution of 680 x 480 pixels was used for computational speed.

Since the Raspberry Pi 2 only has 1 CSI interface, each camera has its own dedicated Raspberry Pi and camera module. The Python picamera library was used to control the cameras and take pictures and video.

2.3. Power Supply

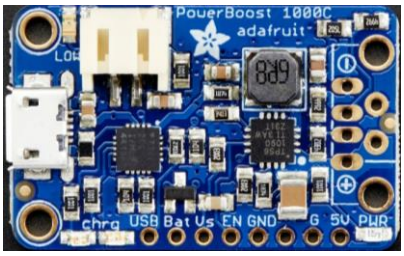


Figure 3: Adafruit PowerBoost 1000C

To make this device portable, it was necessary to have a standalone power source for each Raspberry Pi. A 3.7 volt, 2200mAh Lithium-ion battery was used to power each Raspberry Pi. Since the Pis each run on 5V DC power, an Adafruit PowerBoost 1000C was used to convert the Lithium-ion battery's power to something acceptable to power the Raspberry Pis. The PowerBoost is a DC/DC boost converter especially useful for powering small computers and microcontrollers with Lithium-ion and Lithium polymer batteries.

Since the power input to the Raspberry Pi is a micro-USB connection, a USB plug was soldered onto the board and a USB to micro-USB cable was used to connect the board to a Raspberry Pi. Additionally, an ON/OFF switch was soldered to easily apply and cut power to the Raspberry Pi.

2.4. Camera Trigger Circuit

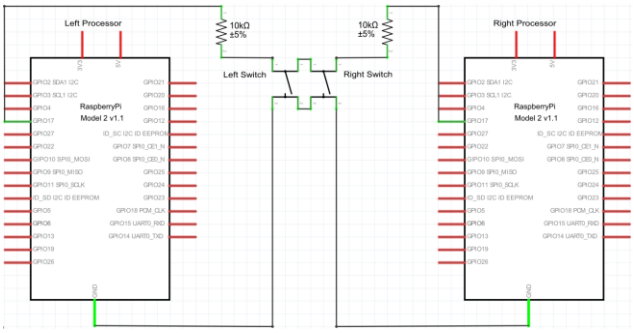


Figure 4: Trigger Circuit Diagram

Rather than using command line instructions to prompt the Raspberry Pis to take pictures, a pushbutton was used to send a trigger pulse to both Pis simultaneously. As shown in Figure 4, a resistor and momentary switch are connected to a GPIO pin on each Pi. In software, the pin is set to high. When the switch is enabled, the pin is sent to ground. When this change in voltage is sensed, a command is sent to the camera to capture an image. Since both inputs and both outputs of the switches are connected to the same nodes, pushing one button will trigger both Pis simultaneously. The difference in trigger timing is dependent on the time it takes for each Pi to interpret the change in pin voltage.

2.5. Enclosure

A camera baseline of 10 cm was chosen for this project. The enclosure was 3D printed with polylactic acid (PLA) filament using a Makerbot Replicator 2. Figure 5 shows the front, top and bottom, and back pieces that make up the enclosure. The pieces are held together with M2 bolts.

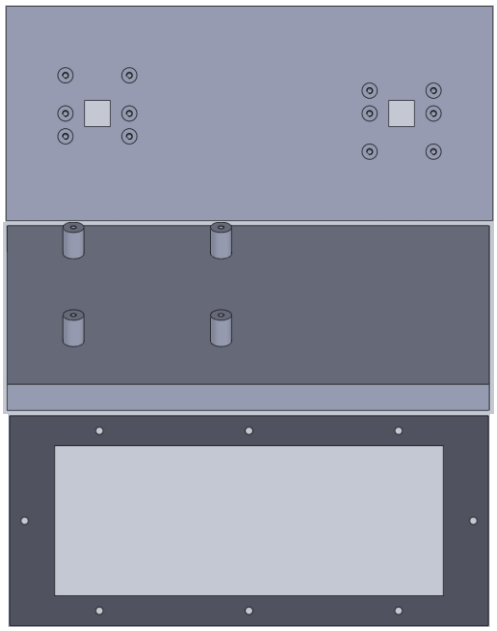


Figure 5: Enclosure Pieces

3. Processing Steps

Once the images are captured from the device, they undergo the processing steps outlined in Figure 6:

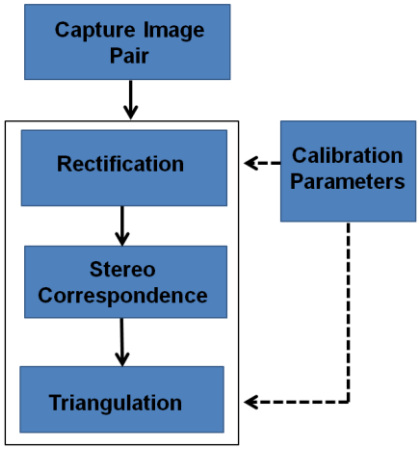
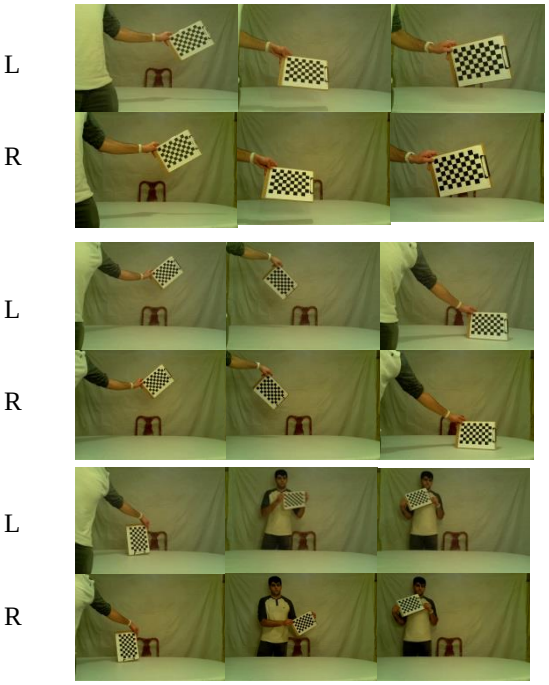


Figure 6: Outline of Processing Steps

3.1. Calibration

A camera calibration method follows that outlined by Zhang [1] and Heikkila [2] and created for MATLAB by Bouguet [3]. It uses a series of images with a checkerboard pattern at different areas in the camera's field of view to determine camera pair's intrinsic and extrinsic parameters. Figure 7 shows the 18 images used for calibration and their mean projection errors:



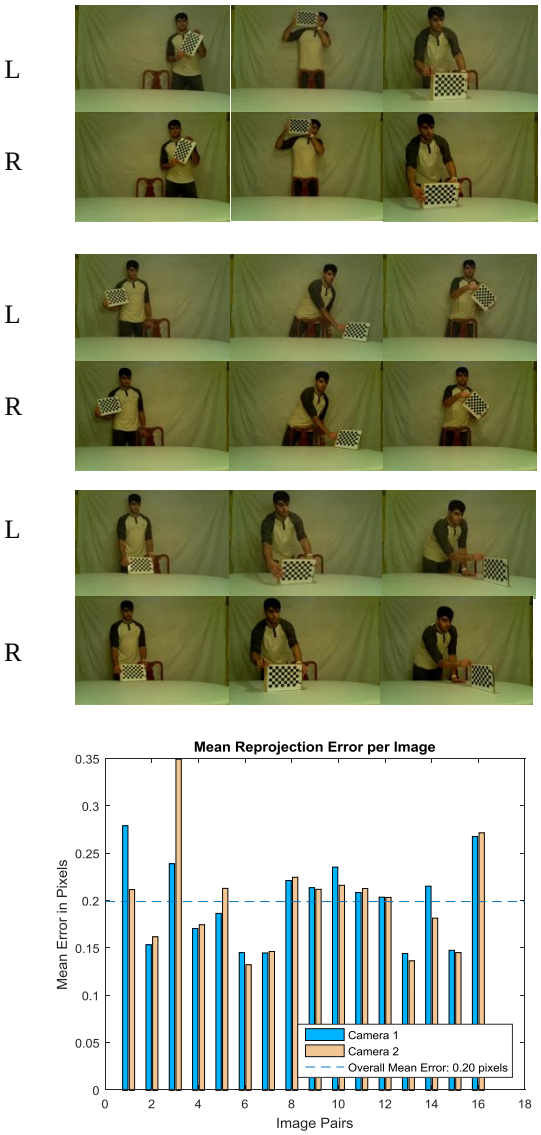


Figure 7: Calibration Images and Reprojection Errors

The intrinsic parameters determined were the radial and tangential distortion coefficients, focal length, and image centers for each camera. The extrinsic parameters determined were the rotation, translation, fundamental and essential matrices used to determine the position of a pixel in the right camera relative to the left camera.

3.2. Image Rectification

The image rectification process has two steps: first, it removes the lens distortions that are measured from the calibration step. Second, the images are transformed such that they are coplanar. This applies the epipolar constraint to the stereo pair which enables a faster and more accurate

stereo correspondence. The MATLAB function *rectifyStereoImages()* was used to perform the rectification.

3.3. Stereo Correspondence

The Semi-Global Block Matching Algorithm outlined in (Hirschmüller, 2005) was used to compute the stereo camera's disparity map. It uses a sum of absolute differences (SAD) similarity measure given by the following formula:

$$SAD(i,j,d) = \sum_{\mu=-w}^{w} \sum_{v=-w}^{w} |I_l(i+\mu, j+v) - I_r(i+\mu, j-d+v)| \quad (1)$$

where I_l and I_r denote the left and right image pixel grayscale values, w is the window size, i and j are coordinates of the center of the window for which the similarity measures are computed, and d is the disparity range.

Once the SAD is computed for all pixel and disparity values, a similarity accumulator is constructed for each pixel, indicating the most likely disparity. A search in the SAD for all disparity values in a range defined for d_{min} and d_{max} is performed for each pixel. The disparity value is finally given under the following condition:

$$D(i,j) = arg \min_{d \in [d_{min}, d_{max}]} SAD(i,j,d) \quad (2)$$

A disparity range of 0 to 64 and window size of 15 were used to calculate the disparity maps. The MATLAB function *disparity()* was used to implement the stereo correspondence algorithm.

3.4. Triangulation

Once the disparity map was computed, a triangulation step was done to project the position of the correspondence from pixel coordinates in the left image to three dimensional space. The MATLAB function *reconstructScene()* was used to perform this task.

4. Experimental Results

4.1. Hardware Assembly

The hardware was built and was successful in generating disparity maps of scenes under varied conditions. Figures 8-10 show images of the final hardware assembly:

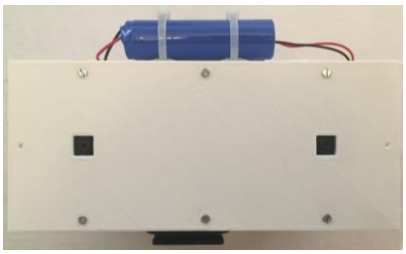


Figure 8: Front View

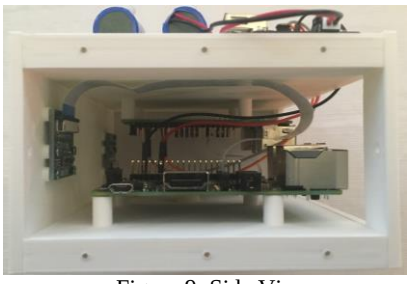


Figure 9: Side View

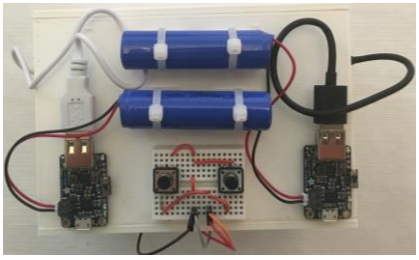


Figure 10: Top View

Images were retrieved from each Raspberry Pi via an Ethernet cable and the disparity map calculation was done offline on another computer running MATLAB.

4.2. Disparity Maps

The constructed stereo camera was tested indoors and outdoors to confirm its use in environments with varied lighting conditions. Figure 11 shows a collection of images taken from the left camera and the resulting disparity map from the pair of images.

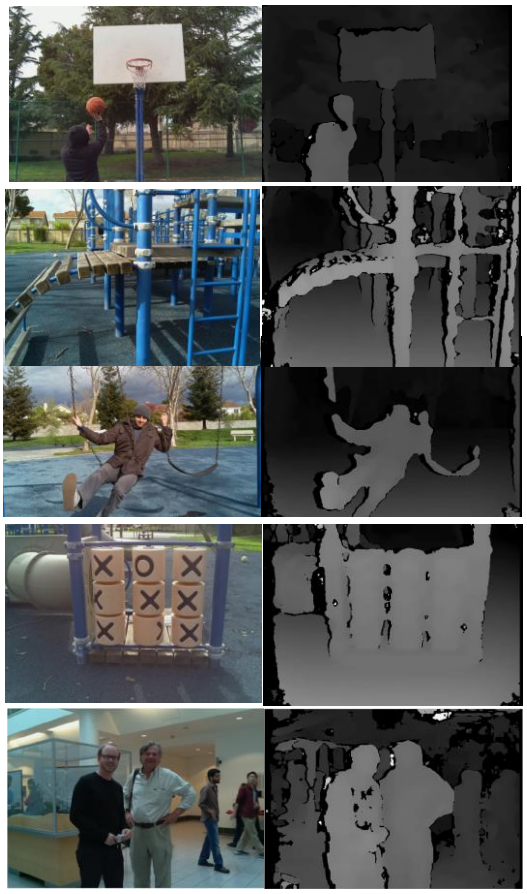


Figure 11: Left Camera Images and Disparity Maps

4.3. Accuracy of Depth Calculations

An additional test was done to determine the accuracy of the depth calculation from the stereo camera. With the camera stationary, a person in the camera's field of view moved away from the camera in a straight line, from a distance of 4 feet to 16 feet in foot increments. Images were taken at every increment.

Once an image was captured, the Viola Jones face detection algorithm was used to determine the pixel coordinates of the person's face in the left camera image. Using the calculated disparity map of the stereo pair, the coordinates of the face's centroid were projected in three dimensional space (as expressed in Section 3.4), and the physical distance was noted. Figure 12 shows an example image with the bounding box of the face detected with the corresponding measured real world distance.

Figure 12: Example Image and Test Results

As shown in Figure 13, the mean absolute error between the measured and actual distance was roughly 5.8 inches.

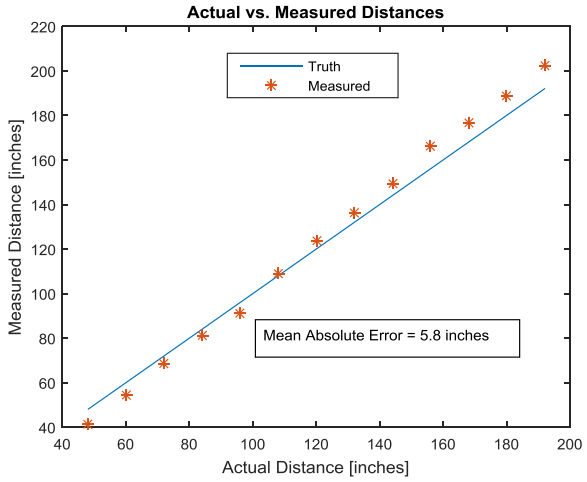


Figure 13: Depth Accuracy Test Results

4.4. Disparity Map from Video

Given videos taken from the stereo camera, a video disparity map was created by calculating a disparity map on a frame by frame basis. Three videos are included with the code for this project: a person shooting a basketball, a person on a playground swing, and a person on a tire swing.

5. Improvements and Future Work

A further improvement to this project would be to natively run the stereo vision algorithm on the Raspberry Pis. Although the computational time would drastically increase, the project would truly be standalone and portable. Utilizing the OpenCV library for Python would allow comparable functionality to that of the MATLAB functions used at this stage of the project. Adding a small touchscreen would also enable the user to see the disparity map on the device.

Furthermore, since we have two Raspberry Pis at our disposal, implementing a distributed computing architecture could provide the needed computational power to do the stereo algorithm onboard in an efficient manner.

6. Acknowledgements

I would like to thank Professor Gordon Wetzstein for teaching an amazing class. His encouragement and direction was instrumental in completing the project.

References

[1] Zhang, Z. "A Flexible New Technique for Camera Calibration". *IEEE Transactions on Pattern Analysis and Machine Intelligence*.Vol. 22, No. 11, 2000, pp. 1330–1334.

[2] Heikkila, J, and O. Silven. "A Four-step Camera Calibration Procedure with Implicit Image Correction." *IEEE International Conference on Computer Vision and Pattern Recognition*. 1997.

[3] Bouguet, J. Y. "Camera Calibration Toolbox for Matlab." Computational Vision at the California Institute of Technology. Camera Calibration Toolbox for MATLAB.

[4] Hirschmuller, H., *Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information*, International Conference on Computer Vision and Pattern Recognition, 2005.