The Development of an Autonomous Balancing Robot Using Only Stereo Vision

Mary Scaramuzza and Ryan Vink
Computer Science
St. Olaf College
Northfield, MN 55057
scaramum@stolaf.edu

Abstract

We are developing a balancing robot able to autonomously explore an indoor environment using only stereo vision for navigation. In this paper, we consider the problem of SLAM (simultaneous localization and mapping), but without the use of laser or ultrasound. Our strategy starts by calibrating the cameras in order to later triangulate the data points. From there we deinterlace the images to reduce the noise created by using lower quality cameras. In order to create a disparity map, the algorithm we use requires us to rectify the right and left images. From the left image, a segmentation is created and used to create the disparity map. For map construction we are using the SLAM6D open-source package developed at the University of Osnabrueck [2]. Our experiments have shown the importance of using good calibration and color data.

Introduction

Much of the simultaneous localization and mapping (SLAM) research uses lasers or UV sensors to gather data. We attempt to solve the same problem with using only stereo vision. By looking at this problem from a different angle we hoped to gather more data in a faster manner. With the extra data, our map would have more detail to help the robot navigate in an unknown environment.

In this paper we will discuss the process that is applied to a stereo pair (figure 1). From there a detailed outline of the different set-ups is provided, along with an explanation of our findings. We start by calibrating the cameras.

Figure 1: Stereo pair taken on the second floor atrium of Regents Hall of Natural Sciences, St. Olaf College.

Calibration

The calibration we applied is an implementation of openCV. It is based on Zhang's two-camera calibration algorithm [1]. The algorithm uses several images containing a black and white checkerboard (figure 2) at different angles and distances. It is absolutely necessary for images to not only be at different angles, but also to be at different depths; the reasoning is discussed further down. The information gathered by the calibration is used multiple times throughout stereo-pair processing.

Figure 2: 8x6 internal grid calibration target.

Deinterlaced Images

Interlacing is the process of using two, alternating sets of scan lines to record video data. Cameras use interlacing to double the refresh rate, as opposed to progressive scan cameras. Interlacing causes a lot of noise if the cameras are moved too quickly, especially when making turns, and pre-process deinterlacing can reduce this noise.

Due to amount of noise created by the security cameras, it was necessary for us to deinterlace the images before rectification. The stereo-pairs were separated into their odd and even scan lines respectively. Because of the calibration parameters, the images could not just be halved vertically. The rectification (to be discussed further down) is dependent upon the calibration images. This caused us to duplicate a single set of scan lines per image. For rectification we decided to use the even scan line images; this was under the assumption the cameras started with the even scan lines.

Rectification

It is necessary to first rectify the images to produce a proper disparity map. We used an openCV rectification implementation with standard parameters [1]. The rectification was performed on the images above (figure 1), resulting in the images below (figure 3). Based on the calibration the rectification works using a color search algorithm to try to best match the scan lines from each stereo pair.

Figure 3: Rectified stereo pair from figure 1.

Segmentation

After rectification the left image of the stereo pair is segmented before the disparity can be created. The segmentation is necessary for the stereo algorithm. The result is displayed below (figure 4).

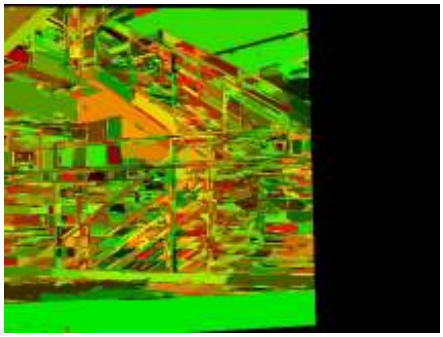

Figure 4: Segmentation of the left image.

Disparity Map

The disparity map is created using the left rectified image, the right rectified image, and the segmented image. The disparity map is created using a color based stereo algorithm. The algorithm is dependent upon the rectified images. The disparity map is needed to later triangulate each point of the image.

Figure 5: Disparity map from the images in figure 1.

Triangulation

From our disparity map, we triangulated our points. Each point corresponded to a pixel in the image. Since the disparity map was based off the left image, the x and y values of any given pixel in the disparity map. The right y value was calculated by using the disparity value of that pixel. The right x value remained the same as the left value. A single x, y, and z value were returned from the function. The same openCV program is used again to achieve this step [1].

SLAM6D

Those values were placed into .3d files to be passed to the SLAM6D software [2]. The z values were also outputted into a separate file to create z-maps. The SLAM6D program does not accept negative values, so we created a single, green point which contained all

of these errors. This aided in debugging our point cloud, and allowed us to view exactly where each of the scans were displayed. The pose information was hard coded it into the necessary file due to the lack of an odometer while taking pictures. This eliminated any roll and pitch changes, since we assumed that any of the indoor environments which were photographed had relatively flat floors. Most of our images were taken in a straight line; however we included a few 90-degree turns to fully test the SLAM6D display.

The SLAM6D (Simultaneous Localization and Mapping with 6 DoF) program that we used was developed at the University of Osnabrueck [2]. It contains software to unify different dot clouds on a single coordinate system, as well as several different ICP minimization algorithms. This required us to make .3d and .pose files. The .3d files contained the number of points followed by a list of points as denoted by their x, y, and z values. The .pose files contained the pose of the robot for the corresponding .3d file. It was formatted as x, y, and z on the top line, and the roll, pitch, and yaw on the second line.

SLAM6D also provided us with a viewer. Since it was originally designed for data from a laser, all dots were displayed in a single shade of gray. We decided to add color to this viewer. To associate color with these newly triangulated data points, we were able to modify the SLAM6D program to ignore the additional color information, which was placed right after each data point in the 3d files, until the point cloud was displayed. We successfully displayed a multi-colored point cloud, which corresponded to the left rectified image (figure 6). To pass the proper color information, the color has to be between the value of 0 and 1. The color was represented as a value between 0 and 255 in our left, rectified image. This forced us to implement a minor conversion by dividing by 255 in each of the color channels.

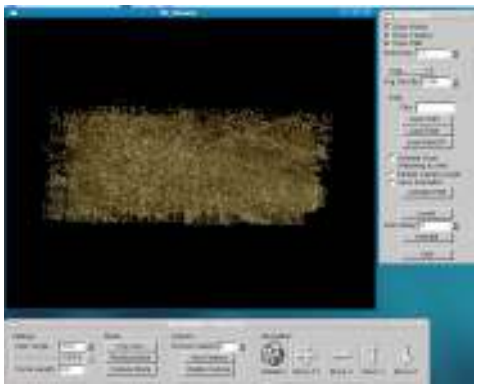

Figure 6: The color point cloud displayed on the SLAM6D program.

Experiments:

Set-up 1

The images were captured using two Sony SuperExwave security cameras set parallel to one another with the center of the lenses roughly 8 inches apart and 8 inches from the front of the a 2'x2'x3/4" wooden board. The cameras sent video images to a Linux box via a wireless transmitter; the entire set-up is run by a home built 12-volt battery. The wooden board was placed on rolling chair.

Figure 7: Set-up one.

Set-up 2

The cameras are set-up exactly the same as set-up 1. Set-up two exchanges the wireless transmitters for a two prong RCA cable to connect the cameras to the Linux box.

Set-up 3.1 & 3.2

Two Sony MVC-CD400/CD250 digital cameras were mounted parallel to each other on a 2'x2'x3/4" wooden board. The cameras were placed mounted roughly 10 inches apart at the same distance from the front of the board. Set-up 3.1 had the focal lengths set zoomed out as far as possible, and set-up 3.2 had the cameras at a mutual zoom. The board was then placed on a rolling chair (see figure 8).

The experiment consisted of taking anywhere between 20 and 35 calibration photos. The cameras were held stationary for the duration of gathering the data for calibration. The photos consisted of stereo pairs where each image has the entire 8x6 internal grid (figure 2). Each new stereo pair has a different angle or distance from the camera than the one before (the new angle and the different distances are important for the rectification).

For set-ups 1 and 2 the cameras were moved in a strictly forward direction taking images every 3 seconds. These images underwent the deinterlace process, contrary to the photos taken by the digital cameras in set-up 3.

For set-up 3.1 and 3.2 the images were taken manually. The distance traveled between each stereo pair was kept consistent for the ease of creating the .pose files. Turns were kept at exactly 90 degrees and only occurred once per data set.

Figure 8: Set-up 3.1 and 3.2.

Discussion

Throughout we have discussed the handling of data sets and the gathering of the data sets; this part of the paper will focus on the information given to us from the data.

The first time through the entire process we used set-up 1. The biggest issue we saw from this experiment was the noise we attributed to sending video over the wireless transmitters. After trying to rectify we quickly saw the data was not going to be good enough to generate a proper disparity map. This led us to experimental set-up 2. The addition of the RCA cable reduced much of the noise; however, we still saw a lot of issues with the rectified stereo pairs. After seeing poor rectification we knew the disparity maps produced would be completely useless. It should be noted here that the calibration errors did not change a significant amount between the wireless transmitters and the RCA cable.

When we started deinterlacing the photos between calibration and rectification, the rectification became better but still not usable. The first attempt at deinterlacing included the calibration images and reduced the vertical size of the image. This proved to be too difficult because calibration assumed the boxes are square. After realizing the calibration problem, the images, not including the calibration images, were deinterlaced as described above. With the best rectification thus far, we attempted to generate a disparity map. This map was the first attempted at using an openCV program to generate a disparity map. This attempt did not give anything close to a usable disparity map. Then we attempted to use St. Olaf's preexisting stereo-algorithm which gave much better stereo results; however, the disparity map generated was still unusable for getting proper triangulation results.

These results pointed us to the original images. We saw the cameras were giving two very different colors for the same point on a given surface. The different colors made the stereo-algorithm give the wrong results because it is based on finding color similarities between the rectified images.

The large magnitude of differences between the colors gathered by each camera forced us to gather higher quality pictures. The use of digital cameras, as described in set-up 3.1,

gave better rectification results and subsequently, better disparity maps. The disparity maps created still were not at a high enough quality to pass to the triangulation function. The high quality data however did give us a chance to go back and look at the results we were getting from the calibration.

When looking at the rectified images we noticed the foreground of the images (the bottom 50 – 60 scan lines) had scan line differences of +/- 8 for well-defined points between the rectified stereo pair. At the same time we were seeing points in the background having a magnitude of +/- 2 for well-defined points. The results implied an issue with the calibration of the cameras. The rectification did not have enough information to rectify the bottom third of the image.

The calibration issues brought about experimental set-up 3.2 to try to overcome the rectification at the bottom by trying to place the cameras at the same zoom point. Zooming caused even more problems. When trying to calibrate and rectify the zoomed images we ended up with very high distortion parameters due to a difference in focal lengths. The rectified stereo pair ended up having roughly 4/5ths of the image as black.

Even with the poor disparity maps SLAM6D did give some as semblance of the real world. The reason we are not seeing the exact same thing we are seeing in the real world is due to the disparity maps having negative values. SLAM6D has no way of handling negative values so we sent it zero values that do not do anything when viewed. The negative disparity values are a result of the issue of the rectification.

Conclusion

In this paper we have presented a possible solution to SLAM using only stereovision. The robot now has the groundwork for it to be able to map an area. We are confident, when given the proper information we will be able to create color 3D maps of an unknown environment.

The stereo will be a great way to gather large quantities of data, including color. We are still working toward effectively testing the process applied above, as well as implementing alternate methods of rectification.

References

[1] Bradski, Gary, and Adrian Kaehler. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, 2008.

[2] Nuechter, Andreas, et al. SLAM6D - Simultaneous Localization and Mapping with 6 DoF. January 2010. <https://slam6d.svn.sourceforge.net/svnroot/slam6d>.