|  |
| --- |
| **Rose-Hulman Institute of Technology** |
| **Status Report Week 7** |
| **CSSE463: IGVC Vision** |
|  |
| Ander Solorzano;Ruffin White;Kurtis Zimmerman;Donnie Quamme |
|  |

Rose Hulman Robotics Team CM 5000

5500 Wabash Avenue

Terre Haute, IN 47803

The team managed to come together and agreed on a plan to accomplish all the deliverables by the expected project due date. The team proposed a rough sketch of the project plan that outlines all of the necessary steps in order to accomplish all the tasks. In addition to the project planning, each member of the team set-up a GitHub account (for all code-related work and deliverables) and a Dropbox account (for image storing) to be used as the team repository. Ander added pictures to the repository that includes pictures from the actual IGVC as well as other field-test images.

**Hardware Mounting**

This week, Ander and Ruffin got the design approval check-off for constructing the mounting bar for the stereoscopic cameras. The design includes a flexible mounting schematic where the spacing between the sensors can be controlled by 2 centimeter increments. The design also includes robust mounting solutions to prevent undesirable augmented movements that would later invalidate the calibration of the cameras after operation. The 2 cameras that we are using are Logitech HD1080p cameras.

**Stereoscopic Calibration and Image Acquisition**

Using LabVIEW as our main programming language, we managed to experiment with native machine and manufacturing vision algorithms to calibrate and test our stereoscopic cameras. We are trying to see how calibration file can be useful for other software applications such as MATLAB.

Ruffin also created an acquisition VI (visual instrument) that allows the user to save and record images from the stereoscopic USB cameras.

The VI includes the following features:

1. Real-time visual feedback for both, left and right, cameras.
2. Histogram representation of HSV and RGB color spaces.
3. Status about the frame rate, frame resolution, and index of the current picture saved.

**Line detection**

We began line detection this week using the algorithm presented in the “GOLD Report” which we previously reviewed. The algorithm works by finding areas of increased relative brightness in order to identify white lines despite the overall brightness or shading in the image. The underlying assumption is that the white lines will be the brightest pixels among the neighborhood of pixels to which they belong. Then, given a specified width we expect the lane line to be, the algorithm will pick out pixels which have a greater brightness than the pixels outside that width. When areas of potential white lines are found and identified via a binary image, we then pass that binary image to a Hough transform to find the significant lines, which should correspond to the lane lines in the grass.

Currently, this method struggles because we have not combined grass detection with this algorithm. Areas of patchy grass sometimes have a brightness similar to the lane lines, so if we can filter the majority of those out, the line identification strength can be greatly improved.

**Grass Filter**

Grass filter work began with attempting to classify points manually in HSV space, similarly to the way the fruit detection project was completed. We decided that this method, while fast, was not sufficient for this project due to the fact that it could not identify lines that were further away from the robot, and spots on the ground that did not have much grass were frequently identified as line pixels. Dilation and erosion were not enough to overcome these problems. Figure 1 contains images that are representative of the problem described. Some lines were identified as substantially grass, and other regions of grass that were patchy were identified as lines.

|  |  |
| --- | --- |
| img14681 | filteredImg |
| Figure 1: Demonstrating problems with manual pixel classifications. | |

After deciding a manual classifier was not enough to complete a grass filter we began work on a SVM to complete the grass filter. This will classify each pixel in RGB space. The first part of this work is acquiring lots of pixels of known type (grass or non-grass) in order to train the SVM. To further this goal we used ginput to allow a user to generate selections from each image that are grass pixels and others that are not.

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |