|  |
| --- |
| **Rose-Hulman Institute of Technology** |
| **Status Report Week 9** |
| **CSSE463: IGVC Vision** |
|  |
| Ander Solorzano;Ruffin White;Kurtis Zimmerman;Donnie Quamme |
|  |

Rose Hulman Robotics Team CM 5000

5500 Wabash Avenue

Terre Haute, IN 47803

**Improving Grass Filter using k-means Approach**

We attempted to increase the accuracy of our grass filter. Previously, the grass was filter using an SVM that looks at the RGB values of a single pixel for its features. In our efforts to increase accuracy we modified how many pixels in a cluster the SVM used, as well as using various other features such as color moment. Unfortunately, varying these parameters did not increase the accuracy of the grass filter.

We then attempted to filter grass using a k-means approach. In this approach, we applied a k-means to a number of pixels that are known to be grass, to generate a set of average grass classes. Then we applied k-means to a test image. Whichever means were significantly close to the known grass means were called grass, and added to the mask. This method provided mixed results. For some images it worked fairly well, as shown on the picture below and to the left. However, on other images it worked very poorly, marking all pixels in the image as grass pixels. This is obviously unacceptable, and so we have determined that k-means is likely not a good way to generate a grass filter.

|  |  |
| --- | --- |
|  |  |
| Figure 1: Sample of image that worked fairly well for applying a grass filter via k-means approach. | |

**Experimentation with Lane Detection**

This week, we experimented with a couple alternative methods for white line detection to deal with our perspective issues. An issue we have seen is that lines deeper in the background may not appear as bright or as wide simply due to the point of view. Thus, we tried varying the brightness/width thresholds based on position in the image. Unfortunately, the inconsistency in the orientation of lines still prevented detection from being completely accurate. In fact, the original method (demonstrated last week) remained the preferred method. We still believe that the reoriented perspective will help particularly with the issue of lane line width, as perspective will not be able to vary the perceived thickness.

**Red Flag Detection**

Another primary obstacle to be detected consists of small red flags. We set up a straightforward red flag detector based on the hue in HSV space, much as was done in the fruit finder, dilating with a 3x3 square to account for edges and ensure that we overestimate the size of the flag. Although we don’t have any sample images consistent with the type of red flags used in the competition, we were able to run the detector on some slightly similar flag images with promising results. Because of the prominence of the red against the grass, flag detection should be reliable, varying only in situations with small flags or flags which are not fully visible due to orientation, wind, etc.

|  |  |
| --- | --- |
| cid:image001.png@01CE0581.EA523820 | cid:image002.png@01CE0582.49745090 |
| Figure 2: Sample of red flag detection using HSV color space. | |

**Improving Barrel Detection Techniques**

In order to improve some of the results acquired from using pattern matching techniques, the team decided to modify the template and create a monochromatic template for each of the bands (i.e. solid grey and solid white barrel template). The goal was to acquire more reliable results for many types of barrels that vary due to the amount of lighting, size, and image view point. Initially, we decided to take a band of the barrel instead of an entire barrel in order to allow the unique “stripped” pattern to be more easily detected.

|  |  |  |
| --- | --- | --- |
| C:\Users\solorzaa\Desktop\Barrels\Templates\barrelTemp3.png | C:\Users\solorzaa\Documents\RHIT\RHIT SR\Winter Term\CSSE463\IGVC Vision\MatLab\Barrels\Templates\barrelTemp1.png | C:\Users\solorzaa\Documents\RHIT\RHIT SR\Winter Term\CSSE463\IGVC Vision\MatLab\Barrels\Templates\monoTemp1.png |
| Figure 3: Selected template transformation procedure for pattern matching detection of construction barrels. | | |

|  |  |
| --- | --- |
| C:\Users\solorzaa\Desktop\Barrels\barrel4.jpg | C:\Users\solorzaa\Desktop\Barrels\SR Results\figure2.JPG |
| C:\Users\solorzaa\Desktop\Barrels\barrel4.jpg |  |
| Figure 4: Barrel detection improvements. The images on the left column are the initial images and the images on the right side represent the barrel acquisition images. The bottom right image shows improvements in the image recognition by detecting “barrel-like” patterns close or within the barrel. Unlike the top right image, the bottom left shows no images on the side. | |

In order to increase the noise detected in the bottom-left image from the previous figure, the team decided to apply a “skyscraper barrel mask” to the image wherever a barrel was detected. The “skyscraper barrel mask” would start from the point that the barrel was detected and then apply several masks (of similar size and shape) from the detected barrel to the top of the image. This would create blind spots past the barrel but since the robot cannot jump over the barrels or fly and since the robot should only be aware of its close immediate environment, the robot doesn’t need to know what’s beyond the first obstacle until after it passes the first obstacle. This would reduce computation time in our algorithm and provide faster reaction time for our robot.

The goal of the team, with respect to barrel detection, would be to implement this algorithm and have it ready for demonstration. Although this newly thought part of the project has not yet been implemented, the figure below shows the idea of what the team is trying to pursue.

|  |
| --- |
|  |
| Figure 5: Image showing the *idea* behind the “infinite skyscraper barrel” approach. *This is the next tasks for the team.* |

**Improving the Image Distortion**

During this week, the team had continued to complete the image acquisition and perspective distortion routine utilizing stereoscopic vision and perspective distortion matrices. While generating the initial calibration files, the team was confronted with the severe lack of identifiable uniform objects as a switch to serve for the point based calibration routine. Utilizing an old tripod for simulating the cameras angle and elevation as anticipated within the robotic superstructure, the team used some pizza boxes along with measured distances to serve as the reference coordinates to which the image coordinates would be map to.

The figure below represents the result of the image calibration. The tedious process of point based calibration is performed by marking and identifying specific points and coordinates within the image that represent real space dimensions. In this instance, we chose the four ends of the pizza boxes.

|  |
| --- |
|  |
| Figure 6: Image after calibration has been acquired. Here you can see the pizza boxes used in point-based calibration routine. |

When generating the point based calibration file, is important to take detailed measurements, regardless of the absurd use of pizza boxes as one’s calibration tool. A small deviation in error can result in a large skewing effect distorting the image geographically further in the upper background regions. Using the tiles of the carpet and taking measurements in imperial units, the coordinates are recorded and converted to metric with the floor layout being quickly sketched out using a modeling tool for easy future depiction or for more numerous points later if required.

|  |
| --- |
| C:\Users\whitemrj\Desktop\Cal setup.png |
| Figure 7: Metric measurements of image using a modeling tool. |

After sequentially specifying each marked point of interest and its respective real space coordinates, the distortion model is calculated. In this case we use the rather simple and less accurate method of division rather than higher-order term computation such as polynomials or other methods due to its efficiency and relatively short computation time. As we require a high refresh rate for faster sensor acquisition, therefore quicker response times, we would prefer to maintain an image processing structure that is minimally computational intensive but still relatively accurate. To check the calibration accuracy, we can take a look at the error map generated by our distortion model. By observing the error range within the bottom right corner, we can see that the magnitudes are severely minute. However, this is the direct representation of the distortion model based on the ideal pixel coordinates as specified previously. There still a great deal of human error in recording the calibration space as well as environmental variables such as the actual camera placement and angle as subject to physical mechanical variances. This level of accuracy is not required within our robot, as the goal is to simply avoid lines rather than a geological surveying their existence.

|  |
| --- |
|  |
| Figure 8: Front-panel view (GUI) of the calibration training interface. |

Figure 6 shows the first rendition of the corrected image. You will at first notice that the image has been mirrored horizontally and is thus upside down. This is due to the respective orientation of the origin correlating with the origin of the image, mainly the Y axis pointing in the same direction of increasing number of the row index as expected with most standards of representing images as matrices. You may also notice the severe skew of the length of the left side first the right side of the corrected image. This is due to the slight rotation and the subject of calibration and its reference in real space or orientation with that of the camera. It could be possible through trial and error to move the camera in a sweeping arc across the subject of reference to correct for this skew and leave the bottommost edge on a level plane. However, subjects that fall within this distance range are of lesser concern than closer objects that will be correctly represented, thus the outermost edges may eventually be masked or ignored when converting detected lines spaces to an occupancy grid representation. This entire process is repeated again in order to generate the second calibration file for the remaining stereoscopic camera, the specific location of the camera within the reference frame of the real space origin will differ, or specifically polarity with respect to its X axis offset, as origin is located directly below and in-between the two cameras.

|  |
| --- |
|  |
| Figure 9: First rendition of the distorted image. |

Now that some simple calibration files have been constructed we can augment the results within our video feedback VI. In order to make the perspective distortion useful for both LabVIEW video acquisition and MATLAB test images, we need to do a bit of overhaul to allow for static image acquisition as well as update the interface to allow for specifying right and left calibration files as well as image source and destination file directories. It is important to note that even with working with relatively small stereo image dimensions of 800x600, that by including perspective correction subroutines, our vision acquisition loop has increased in cycle time from about 30 to 50 ms to now roughly 1.3 seconds. It was also observed that CPU usage raised to peak levels when performing this operation. This foreshadows that severe resolution sacrifices may be necessary in order to reduce computation intensity and improve acquisition speeds, even before considering the additional processing time that will be required for line detection and mapping.

|  |
| --- |
|  |
| Figure 10: Undistorted and distorted real-time image video stream for both cameras in the front-panel GUI. |

Here is a simple block diagram showing the image acquisition process, either acquiring a live video stream, or a specified file. The image parameters are then displayed on the front panel while the stereoscopic images are forwarded on to perspective processing if such cases is enabled. The results are then displayed and can be recorded should the user specify. The periodicity of the entire while loop is set for 3 seconds, this is to allow for less strenuous CPU usage.

|  |
| --- |
|  |
| Figure 11: Block diagram showing the image acquisition process. |

Here you will observe a high resolution depiction of the corrected perspective for one of our calibration images. You will notice the perfectly rectangular features of the centralized pizza boxes, as would be expected since they were the focus of our calibration efforts.

|  |
| --- |
| C:\Users\whitemrj\Desktop\New folder\2013-02-08_01-02-11-.953__L.jpeg |
| Figure 12: High resolution distorted image showing the centralized pizza boxes. |

|  |
| --- |
|  |

GitHub Repository Location: <https://github.com/rhrt/IGVC-2013-Imaging>