### CHANGE THIS TITLE

*Donnie Quamme, Ander Solorzano, Ruffin White-Magner, Kurtis Zimmerman*

Department of Computer Science

Rose-Hulman Institute of Technology  
{quammebd, solorzaa, whitemrj, zimmerka}@rose-hulman.edu

#### Abstract

Insert abstract here.

**1. Introduction**

Insert intro here.

2. METHODOLOGY

We now present the four major components of the robot’s acquisition and recognition process, presented one-by-one with individual results following each step.

**2.1. Perspective correction**

In order for the line detection algorithm to detect relevant lines with respect to the reference of the ground plane, the acquired image from the robot’s perspective must be corrected for the planar distortion when observing the IGVC field from low elevation. A more suitable representation of lines and its locally geographical figures would be from a bird’s eye (top) view. Logically being within the same dimension as the robot will inevitably express its own occupancy grid, this will allow for easy conversion between raw data and geographical features such as describing avoidable obstacles and potential paths to sub goals.

Shown in Figure #, after a sample calibration images has been acquired, the tedious process of point-based calibration and is performed by marking and identifying specific points and coordinates within the image that represent real space dimensions. In this instance, we have simply chosen the four ends of the pizza boxes as identifiable uniform objects as switch to serve for the point-based calibration routine. By utilizing a tripod to simulate the camera’s angle and elevation within the robotic superstructure, we measure the distances to serve as the reference coordinates to which the image coordinates will eventually map.

|  |
| --- |
|  |
| **Fig. #.** 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, it is important to take detailed and consistent measurements, regardless of the calibration tool; a small deviation in error will result in a large skewing effect, further distorting upper background regions within the image. In Figure #, the floor layout is sketched out using a modeling tool for easy future depiction or for extracting more futuristic points later if required.

|  |
| --- |
| Cal setup |
| **Fig. #.** 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. As a real-time operated navigational robotic system requires a high refresh rate and its perceptual environmental awareness, faster sensor acquisition and processing is a direct constraint for system response times. It was important to maintain processing structure that is minimalistic in computational intensity but still relatively accurate. In this case we use the rather simple and less mathematically 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. It is possible to make this assumption by disregarding potential discontinuities and higher-order effects such as camera lens distortion. Such assumptions would be harder to render when considering the use of different hardware such as fisheye lenses when attempting to achieve a wider angle of peripheral perception.

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 of Figure #, we can see that the magnitudes are severely minute. This is the direct representation of the distortion model based on the ideal pixel coordinates we specified, but there is 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. Extreme accuracy is not required within our robot, as the goal is to simply avoid lines rather than geologically surveying their existence. However, great care must be exerted when implementing the camera fixtures upon the robotic platform to ensure that unanticipated camera vibrations or displacement is minimized when traversing rough terrain in order to ensure reasonable and consistent results.

|  |
| --- |
|  |
| **Fig. #.** Front-panel view (GUI) of the calibration training interface. |

Figure # exhibits the first rendition of perspective corrected calibration image. Notice that the image is mirrored horizontally and thus appears upside down. This is due to the respective orientation of the origin correlating with the origin of the image, mainly due the Y axis pointing in the same direction of ascending value of the row index as expected with most standards of representing images as matrices. Also notice the severe skew of the length of the left side versus 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, as reaffirmed within the line detection section of this report. It is such that 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 second and remaining stereoscopic camera.

|  |
| --- |
|  |
| **Fig. #.** First rendition of the distorted image. |

Once the calibration files have been constructed, we can implement the results within our video feedback virtual instrument, or VI. In order to make the perspective distortion useful for both LabVIEW video acquisition and Matlab test images, we overhaul our acquisition process to allow for static image as well as methods for programmatically specifying right and left calibration files as well as image source and destination directories. It is important to note that even when working with relatively small stereo image dimensions of 800x600, that by including perspective correction subroutines, our vision acquisition loop is subjected to and increased cycle time from about 30 to 50 ms per frame to roughly 1.3 seconds. It is also important to know that average CPU usage also increased from around 30 to 40% to 95 or 100%.

|  |
| --- |
|  |
| **Fig. #.** Undistorted and distorted real-time image video stream for both cameras in the front-panel GUI. |

Figure # shows our high-level block diagram, depicting the beginning image acquisition and perspective correction process. By reading the user-defined input parameter, the loop continues to either acquire a live video sample from the USB camera hardware, or read in an image from a specified directory. The image parameters are then displayed on the front panel while the stereoscopic images are forwarded on to perspective processing if enabled, where the results are then displayed and can be recorded. The periodicity of the entire loop is set for 3 seconds to allow for less strenuous CPU usage. That time could be reduced if further hardware improvements are made.

|  |
| --- |
|  |
| **Fig. #.** Block diagram showing the image acquisition process. |

In Figure #, observe the finished product of our corrected perspective for one of our previous calibration images. Notice the perfectly rectangular features of the centralized pizza boxes, as would be expected since they were the region of interest for our prior calibration efforts.

|  |
| --- |
| 2013-02-08_01-02-11-.953__L |
| **Fig. #.** High resolution distorted image showing the centralized pizza boxes. |

**2.2. Grass filtering**

**2.3. Line detection**

An integral portion of the robot’s navigation is identifying and remaining within the bounds constrained by white lines. Based on a lane line detection algorithm developed in the GOLD report (ref #), we developed a process for detecting white lines on a grassy field.

Whereas in (ref # of GOLD Report), the lines being detected were bright white lane lines on a significantly darker pavement background, the lines in this competition are painted on a grassy background, so the contrast is significantly reduced. Additionally, the material difference between pavement and grass meant the lines are not as uniformly solid in our test images.

One last difference between our scenario and that from the GOLD Report is the type of lines we may encounter. Whereas lane lines on a road run parallel and – within some short distance – are completely vertical, within a small error, the lane lines on the competition field need not be parallel, may arc significantly, or – as in the case of a turn – may run horizontal.

|  |  |
| --- | --- |
| img39685 |  |
| **Fig. #**. The types of lines to be detected include solid white straight lines, dashed lines, and arcs. The lines are not guaranteed to be uniformly distinct, as shown in these two images. | |

However, the basics of the lane line detection algorithm can be further enhanced to work with an acceptable degree of accuracy. While the original algorithm makes just one pass, scanning horizontally for significant changes in brightness, we add on a vertical pass to account for horizontal lines, arcs, and even diagonal lines.

To begin setting up, we first convert the image in RGB space to HSV space, using the value band to represent brightness. As is shown in Figure #, while not perfectly consistent, HSV space provided the most consistent brightness indicator, as compared to two other calculations found in (ref #), distance in RGB space as well as a more refined weighted distance measure.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a) RGB Distance | (b) Weighted RGB Distance | (c) HSV Value Band |
| **Fig. #.** Available measures for perceived brightness in an RGB image | | |

Then, instead of doing two iterative passes over each image, we applied a filter which accounts for the same calculation that the GOLD Report computes, accounting now for our vertical pass as well. In Figure #, the matrix shown can be expanded with more zeros to account for the expected line width. Applying the filter will amplify all pixels which are candidate line pixels, while grass field pixels will be reduced to lower intensities. The result of this filter is shown in Figure #(b).

Because of dead grass and patches of dirt, some grass pixels are misclassified as line pixels, so we utilize the power of the grass filter to remove as much of the grass background as possible, as shown in Figure #(c). Lastly, applying a simple Hough line transform to the remaining binary image, we end up with a binary image marking all remaining strong candidate lines, the final result given in Figure #(d).

|  |
| --- |
| (a) |
| (b) |
| (c) |
| (d) |
| **Fig. #.** Starting with the original color image, three primary steps are applied to detect white lines. |

As shown in the resulting image in Figure #, our algorithm also worked well on arcs, which we did not initially anticipate, so no further refinements had to be made to account for different shapes of lines.

However, due to perspective in each of the images, faded lines and distant lines could not reliably be picked up by the line detection algorithm, like those shown in Figure #. While not ideal, the approach is acceptable for two reasons: first, distant lines are not an immediate risk to the robot and can likely be picked up by the robot if approached; and second, the perspective mapping mentioned earlier will improve the line detector’s ability to pick up lines which appear smaller because they are further in the distance.

|  |  |
| --- | --- |
|  |  |
|  |  |
| **Fig. #.** Lines which appear faded or distant cannot reliably be picked up by the line detector due to the brightness threshold and expected line width. | |

**2.4. Barrel detection**

3. CONCLUSIONS

Using a four-step pipelined process, the groundwork is in place to allow for accurate and efficient image acquisition and processing for the robot’s image-based vision. Because each component has been developed independent of one another, each filter acts much like a plugin, so components can be added or removed as deemed appropriate for the final competition setup.

The image calibration technique developed for remapping perspective provides a more advantageous view for line detection, and with significant hardware and algorithmic improvements, could help streamline the consistency of lane lines across all images, regardless of orientation or distance in the image’s view. At the same time, because barrel detection relies more heavily on the original perspective of the image, those two components could run in parallel to minimize total execution time for one frame without reducing any accuracy.

The same can be said for grass filtering and line detection. While both need to be applied to the same perspective, because the grass filter can take slightly longer due to its underlying SVM processing, the first two steps – computing brightness and applying the relative edge filter – can be started while the grass filter is still running, again without making any sacrifice to accuracy.

Because all components were tested on a non-ideal set of test images, accuracy is expected to improve dramatically when the components are applied to images taken from the competition field. In particular, the lane lines are expected to be much more uniformly white and bright, and the grass should be more consistent in its coloring and uniformity. The one downfall with this knowledge is that the grass filter may need to be reconfigured to account for the differences in grass pixels. Because the SVM was trained on grass pixels from the non-ideal images, there is the possibility that more ideal grass pixels will not actually be classified correctly. Fortunately, referring back to Figure #(b), line detection can still be computed with very good accuracy even without the grass filter, so no serious drop will occur in line detection.

4. FUTURE WORK

There are several areas that can be expanded upon and improved to enhance the robot’s functionality and accuracy.

In the near future, the most pressing need is to combine all of the developed features to be used in tandem. The most effective sequence of filters would be the following ordering:

1. Acquire camera image(s).

2. Remap perspective to top view.

3. Detect and remove construction barrels.

4. Detect and remove grass pixels.

5. Detect white lane lines.

All information obtained would then be combined in an occupancy grid to map the robot’s perceived world at that time.

In terms of line detection, there are a couple of short-term enhancements and long-term enhancements that could be made to improve accuracy and computation time. In the short-term, further experimentation with morphological operations to improve the shape of detected lane lines could improve some of the lines which are not picked up as connected components; morphology may also be helpful in removing extra undetected grass pixels prior to detecting lines. Additionally, to deal with issues of reduced brightness of distant pixels, the brightness thresholds could be adaptive based on the position of the pixels within the overall image.

In the longer term, computation from frame-to-frame could be improved by interpolating the position of the lines in the next frame based on the detected lines in the current frame. Especially for a straight line, determining its slope would allow us to interpolate its starting point in the next frame and utilize that information to make line detection more intelligent.

Additional areas of improvement and further investigation would include techniques for reducing computation time and resource requirements. After attempting to re-create similar performance within software, it is easy to recognize the appreciable performance hardware implementations can bring when observing GOLD Report’s profound systemwide refresh rate of over 20 Hz. It would thereby be advantageous to invest time to port our own developed code into something executable on a real-time platform. Sn FPGA would be ideal for the high-speed and repetitive tasks and mathematical calculations; such could be achievable by utilizing pre-existing platforms like NI’s cRIO. Additional tunable parameters could include reducing the native image resolution when performing image acquisition; reducing the number of dimensions or color spaces when performing distortion calculations, as well as reducing the severity of perceived distortion could help ease the computational overhead of perspective correction process.

5. REFERENCES