Design and Engineering of an Autonomous Line Following Pacer for Application in Track and Field

Robert Miranda

July 16, 2018

1 Abstract

A fully automated line following robot was constructed from a $\frac{1}{10}$ scale model RC Car. Specifically, the line follower was intended to be used in a track and field environment, acting as a pacer for a track race or workout. The pacer uses a Raspberry Pi as an image processor to determine its exact position relative to a standard line. A Proportional Integral Derivative (PID) algorithm is used to control the direction of the car based on this input. The pacer can accurately follow lines, although because of its limited range of view it suffers from not being able to self correct if it loses sight of the line.

Contents

2 Motivation and History

The hardest part of being a track and field athlete is being able to run the right pace. The seconds between running too fast and running too slow are a daily struggle in practice, and are even more important in races. But while every other aspect of the sport, from the rubber of the tracks to the carbon fiber in 3D-printed shoes, has been updated with modern technology, athletes must still rely on a 'gut and feel' approach to pacing.

The goal of this project is to provide a reliable, mechanical alternative to a human pacer. This pacer would be a line following robot, specifically targeting the lane lines of a track. It would be able to keep a constant pace, and would be able to accurately follow the curve of the track.

Line following has become a standard challenge in robotics, with several organized competitions dating back until the early 2000s. Competitors are judged for their accuracy in following the course, as well as the speed of their run. However, the pacer would not need to perform at this level, as line following competitions often ask robots to navigate pseudo-random paths with sharp turns. The pacer would only have to navigate the smooth oval turns found on a track.

Line followers have very practical industry applications. For one example, a line following technology could replace traditional conveyor belts in a warehouse system, similar to Amazon's autonomous system. Additionally, the technology has obvious applications towards autonomous vehicles, although clearly a car that drives over the center line would not be ideal. Nevertheless, a successful line follower would provide valuable insight into how to program a self-driving car. But line followers can be used in much less dynamic situations, such as a personal home assistant, helping elderly or disabled residents move around their home in much the same way as a stair lift does today. Or in a public setting such as a mall or museum, providing a virtual tour or offering free samples.

However, line followers have only a very limited application in track and field. Currently, humans make up nearly 100% of all pacers, with professional pacers sponsored at every professional meet from 800m to a Marathon. And as expected, humans are not perfect, and often poor performances and slow races are blamed on a pacer missing their splits. As of now, the only applications of autonomous pacers seem to be in individual projects. There are several designs online, each with questionable specifications and no broad use. Puma did release a model named the Puma BeatBot, but the model is only available for professional athletes sponsored by Puma meaning the majority of the runner market still has to rely on a human pacer.

A successful project would entail solving the pacing problem. A pacer would be able to perform the needs of any workout, setting a desired pace and distance for an athlete to follow. It might even replace sponsored pacers in professional races. Hopefully it could lead Menlo's track and field team to a WBAL League Championship.

3 Theory of Operation

The pacer uses image analysis with a Proportional Integral Derivative (PID) controller algorithm to follow a given line. The pacer begins by taking a constant video feed of the area immediately in front of the pacer with the main Raspberry PiCamera. This image is constantly updated as the pacer moves, and the data is sent to the Raspberry Pi where the image is processed.

The Raspberry Pi processes a single frame at a time, taken from the live video stream. The image is converted into a 640 x 480 x 3 matrix of values, containing the red, green, and blue (RGB) color values for each pixel in the image. This image is first simplified to reduce noise and improve processing.

First the RGB image is converted to a grayscale matrix. Then a Gaussian Blur is performed on the matrix, whereby each pixel value is taken to be an average of all the pixels around it. This reduces noise in the image, which significantly improves the performance of future processing. Specifically, the Gaussian Blur is defined in integral form over the image by

$$\iint_D G(x,y)\,dA = \iint_D \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\,dA$$

where x and y are the distance from the current pixel, and σ is an input standard deviation parameter. Intuitively, this means that neighboring pixels have a much greater impact than pixels that are further away. In application, it is often more computationally effective to only calculate a Gaussian Blur two standard deviations away from a given pixel.

Now that the image has been grayscaled and blurred the Raspberry Pi begins performing a contour analysis. The concept is that the largest contour in the image will be the intended line, so the algorithm searches for the center of the largest contour. Normally this would involve taking a double integral over the region of the contour, calculating a weighted average to find the center of the figure.

$$M_x = \frac{\iint_D x\,dA}{\iint_D dA}$$

However the algorithm cannot do this because it would be extremely inefficient. Specifically, the algorithm defines a contour as the largest continuous curve of equivalent pixel colors. (This is equivalent to treating the matrix as a function on x and y and finding the largest level curve on the surface.) This path is then stored as the contour, not the area it encloses. To approach this problem the algorithm invokes Green's Theorem to calculate the center of the contour while only processing the boundary.

Specifically, Green's Theorem relates a double integral over a region of a continuous, smooth function to the line integral around the boundary of the region.

$$\oint_c < L, M > \cdot d\vec{r} = \iint_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}\right) dx\,dy$$

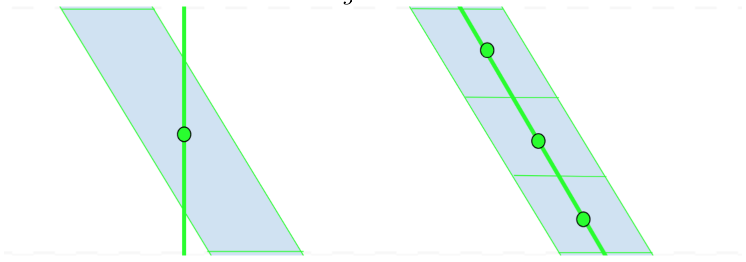
Where $\vec{F} =< L, M >$ is a vector valued function. Thus the line integral of the divergence of a potential function around the boundary of the region is equal to the curl of function integrated over the region. This allows the original double integral for the center of the contour to be rewritten as a line integral around the contour (which is stored by the contour detection algorithm) by taking $\vec{F} =< 0, \frac{1}{2}x^2 >, < 0, x >$.

$$M_x = \frac{\oint_c < 0, \frac{1}{2}x^2 > \cdot \bigtriangledown\ dA}{\oint_c < 0, x > \cdot \bigtriangledown\ dA}$$

However, defining a single contour over the whole image poses several problems. For instance, while taken over the whole image there is a greater probability that the algorithm will identify a contour other than the line to analyze. Alternatively, the line may pass through the frame at an angle, or it may curve, but if the center aligns with the center of the image the pacer will blindly proceed straight when it should turn. To take advantage of this information, the image is split into several horizontal sections.

Each section undergoes the described contour detection algorithm to find the center of the 'line' in the section. This reduces the risk that the algorithm will misidentify another contour as the line. In addition, the use of sections allows the pacer to respond to changes in the direction of the line, such as a curve or a line that passed through the image at an angle, but has a center aligned with the center of the image.

Figure 1: Advantages of Using a Linear Regression from Multiple Contour Sections. Notice that by splitting the second line into three sections vertically, the line can be better approximated than just by using one contour across the whole image.



However, because of the limited range of view of camera the pacer faced difficulties following a curved line, especially at high speeds. This is improved by fitting a linear regression to the fitted contours to predict where the line will be ahead of the field of view of the camera.

Specifically, a linear regression is fit using a least squares regression algorithm. A best fit line of the form

$$y \approx b_1 x + b_0$$

is defined. Then a cost function is defined by summing the squared errors of the best fit line across all of the data points.

$$J(b_0, b_1) = \sum_{i=1}^{n} = [y_i - (b_1 x_i + b_0)]^2$$

This cost function is minimized by setting $\nabla J = \vec{0}$ and solving for the corresponding values of b_0, b_1, thus defining a best fit line through the data. Specifically

$$\vec{b} = \begin{pmatrix} b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} \sum\limits_{i=1}^{n} x_i^2 & \sum\limits_{i=1}^{n} x_i \\ \sum\limits_{i=1}^{n} x_i & n \end{pmatrix}^{-1} \begin{pmatrix} \sum\limits_{i=1}^{n} x_i y_i \\ \sum\limits_{i=1}^{n} y_i \end{pmatrix}$$

The range of the image is 480 pixels in the direction of the pacer's motion. Thus the linear regression is evaluated at x = 960. This predicts the location of the line ahead of the field of view of the pacer.

This method is susceptible to over fitting the line in two ways. First, fitting too many sections to the image, and second fitting too many variables to the regression. Note that as the number of sections increases, the size of each section correspondingly decreases. Thus it is more likely to misidentify a contour that is not the line as the number of sections increases. Additionally, using a regression with too high a degree is also prone to over fitting the data. For instance, it is easy to see that if the image is cut into n sections and an $(n-1)^{th}$ degree polynomial is used the polynomial will pass through each contour point. (Note that this is just an application of Lagrange Interpolation) Originally, four sections with a quadratic regression were used, but this was found to over fit the data. A linear regression with three sections was found to perform better.

The distance from this projected line location to the pacer's direction, assumed to be center of the front facing camera image, is fed to the micro controller. Specifically, the controller uses a Proportional Integral Derivative (PID) algorithm to adjust the pacer's direction relative to the line.

The proportion term is perhaps misleading: it simply represents the current distance from the pacer's direction and the line. Clearly the pacer should turn towards the line, and correspondingly a larger separation should create a larger change of direction from the pacer, acting almost as a restoring force to keep the algorithm on equilibrium over the line. However, this causes the pacer to oscillate over the line. (As an analogous example, consider the ideal behavior of an oscillating spring acting under Hooke's Law.)

To solve this a derivative term is needed. This weights the rate of change of the error term, and acts as a dampening constant. In effect, it smooths the harmonic oscillations until the pacer comes to equilibrium on the line. The final term used in the integral term, which simply adds the total value of all the error terms. Although often excluded, the integral term can fix steady state errors, where the pacer is stuck a constant distance from the line. Because the error is constant, the derivative term will not kick in, but the growing integral term will eventually bring the pacer back to the center.

The PID controller is extremely sensitive to the conditions of use, and thus must be tuned precisely to the specific parameters of the pacer. Initially, the proportional term is slowly increased until the pacer oscillates over the line; then the derivative term is increased until the oscillation is sufficiently damped. This process is repeated for sufficiently desired results. Finally, the integral term is increased when the presence of steady state errors are found.

The final direction is set as a linear combination of the PID terms. Specifically with weights $\vec{\lambda} =< \lambda_p, \lambda_i, \lambda_d >$, the direction is set as

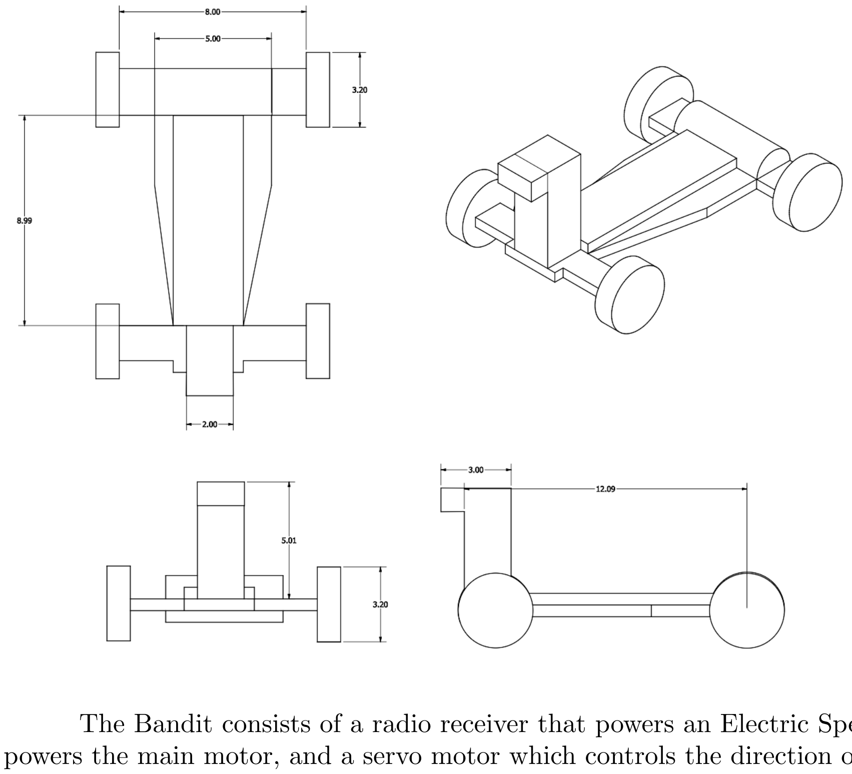
$$d =< P, I, D > \cdot \vec{\lambda}$$

4 Design

The design of the pacer relies on using the preexisting functionality of an RC car to control speed and direction. Specifically, the pacer was based off of a Traxxas Bandit XL-5.

Several mechanical changes were also made to optimize the Bandit for performance as a pacer. A camera mount was designed and laser cut with a Maker Case style interlocking design. It positions the Raspberry Pi Camera at a 45° relative to the horizontal. Additionally, a frame was constructed around the main body of the Pacer, making it more sturdy upon impact. Additionally, several LED's were used to make a headlight for the Pacer above the camera mount, which improved performance in low visibility situations.

Figure 2: Isometric and Orthogonal Views of a CAD Model of the Modified Bandit



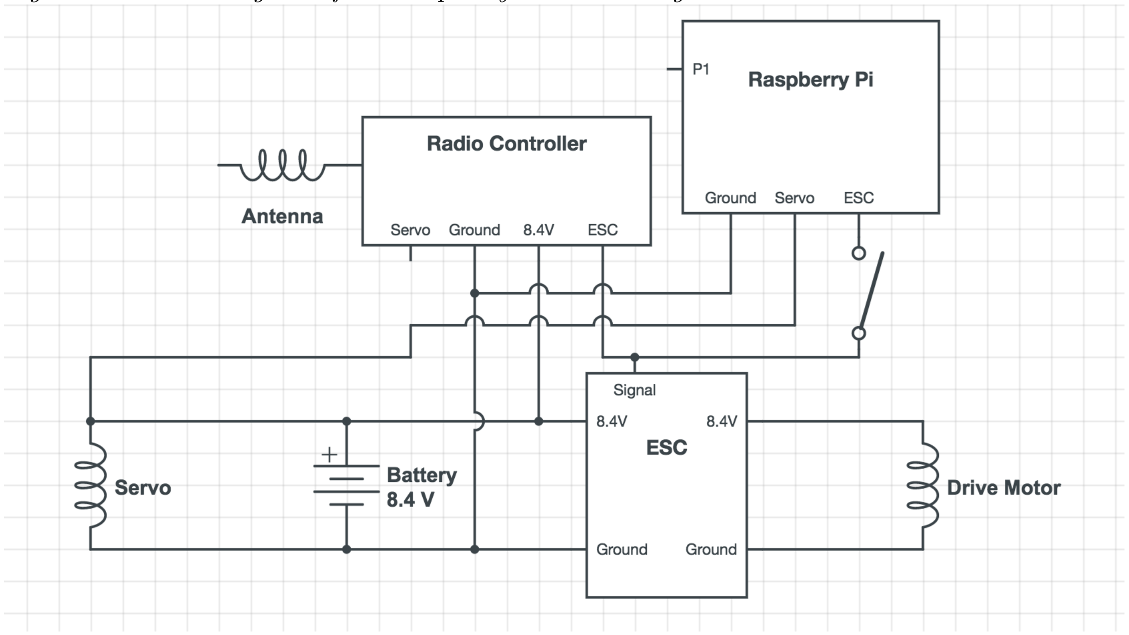
The Bandit consists of a radio receiver that powers an Electric Speed Control, (ESC) which in turn powers the main motor, and a servo motor which controls the direction of the front wheels. The Raspberry Pi acts in place of the radio controller, powering the ESC and servo in place of the radio transmitter system.

Specifically, the radio receiver sends Pulse Width Modulation (PWM) signals with varying duty cycles to control the ESC and servo. The receiver sends a square wave with a period of 100hz and a varying duty cycle ranging from 10% to 20% of the total period.

For instance, a 100hz PWM signal with a duty cycle of 10% from the servo channel corresponds to a full left turn, while a duty cycle of 20% corresponds to a full right turn. Similarly, a 15% duty cycle from the ESC channel corresponds to the lowest possible motor speed, while 20% corresponds to the maximum throttle control. To measure each signal, the PWM channel was connected to an oscilloscope. Then the corresponding control on the radio transmitter was triggered, whether the control for the left turn, full throttle, etc. This signal was then interpreted by the radio receiver, and a PWM signal was sent to the corresponding ESC or servo. However, this signal was intercepted and measured by the oscilloscope, and provided a full mapping of the controls of the Bandit.

The Raspberry Pi is connected to the ESC and servo in place of the radio receiver via the General Purpose Input Output (GPIO) pins. Each emits the desired PWM signal to control the pacer, allowing for the Raspberry Pi to have full control over the Bandit. However, the radio receiver cannot be fully removed from the circuit. The original circuit design for the Bandit involved the power source for the servo traveling from the battery, through the ESC, then through the radio receiver, and finally to the servo. Although this series circuit is not ideal, it is beyond the scope of the project to redesign the Bandit, and the radio receiver is left in the circuit.

Figure 3: Circuit Diagram of the Raspberry Pi Controlling the Servo and the Motor



The pacer also relies on the Raspberry Pi to control the speed of the pacer. Because the speed of the Bandit varies as the onboard battery drains, the speed cannot be accurately controlled by modulating the input of the Raspberry Pi. Therefore, the Pacer directly measures its speed in real time by measuring the period between rotations of the wheel.

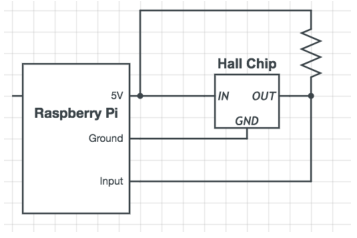
Specifically, the pacer has a build in magnet placed on one of the rear (driven) wheels. The magnetic field generated by this magnet is then measured by a Hall Chip every revolution.

The Hall Chip relies on the magnetic component of the Lorentz Force on moving charges in a magnetic field.

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B})$$

Specifically, the magnetic component of the Lorentz Force is calculated as the cross product of the velocity of the moving charge and the magnetic field. Note that because of the definition of the cross product, this force is directed perpendicular to both the velocity of the charge and the magnetic field. This phenomena, known as the Hall Effect, means that a moving charge passing through a magnetic field will tend towards the direction of the Lorentz Force, creating a potential difference across the path of the charge. The Hall Chip sends a current and measures the corresponding electric potential difference across both ends of the Hall Chip. This voltage difference allows the Hall Chip to act as a controlled transistor, allowing current to flow when a magnetic field is detected, and preventing current flow when no magnetic field is present.

Figure 4: Circuit Diagram of the Hall Chip Configuration.



The Hall Chip is positioned such that it can only detect the magnetic field of the wheel magnet at a certain position on each rotation. Thus by measuring the signal from the Hall Chip the period of the wheel can be used to calculate the speed.

Specifically, the linear velocity is related to the angular velocity of a wheel by the radius of the wheel.

$$\vec{v} = r\vec{\omega} = r\frac{2\pi}{T}$$

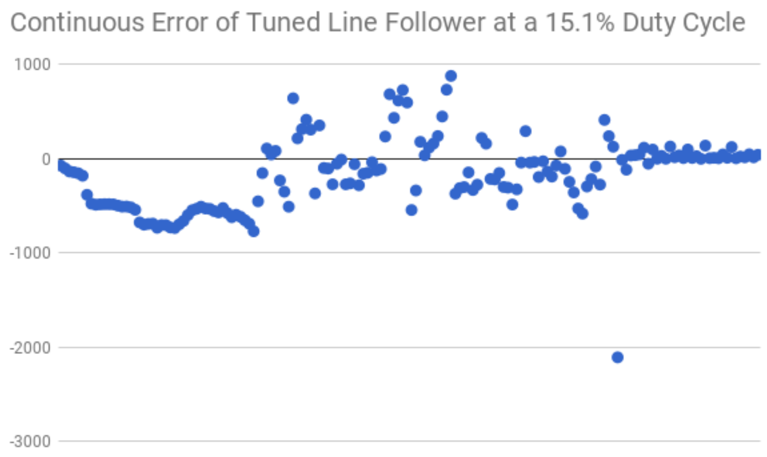
where T is the measured period of the rotation of the wheels. This allows the pacer to keep a real time measurement of the velocity, and adjust itself to keep a constant velocity.

One potential problem in this measurement is posed by the bounce time of the Hall Chips. Although the Hall Chip is positioned at a certain point so as to only detect the wheel magnet at a certain position on each rotation, there is a danger that the processing speed of the Raspberry Pi will be fast enough to record two or more signals per one rotation. (Thus the Raspberry Pi receives and processes two signals from the Hall Chip before the wheel completes another rotation) The bounce time is a delay period after a measurement has been made before a following measurement can be made. Specifically, the bounce time is set for half of the desired period. That is to say that for a given velocity, the corresponding period of rotation for the pacer moving at set speed is calculated, and the bounce time is set for half of this time. This prevents the Raspberry Pi from miscounting the number of rotations and allows for an accurate measurement of the real time velocity.

5 Results

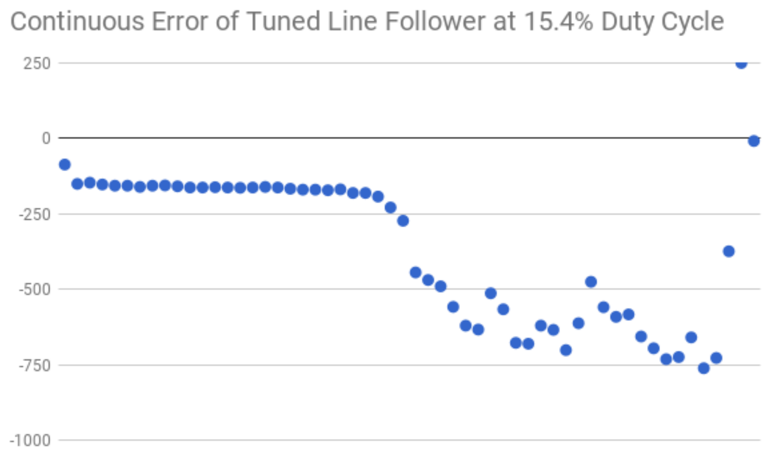
The pacer was tuned to accurately follow a test curve at low speeds. The final tuned constants had a value of $P = 0.000001$ and $D = 0.00000015$. Note that steady state errors were not found during testing and so the integral constant was not tuned.

Figure 5: Continuous Error of the Tuned Pacer running at a Duty Cycle of 15.1%.



The data in Figure 5 corresponds to a test run over a track consisting of a long straight, a constant curve, followed by a second long straight. The pacer is run with a 15.1% duty cycle output to the motor. Recall that the motor's duty cycle ranged from 15% to 20%, so this test was performed at one of the slower speeds. Notice that on the straights the error was constant, meaning the pacer followed the line steadily. On the curve the error fluctuates and changes sign, showing that the pacer oscillated between both sides of the line. However, the pacer was always able to keep the line within view, and return to an equilibrium position. It is critical that this error does not exceed the range of view of the main camera, because without a view of the line the pacer would be unable to continue navigation. This was found to be a problem at faster speeds.

Figure 6: Continuous Error of the Tuned Pacer running at a Duty Cycle of 15.4%.



This data corresponds to a test run over the same track as in the 15.1% duty cycle test; however, this test was run at a motor duty cycle of 15.4%. Because of the faster speed, small oscillations in the pacer could not be corrected fast enough, and the pacer continued to wander off course and eventually lost the line. This suggests that at different speeds the PID constant would have to be re-tuned to match the pacer's new calibrations.

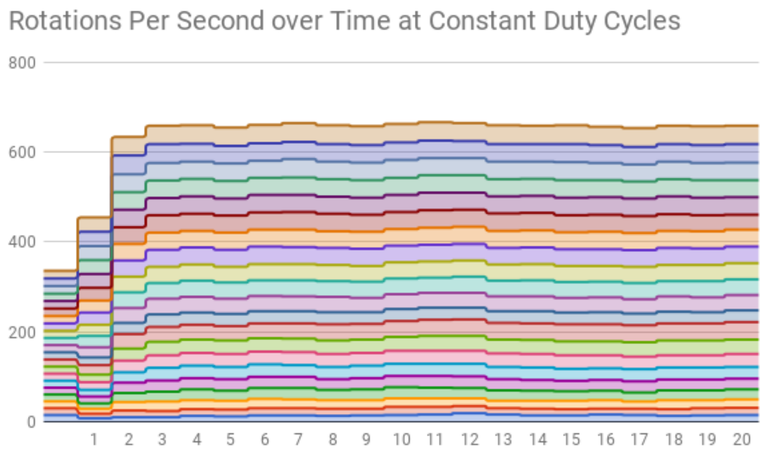
The raspberry pi was also found to be able to accurately control the speed of the pacer by varying the duty cycle output to the motor. Initially the duty cycle was predicted to correspond to the angular acceleration of the motor, however this was not found to be the case in applied tests.

The duty cycle is sent through the ESC, which integrates the varying duty cycle into a constant current of variable intensity. At a higher current, the motor outputs more torque, which should result in angular acceleration in accordance with the rotational form of Newton's Laws.

$$\tau = I\alpha$$

However, in practice, especially when in contact with the ground, this increase in torque is almost instantly counteracted by friction.

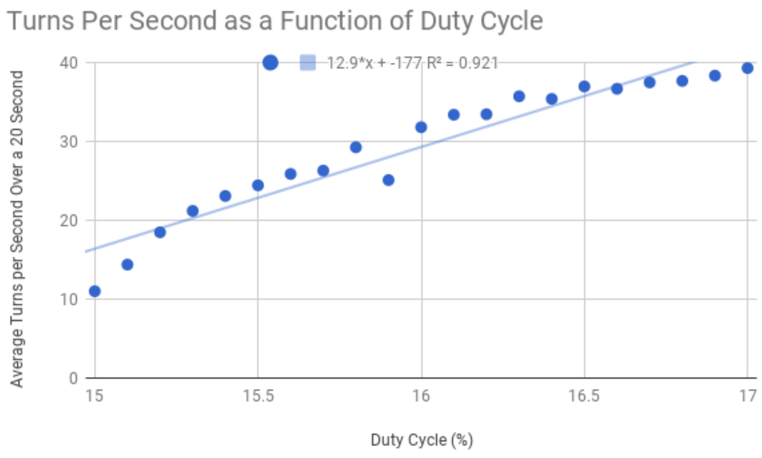
Figure 7: Rotations Per Second at a Constant Duty Cycle.



To test this the pacer was set to run at a constant duty cycle over twenty seconds. Over each second the number of wheel rotations was recorded. If the duty cycle had corresponded to the angular acceleration of the wheels, then the amount of turns each second would have increased, even with a constant duty cycle. However, this was not found to be the case. Figure 7 shows the number of turns each second for a twenty second period for the duty cycles 15.0% to 17.0%, stacked vertically in an area chart.

Notice that in the first three seconds the amount of turns per second increases for every duty cycle. However, after that the amount of turns per second remains relatively constant for the remaining seventeen seconds. If the duty cycle has corresponded to the angular acceleration of the wheels, this trend from the first three seconds should have continued across the entire test period.

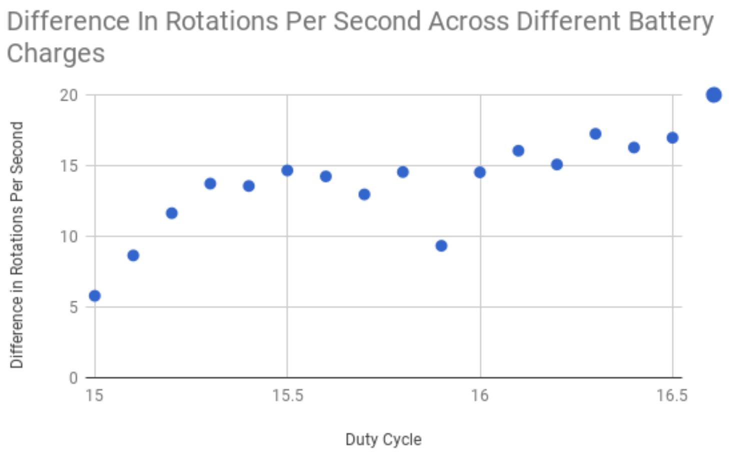
Figure 8: Turns Per Second as a Function of Duty Cycle



The angular velocity of the wheels of the pacer was plotted as a function of duty cycle. This relationship was approximated as linear with an $R^2 = 0.921$, meaning that 92.1% of the variation in the motor speed was explained by the linear model. This allowed the pacer to accurately control its speed as a function of duty cycle, by using the aforementioned regression and converting its angular speed to linear speed.

One final concern was the dependence of the pacer's speed on battery charge. As charge drained from the battery its potential difference lowered and the current it could produce decreased, thus leading to the car slowing down as it lost charge.

Figure 9: Difference in Rotations Per Second Measured At Two Different Charge Levels



Because there was no way to measure the charge in the battery while the pacer was running, the rotation rate was measured separately at different charge levels, one with high charge and one with low charge. Figure 8 shows the difference in the average number of rotations per second across the twenty second time period per duty cycle. Discounting the lower duty cycles, the loss of charge appeared to slow the rotation rate by fifteen turns each second. Thus by assuming that the loss of charge slows down the pacer independent of the duty cycle, the pacer can assess its charge level and actively correct for this. This is because only the constant coefficient of the previously derived linear regression need be adjusted.

$$\omega = 12.9 * d + C$$

Where ω is the angular velocity of the wheels, d is the duty cycle, and C is a charge constant. In the original charge level used to derive the equation, $C = 1.77$. However, at different charge levels, a different charge constant is used. Additionally, it is easy to calculate the charge constant by measuring the actual angular velocity and comparing it to the calculated value given by the equation.

It should also be noted that if the decrease in speed was not independent of the duty cycle, that is if both the slope and intercept of the regression depended on the charge, there would be no way for the pacer to calibrate itself for this loss in charge. It would involve solving a system with two unknowns, the affect of charge on the slope, and the affect of charge on the intercept.

In this way the pacer was able to follow a line and accurately control its speed: the intended goals of the project.

6 Conclusion

The pacer achieved its goal of accurately following a line and being able to modulate its speed. Additionally, it proved that image processing could be used to accurately navigate and control a line following algorithm.

However, the image processing was not fast enough to directly control the line following algorithm.

In the limited settings of a known track, such as the pacer would experience in a track and field environment, the pacer was able to predict where the line would lead by fitting a linear regression to the visible section of the line. This assumption, that the line would be continuous and not contain sharp curves, allowed the pacer to accurately follow such lines. However, this assumption would not be satisfied in a dynamic environment, such as in the process of a self driving car.

The pacer also suffers from a limited field of view. Once the pacer has lost sight of the line it cannot continue using the line following algorithm. The raspberry pi camera suffers two major drawbacks, specifically it has a narrow field of view, and it has a short connecting cable, meaning that the camera cannot be positioned to have a large field of view. Increasing the field of view would have two benefits. First it would make it harder for the pacer to lose the line. Additionally, if the pacer had a larger field of view it would not need to rely on a regression to predict the location of the line beyond its field of view. This could negate some of the problems previously mentioned that would limit the application of the line follower to more dynamic environments, although further testing is still needed.

Additionally, the project found a lot of success in using the raspberry pi to control the RC Car. The method of interpolating the necessary duty cycles with the oscilloscope was efficient and easy, and could be used to control and system operating on PWM signals.

7 Next Steps

Although the project achieved its main goals of line following and speed regulation, it has not reached a point where it can apply these to a track and field environment. It was shown in testing the the PID constants need further refinement, specifically for testing the pacer at higher speeds. Additionally, the pacer was tested on a track at low speeds, however, it was not always able to distinguish the line. Both the red of the track and the white of the line were both read as light colors, and the pacer often failed to distinguish them. This may have been because of the glare off of the track although further testing is needed. A potential solution could be to filter out the red color values from the initial RGB measurement of the image, so as not to look at the red of the track at all. This would allow the pacer to read the red of the track as dark, while seeing the white (composition of green and blue) of the lines as light. Additionally, fail safes would need to be put in place for use with subjects. However, the pacer shows a lot of promise, and a successful transfer onto the track is expected.

Acknowledgments

Special thanks are owed to Dr. Dann, Mr. Ward, and Mr. Allard for specific feedback and advice on the project. Additionally, more thanks are owed to Ms. Lax, Mr. Formato and Prof. D. Auroux for support in mathematics and physics methods. Constant feedback and motivation were provided by L. Tewari, R. Woodson, T. Yang, and A. Zheng.

Bibliography

References

[1] Agarwal, Tarun. ”Line Follower Robots - Controlling, Working Principle and Applications.” El-Pro-Cus. Accessed February 5, 2018. https://www.elprocus.com/.

[2] Anonymous post to Algorithms and Stuff web forum, ”Fastest Gaussian Blur (In Linear Time).” Accessed February 5, 2018. http://blog.ivank.net/.

[3] Coxworth, Ben. ”Puma’s BeatBot Races Runnners.” New Atlas, May 4, 2016. Accessed February 5, 2018. https://newatlas.com/.

[4] ”Fastest Line Follower Challenge.” Technoxian Championship. Accessed February 5, 2018. https://www.technoxian.com/.

[5] ”Hough Line Transform.” OpenCV. Accessed February 5, 2018. https://docs.opencv.org/.

[6] Pakdaman, Mehran, and M. Mehdi Sanaatiyan. ”Design and Implementation of a Line Follower Robot.” Computer and Electrical Engineering, 2009. http://ieeexplore.ieee.org/.

[7] Rosebrock, Adrian. “OpenCV Center of Contour.” PyImageSearch, 29 Jan. 2016, www.pyimagesearch.com/2016/02/01/opencv-center-of-contour/.

[8] Shead, Sam. ”Amazon Now has 45,000 Robots in its Warehouses.” Business Insider, January 3, 2017. Accessed February 5, 2018. http://www.businessinsider.com/.

[9] “Structural Analysis and Shape Descriptors¶.” Structural Analysis and Shape Descriptors - OpenCV 2.4.13.6 Documentation, Open CV, docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html.

[10] ”Traxxas Bandit XL-5.” Traxxas. Accessed February 5, 2018. https://traxxas.com/.

[11] Wescott, Tim. “PID Without a PHD.” Embedded, 1 Oct. 2000, www.embedded.com/design/prototyping-and-development/4211211/PID-without-a-PhD.