READING BETWEEN THE LINES: ROCHESTER INSTITUTE OF TECHNOLOGY'S IDE CUP

Chris Abajian, Rebecca Reich

Students at Rochester Institute of Technology

ABSTRACT

Youthful students are no longer the only ones taught to read between the lines. Intelligent robots were created as part of the IDE Cup with the sole goal of autonomously racing around a track as fast as possible. With knowledge of microcontroller development, external peripherals, and control system theory, a motorized self-driving miniature car was created. Its objective was to read and process a white racetrack with black edge lines using a single input line scan camera. The authors developed this car after several weeks of tuning and improving the control system to eventually compete against other teams and take home the gold medal. The intelligent car was a success and displayed the positive effects of teamwork and application of knowledge between two computer engineering students.

***Index Terms*—** Intelligent racing, autonomous car, control systems, IDE Cup

1. INTRODUCTION

An intelligent motorized racecar may seem like an unimportant, fun project to an average person. However, projects like this are the backbone of innovation for the future. Intelligent systems are a guaranteed future [1] in our society and any development in this field is progress. While seemingly nonessential, a self-driving racecar like the one developed by the authors possesses the essential basics to line following and autonomous movement that can be expanded to further applications.

In the IDE Cup, restrictions are imposed on the competitors to maintain a level playing field. The most important of these restrictions is the following clause: "No modification to wheels or chassis." [2] The authors learned from teaching assistants at the start of this endeavor that one of the largest factors at play is the friction between the track and the wheels. It was suggested that the dust accrued on the wheels after each run significantly reduced the force of friction and should be removed before the next run. One popular method for this was the use of tape to quickly remove the dust particles. Another important factor at play is the processing of the line scan camera data. The line scan camera is very sensitive to light and even a slight change in luminance around the track can lead to a catastrophic malfunction of a control system that does not account for this.

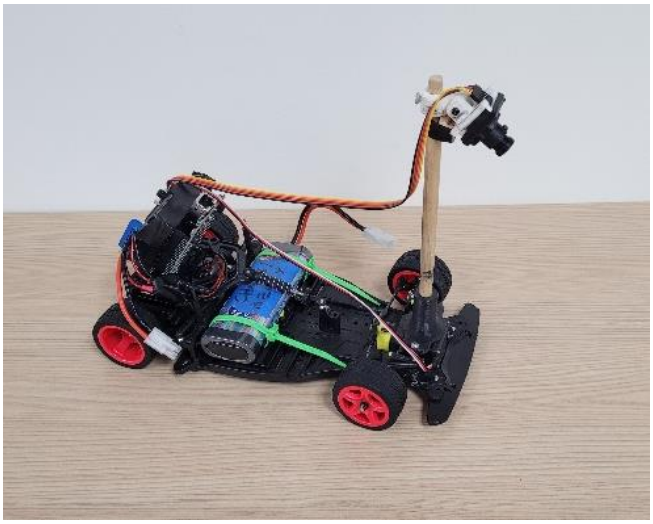

Fig. 1. Assembled IDE Cup car on race day.

With the knowledge of two key factors listed above, the authors proposed solutions to mitigate the influence of these factors on the car's consistency. First, before each day the car was tested, the wheels were thoroughly cleaned using water and an abrasive sponge to prevent the buildup of dust in the wheel treads. Between runs on the racetrack, tape was used to quickly remove loose dust and absolve friction as a limiting factor for consistency. A second tactic employed was to utilize dynamic thresholding when converting raw line scan camera values for detecting edges. Convolution techniques were proposed at the start of the project however several digital filters were required before an optimal dataset was obtained: three smoothing filters and one derivative filter. Each filter implementation required a loop with the number of iterations equal to at least three times the length of the camera data vector. With concern to the number of CPU cycles required to process the data, a dynamic thresholding procedure was implemented. The use of a threshold can convert the raw camera data to a binary vector representing the camera seeing a track or no track. This greatly simplified the number of iterations required to process the data and detect edges on the track.

After the line data is processed and the edges determined, the control algorithm steered the car based on the error

between the expected center of the car and the current estimated center. Speed was also adjusted based on this error as well as previous errors. Without the use of encoders to empirically measure the car's velocity, the current speed of the car was estimated using a summation of the previous positional values.

The rest of this paper is organized as follows. After the introduction, Section 2 describes the background of the control system developed as well as how onboard timers were used to control the car. Section 3 details a more technical look at how sensor data is processed and the control algorithm that interprets this data. Section 4 presents the final tuning and race day results, and Section 5 concludes the paper.

2. BACKGROUND

2.1. Analysis of Previous Strategies

Inspiration for the dynamic thresholding approach the authors implemented stemmed from experimental testing and implementations from previous IDE Cup contenders. To combat the light sensitivity of the line scan camera, high school students from the Czech Republic isolated the line scan camera and utilized an LED array to illuminate the track immediately in front of the car [3]. An LED array was considered in the final design of the car, however after testing the additional light showed no visible impact on the line scan camera's output.

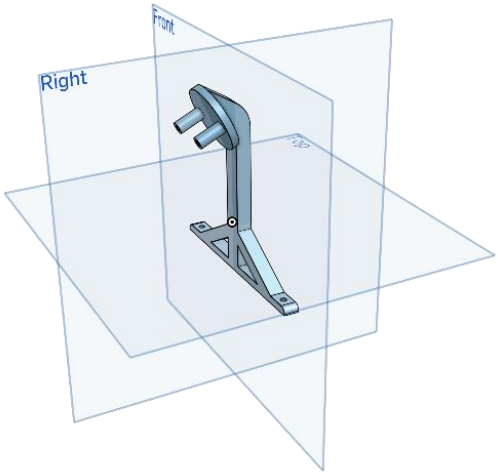

Fig. 2. 3D printed LED mount for the NeoPixel Jewel designed using OnShape.

One key area where the LED array would dramatically help was creating even lighting conditions across a single section of the track. When the left or right edge of the track was brighter than the other, the threshold used would filter out the track on the dim lit side. To combat this, two dynamic thresholds were calculated: one for the left side of the camera and another for the right.

A bonus challenge for the competition was to stop at the starting line after completing a lap. This trivial-seeming task was difficult to implement due to the width of the starting line pattern. At high speeds, the camera would not have enough time to generate a clear pattern. One team from Switzerland added a second line scan camera at a different angle to process more of the track [4]. A steeper camera angle would make the narrow starting line more visible without light bleed from the surrounding track.

2.2. Basic Strategy for Speed

Provided the simple task of navigating a track as quickly as possible, a basic strategy to drive fast down the straights and slow around corners was theorized. The estimated position of the car on the track was calculated every new camera data sample. The estimated position directly influenced the motor duty cycles and the position of the servo responsible for steering. When centered this error would be small, and the motor duty cycles would be greatest. Around corners the error would be large, and the motor duty cycles would be smallest.

2.2. Control System

A proportional-integral-derivative (PID) controller was used to control the servo and motor duty cycles. A PID controller uses current and previous error to continuously adjust the drive components. A generic PID formula is shown:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \qquad (1)$$

The proportional, integral, and derivative components in (1) determine the next control variable. The proportional term is a constant adjustment based on the current error. The integral term is an adjustment based on the accumulated error. This helps the system respond more smoothly and less sporadically. Finally, the derivative term accounts for large error changes between the previous and current error. For the purposes of the car's control system, two PID systems were implemented: one for steering the servo, and one for adjusting the motor speeds.

2.1. Using Onboard Timers

The FRDM K64F microcontroller used features several onboard timers. Two timers were incorporated into the design of the intelligent-driving car: the FlexTimer Module (FTM) and the Periodic Interrupt Timer (PIT). The PIT was used to generate an interrupt used to start reading values from the camera via the onboard analog-to-digital converter (ADC). The PIT simply counts down from a specified starting value and generates an interrupt when the timer reaches zero. This interrupt is used to enable interrupts on an FTM which reads pixel values from the ADC and indicates a new clock period for the GPIO acting as a clock for the camera.

One benefit of the FTM is the fact that it is multichannel meaning multiple timers can be configured. Five more FTM's are configured to generate PWM signals for the servo and both motor duty cycles (in both directions).

3. PROPOSED METHOD

3.1. Assembling the Car

The assembly of the car was simply connecting the motor shield to the microcontroller and the motors, servo, and camera to the shield. 3D printed mounts were provided to affix the camera to the chassis.

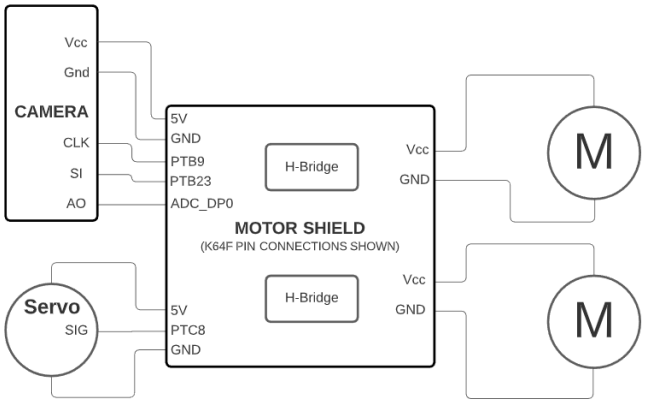

Fig. 3. Wiring schematic showing the connections between the camera, servo, both DC motors, and the motor shield with K64F pin connections shown.

An estimated Bill of Materials (BOM) is shown in Table I. The provided parts were referenced from the 2021 NXP Cup cars, however these components show similar capabilities to those used for the 2020 IDE Cup.

TABLE I BILL OF MATERIALS FOR THE 2021 NXP CUP CARS

Part	Description	Price	Qty
Line Scan Camera	CMOS linear sensor array of 128 pixels & 1 adjustable lens, w/ TAOS 1401	$77.37	1
Servo Motor	Futaba-S-3010	$28.99	1
DC Motor	7.2V, 220mA, Stall Torque 80g.cm min	$26.19	2
Motor Driver (H-Bridge)	FRDM-KL25Z shield	$102.65	1
Battery	7.2V NiCd, ≤3000mAh	$26.17	1
Car Chassis	2021 DFRobot Car Chassis	$103.55	1
Microcontroller	Freedom K64F	$41.18	1
Total	$406.10		

3.2. Obtaining Camera Data

The line scan camera was positioned and angled to optimize both the clarity of pixel data and the range of pixel data. At a shallower, straight-looking angle, the camera can read line values farther away from the car and provide time for the control system to accelerate and, more importantly, decelerate. However, such an angle would decrease the image clarity and make it difficult to process edges, particularly the starting line pattern. The angle was adjusted to account for both considerations and connected to an oscilloscope to properly focus the lens. An unfocused camera lens resulted in less distinction between light and dark parts of the track.

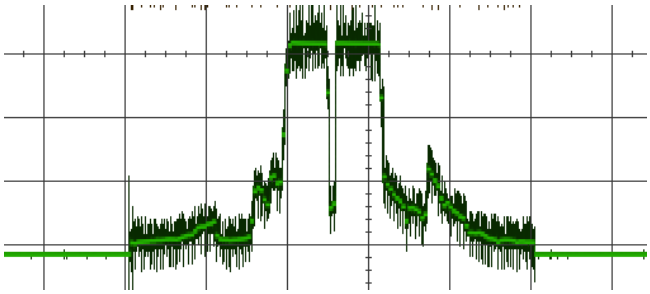

Fig. 4. Oscilloscope trace of a well-focused camera with clipping due to too high of an integration period. Camera is pointed at a thin white piece of paper with a single black line in the center.

The line scan camera was observed to process valid data between minimum and maximum indices. This was tested with the camera pointed directly at a white plane spanning several feet in the horizontal axis. The camera output was observed to produce low ADC values in the first and last 20 pixels. Low ADC values within the valid range indicated a low-light level image whereas a high ADC value indicated a white image, or the track. The camera integration period could be increased to further charge the capacitors and produce a more accurate image at the expense of sampling time. Ultimately, a 2.5 millisecond integration period was used to obtain 400 camera samples per second. More camera samples resulted in a faster response around corners.

3.3. Processing Camera Data

The raw camera data was processed to obtain the number and location of edges in the line. This was accomplished using a dynamic threshold calculated at each camera data cycle.

With the binary vector as shown in Fig. 5 generated, both rising and falling edges were detected and the combined edge count was summed. Before entering the centering algorithm, the estimated position on the track was calculated using the number of edges and location of these edges. If no edges were observed within the valid window, a full black image was observed. This state was occasionally entered around a sharp turn where the car has yet to straighten out.

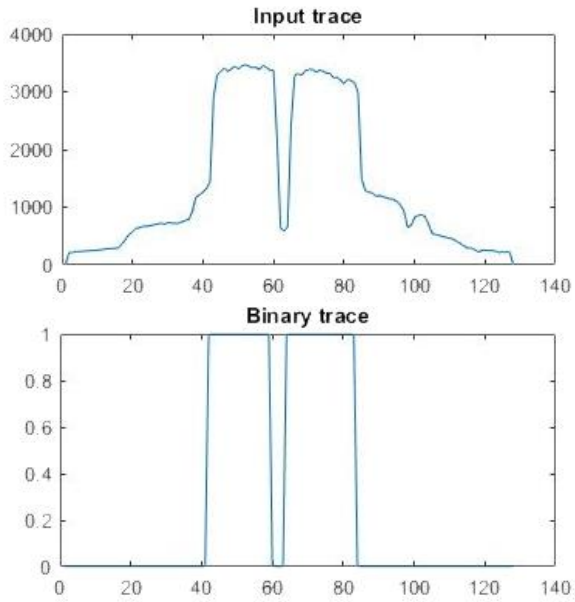

Fig. 5. Matlab plot of raw (top) digitally converted camera values and the resulting binary trace (bottom) after thresholding. Camera is pointed at a thin white piece of paper with a single black line in the center.

To prevent catastrophic failure, the previous position is used to hopefully realign the car.

If two edges were observed, the position was estimated as the center point between the two indices. If four edges are observed, the car probably entered an intersection at a crooked angle. The widest track width between a rising and falling edge was assumed correct and used to calculate the center. Finally, if more than four edges are seen, the camera has either malfunctioned or the starting line was detected. In the event of the latter case, the car stopped if this feature was enabled.

```
// calculate error
error = CAM_CENTER - (max_idx+min_idx)/2;

// estimate position on track
if (min_idx == -1 && max_idx == -1) {
  // START DETECTION
  position = POS_START;
} else if (min_idx == 0 && max_idx == 0) {
  // looking at black, use last position
  position = last_pos;
} else if (min_idx <= CAM_LOW+10 && max_idx >= CAM_HIGH-10)
  position = POS_INTERSECTION;
} else if (abs(error) <= abs_error_thresh) {
  position = POS_CENTER;
} else {
  position = POS_CORNER;
}
```

Fig. 6. Algorithm for calculating the car's current position on the track. If the current position could not be determined, the last position was assumed.

3.4. Centering Algorithm

With the estimated position of the car on the track, the car is centered by the controller using a simplified PID equation. In the case of steering, a simple proportional term is used to calculate the duty cycle of the front servo required. The integral and derivative components were not implemented due to the time required to fine tune these parameters versus the miniscule payoff. A positive positional error indicates the car is located on the right side of the track and the servo duty cycle is decreased to turn left. A negative positional error steers the car to the right.

At higher positional errors, the car turns more aggressively using differential drive. On right turns, the left motor has a higher duty cycle than the right and vice-versa on left turns. This enabled the car to enter turns faster without going off track.

3.5. Variable Speed Control

The motor control equation included both proportional and integral components. Initially, only proportional control was used however at higher speeds, the motor duty cycles were not small enough to sufficiently brake the car around a turn. To compensate for this, the integral term was added using a summation of the current estimated speed of the car. This speed counter is incremented when the positional error is small and predicted to be in a straightaway, and decremented when the error is large and predicted to be in a corner. This system can get out of control if, for example, the car spends more time on straightaways than corners. To prevent this, the speed accumulator was clipped at minimum and maximum values.

An added benefit of this accumulator was recouping lost speed around turns. The speed accumulator would go negative if the car were in a turn for a long period of time, which boosted the motor duty cycles. This was particularly useful when exiting corners and, more importantly, in scenarios where sustained cornering was prevalent. If the car was in a windy part of the track, the overall speed was not hindered as much due to these boosts. A neat side effect of this was rapid drifting around entire loops when the wheels were dusty.

4. RESULTS

4.1. Tuning for the Track

The final track layout was unknown until race day. Upon seeing the track, teams were granted five minutes to tune the parameters for their control algorithms. The authors developed several modes for the car depending on the desired speed. The medium speed mode was found to work well on the track when tuning, however the fast mode encountered a false positive on a gap between two tracks thinking it was a starting line. After practice was finished, the fast mode was modified to reduce the dynamic threshold and potentially sacrifice the start detection in favor of a clean run. The final equation used for servo duty cycle control is shown:

$$duty_{servo} = 7.5 - 0.1 * error \quad (2)$$

Unlike (2), the motor duty cycles combined proportional control as well as integral control. The duty cycle equation for the left motor subtracted the proportional term:

$$duty_{m,left} = 40 - 0.6 * error - 0.2 * sum \quad (3)$$

The right motor equation added the proportional term:

$$duty_{m,right} = 40 + 0.6 * error - 0.2 * sum \quad (4)$$

With a positive error, the car is estimated to be on the right side of the track. Thus, (3) would yield a lower duty cycle than (4) and differential drive between the motors would assist the servo in turning left. The reverse would occur with a negative error. In either case, the integral sum was subtracted. This sum increased with the car centered and decreased with around corners with a minimum and maximum value of -100 and 600, respectively. A sum value of -100 would increase the motor duty cycles by 20% around prolonged turns.

4.2. Final Race Results

The author's car was the first to race and set the precedent for the rest of the cars. The fast mode proved to be stable and made it around the track on the first valid attempt. In hindsight, it may have been beneficial to further increase the maximum speed, but there was a risk of braking too hard around turns at greater speeds.

The author's team finished in first place with a time of 21.11 seconds. This was a narrow victory of only four milliseconds from second place. Several other cars showed promising steering and control algorithms but went off track on the first attempt and thus the speed was decreased for subsequent attempts. This appeared to be a downside of the three-mode approach and perhaps instead of having a mode for fast, medium, and slow, the three modes should be faster, fast, and medium, where the first is almost expected to fail. Many teams expressed concern about finishing at all and as such more conservative speeds were selected.

TABLE II TOP 7 FINAL TIMES ON RACE DAY

Team	Run 1	Run 2	Run 3	Final
13	21.11	-	-	21.11
18	X	21.15	-	21.15
17	X	21.36	-	21.36
14	22.54	-	-	22.54
1	X	24.02*	-	23.02
9	23.34	-	-	23.34
2	24.64	-	-	24.64

4.3. Future Improvements

The number one failure on race day for the team was the failure to stop at the starting line after completing the lap. This was determined to be the result of a high speed and decreasing the threshold to prevent false positives. In future iterations of the car, and LED array would have been beneficial to guarantee even lighting conditions. One team also pondered the idea of using two OPB745 photo transducers to detect the starting lines and even went as far as designing and printing PCB's for it. Ultimately, the team did not use this functionality.

5. CONCLUSION

The intelligent self-driving car came to fruition and successfully completed a lap around the final racetrack faster than any other car that day. The project was a success and pushed the author's engineering capabilities farther than previous projects. Knowledge and adaptation of control systems was used alongside microcontroller development skills to interface external sensors and peripherals together. Working as a team helped keep the authors grounded and aligned with the task when a train of thought went too far off scope. Future adaptations of the IDE Car could introduce a second camera sensor or even a 2D camera for more data processing capabilities.

6. REFERENCES

[1] López de Mántaras Artificial Intelligence Research Institute (IIIA), Ramón. "The Future of AI: Toward Truly Intelligent Artificial Intelligences." *OpenMind BBVA*, 27 Feb. 2019, www.bbvaopenmind.com/en/articles/the-future-of-ai-toward-truly-intelligent-artificial-intelligences/.

[2] L. Beato, "Lecture 19a NXP IEEE Paper," on *RIT MyCourses*, Online, slide 11, 2020.

[3] T. Havran, J. Charvát, M. Šturala, "Technical Report Valasi", https://community.nxp.com/t5/The-NXP-Cup-Technical-Reports/Technical-Report-Valasi-pdf/ta-p/1123279 (online), pp. 6, 2015.

[4] C. Kaufmann, M. Rauscher, V. Frangi, "EMEA NXP Cup 2017 ARCar - Switzerland", https://community.nxp.com/t5/The-NXP-Cup-Technical-Reports/EMEA-NXP-Cup-2017-ARCar-Switzerland-pdf/ta-p/1110396 (online), pp. 3, 2017.