Antlion: A Self-Driving Vehicle
(May 2017)

Alexander Howard and Abhimanyu Gupta

***Abstract* — The Antlion is the fifth place winner of the NXP cup at ImagineRIT. Designed for the purpose of following an unknown track, the Antlion follows a white track whose path is defined by two black lines around the edges. The Antlion was designed to detect these black lines using a line-scale camera and follow the track at the highest possible speed. This paper describes the methodology for the detection of the track's edges, processing of this information and the servo/motor control used to drive the Antlion around the track.**

***Index Terms* — Automation, Signal Processing, NXP, Automated Vehicle, Model Car**

I.INTRODUCTION

In an age of technology, where self-driving cars are already a reality, safe methods of driving an automated vehicle within a predefined area are of the utmost importance. While the Antlion is a model car, and it performs on a much smaller and simpler scale than most self-driving vehicles in development, the methods it uses can translate to a larger, more complex vehicle

The Antlion competed in the NXP cup at ImagineRIT, a competition where similar model automated cars race around a track, whose layout was unknown prior to ImagineRIT. The path the track takes was defined by two black lines that marked the outer edge of the track. The model cars involved in the competition were designed to detect these lines and stay within them for the entirety of the track. The cars were also designed to traverse

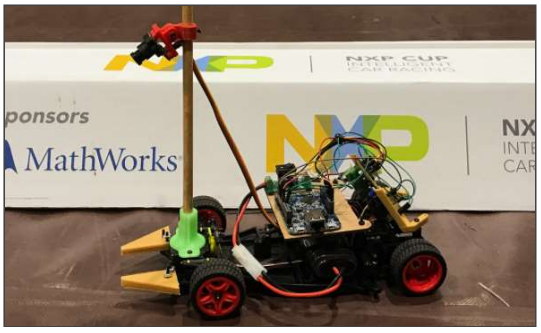

Fig. 1. The Antlion as photographed on the day of the NXP Cup Race at ImagineRIT

the track as fast as they possibly could while staying on the track. A car that didn't keep at least two wheels on the track at all times, or failed to follow the track as intended, was considered disqualified. Since all cars were designed under the same guidelines, some similarities between methodologies can be expected.

In addition, all cars were provided with similar equipment; a basic chassis; two DC motors, one for each of the back wheels; one servo motor, which controls the turning angle of the front wheels; a Line-Scan Camera, which reads the brightness of a line in front of the car, returning a 128 pixel array; and a 7.2 volt battery. Each car was controlled by a K64F microcontroller board, which interfaced with an FRDM TFC board to control the Motors and read the camera. Cars could then be outfitted with 3D printed parts, the designs of which were provided. Additional parts could be added as desired. By regulating equipment, the competition is determined by method, rather than the quality of the equipment used.

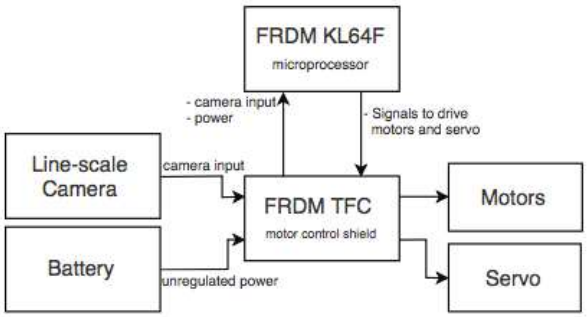

Fig. 2. Diagram of Interfaces between Microcontroller, Motor Control Shield, and on-board equipment

Each car would require a method that allows them to understand the track and follow it at the fastest speed possible. Each car would use the basic methodology of reading from the Line-Scan Camera and using the resulting one-dimensional signal to control the speed of the DC motors and the position of the servo.

The Antlion's method of following the track involved used the signal from the Line-Scan Camera, which was filtered twice; once to smooth the signal, then again to differentiate it. Using this differentiated signal, the car's offset from the middle of the track was calculated. Based on this offset, the Antlion changes the position of a servo connected to its front wheels and the speed of the DC motors connected to each of its back wheels.

This paper will go into greater detail on the methodology used by the Antlion. Background on some of the methods used are provided in Section II, more details on methods is provided in Section III, and the results of the methods are provided in Section IV. These methods could later be expanded upon to create a larger, more complex automated vehicle.

II. BACKGROUND

The Antlion made use of various signal processing and motor control methods. Antlion used multiple methods of filtering on the single Line-Scan Camera to improve readability. Chung et al. [2] used a second camera to get a more reliable set of data while Cheng et al. [1] use a 2-D camera along with a MPU 9250 gyroscope to get better input data. The FlexTimer Module on the K64F microcontroller was used to control the motors and servo, while the PID control methods were used to smooth control of the motors and servo. These methods are outlined and defined here.

Table 1. Connection Mapping between the K64F microcontroller, and the FRDM TCF motor shield. Also shown are the components controlled or read by the connections

K64 Pin	Signal	Motor control board - Pin	Description
P5 - 9V_VIN	V+ Battery	J9-16	POWER
GND - 14	GND	J9-14	POWER
P3V3	3.3V	J9-8	POWER
P3v3	3.3V	J9-4	POWER
	Motor		
Port A - 2	A-IN1	J1-5	Motor control signal
Port A - 1	A-IN2	J1-7	Motor control signal
P3V3	Enable	J10-3	Motor enable
Port C - 3	B-IN1	J10-12	Motor control signal
Port C - 4	B-IN2	J10-10	Motor control signal
	Servo		
Port C - 8	servo-pwm	J10-2	PWM signal for servo
	Camera 1		
Port B - 23	SI	J2-19	Camera SI pulse
Port B - 9	CLK	J2-20	Camera clock
ADC0 DP0	AO	J2-4	Camera analog output

A. FIR

The signal from the Line-Scan Camera starts out too noisy to effectively use. As such, the Antlion uses FIR, or Finite Impulse Response filters. These filters are used over a set of discrete data to modify it in some way. The Antlion used two sets of filters; a smoothing filter and a differential filter.

The smoothing filter takes a noisy signal and flattens it, removing noise. A smoothing filter sets the value of a point based on the average of several other points in the signal. Usually, these other points are localized to the

modified point, but the averaging window can be anywhere in the signal.

The differential filter differentiates the signal at the desired point, giving the slope of the signal at that point. Like the smoothing filter, this filter is localized to the desired point of differentiation. The filter takes a sum of a number of points to one side of the differentiated point and subtracts the sum of an equivalent number of points to the other side of the differentiated point.

B. The FlexTimer Module

The motors and servo of the car are controlled by a square wave function. The speed of the DC motors is controlled by a duty cycle anywhere from 0% to 100%. At 0% duty cycle, the motors don't move, while at 100% they move at max speed. The servo's position is controlled by a duty cycle from 5% to 10%, with a 7.5% duty cycle corresponding to a centered servo.

In order to generate these waveforms, the K64F uses several on-board FlexTimer Modules, or FTMs. The modules keep a counter which continuously counts up at either the system clock, or a scaled down speed of the system clock. Once the counters reach their maximum, they reset. The FTMs can be set to toggle an output when this occurs. Additional fields, called moduli or MOD values, can be set on the FTMs. When the counters reach these values, the output is toggled as well.

The output of one of the FTMs was connected to the servo of the car, while four outputs were connected to the DC motors, two to each motor, one on the positive terminal and one on the negative terminal. By setting one terminal to a 0% duty cycle, the other terminal can be used to control the speed. Swapping which terminal is set to 0% and which controls the speed changes the direction the motor spins.

C. PID

The Antlion makes use of PID, or Proportional-Integral-Differential, control to smoothly and efficiently transition the position of the car's servo and the speed of the DC motors. Krase et al. [3] use the P(proportional) and I(Integral) part of the PID control, but upon testing it is found that by including differential, the desired voltage can be reached faster.

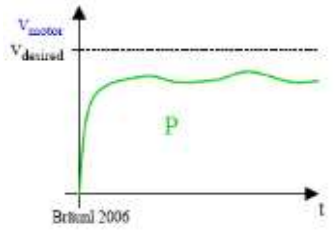

Fig. 3. Voltage output with proportional control

$$e[n] = V_{desired} - V_{current} \quad (1)$$

$$V[n] = V[n-1] + k_p(e[n] - e[n-1]) \quad (2)$$

Equations (1) and (2) show how to apply proportional motor control. By calculating the error between the desired voltage and the current voltage, then adding a proportion, k_p, of that to the current voltage, a smooth increase up to the desired voltage, although the maximum actual voltage is usually lower than the desired voltage.

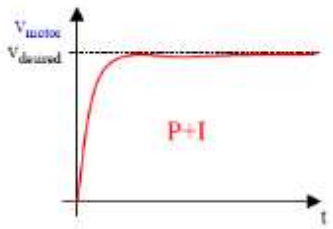

Fig. 4. Voltage output of combined Proportional and Integral control.

$$V[n] = V[n-1] + k_p(e[n] - e[n-1]) + k_i(e[n]) \quad (4)$$

Adding Integral control as in (4) brings the output voltage closer to the desired voltage.

It also serves to smooth the output. k_i in (4) above proportionally controls the effect of the Integral control on the output

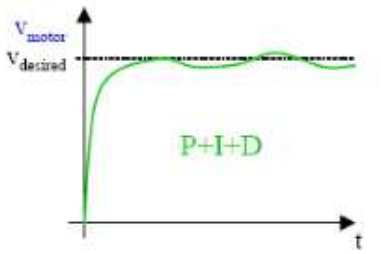

Fig. 5. Voltage output of combined Proportional, Integral, and Derivative control

$$V[n] = V[n-1] + k_p(e[n] - e[n-1]) + k_i(e[n]) + k_d(e[n] - e[n-1]) \quad (5)$$

Proportional and Integral control alone is slow to reach the desired voltage. By adding Derivative control, the output can be brought to the desired voltage much faster. The Derivative control also has a proportional variable, k_d.

III. PROPOSED METHOD

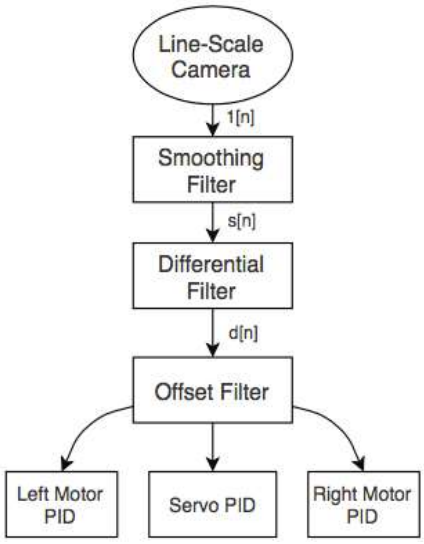

Fig. 6. Block Diagram of Line Scan Signal Processing from Line-Scan Camera to PID

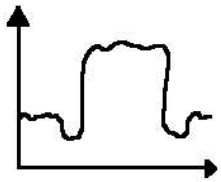

Fig. 7. Drawn diagram of the output of the Line-Scan Camera over a straight portion of track

One of the key features of automated driving is the ability to identify the edges of the driving area. The Antlion, as well as the other cars in the competition, identify the track through the use of a signal, l[n], retrieved by the Line-Scan Camera.

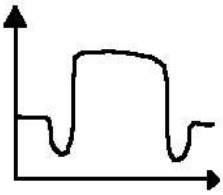

Fig. 8. Drawn diagram of smoothed Line-Scan Camera output

$$s[n] = 0.1 * l[n-2] + 0.225 * l[n-1] + 0.35 * l[n] + 0.225 * l[n+1] + 0.1 * l[n+2] \quad (6)$$

The output of the Line-Scan Camera is particularly noisy at first. Equation (6) above shows the smoothing filter used by the Antlion. This is a special form of smoothing filter known as a weighted average. This weights different points more or less; in this case, the center point is weighted heaviest, while the other points are weighted proportionally to their distance from the center. This improves the smoothing of the filter, favoring the center point over the other points, creating a clearer picture for the system.

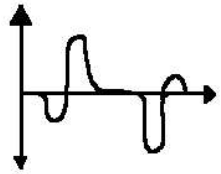

Fig. 9. Drawn Diagram of differentiated, smoothed, Line-Scan Camera output

$$d[n] = s[n-1] - s[n+1] \quad (7)$$

Equation (7) above shows a weighted differential filter which takes the output of the weighted smoothing filter. Like the weighted smoothing filter, the weighted differential filter favors points closer to the center, creating a more accurate differential output.

$$offset = (64 - \frac{(p_{max} - p_{min})}{2}) \div 32 \quad (8)$$

Taking this differentiated output, the maximum and the minimum points are found, then the midpoint of these two points are found as well. Since the Line-Scan Camera returns a 128 pixel array, the midpoint of the Camera is taken to be pixel 64. The difference between the midpoint of the camera and the calculated midpoint of the track is then normalized to a value between -1 and 1. For a value that takes into consideration the entire range of the camera, normalization would divide the difference between midpoints by 64. However, a value of 32 creates more dramatic results in the offset, and serves to keep the Antlion more effectively centered on the track.

$$Duty_{Servo} = 7.5 + 2.5 * offset \quad (9)$$

Using this offset value, the servo position and motor speeds can be proportionally changed to the Antlion's position on the track. The Servo's position was set in a linear fashion based on the offset, as shown in (9) above. The further from the center of the track the Antlion is, the more the front wheels turn. Since the offset can be either positive or negative, (9) naturally handles both left and right turns.

$$topspeed = maxspeed * (1 - offset^2) \quad (10)$$

$$wheeloff = offset * (maxspeed - topspeed) \quad (11)$$

Using the front wheels to turn the Antlion is not enough, however. Even at maximum turning angle, the Antlion would not make a turn sharply enough to make it around the track. Since each of the back wheels have their own DC motors, they can be controlled independently of each other. By slowing down one wheel and speeding up the other, turning can be improved. Since the car should also slow down around turns, (10) and (11) are used to manipulate the top speed of each wheel, and the wheel offset used for turning. When turning left, the wheeloff variable is subtracted from the left wheel and added to the right wheel, while the reverse happens when turning right. On a straightaway, the Antlion goes at maximum speed.

Just slowing down on the turns is not enough, to make sure Antlion can make the turn. It reversed the flow of current in the motors with the intention of braking right before entering a turn. This is determined by the value of the *offset*. When offset is between 0.1 and 0.2, the current flows in reverse direction for an instant, resulting in sudden drop in velocity.

Liu et al. [4] also made use of a current feedback to determine if their car was entering or exiting a ramp. The Antlion did not make use of Motor current feedback.

Both the calculated servo position and the wheel speeds are taken as the desired output of the Antlion. These values are used in the PID described in Section 2. Every time the FTM counter ticks over, an interrupt is generated. Inside of this interrupt, the PID calculations are performed and the duty cycles for the servo and two motors are set.

IV. Results

The Antlion competed in the NXP Cup at Imagine RIT on May 5. The Antlion competed against nineteen other vehicles that

were provided with the same equipment and competed under the same guidelines.

Of the twenty competing vehicles, seven finished, and of those seven, the Antlion came in fifth, with at time of about nineteen seconds.

Prior to the competition, the weights of the smoothing filter the the proportions of the PID control function were calibrated manually. This was done by configuring them and running the Antlion on a short portion of track to see how it performed. The values of $k_p = 0.55$, $k_i = 0.1$, and $k_d = 0.25$ were found to work effectively for the Antlion.

The final track was composed of several turns, an intersection, an S curve, and a hill. The Antlion stayed centered on the track for all of the straightaways. It also handled the intersection, the S curve, and the hill with ease.

On some of the turns, however, the Antlion faced difficulty, particularly after long stretches of straight track. This is because, on long stretches of straight track, the Antlion picks up speed, and by the time it reaches a turn, it's moving too fast to stay on the track. On one occasion, it nearly went off the track.

The turning difficulty also served to slow the Antlion down. The Antlion drifted more than turned on turns, and it lost out on the top speed it could have driven at, since going too fast would have sent it off the track.

V. Conclusion

While the Antlion completed the provided track and finished fifth of seven finishers, it's methodology could be improved upon.

Since a large part of the Antlion's trouble turning came from speed, braking could have been done intelligently, braking for longer the longer the Antlion had been going on a straightaway. This would improve turning performance and allow the Antlion to travel on straightaways at higher speeds for longer times.

More equipment could be added to the Antlion as well. In particular, second Line-Scan Camera could be added to provide a second signal to filter and combine with the first Camera. The Camera could also be used to search for secondary characteristics of the track.

Current feedback from the motors could be read by the K64F microcontroller through the FRDM TCF motor shield. Using this feedback, the actual speed of the motors could be understood, improving on the PID and detecting the when the Antlion slowed down, such as going over the hill.

References

[1] Cheng, Gang, Wang, Haonan, and Fan, Bin. *USTB SMARTCAR Team Technical Report*. Rep. community.nxp.com - NXP, 8 Oct. 2015. Web. 10 May 2017.

[2] Chung, Andrew Yunseok, Samuel Dawson, and Justin Laguardia. Tem_Dinky.pdf *NATCAR Final Report*. Rep. community.nxp.com - NXP, 29 Mar. 2015. Web. 10 May 2017.

[3] Krase, Ian Lee, Daniel, and Gan, Nicholas. team9.pdf *EECS 192 Progress Report*. Rep. community.nxp.com - NXP, 29 Jun. 2015. Web. 11 May 2017.

[4] Liu, Weitang, Mason Lee, Brian Jae Kim, and Weijie Zhang. Team_The_One.pdf *NATCAR Final Report*. Rep. community.nxp.com - NXP, 4 Jun. 2015. Web. 12 May 2017.