Rapid Autonomous Car Control based on Spatial and Temporal Visual Cues

1st Surya Dantuluri
Monta Vista High School
Cupertino, United States of America
dsuryav@gmail.com

***Abstract*—We present a novel approach to modern car control utilizing a combination of Deep Convolutional Neural Networks and Long Short-Term Memory Systems: Both of which are a subsection of Hierarchical Representations Learning, more commonly known as Deep Learning. Using Deep Convolutional Neural Networks and Long Short-Term Memory Systems (DCNN/LSTM), we propose an end-to-end approach to accurately predict steering angles and throttle values. We use this algorithm on our latest robot, El Toro Grande 1 (ETG) which is equipped with a variety of sensors in order to localize itself in its environment. Using previous training data and the data that it collects during circuit and drag races, it predicts throttle and steering angles in order to stay on path and avoid colliding into other robots. This allows ETG to theoretically race on any track with sufficient training data.**

***Index Terms*—Recurrent Neural Networks, Convolutional Neural Networks, Long Short-Term Memory**

I. Introduction

Monta Vista High School's El Toro Grande 1 is a proof of concept, answering the question on whether Machine Learning methods can be applied to autonomous driving to rival traditional computationally expensive computer vision, path planning, and localization algorithms. This report explains these methods as well as the hardware innovations necessary to execute such new software methods. This report is an example of how the International Autonomous Robot Racing Challenge (IARRC) promotes advancement in research of autonomous vehicles at the secondary and university school levels.

Vehicle Name: Monta Vista High School's Robotics Team has a tradition of naming their robots as *El Toro* (English translation: The Bull). Most of *El Toro* robots are used for For Inspiration and Recognition of Science and Technology (FIRST) robotics competitions, held primarily in the USA. Every *El Toro* robot is hardcoded to perform tasks during an autonomous period during FIRST robotics competitions and are manually controlled for the rest of the match. Our robot for the IARRC competition is relatively nimble and a lot more autonomous when compared to other *El Toro* robots. As a result of the tradition and how advanced this robot is in terms of hardware and software, the name *El Toro Grande 1* (English translation: The Big Bull) is a fitting for this robot.

II. Platform Design

A. Platform Overview

ETG is made up of 3 networks: Sensor Network, Vision Network, and the Engine Network. Among all these networks, the Quad Core 1.2GHz Raspberry Pi 3 conducts computational flows among all three concurrently. The upper structure of ETG is 3D printed from a CAD model.

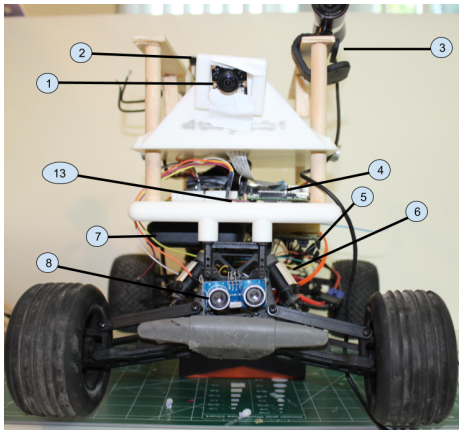

Fig. 1. Front view of robot components. Cross-referenced in Tables 1,2, and 3

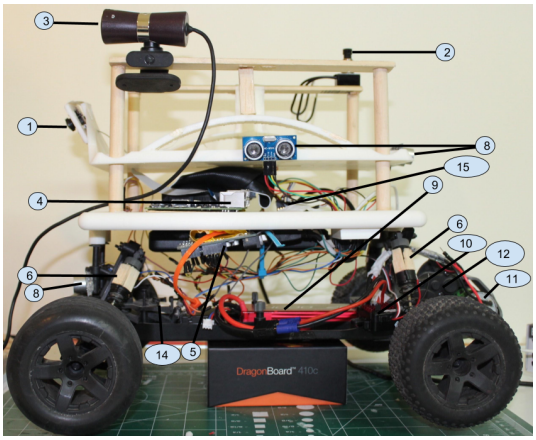

Fig. 2. Side view of robot components. Cross-referenced in Tables 1,2, and 3

B. Sensor Network

There are five sensors mounted around ETG. Four of these sensors are ultrasonic sensors placed on all four sides of ETG. All the ultrasonic sensors are used digitally and use a threshold to indicate whether the ultrasonic sensor will record data or not. This means that the sensors indicate something is present

50cm away with recording ”1” in the data. If something is more than 50cm away from any of the ultrasonic sensors, the ultrasonic sensors will not record a ”1” in the data, rather, a ”0”. The threshold has been put in place as a consequence of the noisiness of the HC-SR04 Ultrasonic Sensors. A SparkFun MPU-9250 IMU Breakout board is used to collect acceleration data as indicated in figure 3

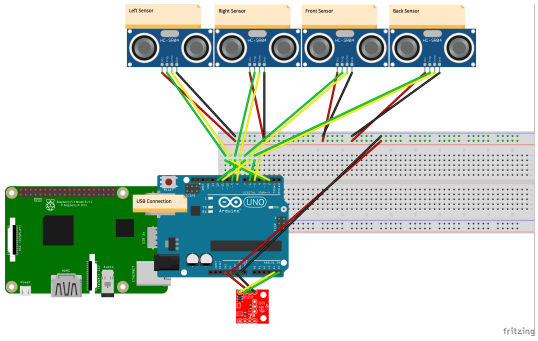

Fig. 3. Sensor Network

TABLE I
SENSOR NETWORK

#	Sensor	Type	Function
8	Ultrasonic Sensor - HC-SR04	Distance	Localization
13	MPU-9250 IMU	Positioning	Localization

C. Vision Network

There are two crucial components to the Vision Network, the Wide-Angle Raspberry Pi Camera and the HP 4110 Webcam as shown in figure 4. The Wide-Angle Raspberry Pi Camera (Pi Camera) occupies most of the weight while training on DCNN/LSTM while the HP Webcam provides a stream of images of the traffic light to a trained Convolutional Neural Network model. Both cameras supply images to the Raspberry Pi 3 for throttle values and steering angle prediction.

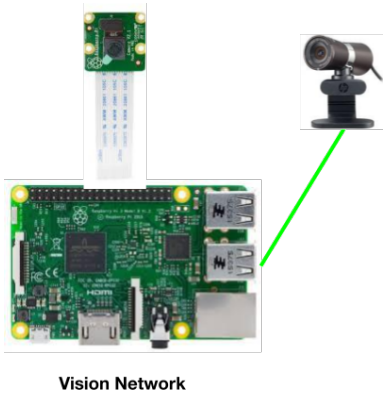

Fig. 4. Vision Network

TABLE II
VISION NETWORK

#	Camera	Type	Function
1	Wide Angle Fish-Eye Pi Camera	Camera	Track Detection
3	HP 4110 Webcam	Camera	Traffic Light Detection

D. Engine Network

The Engine Network has been completely refreshed from the stock configuration given in the original RC Car. This new network is equipped with a 27 turn motor, which significantly increases torque when compared to the stock 15 turn motor. A 4 Amp LiPo battery is used, rather than the stock 1.8 Amp NiMH battery because of the voltage drop off as shown in figure 5. Using a NiMH battery, throttle PWM values are inconsistent because PWM values tend to be higher at the end of the battery cycle in order to maintain the same velocity, whereas using LiPo batteries, throttle PWM values stay relatively the same throughout the battery charge cycle to maintain a constant velocity. Having a consistent throttle value cleans the data so that the LSTM will not have to find out this vague pattern in the data. A 60A ESC and a 5V BEC are used to receive PWM signals from the PCA 9685 and control the motor. A HITEC HS-645MG Servo has been mounted on the car to increase steering range and to increase K_s in the Ackerman steering geometry equation as shown in equation 1. By increasing K_s, ETG is capable of making harder turns left or right on difficult corners of the race. The Ackerman steering angle equation is described by Kim [3] as follows; θ_t is steering angle in degrees, v_t (m/s) is the velocity at time t. K_s is the steering ratio between the turn of the steering and the turn of the wheels, as described before. K_{slip} represents the relative motion of the steering wheel with respect to the road and d_w is the length between the front and rear wheels.

$$\theta_t = f_{steers} = (u_t)d_w K_s(1 + K_{slip}v_t^2) \quad (1)$$

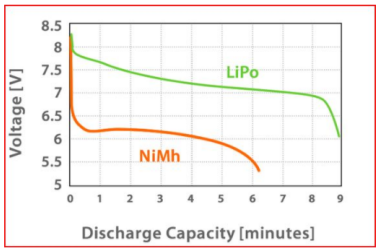

Fig. 5. LiPo and NiMH voltage dropoff curves

E. More Electro-Mechanical Engineering

The stock ECX 1/10 Circuit 2WD has been heavily modified as described in the Engine Network subsection. Some modifications to the chassis are:

- Damped suspension

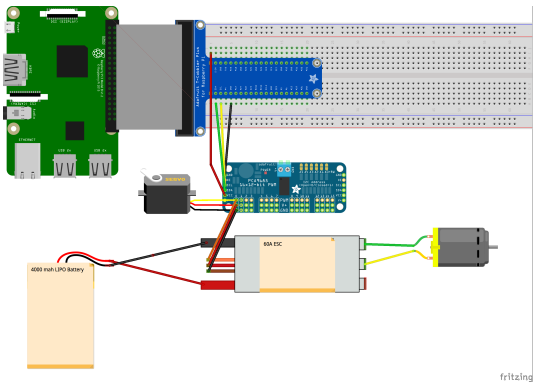

Fig. 6. Engine Network

TABLE III
ENGINE NETWORK

#	Part	Type	Function
10	60A Dynamite ESC	ESC	Motor Control
14	HITEC HS-645MG	Servo Motor	Traffic Light Detection
9	4000 MAH LiPo	Battery	Power supply
15	PCA9685	PWM Control	PWM Output
11	27T Brushed Motor	Brushed Motor	Throttle

  - Stock suspension allowed only 0.5 kg of load on the vehicle. By placing 2 wooden spacers 2cm long adjacent to suspension coils across the vehicle, load was increased to an upwards of 5 kg.
  - An increased load caused the vehicle to tilt left or right in order to maintain ground force. Simulated in figures 7 and 8.
- Increased tooth count for the pinion gear
  - The stock 15 tooth pinion gear was changed for a 20 tooth pinion gear.
  - A higher tooth count for the pinion gear increases speed but slows acceleration.
  - A higher tooth count is necessary since the 27T motor made the robot noticeably slower from the stock 15T motor. This higher speed allowed the ETG to go at speeds up to 12 m/s.

F. Vehicle Body

The vehicle body was created to be mounted on top of the chassis of the ECX 1/10 Circuit car. It was created using Fusion 360 and printed through 3D Hubs, an online 3D printing service. The parts were shipped from Ohio and were assembled with wooden dowels and Balsa wood. Using the wooden support system, 3D printing costs were reduced by an upwards of 10 times the original price of the CAD model. The original CAD model is shown in figure 9.

III. Hardware Systems Design

A. Processor Framework

Processing is distributed among the Raspberry Pi 3 (RPI3) and the Arduino UNO R3. During the data collection stage,

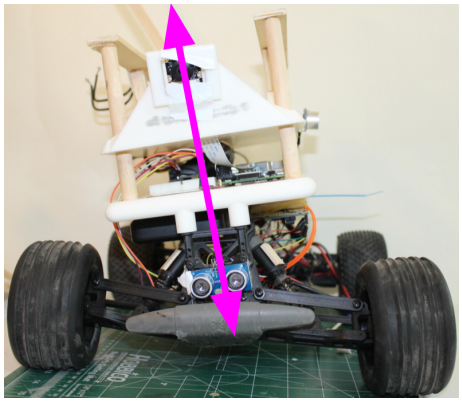

Fig. 7. Line representing the normal force vector, which points roughly north-western, and the gravitational component antiparallel to the normal force vector

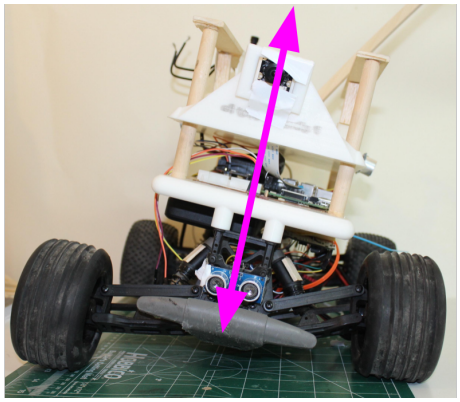

Fig. 8. Line representing the normal force vector, which points roughly north-eastern, and the gravitational component antiparallel to the normal force vector

the RPI3 collects sensor data from the Arduino and stores that sensor data with image data. This data is trained on the DCNN/LSTM on a cloud server to generate a model. This model is run on the RPI3, which uses sensor data and image data to make throttle and steering angle value predictions.

B. Electrical Sub-systems

A 10 amp 5v portable battery charger powers the RPI3. The RPI3 provides power to the Arduino through a USB port and powers the PCA 9685 to generate PWM signals to control the servo and ESC. A 4 amp 7.2v 2 cell LiPo battery powers the ESC which controls the 27T motor.

IV. Perception And Control

A. Software Architecture

The RPI3 is the central processor, conducting all the subsystems to work together. The Sensor Network works as follows; The Arduino UNO is responsible for polling the 4 Ultrasonic sensors located on all four sides around the vehicle as well as the MPU 9250 IMU. Generally, with a required buffer included to effectively poll all sensors within a couple milliseconds, a steady rate of 12 readings per second is achieved. These readings are sent to the Raspberry Pi 3 through the UART

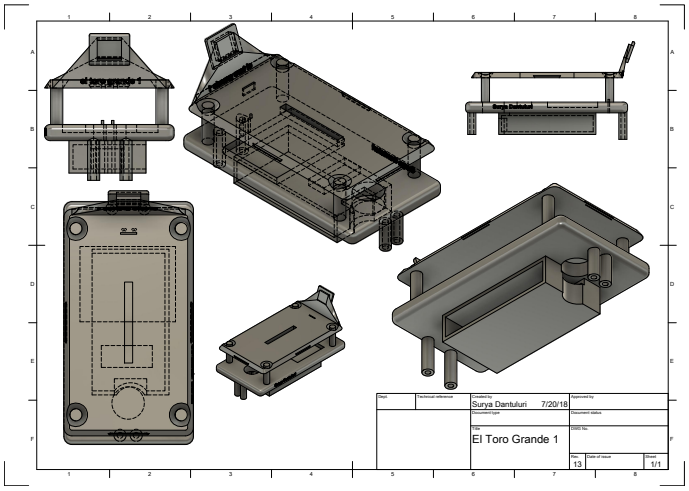

Fig. 9. CAD Drawing for ETG upper body

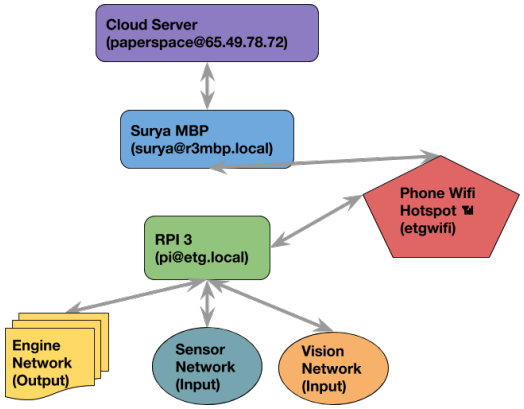

Fig. 10. Software Architecture

serial port using a UDP connection as shown in figure 10. The Raspberry Pi 3 only receives readings when it writes to the Arduino through the serial port, requesting a reading. This occurs every time the main program on the Raspberry Pi 3 goes through the drive loop(explained in the next section). The drive loop on the Raspberry Pi 3 occurs at a rate of around 20-30 times a second. As a result, the RPI3 and Arduino protocol are forced to be done on a separate thread. This allows the drive loop to run efficiently while also getting sensor readings at a steady rate. The RPI3 performs all the high-level processing, including data management and data retrieval. A cloud server preprocesses and runs the DCNN/LSTM model on the data collected by the RPI3. Generally, this takes 30-40 minutes to train on a dataset of 40,000 images. The cloud server consists of 8 CPU cores, 30 gigabytes of RAM, and a Nvidia Quadro P4000. The model uses Tensorflow GPU to train significantly faster. The trained model is then run on the RPI3, which uses all the networks to predict throttle values and steering angles. The Surya MBP node is used to start the car through SSH which is done through the Wifi hotspot as shown in figure 10.

1) Drive Loop: The drive loop is a loop that polls every component in the Engine, Vision, and Sensor network. The drive loop is used for both the data collection stage and running stage. In the data collection stage, the loop polls components as follows: Pi Camera, 4 Ultrasonic Sensors and 1 IMU (run on a eparate thread), steering angle, and throttle value. This data is stored in JSON files with references to the corresponding image (since an image is taken for every iteration of the drive loop). During the running stage, the loop polls the same components except for the engine network. The vision and sensor network are used as parameters for the trained model which outputs throttle and steering angle value predictions.

B. Perception and Planning

1) Object Detection: ETG solely uses the webcam in order to detect objects. For IARRC, a 3 layer Deep Convolution Neural Network is used as shown in figure 13. A 150 by 150-pixel image is formed by scaling the original 720 by 480 image down. This scaled image is put through a ConvNet. Each Convolutional layer takes a local receptive field (we used a stride length of 2) and applies random weights (which it learns over time through backpropagation) to each dimension. Each activation layer ensures that these weights are not negative. Then, a max pooling layer takes the max values within the local receptive field. This block of: Convolutional layer, Activation layer, and Max pooling layer is repeated three times. These weights are then flattened to one dimension (activation and dropout are also added to prevent negative values and overfitting, respectively) so that the CNN can predict on whether the given image contains a traffic light that is red or green as shown in figures 11 and 12. Data is provided from Google Images.

2) Path Planning: As mentioned in the introduction, ETG does not use conventional path planning methods, rather a new neural network model. A DCNN is utilized for predicting steering angles while a variant of the Recurrent Neural Network, a Long Short-Term Memory System is utilized for throttle value prediction.

C. Trajectory and Velocity Control

1) Steering Control: A Deep Convolutional Neural Network is used, similar to the network used for the object detection. The CNN as shown in figure 15 has a batch normalization and max-pooling layer of a stride length of 2 for every layer. The encoder extracts features from the first part of the DCNN before being decoded into vectors that eventually are flattened to a vector of length 10. Steering control is dependent on which of these values are turned on. If the first value is 1, then the steering is turned 100% to the left, whereas if the last value is 1 then the steering is turned 100% to the right.

Training Methods We utilized OpenCV and PyGame for light-weight image processing and image pre-processing.

Fig. 11. Green prediction

Fig. 12. Red prediction

When we changed colorspaces and added a variety of tuned masks to brighten colors (such as yellow and white for the race track) we found out that our model did considerably better as shown in figure 14. Unfortunately, our RPI3 ran into memory issues even when cache storages were cleared. This limited us to use our original RGB image during the entirety of the IARRC 2018.

2) Throttle Control: We found out that a Long Short-Term Memory System worked better than a DCNN through trial and error. Our LSTM system is a RNN variant that updated its weights through backpropagation (like the previous DCNN models) through the Time Distributed Layer. Our LSTM model consists of three gates within each unit (we use 2 units).

- **Input Gate:** Decides what values from input it should update
- **Output Gate:** Decides what values it should output based on input and memory of the unit
- **Forget Gate:** Decides what values it should through away from the unit

A better representation of these gates are shown in figure 15

To prevent overfitting we add Dropout at a value of 0.1 and a Rectified Linear Units (ReLU) layer to prevent negative values.

Using a LSTM model, the throttle took the previous sequence of images and other metadata in order to produce

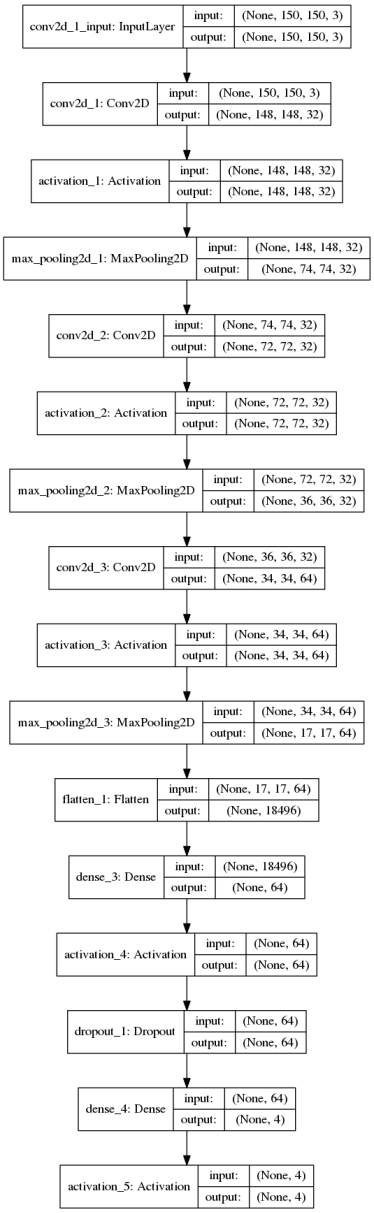

Fig. 13. Deep Convolutional Neural Network used for Traffic Light recognition (Diagram made by built-in Keras function)

predicted throttle values that were not drastically different from the previous time step. This meant the car sped up and slowed down with a difference of no more than 20 units in the scale of 220 (min PWM value; 0 m/s) to 420(max PWM value; 12 m/s) PWM values. Using a traditional DCNN as used before, throttle values were erratic and broke several components when testing since predictions were not based on top of one another. Predictions for both the LSTM and DCNN model (for throttle control) did have erratic values, however, the LSTM ensured that these erratic values were dropped through the Forget Gate while the DCNN used these values.

Optimization Techniques We used the Adam optimization

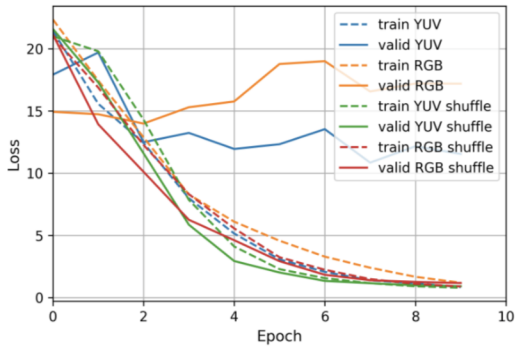

Fig. 14. Loss graphs when changing colorspaces

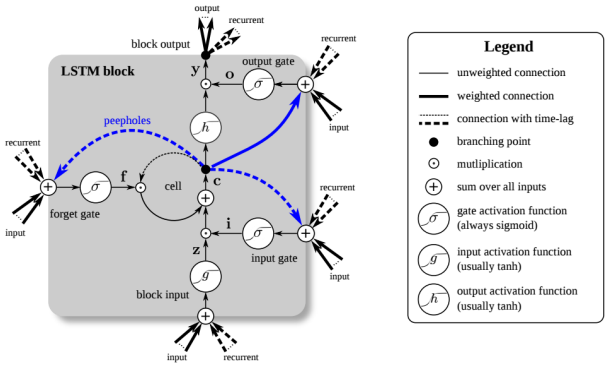

Fig. 15. Representation of One LSTM [1]

algorithm, as Mohandas [4] explains, Adam is used to combine averages of previous gradients computed through backpropagation at different moments to better adaptively update weights of the model. We also used the Mean squared error function as shown in equation 2, by Hao [2], to determine the loss or accuracy between the validation and training dataset.

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{2}$$

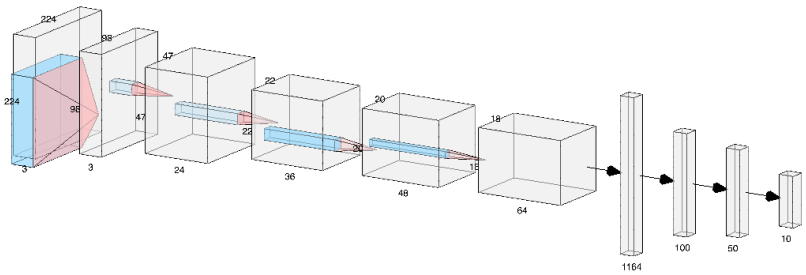

Fig. 16. Deep Convolutional Neural Network for Throttle Control (Diagram made by hand)

V. Vehicle Behavior

A. Line Detection

Without OpenCV, ETG can identify lines purely based off training data. Visualizing salient objects detected by the model, we find this out in figures 17 and 18

Fig. 17. Line detection

Fig. 18. Clearer line detection

B. Target Region using Lines

From the instance in figure 18, ETG starts turning left. As it is, the DCNN focuses its attention toward the right side of the viewing window, indicating that steering is predicting values that turn the vehicle toward the right as indicated in figure 19. Similar behavior is seen in figures 20 21 where the model is trying to stay near the detected lines.

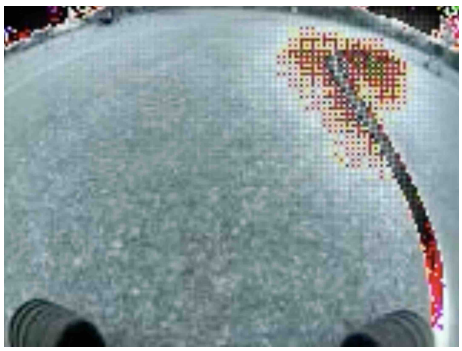

Fig. 19. Line region detection

Fig. 20. Line region

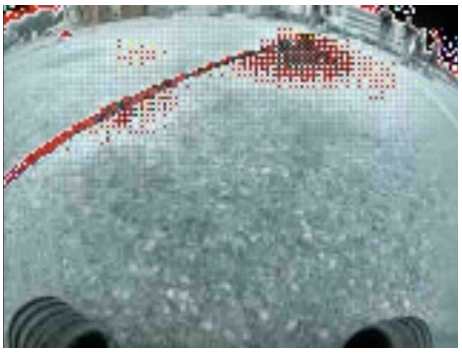

Fig. 21. Line region detection

C. Target Region Without using Lines

Even without detecting the lines, the trained model (surprisingly) finds out regions of interest. This is apparent in figures 22 and 23 where the heatmap focuses interest on the track, not the lines themselves.

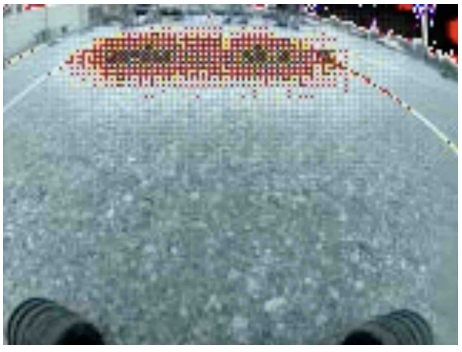

Fig. 22. Line region detection

VI. Conclusion

ETG works better than expected on courses its never seen before. As a proof of concept, it has some flaws, but proves that machine learning methods can rival traditional computationally expensive computer vision, path planning, and localization algorithms. Memory and caching issues are still a big limitation to the RPI3, which may be fixed with software updates or may be a hardware limitation. LSTM systems have proven to exceed DCNN models with throttle control and vise versa for steering control.

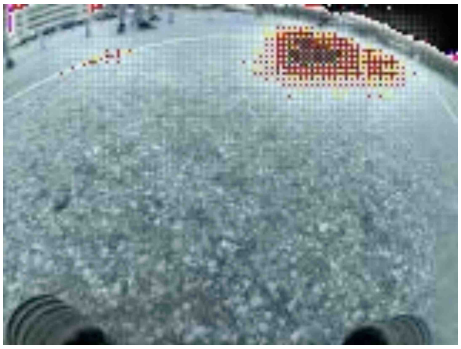

Fig. 23. Line region detection

A. Appendix

The Monta Vista High School Team (Surya Dantuluri) competing in the 2018 International Autonomous Robot Racing Challenge are presented in Table 4:

TABLE IV
Monta Vista High School Team

Name	Role	Hours
Surya Dantuluri	Embedded Software	500
	Embedded Hardware	
	Steering Control	
	Software Integration	
	Vision Systems	
	Vision Systems Design	
	Electro-Mechanical Design	
	Machine Learning Software	
	Electrical and Instrumentation	

TABLE V
Cost Estimate

Item	Actual Cost (USD)	Cost to Team
ECX 1/10 Chassis	160.00	160.00
Laptop	1300.00	Personal Donation
Raspberry Pi 3	40.00	40.00
Arduino UNO R3	20.00	20.00
3D Printed Parts	90.00	90.00
Sensors	50.00	50.00
Portable Power Bank	40.00	40.00
5 Batteries and Charger	300.00	300.00
Motors (Brushless and Servo)	70.00	70.00
ESC(s)	80.00	80.00
Wires and Etc.	50.00	50.00
PCA 9685	20.00	20.00
Total	**2220.00**	**920.00**

References

[1] Greff, K., Srivastava, R., Koutnk, J., Steunebrink, B. and Schmidhuber, J. (2018). LSTM: A Search Space Odyssey - IEEE Journals & Magazine. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/7508408/ [Accessed 21 Jul. 2018].

[2] Hao, Z. (2018). Loss Functions in Neural Networks — Isaac Changhau. [online] Isaac Changhau. Available at: https://isaacchanghau.github.io/post/loss_functions/ [Accessed 21 Jul. 2018].

[3] Kim, J. and Canny, J. (2018). Berkeley EECS Publications. [online] People.eecs.berkeley.edu. Available at: https://people.eecs.berkeley.edu/ jfc/papers/17/iccv-final-interpretable_learning_for_self_driving_cars.pdf [Accessed 21 Jul. 2018].

[4] Mohandas, G. (2018). What is an intuitive explanation of the Adam deep learning optimization algorithm?. [online] Quora. Available at: https://www.quora.com/What-is-an-intuitive-explanation-of-the-Adam-deep-learning-optimization-algorithm [Accessed 21 Jul. 2018].