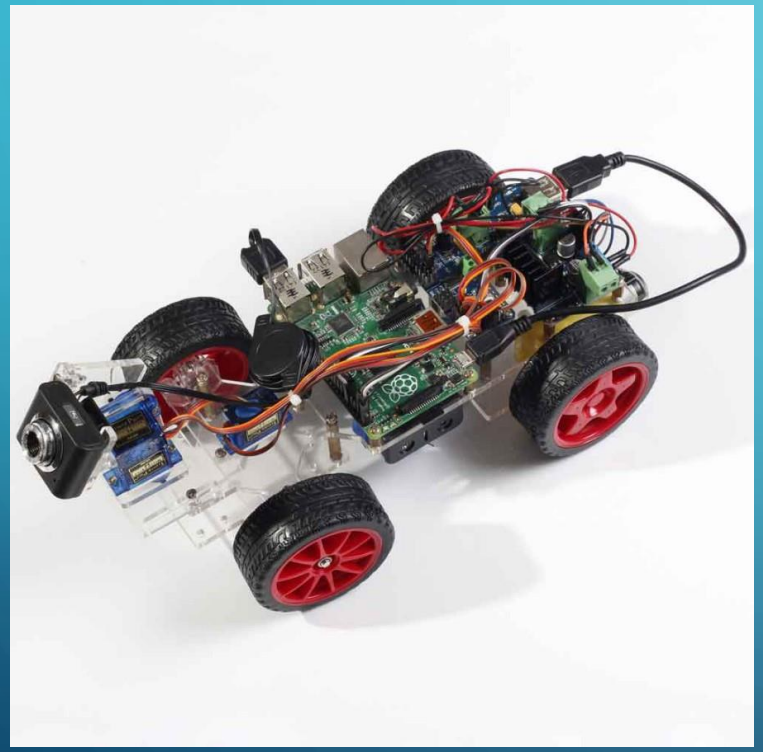
SELF-DRIVING VEHICLE

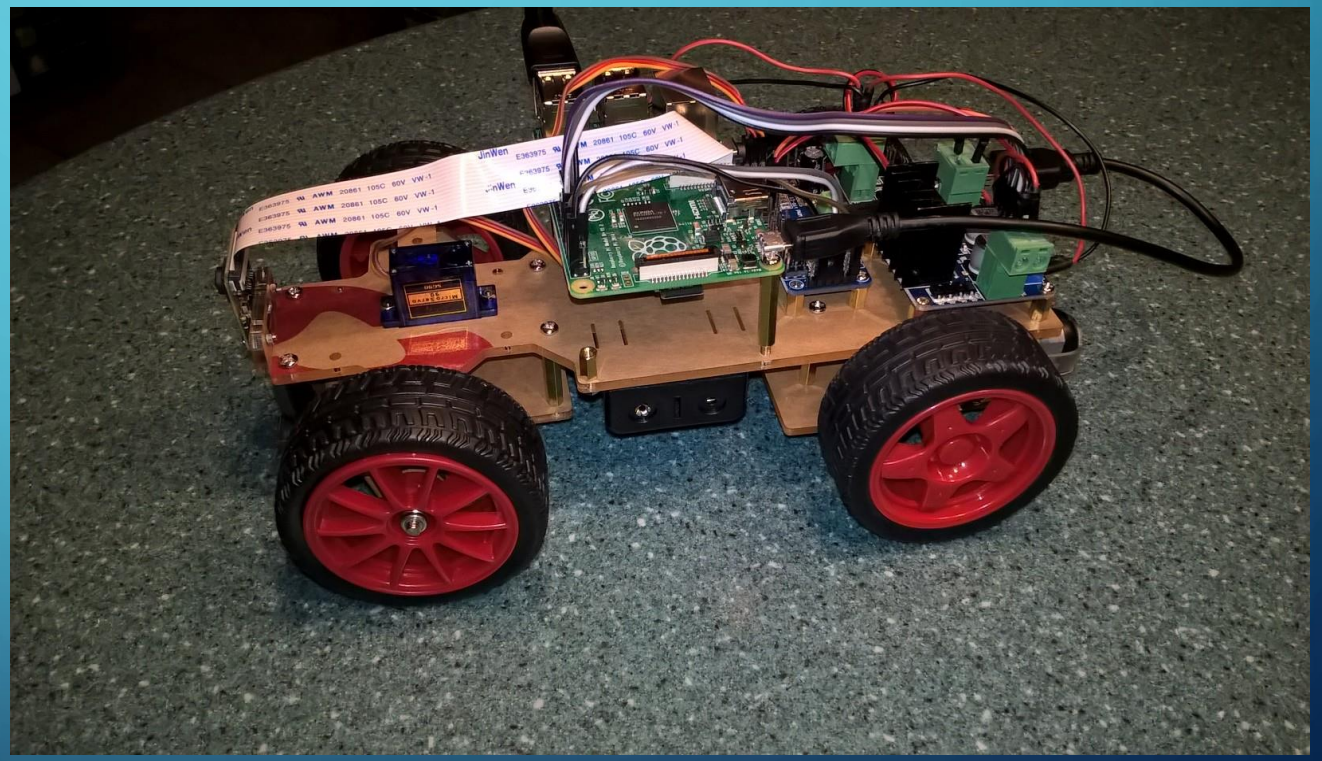
LESSONS LEARNED FROM A REAL-TIME IMAGE RECOGNITION PROJECT

Gene Olafsen

HARDWARE

Modified SunFounder Rasperry Pi video car kit





SINGLE-BOARD ARM-BASED COMPUTER

RaspberryPi 3

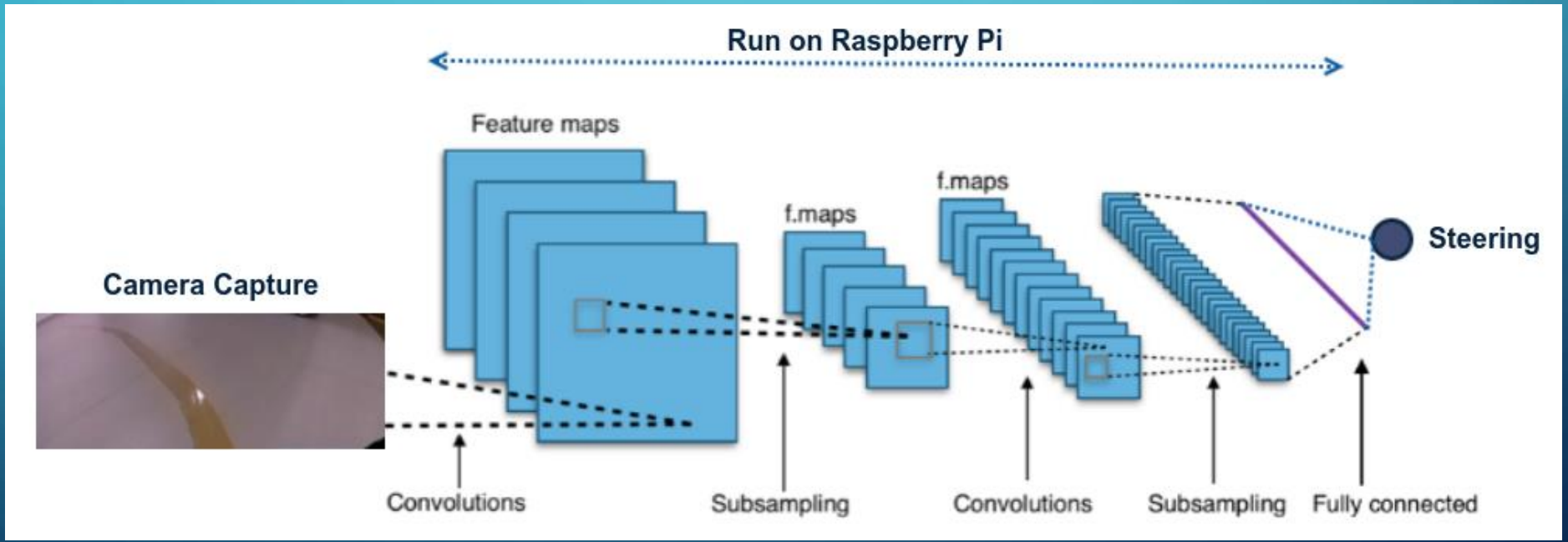
SoC: Broadcom BCM2837
CPU: 4× ARM Cortex-A53, 1.2GHz
GPU: Broadcom VideoCore IV
RAM: 1GB LPDDR2 (900 MHz)
Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage: microSD
GPIO: 40-pin header, populated
Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

SOFTWARE

- Python
- TensorFlow
- Keras
- OpenCV

DEVELOP A MODEL

- Follow a line...

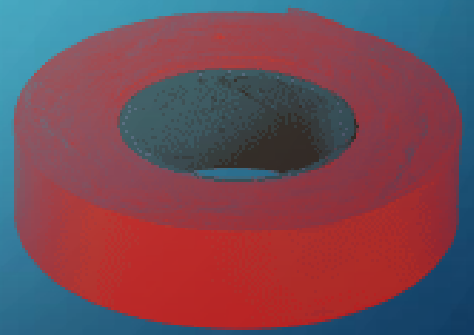


DRIVE WITH XBOX CONTROLLER

- https://github.com/FRC4564/Xbox

DEFINE A PATH

- Rope or Tape



DATA COLLECTION

- Images and Steering Angle
- 

433

PULSE WIDTH MODULATION AND SERVO

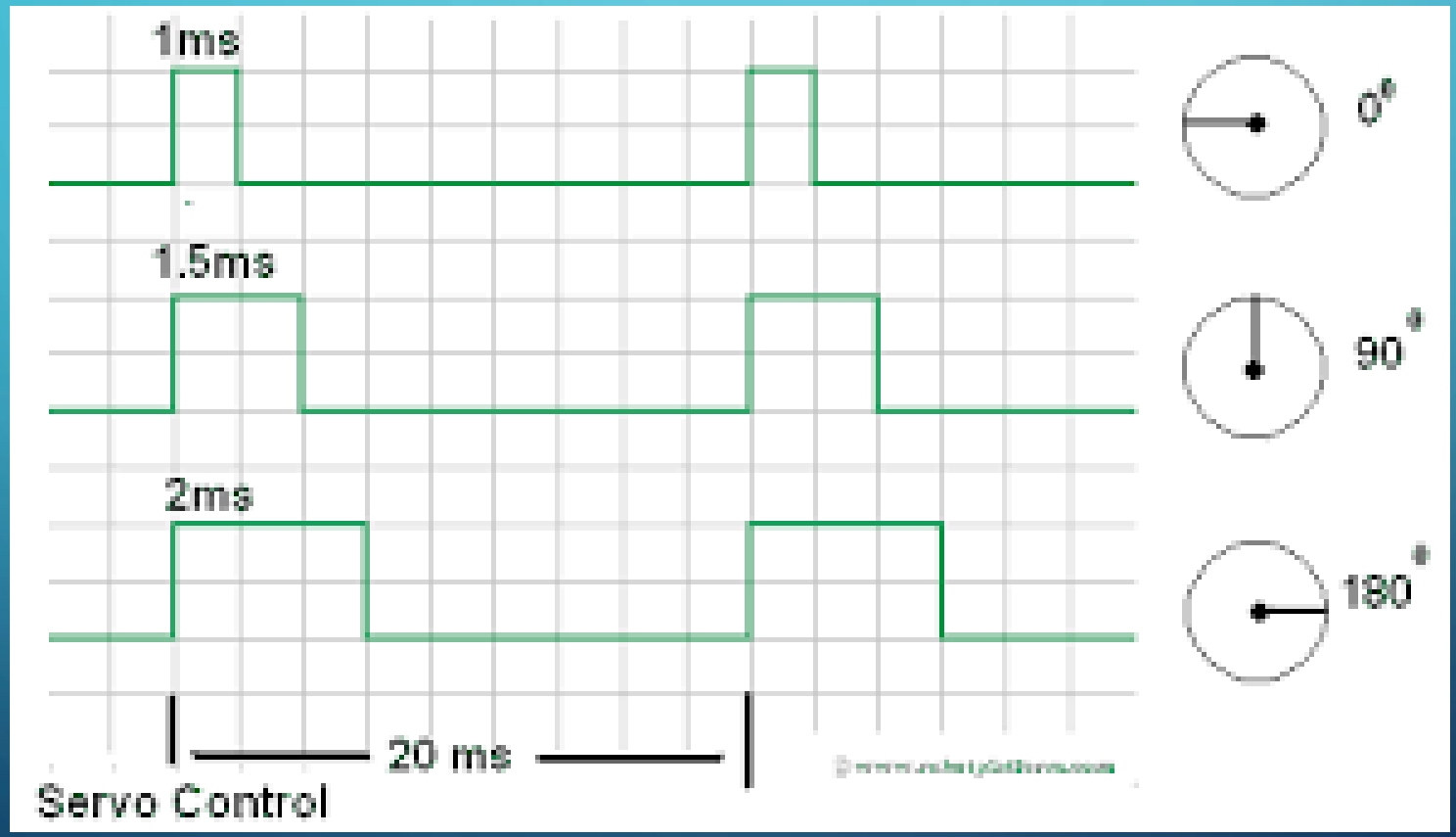
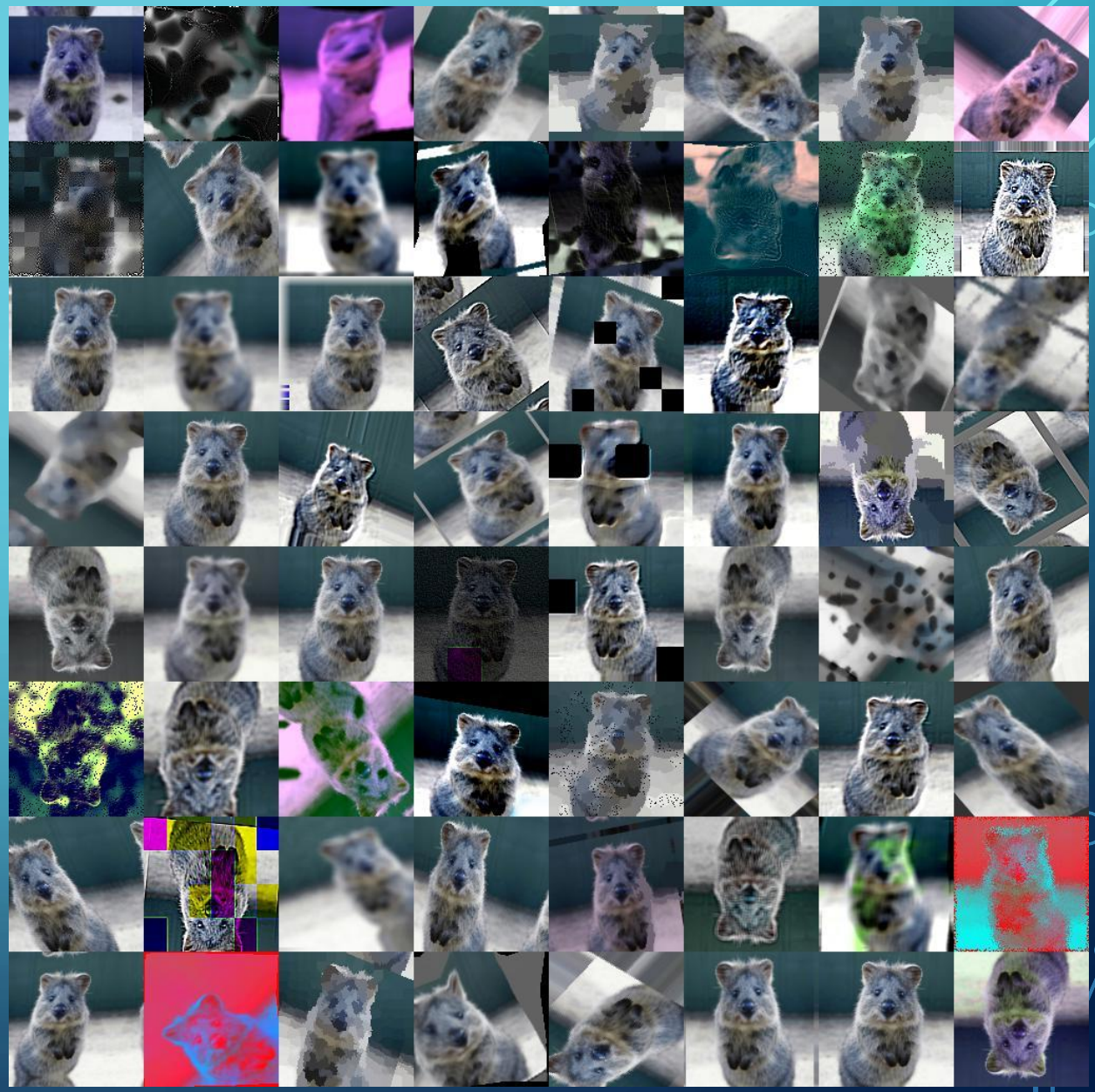


IMAGE AUGMENTATION

- https://github.com/aleju/imgaug



CODE

```
sometime = lambda aug: iaa.Sometimes(0.3, aug)
sequence = iaa.Sequential([
  # blur images with a sigma between 0 and 3.0
  sometime(iaa.GaussianBlur((0, 1.5))),
  # sharpen images
  sometime(iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))),
  # add gaussian noise to images
  sometime(iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 3.), per_channel=0.5)),
  # randomly remove up to 10% of the pixels
  sometime(iaa.Dropout((0.0, 0.1))),
  # dropout variant
  sometime(iaa.CoarseDropout((0.10, 0.30), size_percent=(0.02, 0.05), per_channel=0.2)),
  # change brightness of images (by -10 to 10 of original value)
  sometime(iaa.Add((-10, 10), per_channel=0.5)),
],
random_order=True # do all of the above in random order
)
```

MODEL

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 80, 160, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 78, 156, 16)       736
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 76, 152, 16)       3856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 19, 76, 16)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 17, 72, 32)        7712
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 68, 32)        15392
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 34, 32)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 34, 4)          132
_________________________________________________________________
flatten_1 (Flatten)          (None, 408)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 409
=================================================================
Total params: 28,237
Trainable params: 28,237
Non-trainable params: 0
```

NETWORK

```
# Construct the network# Const
image_inp = Input(shape=(FRAME_H, FRAME_W, 3))

x = Conv2D(filters=16, kernel_size=(3, 5), activation='relu', padding='valid')(image_inp)
x = Conv2D(filters=16, kernel_size=(3, 5), activation='relu', padding='valid')(x)
x = MaxPooling2D((4, 2))(x)

x = Conv2D(filters=32, kernel_size=(3, 5), activation='relu', padding='valid')(x)
x = Conv2D(filters=32, kernel_size=(3, 5), activation='relu', padding='valid')(x)
x = MaxPooling2D((4, 2))(x)

x = Conv2D(filters=4,  kernel_size=(1, 1), activation='linear', padding='same')(x)

x = Flatten()(x)

x = Dense(1, activation='tanh', kernel_regularizer='l1')(x)

angle_out = x

model = Model(inputs=[image_inp], outputs=[angle_out])
```

COLLABORATE

- Collect the image and steering training data during MW-MLG meetings!
- Train the model.
- Deploy the trained model and see how well it performs.
- Make the training data public (GitHub).
- Modify the model, train and deploy during future meetings.

FUTURE ENHANCEMENTS

- Modulate speed approaching a curve
- Observe 'traffic' signs
- Avoid obstacles

TOOLCHAIN

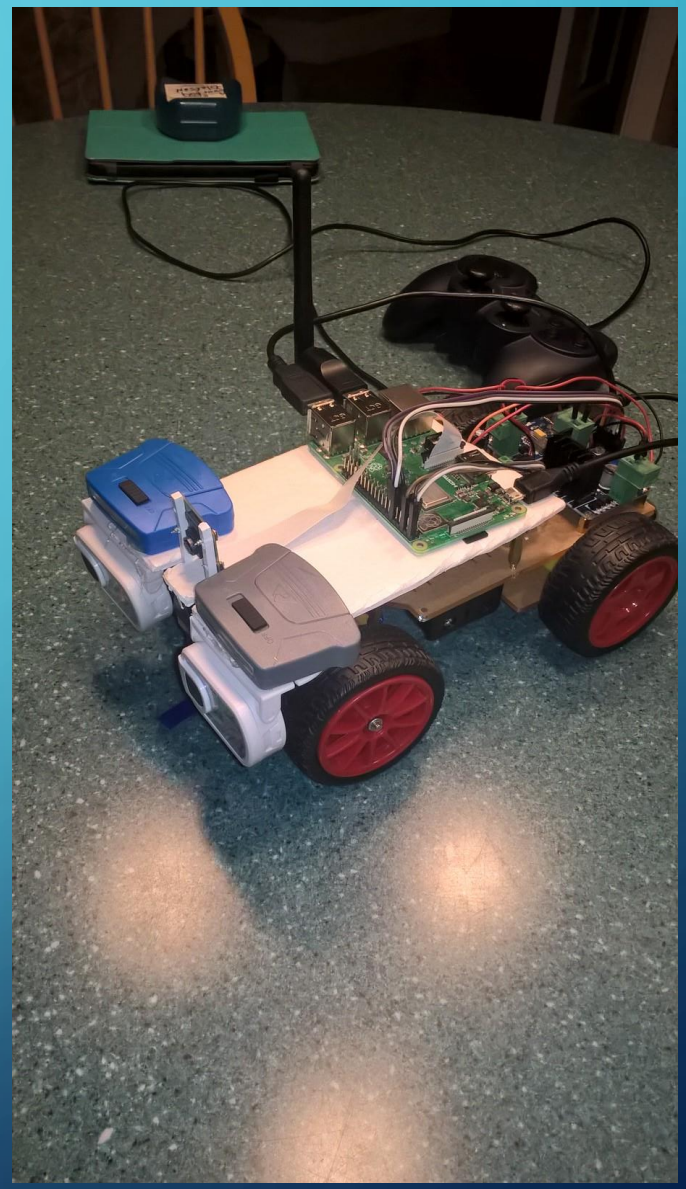
- MobaXterm – SSH Terminal and X-Windows Server
  - https://mobaxterm.mobatek.net/
- Linnode – hosting
  - https://www.linode.com/
- CodeAnywhere – Browser-based SSH, Text Editor
  - https://codeanywhere.com/
- Git – Source/Data Repository
  - https://github.com/

ISSUES AND LESSONS LEARNED

- While this project is inspired by self-driving model car projects that already exist, there is nothing for the vehicle platform or controller which was selected.

IMAGE CAPTURE

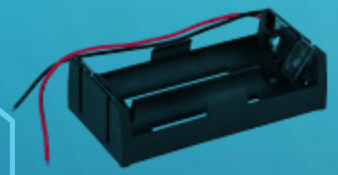
- Initial attemps to capture training data failed miserably because images produced were too dark.
- To correct the issue, a number of very bright LED 'headlights' were added to the vehicle.



POWERING THE PLATFORM

- Developing the capture software and ultimately the program which will load the model and predict the steering angle must be debugged on the actual test vehicle.
- For the time where engaging the motors is not necessary, the Raspberry Pi can be powered from another computer's USB port.
- Testing the data capture with the steering and drive motors functioning reduces battery life to about 20 minutes.
- Even with multiple batteries, it was difficult to work in a 20 minute window.

POWER DISTRIBUTION



Dual 18650 Li-Ion Cells

AC Power Adapter

7.4V

DC-DC Converter

5V

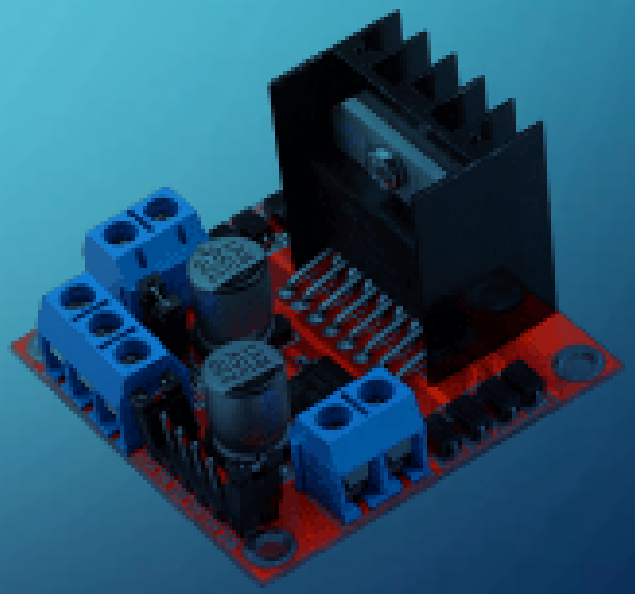
Raspberry Pi

Servo Controller

DC Motor Driver

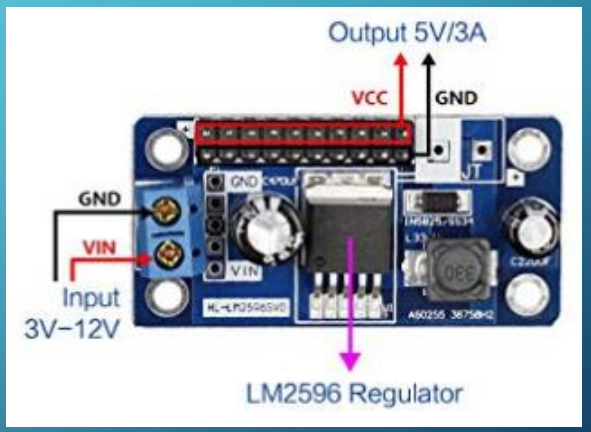
DC MOTOR DRIVER

- The driver is built based on 11 L298N. As a high-voltage and large-current chip for motor driving, encapsulated with 15 pins, the chip has a maximum operating voltage of 46V and an instant peak current of as high as 3A, with an operating current of 2A and rated power of 25W. Thus, it is completely capable of driving two low-power DC motors.



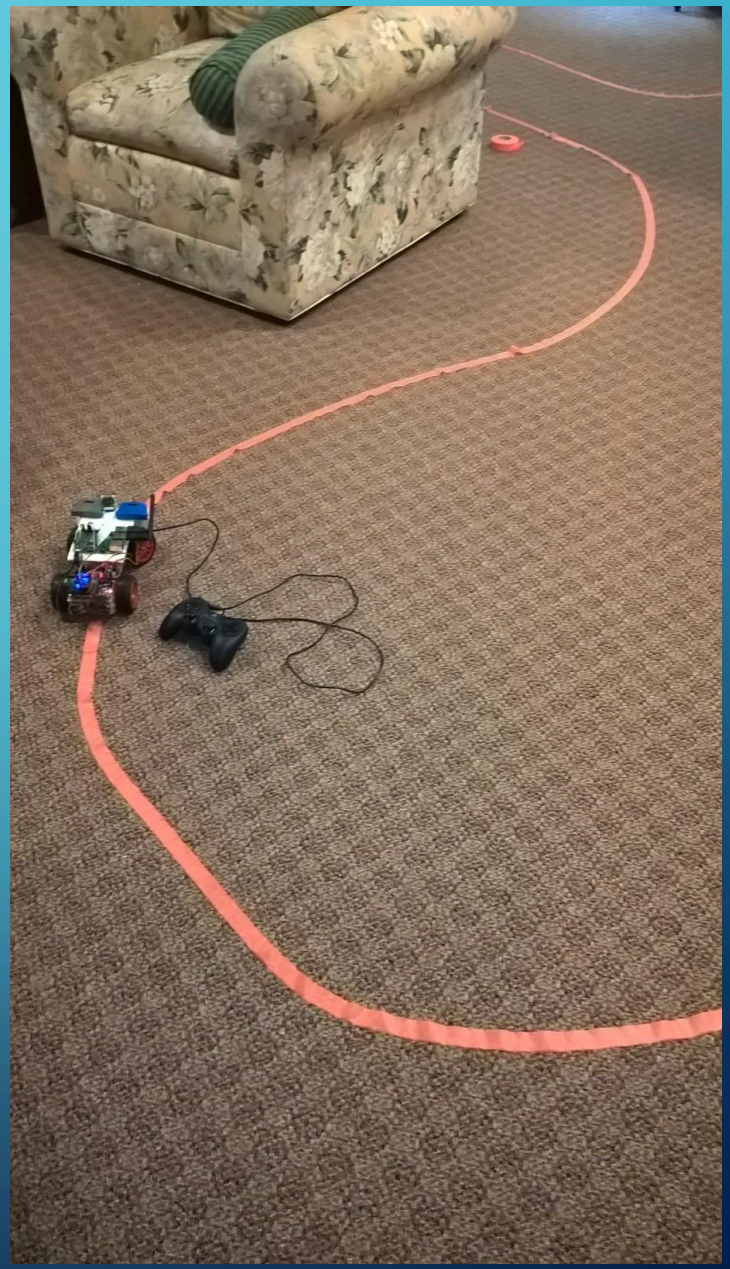
STEP-DOWN DC-DC CONVERTER

- Based on the chip XL1509, the module converts the battery output of 7.4V to 5V, so as 10 to supply power to Raspberry Pi and the servo. As a DC to DC converter IC, the chip has an input voltage ranging from 4.5V to 40V and generates an output voltage of 5V with a current of as high as 2A.



INITIAL TRAINING ATTEMPTS

- I decided originally to provide a long continuous line on which to train the vehicle.
- The gamepad controller is used to steer the vehicle as it collects images and steering angle information.

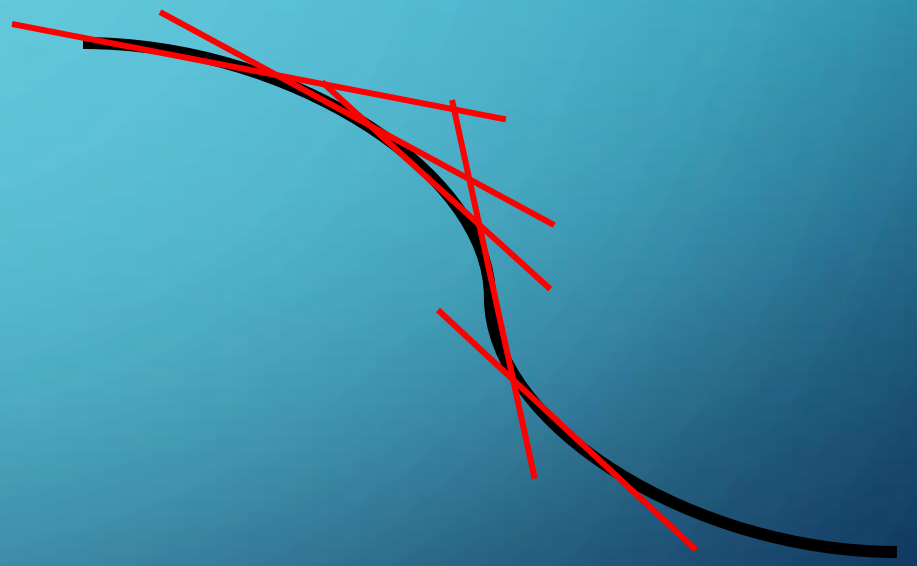


TESTING THE MODEL

- Upon collecting the data, training the model and observing instance execution- the steering response to visual input was ineffective. In fact, there was almost no steering response produced- no matter what input was provided to the camera. The steering servo simple shuttered with a seemingly random response to any input.

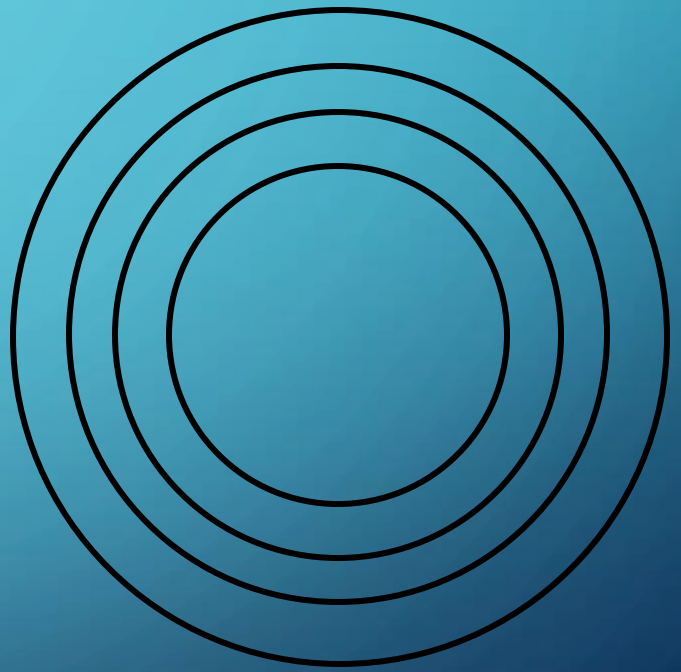
TRAINING DATA ANALYSIS

- Upon collecting the data, training the model and observing instance execution- the steering response to visual input was ineffective.
- The data consisted of mostly 'straight' steering angle values.
- Most of the driving is done in straight lines, with little continuous steering adjustment.

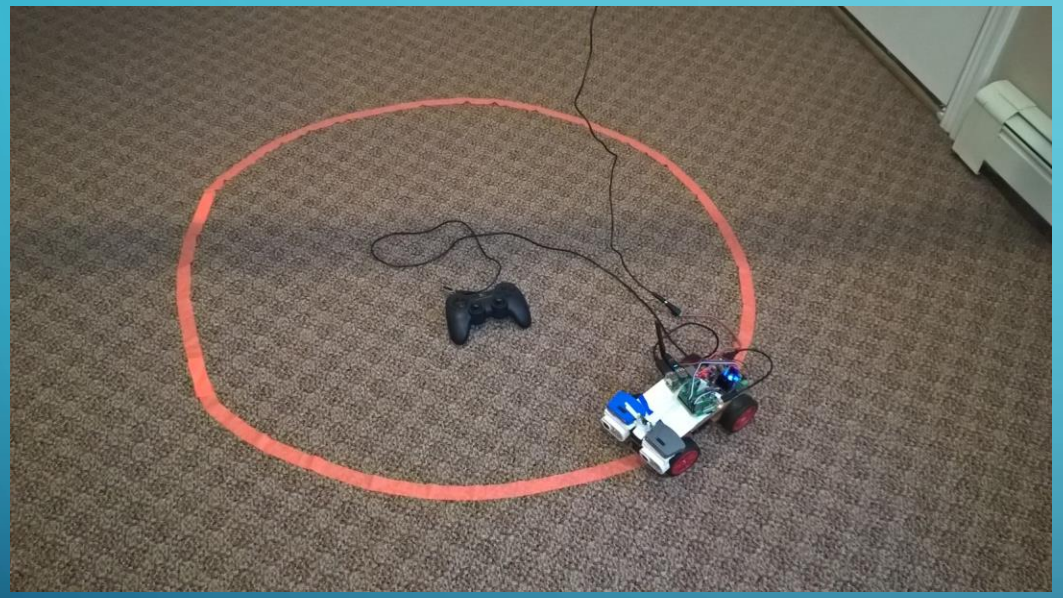


PRACTICE DRIVING CIRCLES

- To capture many training instances with a 'large' steering angle, I decided to train the vehicle on circles of different radii.
- The plan is to be able to apply roughly a continuous input to the variable steering for each of the different radii.



DATA CAPTURE



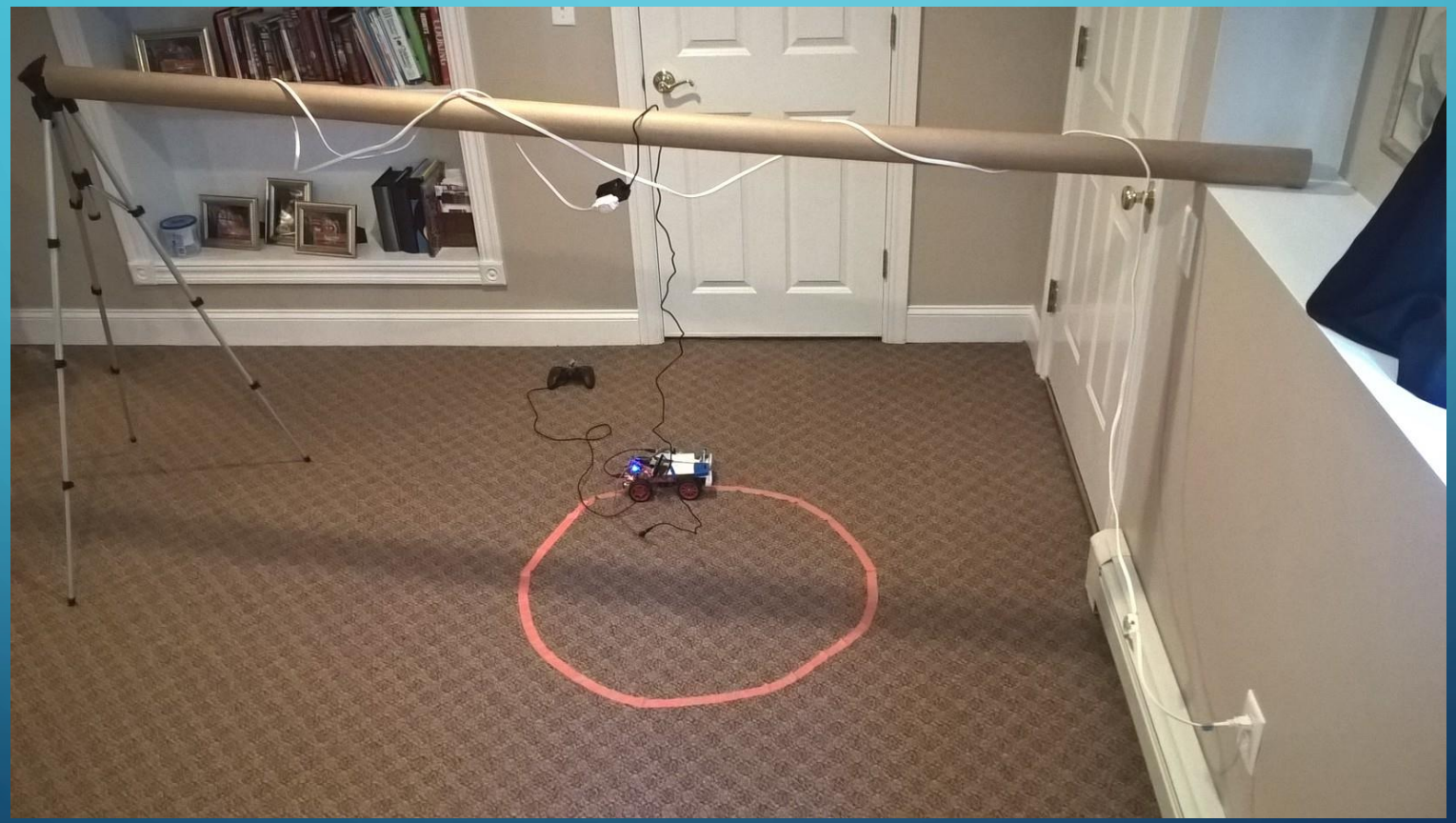
CAMERA POSITION

- The camera was repositioned to 'look' lower toward the ground in front of the vehicle. This change was mde so that the car could respond to sharper turns.

POWERED CAPTURE

- Capturing a large number of training instances requires running the vehicle for extended periods of time... that means, no batteries.
- For each circle- the vehicle is 'driven' clockwise for 2 or 3 revolutions and then counter-clockwise for 2 or 3 revolutions. This procedure captures labeled training data. Finally, the same circle is driven again 2 or 3 revolutions both clockwise and counterclockwise AGAIN to capture labeled validation data.
- The radius of the circle is increased a set amount. (about 2-3 inches)

POWERED CAPTURE RIG

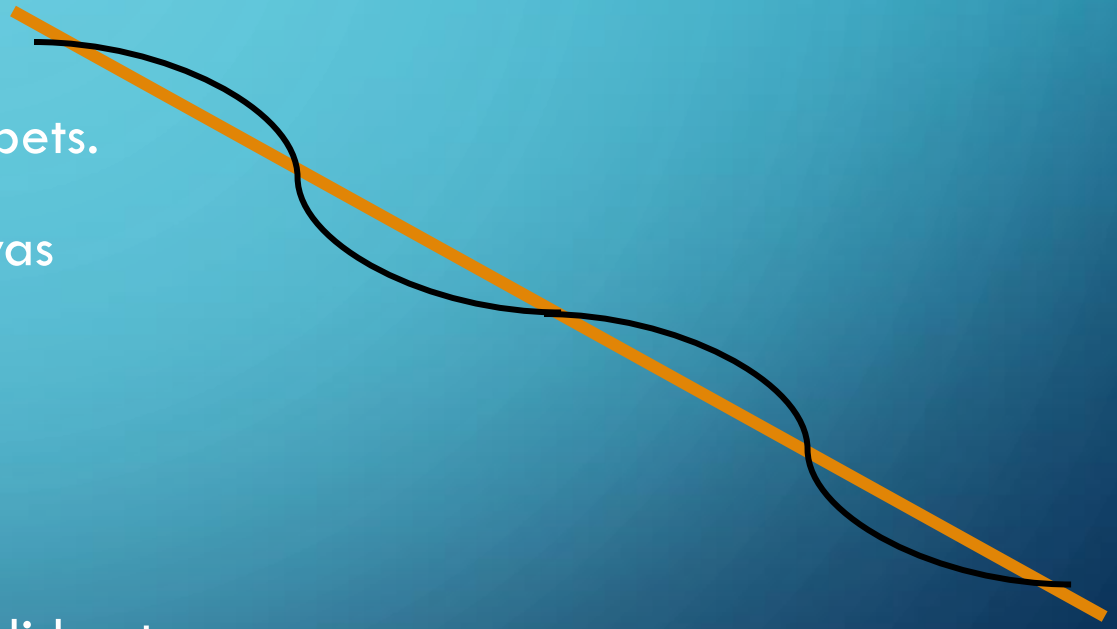


CAPTURE, TRAINING, DEPLOY STEPS

- The data is captured to the RaspberryPi's SD card.
- The training and validation data is uploaded to Github.
- The Linode machine clones the data repository.
- The model is trained and the weights.hdf5 file is uploaded to Github.
- The weight file is retrieved from Github on the RaspberryPi.

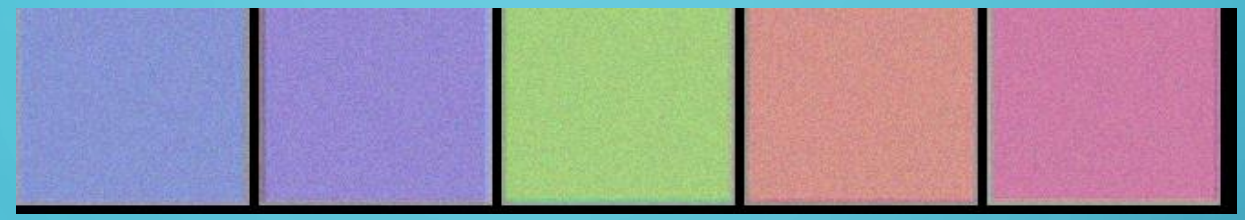
TEST DRIVE OBSERVATIONS

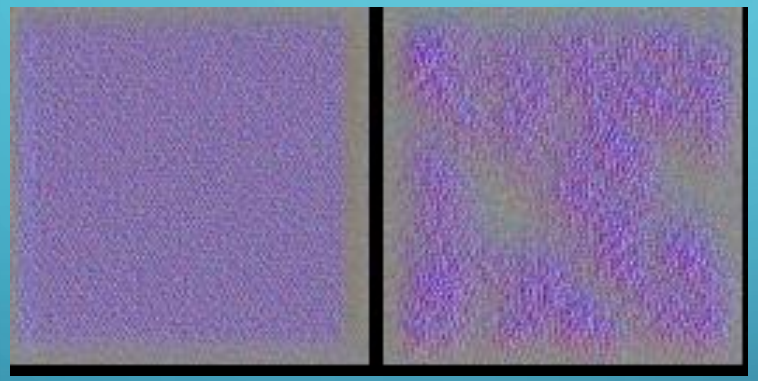
- The car performed very well.
- Able to transition across different color carpets.
- Can drive at speeds faster than the data was captured.
- Faster driving speed revealed a slight 'swimming' motion.
- Probably a result of the training set which did not include any examples of driving straight! This theory will be tested with a new model which includes such training.



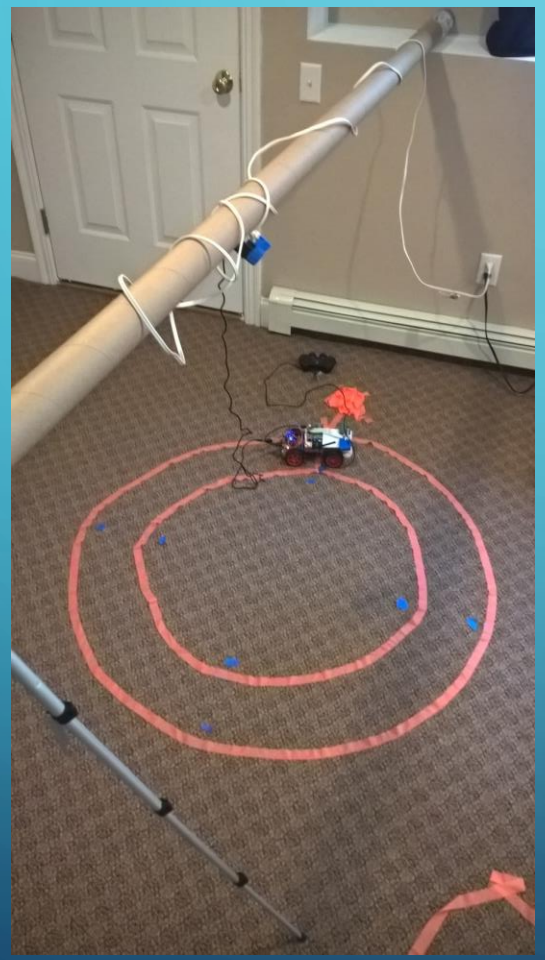
VISUALIZING FILTERS

- Conv1
- Conv2
- Conv3





BEYOND LINE FOLLOWING

YOLO – YOU ONLY LOOK ONCE

- Crash