Behavioral Cloning For Lane Following

Behavioral cloning is a method by which human subcognitive skills can be captured and reproduced in a computer program. As the human subject performs the skill, his or her actions are recorded along with the situation that gave rise to the action. A log of these records is used as input to a learning program.

Behavioral Cloning | SpringerLink
https://link.springer.com/10.1007%2F978-0-387-30164-8_69

Two Case Studies (Research Papers)

1. By: Georgia Tech AutoRally
2. By: NVIDIA

In the context of your specific needs

Behavioral Cloning for Lane following

- Basic tips on how to improve your models ability to work
- Simple things you can do to increase performance
- (Ex: given specs of your processor)

Recall: TIMTOWDI

- **T**here **I**s **M**ore **T**han **O**ne **W**ay **T**o **D**o **I**t
- *Pearl-programming motto*

Recall: Consider the Academic Way

- Find out how very very intelligent people are trying to solve the problem.

Two Case Studies (Research Papers)

1. By: Georgia Tech AutoRally
2. By: NVIDIA

In the context of your specific needs

Behavioral cloning for Lane following

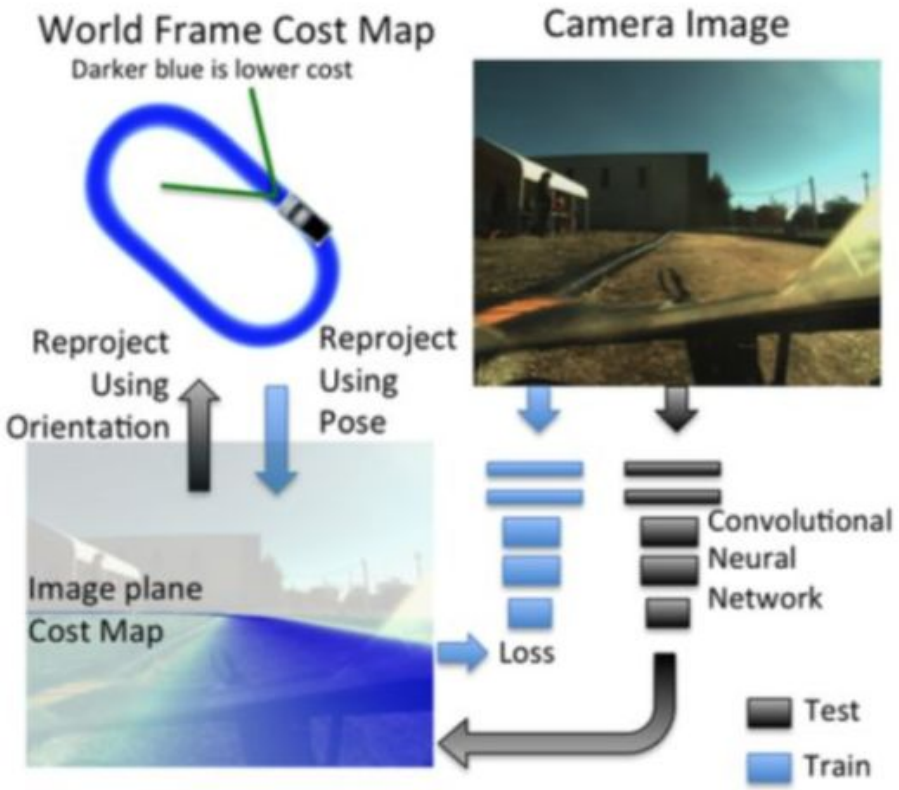
- Basic tips on how to improve your models ability to work
- Simple things you can do to increase performance
- (Ex: given specs of your processor)

RESEARCH PAPER 1

Aggressive Deep Driving: Combining Convolutional Neural Networks and Model Predictive Control

1st Conference on Robot Learning (CoRL 2017),
Mountain View, United States.
arXiv:1707.06404v1 [cs.RO] 17 July 2017

- A novel deep learning approach for analyzing monocular video and generating real-time cost maps for model predictive control which can drive an autonomous vehicle aggressively
- Analysis of the benefits of different representations for the cost maps, with the demonstration that direct prediction of a bird's eye view cost map gives the best performance


World Frame Cost Map
Darker blue is lower cost
Camera Image
Reproject Using Orientation
Reproject Using Pose
Image plane Cost Map
Loss
Convolutional Neural Network
Test
Train

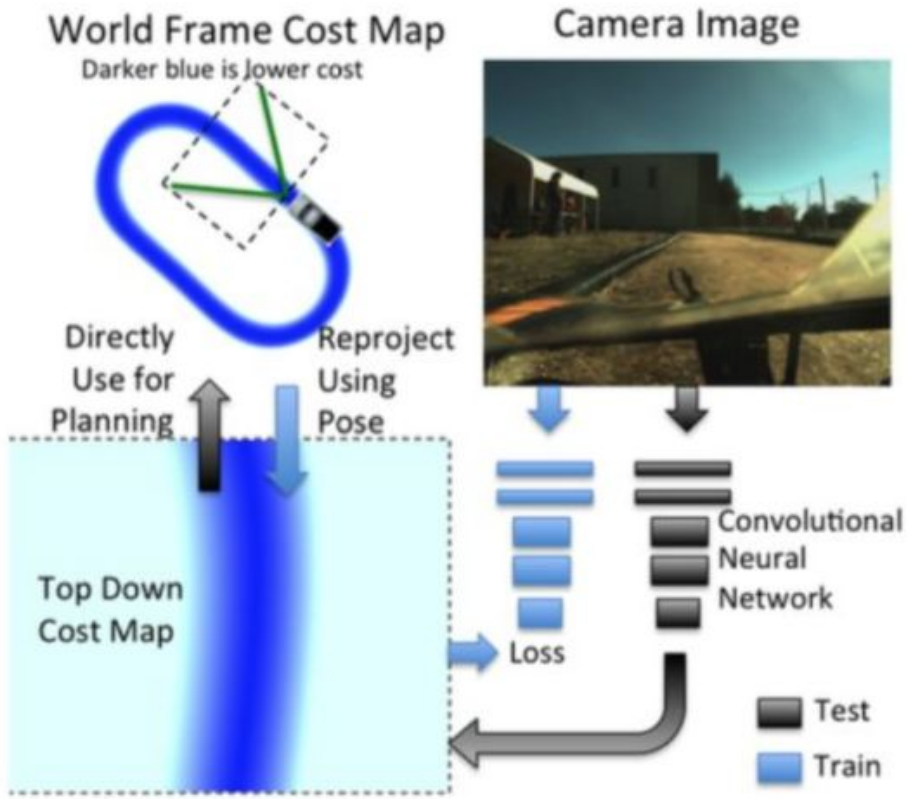
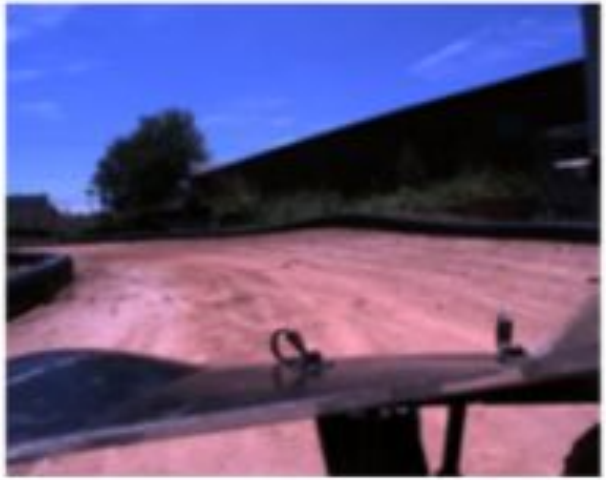
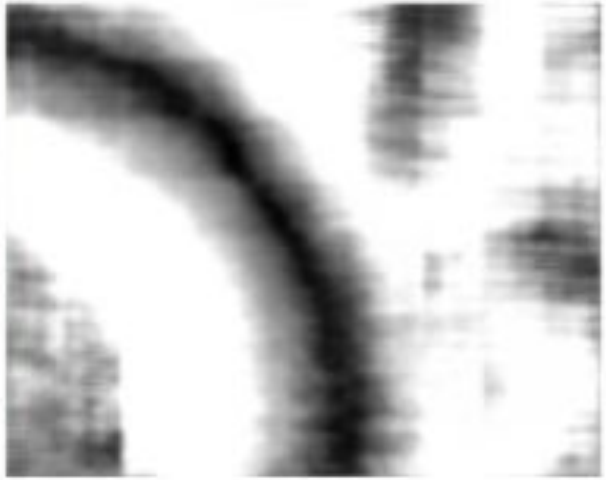
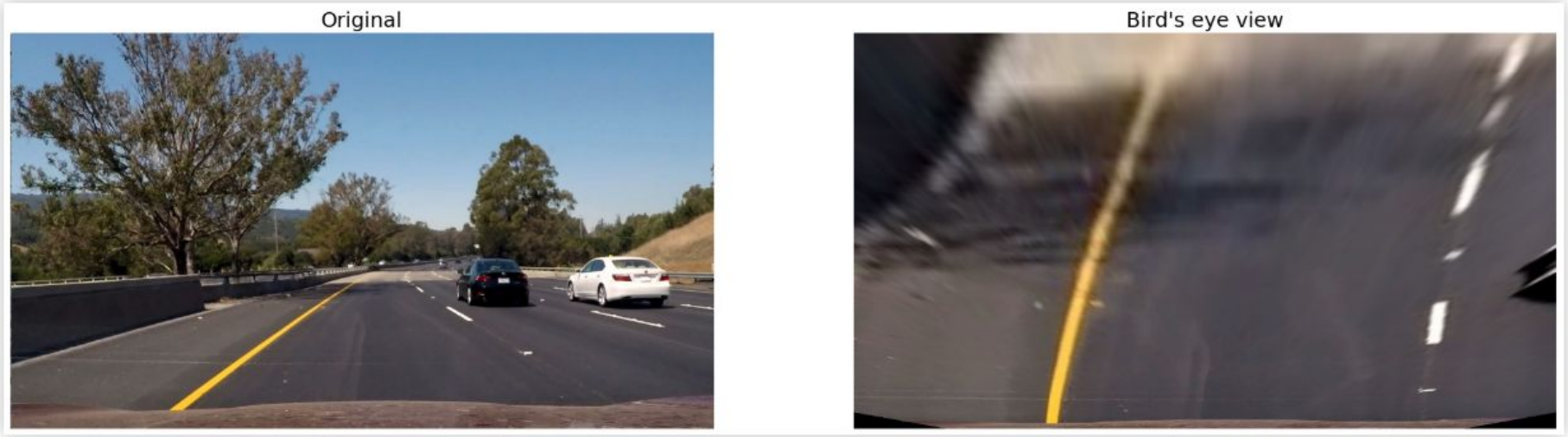

World Frame Cost Map
Darker blue is lower cost
Camera Image
Directly Use for Planning
Reproject Using Pose
Top Down Cost Map
Loss
Convolutional Neural Network
Test
Train

Figure 2: Network architecture with input and training targets. Left: Neural network architecture used to produce top down cost maps. Right: example input image, image plane training target and top down training target, respectively

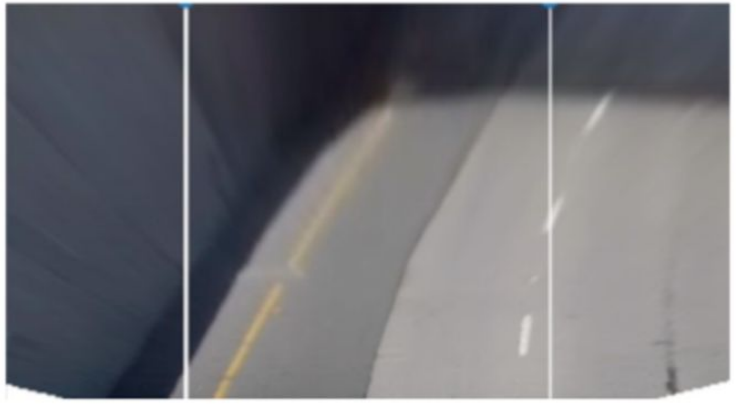



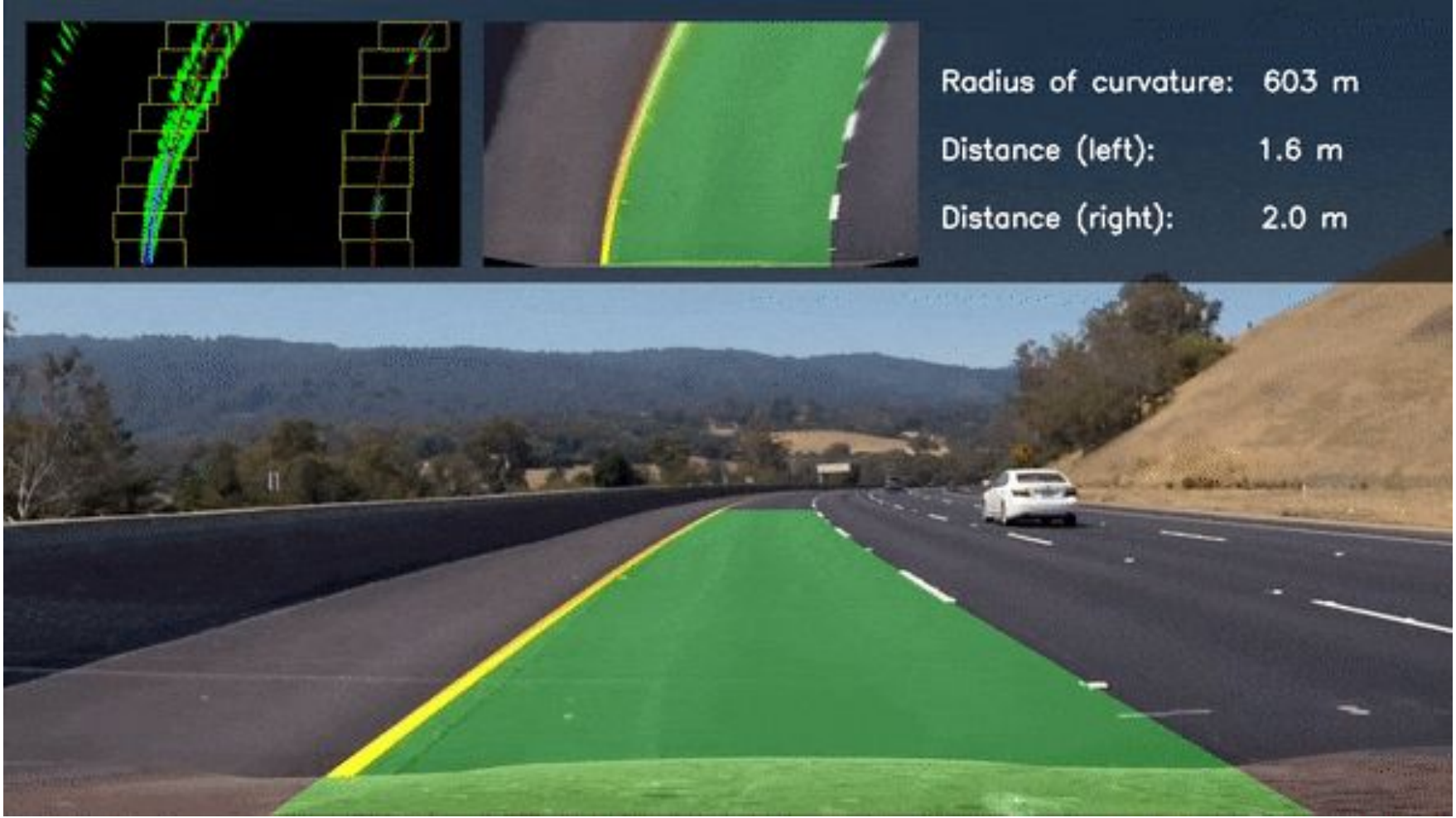

Using the top down network produced significantly more robust, consistent, and overall faster runs than the image plane network. Using the image plane network, it was only occasionally possible to produce runs of 10 consecutive laps (at the slowest speed). Most of the runs lasted between 1 and 5 laps before intervention was required. Usually, this was due to the network not identifying a turn,



From my former classmate: navoshta.com/detecting-road-features/


Fremont Ave
Los Altos





from my former classmate: navoshta.com/detecting-road-features/

```
class BirdsEye(object):

    def __init__(self,source_points, dest_points):
        self.spoints = source_points
        self.dpoints = dest_points

        src_points = np.array(source_points, np.float32)
        dest_points = np.array(dest_points, np.float32)

        self.warp_matrix = cv2.getPerspectiveTransform(src_points, dest_points)

        # Use this matrix to warp img from sky perspective to vehicle perspective
        #self.inv_warp_matrix = cv2.getPerspectiveTransform(dest_points, src_points)

    def skyview(self, img):
        shape = (img.shape[1], img.shape[0])
        warp_img = cv2.warpPerspective(img, self.warp_matrix, shape, flags = cv2.INTER_LINEAR)
        return warp_img
```

github.com/mithi/little-miss-trendy/

```
def debug_skyview(self, img):
    warped_img = self.skyview(img)
    img = make_debug_image(warped_img, self.dpoints)
    return img

def debug_raw(self, img):
    return make_debug_image(img, self.spoints)

def show_raw_dotted(self, img):
    show_dotted_image(img, self.spoints)

def show_skyview_dotted(self, img):
    warped_img = self.skyview(img)
    show_dotted_image(warped_img, self.dpoints)
```

github.com/mithi/little-miss-trendy/

```
def make_debug_image(this_image, points, thickness = 1, color = [0, 0, 0], r = 6):

    image = this_image.copy()

    cv2.line(image, points[0], points[1], color, thickness)
    cv2.line(image, points[2], points[3], color, thickness)

    for point in points:
        cv2.circle(image, point, r, color, -1)

    return image
```

github.com/mithi/little-miss-trendy/

VIDEO: In practice

December 12, 2017 - 18 seconds

- https://www.youtube.com/watch?v=p1y09lydZEk

NEXT: EXCERPTS FROM HIS CODE

from my former classmate: github.com/WolfgangSteiner/RoboCar

- HE DID PERSPECTIVE TRANSFORM!

```
def calc_perspective_transform(self):
    c = self.warped_size[1] // 2
    wp = 280
    hp = 210
    dp = 300

    src_coords = np.array([[269,200],[386,200],[405,281],[248,281]], dtype=np.float32)
    #src_coords = np.array([[257,282],[399,282],[445,349],[209,349]], dtype=np.float32)
    dst_coords = np.array([[c-wp//2, dp+hp],[c+wp//2,dp+hp],[c+wp//2,dp],[c-wp//2,dp]], dtype=np.float32)

    self.M = cv2.getPerspectiveTransform(src_coords, dst_coords)


def warp_perspective(self, img):
    return cv2.warpPerspective(img, self.M, self.warped_size)
```

from my former classmate: github.com/WolfgangSteiner/RoboCar

RECALL: YAGNI (DTSTTCPW)

- **D**o **T**he **S**implest **T**hing **T**hat **C**ould **P**ossibly **W**ork

```
def set_throttle_steering(vel, steer):
    if abs(vel) < 0.1:
        left_vel = steer
        right_vel = -steer
    elif steer < 0:
        left_vel = vel * (1.0 - 0.75 * abs(steer))
        right_vel = vel
    else:
        left_vel = vel
        right_vel = vel * (1.0 - 0.75 * abs(steer))

    set_velocities(left_vel, right_vel)
```

from my former classmate: github.com/WolfgangSteiner/RoboCar

HE TRAINED A MODEL!

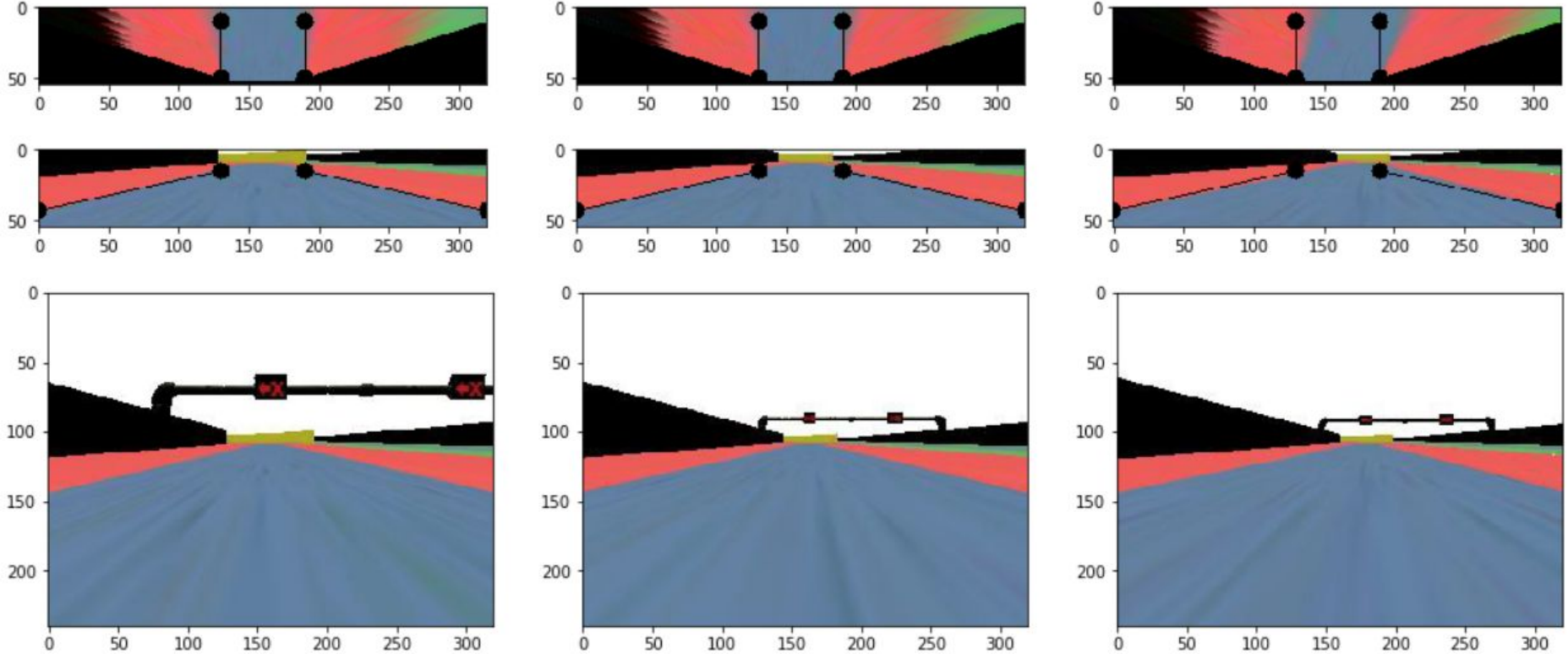
BEHAVIORAL CLONING!

```
    model.add(MaxPooling2D())
    input_shape = []

model.add(Flatten())

for n in dense:
    if n == 0:
        continue

    if dense_dropout > 0.0:
        click.echo(f"Dropout {dense_dropout}")
        model.add(Dropout(dense_dropout))

    click.echo(f"Dense {n}")
    model.add(Dense(n, kernel_regularizer=regularizer(dense_regularization)))
    model.add(get_activation(activation))

model.add(Dense(1))

click.echo("Compiling model...")
model.compile(optimizer=Adam(lr=learning_rate), loss="mse", metrics=[metrics.mse])
```

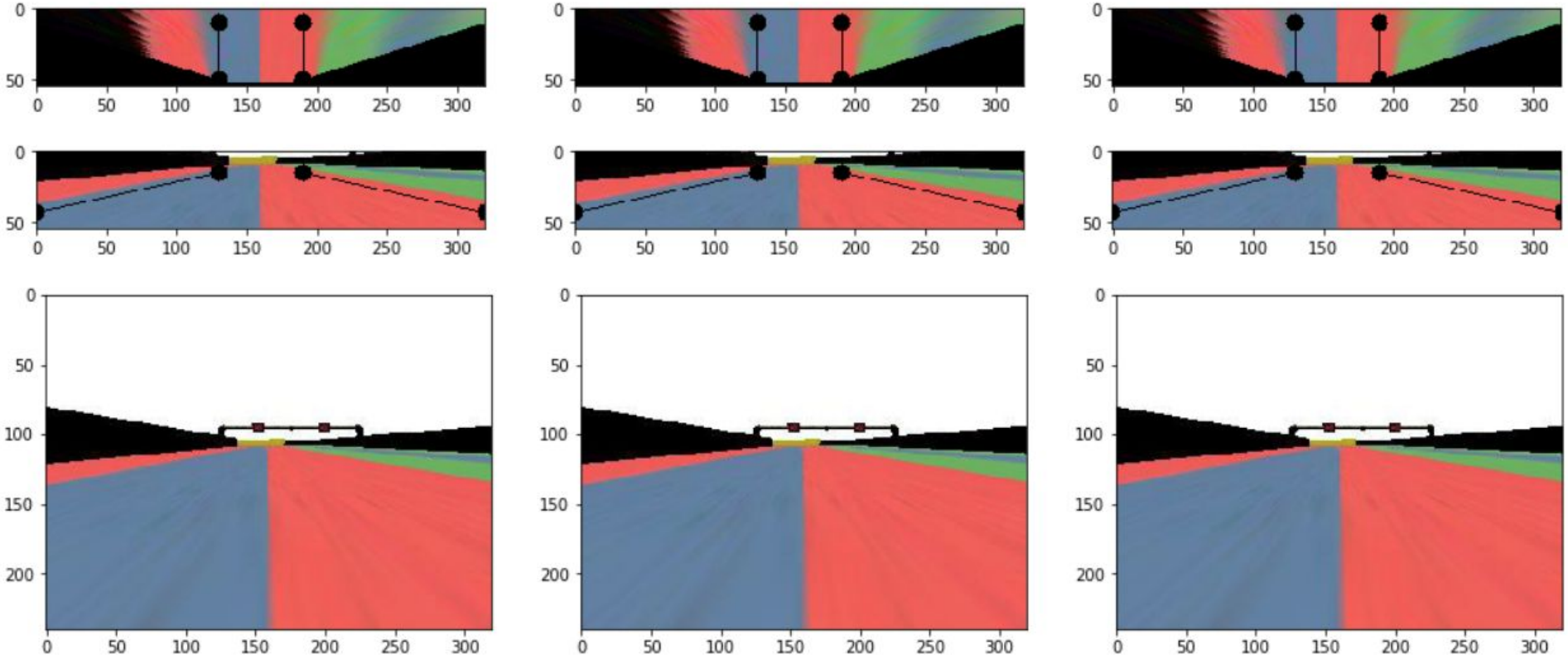
from my former classmate: github.com/WolfgangSteiner/RoboCar

CAMERA POSITION MATTERS

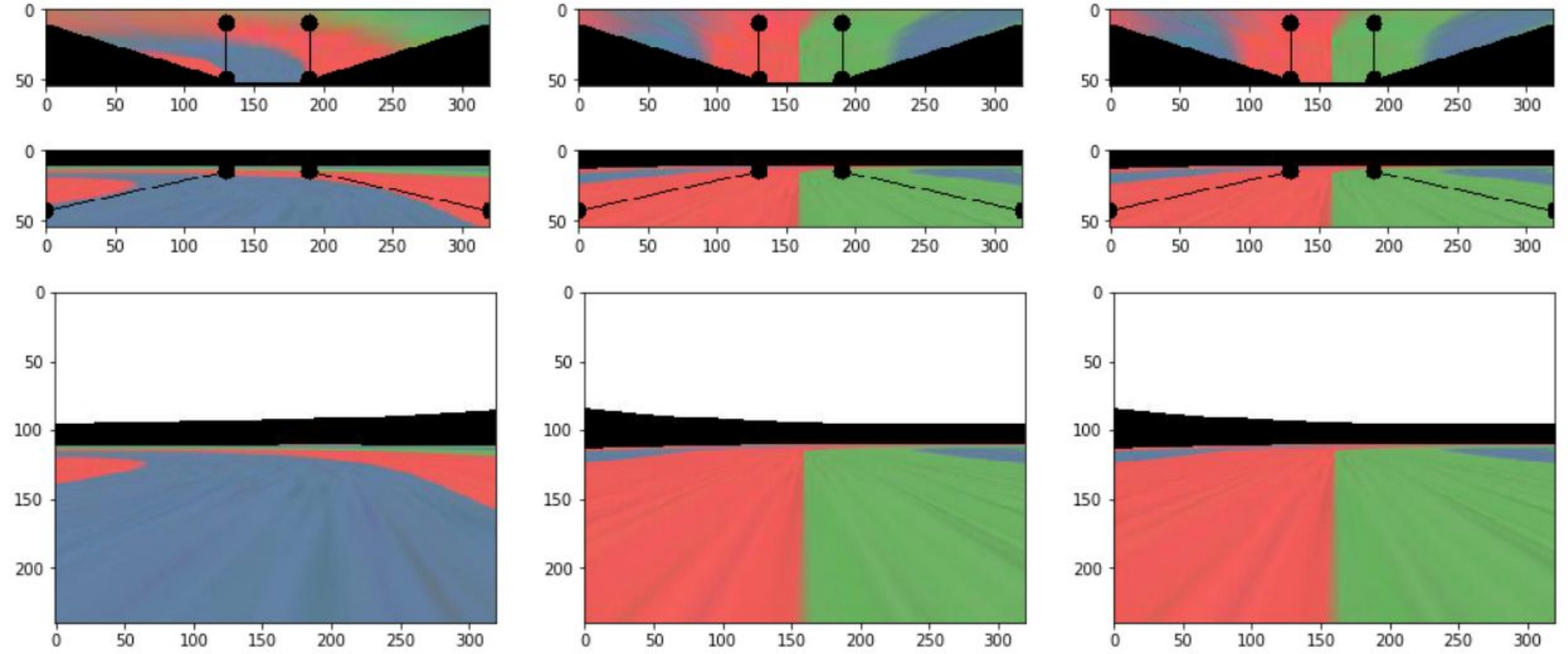
INSPECT YOUR DATA, IS IT USEFUL?
QUALITY IS BETTER THAN QUANTITY



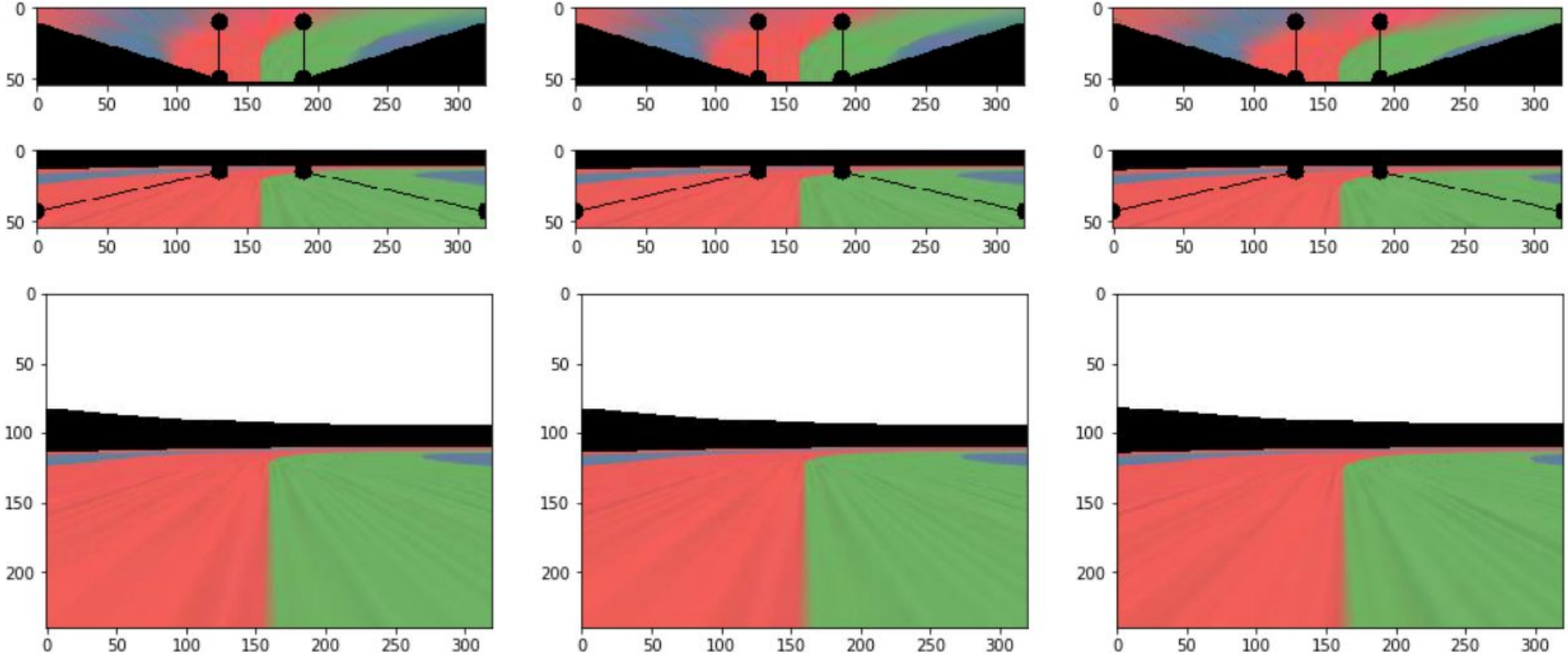
github.com/mithi/little-miss-trendy/



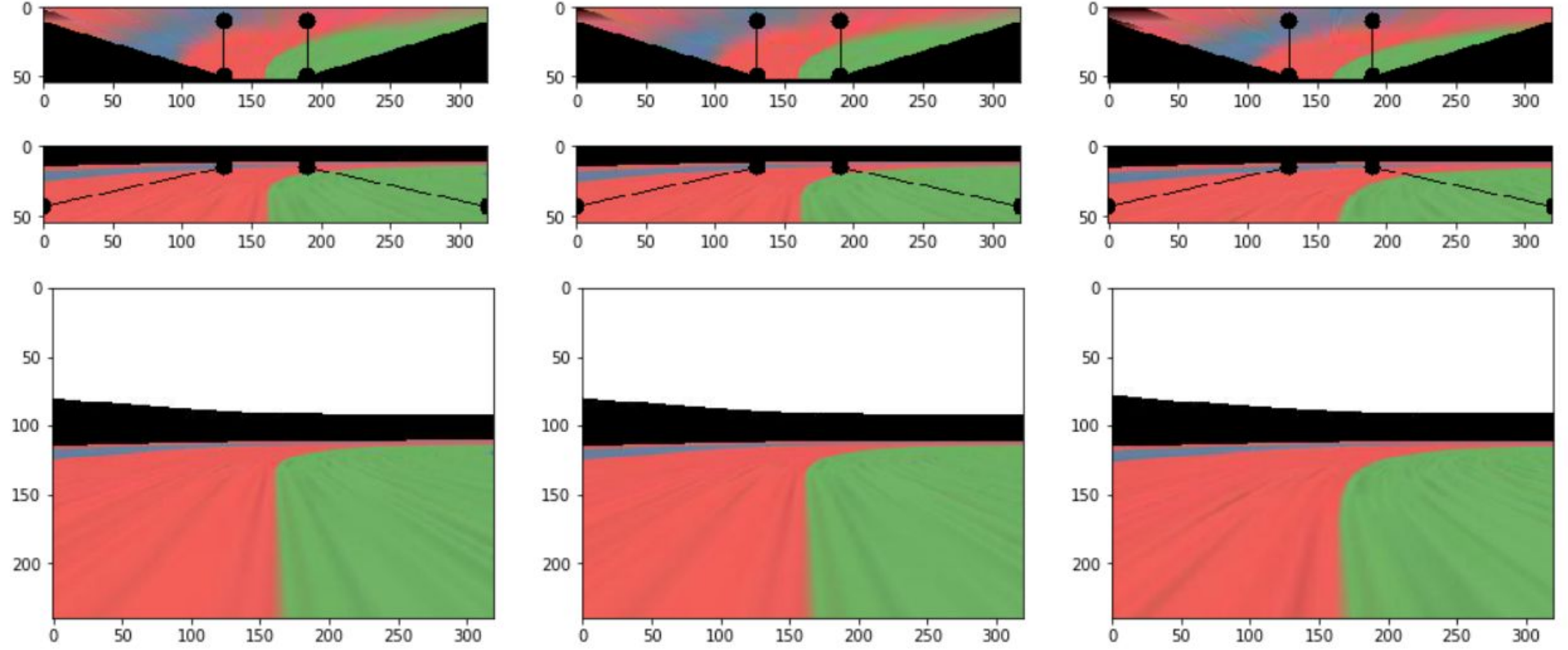
github.com/mithi/little-miss-trendy/



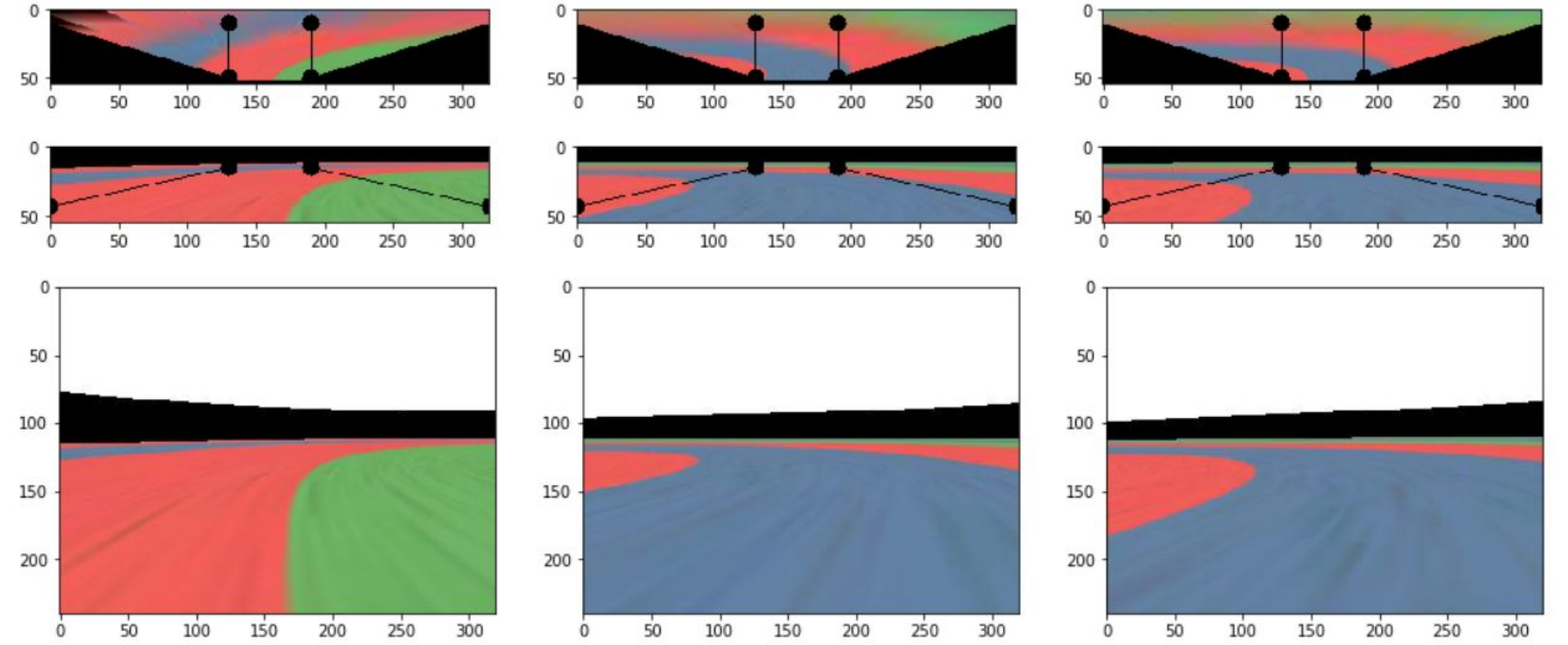
github.com/mithi/little-miss-trendy/



github.com/mithi/little-miss-trendy/



github.com/mithi/little-miss-trendy/



github.com/mithi/little-miss-trendy/

```
# (x, y) lowerleft, upperleft, upperright, lowerright
source_points = [(0, 43), (130, 15), (190, 15), (320, 43)]
dest_points = [(130, 50), (130, 10), (190, 10), (190, 50)]
birds_eye = BirdsEye(source_points, dest_points)
```

```
birds_eye.skyview(image)
birds_eye.debug_skyview(image)
birds_eye.debug_raw(image)
birds_eye.debug_raw_dotted(image)
birds_eye.debug_skyview_dotted(image)
```

github.com/mithi/little-miss-trendy/

RESEARCH PAPER 2

End to End Learning for Self-Driving Cars

NVIDIA Corporation
Holmdel, NJ 07735

arXiv:1604.07316v1 [cs.CV] 25 Apr 2016

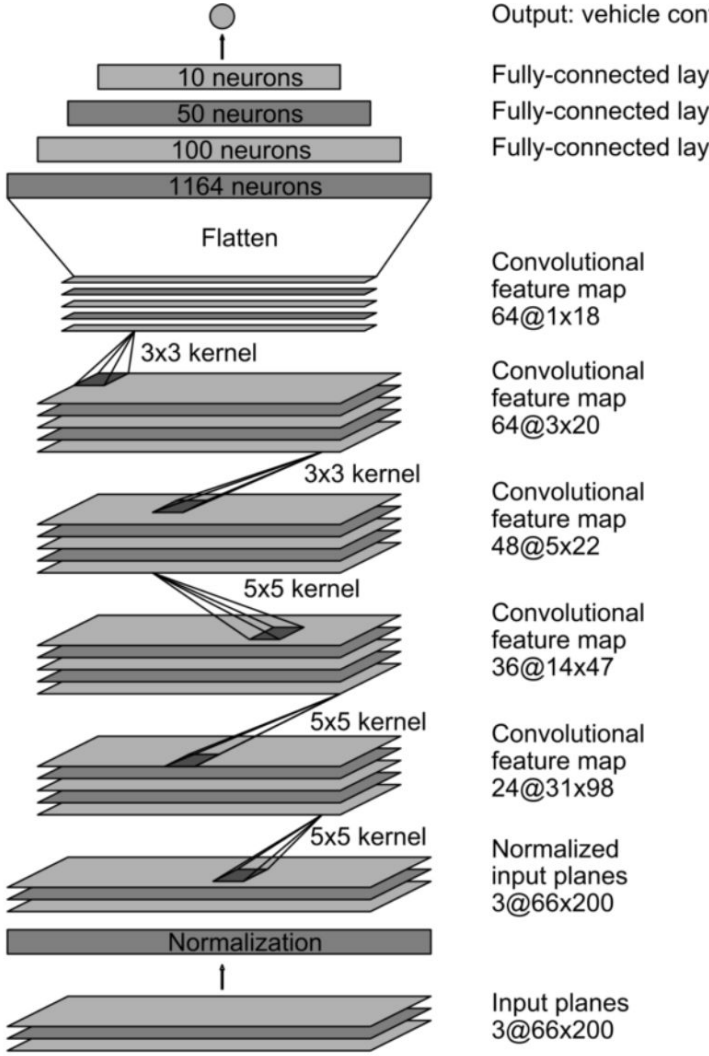
FOCUS ON

- **NETWORK ARCHITECTURE**
- **DATA SELECTION**

NETWORK ARCHITECTURE

- 9 layers,
- 1 normalization layer
- 5 convolutional layers
- 3 fully connected layers.

From:
https://devblogs.nvidia.com/deep-learning-self-driving-cars/



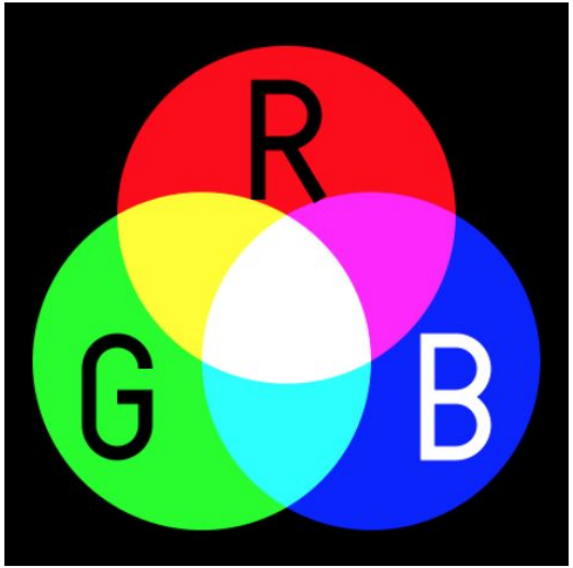
```
def build_modified_nvidia_model():
    model = Sequential()
    model.add(Lambda(lambda x: x/127.5-1.0, input_shape=INPUT_SHAPE))
    model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(36, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(48, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(100, activation='elu'))
    model.add(Dense(50, activation='elu'))
    model.add(Dense(10, activation='elu'))
    model.add(Dense(1))
    model.compile(optimizer = OPTIMIZER_TYPE, loss = LOSS_TYPE)
    return model
```

NETWORK ARCHITECTURE

- **Input image is split into YUV plane**
  - (WHY YUV instead of RGB? Grayscale? No explanation, why 3 channels not 1 channel? no explanation)
  - Something you can experiment with! Color representation!

¯_(ツ)_/¯

COLOR REPRESENTATION



Additive color mixing: adding red to green yields yellow; adding red to blue yields magenta; adding green to blue yields cyan; adding all three primary colors together yields white.

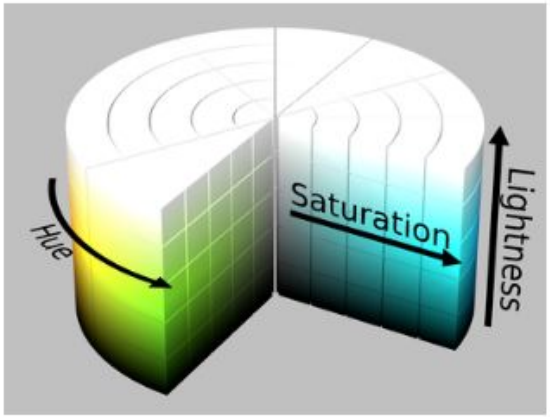


Fig. 2a. HSL cylinder.

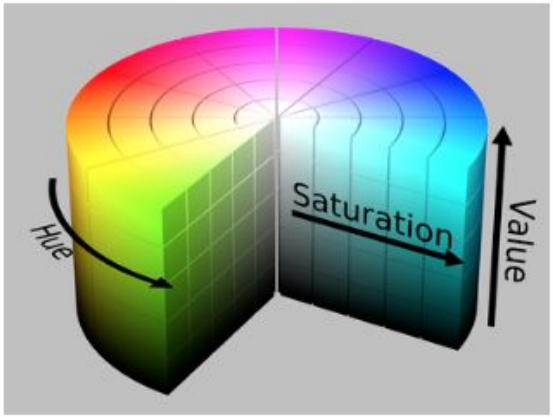


Fig. 2b. HSV cylinder.

COLOR REPRESENTATION

Why do we use the HSV colour space so often in vision and image processing?

54

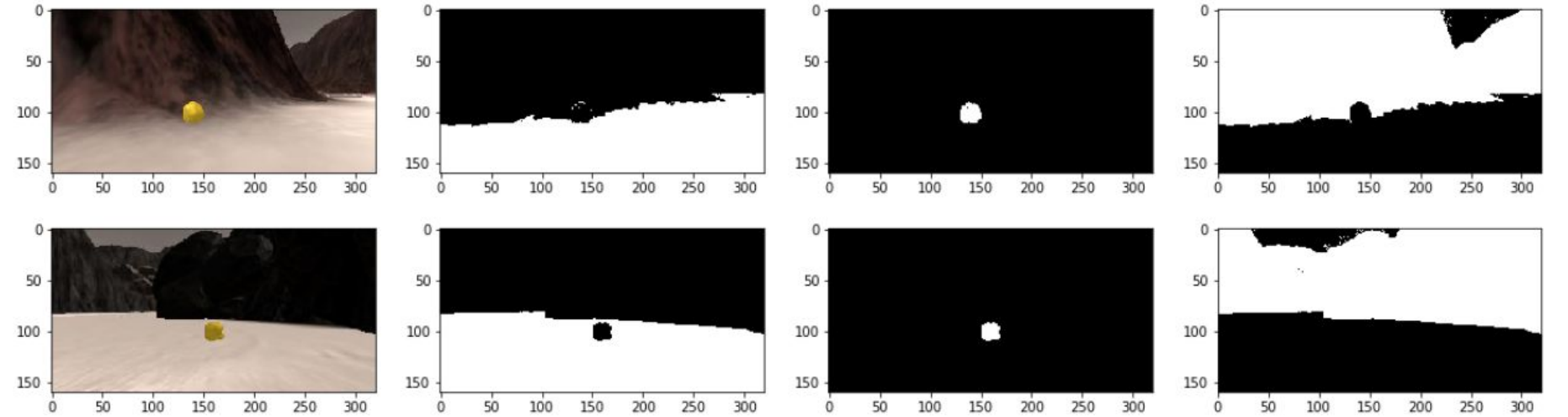
I see the HSV colour space used all over the place: for tracking, human detection, etc... I'm wondering, why? What is it about this colour space that makes it better than using RGB?

image-processing computer-vision

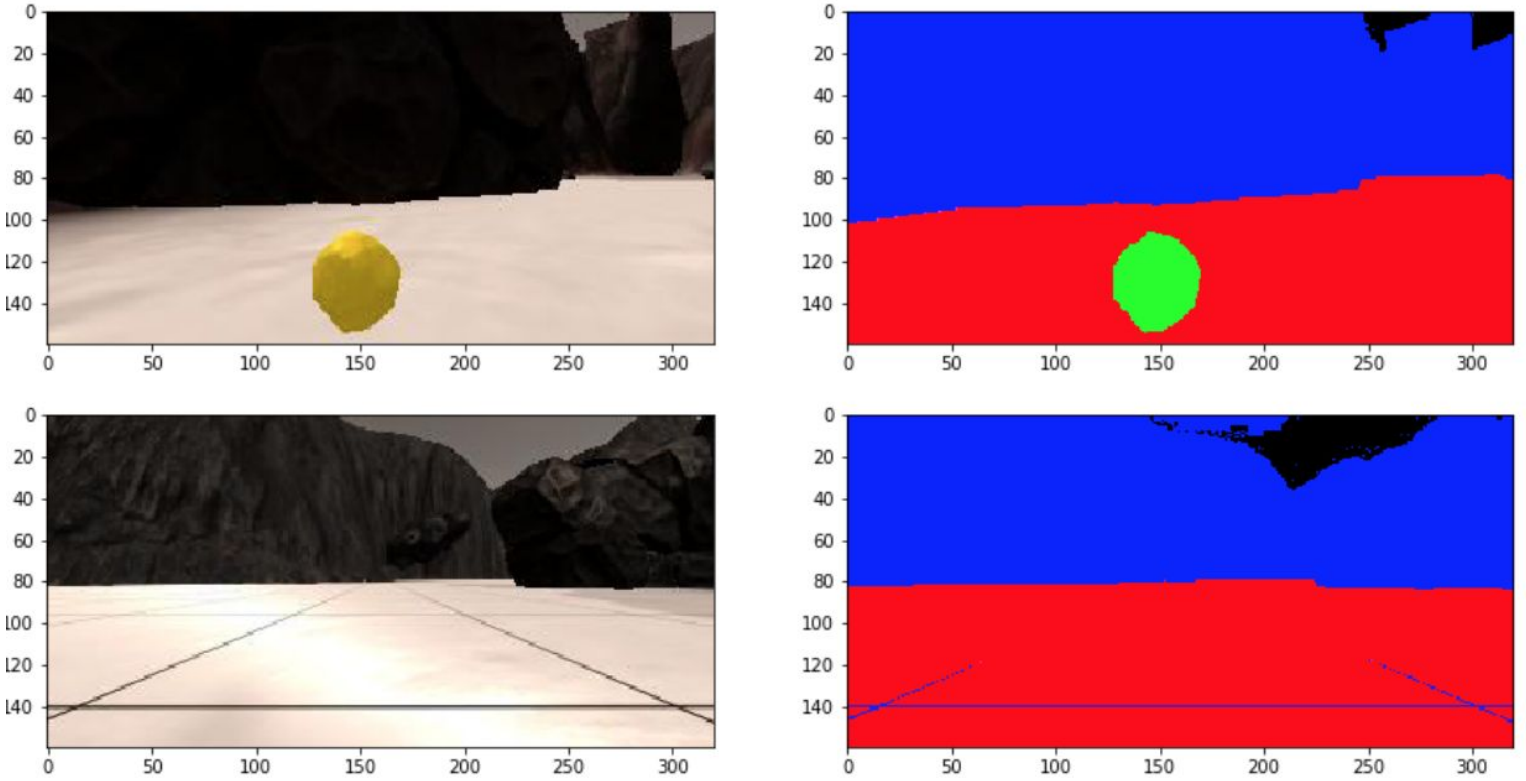
The simple answer is that unlike RGB, HSV separates *luma*, or the image intensity, from *chroma* or the color information. This is very useful in many applications. For example, if you want to do histogram equalization of a color image, you probably want to do that only on the intensity component, and leave the color components alone. Otherwise you will get very strange colors.

In computer vision you often want to separate color components from intensity for various reasons, such as robustness to lighting changes, or removing shadows.

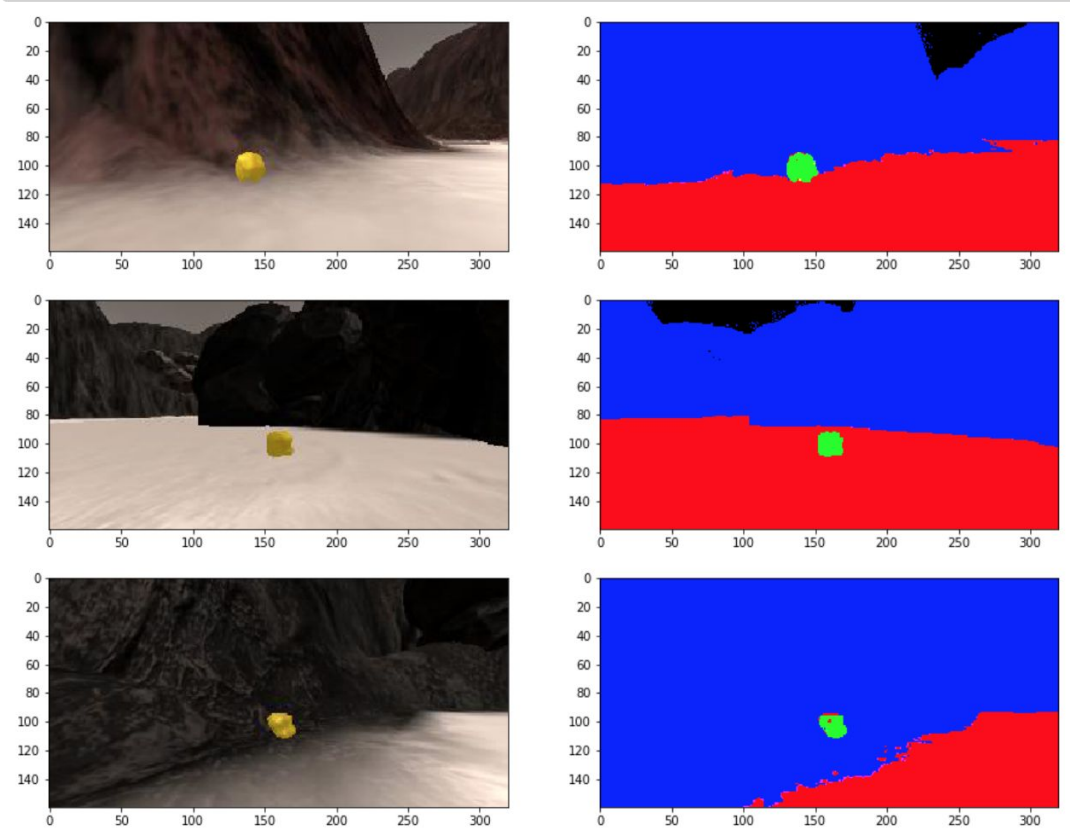
COLOR REPRESENTATION



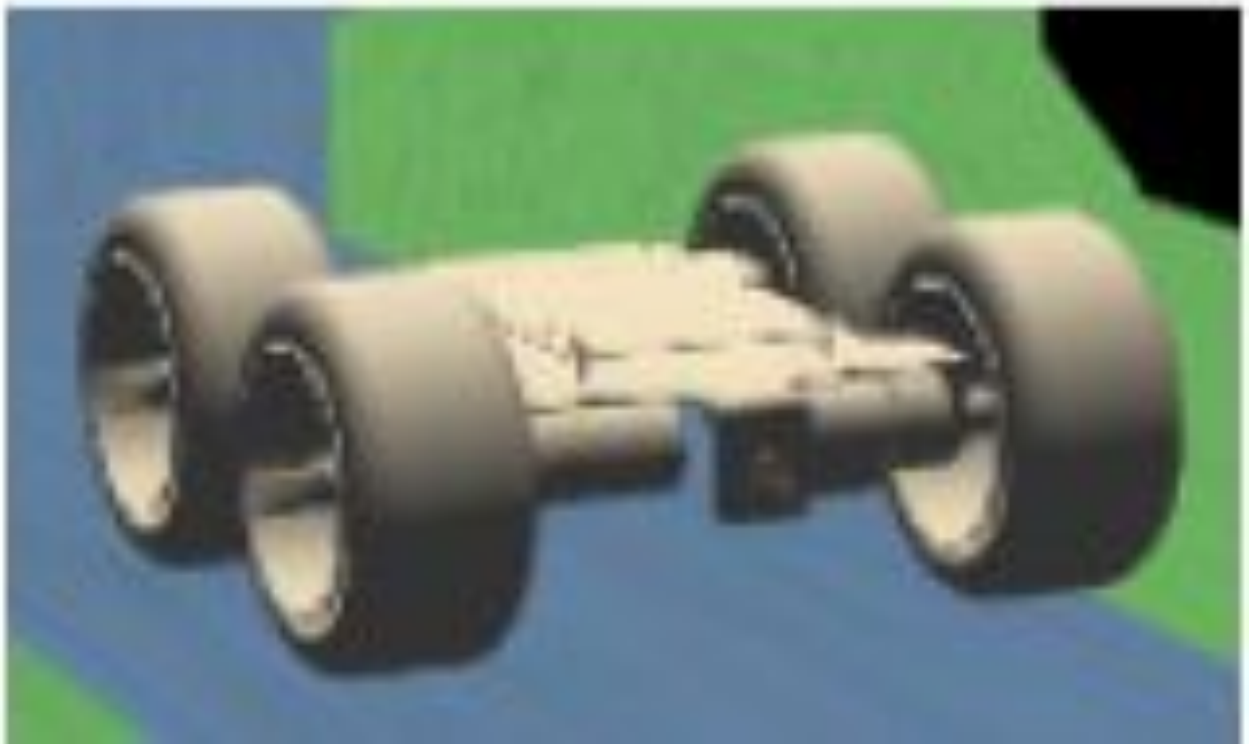
COLOR REPRESENTATION



COLOR REPRESENTATION



HACK! J/K



COLOR REPRESENTATION

```
def filter_hls(img, thresh_min, thresh_max, height = None):

  hls_img = img.copy() # make a copy
  cv2.cvtColor(hls_img, cv2.COLOR_BGR2HLS) # convert image to hls

  # a matrix of zeros same xy size as img, but a single channel
  binary_img = np.zeros_like(img[:,:,0])


  # Require that each pixel be above all three threshold values in HLS
  # within_thresh will now contain a boolean array with "True" where threshold was met
  within_thresh = (hls_img[:,:,0] >= thresh_min[0]) & (hls_img[:,:,0] <= thresh_max[0]) & \
                  (hls_img[:,:,1] >= thresh_min[1]) & (hls_img[:,:,1] <= thresh_max[1]) & \
                  (hls_img[:,:,2] >= thresh_min[2]) & (hls_img[:,:,2] <= thresh_max[2])

  # Index the array of zeros with the boolean array and set to 1
  binary_img[within_thresh] = 1

  if height is not None:
    binary_img[:height, :] = 0

  return binary_img
```

COLOR REPRESENTATION

- To detect `rocks`, navigable terrain (`ground`), and obstacles (`blocked`)

:
```
ground_thresh_min = (0, 100, 70)
ground_thresh_max = (255, 255, 255)

rock_thresh_min = (0, 100, 0)
rock_thresh_max = (255, 255, 70)

blocked_thresh_min = (0, 0, 0)
blocked_thresh_max = (255, 100, 255)
```

COLOR REPRESENTATION

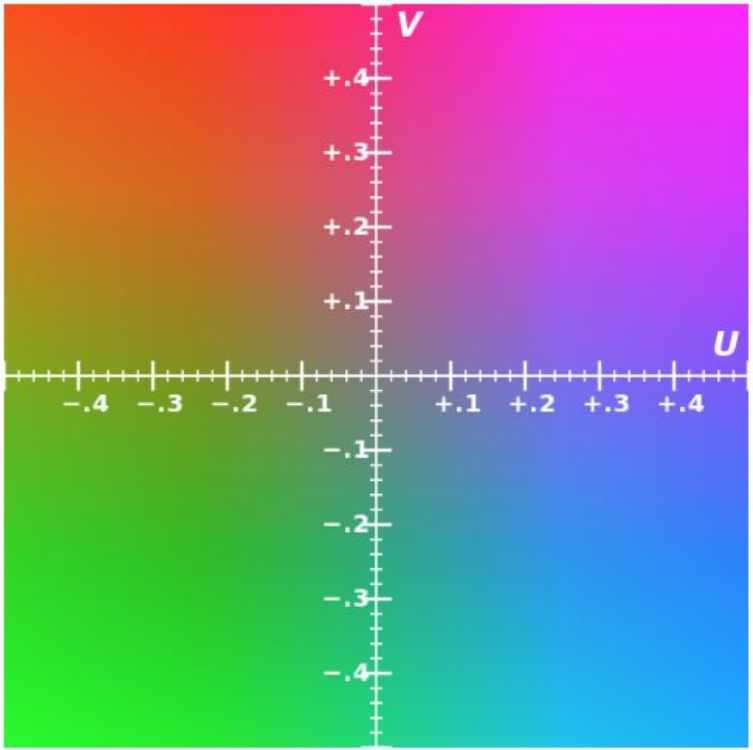
The **YUV** color model represents the human perception of color more closely than the standard RGB model used in computer graphics hardware. In **YUV**, Y is the luminance (brightness) component while U and V are the chrominance (color) components.

COLOR REPRESENTATION

Is YUV better than RGB?

There is some debate over which format is **better** for frame grabbers: **YUV or RGB**. ... **YUV** color-spaces are a more efficient coding and reduce the bandwidth more **than RGB** capture can. Most video cards, therefore, render directly using **YUV or** luminance/chrominance images.

COLOR REPRESENTATION

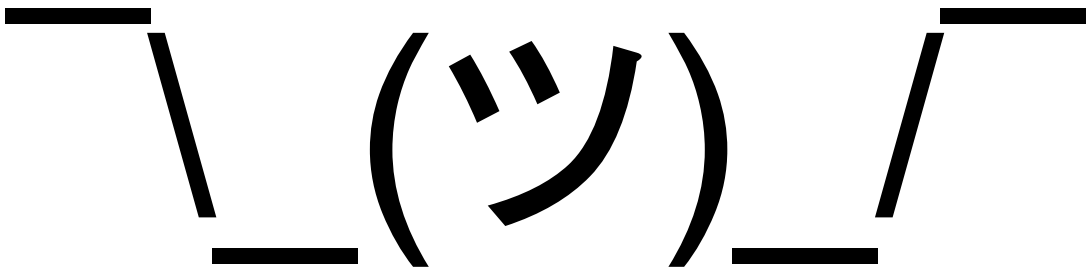


COLOR REPRESENTATION (U and V)

COLOR REPRESENTATION (Y)

NETWORK ARCHITECTURE

- **Trained the weights of our network to minimize the mean-squared error between the steering command output by the network,**
- *(cool MSE! Makes sense. but which optimizer used? Not explained)*
- You can also experiment with this but adam is probably good enough



NETWORK ARCHITECTURE

- The first layer - image normalization.
- normalizer is hard-coded
- not adjusted in the learning process.
- Performing normalization in the network allows to be accelerated via GPU processing.

NETWORK ARCHITECTURE

- - The convolutional layers are designed to perform feature extraction, and are **chosen empirically through a series of experiments** that vary layer configurations. We then use strided convolutions in the first three convolutional layers with a 2×2 stride(subsample) and a 5×5 kernel (filter), and a non-strided convolution with a 3×3 kernel size in the final two convolutional layers.
-
-

```
def build_modified_nvidia_model():
    model = Sequential()
    model.add(Lambda(lambda x: x/127.5-1.0, input_shape=INPUT_SHAPE))
    model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(36, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(48, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(100, activation='elu'))
    model.add(Dense(50, activation='elu'))
    model.add(Dense(10, activation='elu'))
    model.add(Dense(1))
    model.compile(optimizer = OPTIMIZER_TYPE, loss = LOSS_TYPE)
    return model
```

HIDDEN LAYER VISUALIZATION

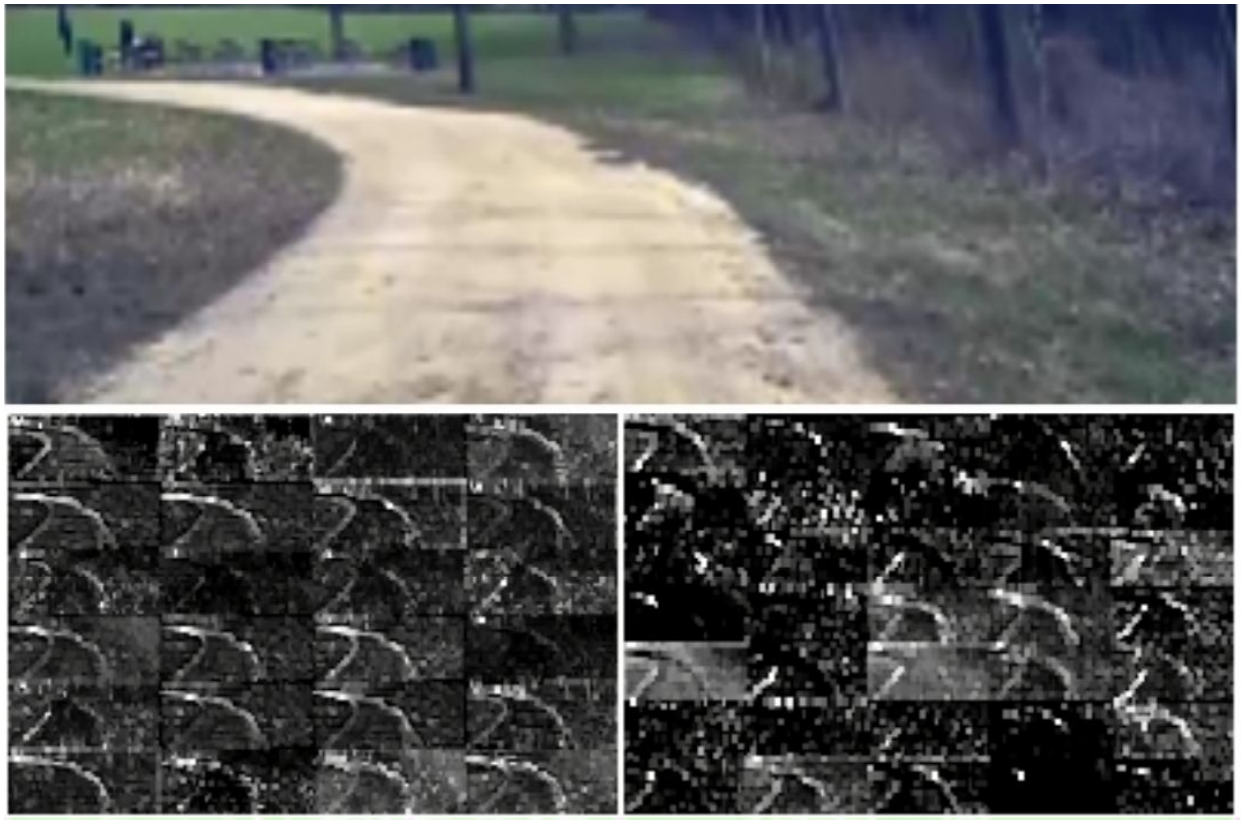
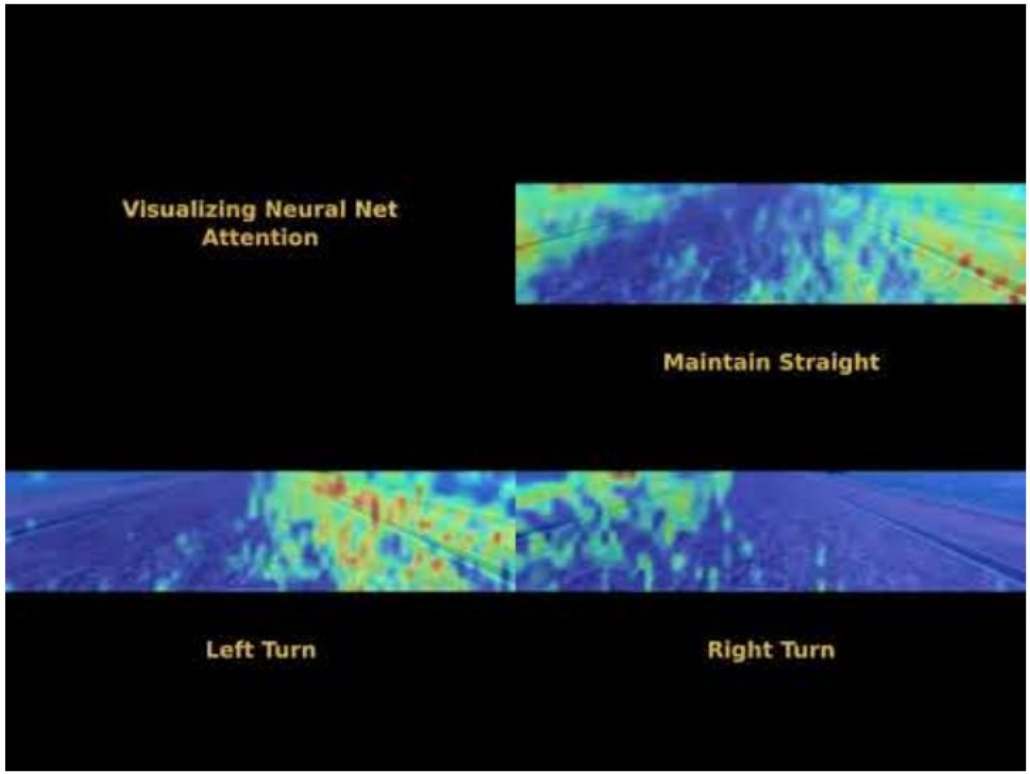


Figure 7: How the CNN "sees" an unpaved road. Top: subset of the camera image sent to the CNN. Bottom left: Activation of the first layer feature maps. Bottom right: Activation of the second layer feature maps. This demonstrates that the CNN learned to detect useful road features on its own, i. e., with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads.

LAYER VISUALIZATION

SEE HIS CODE
github.com/andrewsommerlot/Carnd-Behavioral-Cloning/blob/master/visualize_attention.py



DATA SELECTION

INSPECT YOUR DATA, IS IT USEFUL?
QUALITY IS BETTER THAN QUANTITY

DATA SELECTION

- The first step to training a neural network is selecting the frames to use.
- **sample that video at 10 FPS** because a higher sampling rate would include images that are highly similar, and thus not provide much additional useful information.

TRUE STORY: 2 approaches same architecture

Not so good: (epochs = 10?)

- About **37k** not so good data **(301 MB)**
- **120x320x3** = **115k** features (cropped)
- Model size: 22MB (wobbly performance)

Better: (epochs = 5)

- About **9k** relatively good data **(73.3 MB)**
- **200x75x3** = **45k** features (cropped, resized
- Model size: **5 MB** (smooth performance)

Note: Maybe we can even use just 4k data, and resize it to even smaller!!!

ORIGINAL

Layer (type)	Output Shape	Param #	Connected to
lambda_1 (Lambda)	(None, 66, 200, 3)	0	lambda_input_1[0][0]
convolution2d_1 (Convolution2D)	(None, 31, 98, 24)	1824	lambda_1[0][0]
convolution2d_2 (Convolution2D)	(None, 14, 47, 36)	21636	convolution2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 5, 22, 48)	43248	convolution2d_2[0][0]
convolution2d_4 (Convolution2D)	(None, 3, 20, 64)	27712	convolution2d_3[0][0]
convolution2d_5 (Convolution2D)	(None, 1, 18, 64)	36928	convolution2d_4[0][0]
dropout_1 (Dropout)	(None, 1, 18, 64)	0	convolution2d_5[0][0]
flatten_1 (Flatten)	(None, 1152)	0	dropout_1[0][0]
dense_1 (Dense)	(None, 100)	115300	flatten_1[0][0]
dense_2 (Dense)	(None, 50)	5050	dense_1[0][0]
dense_3 (Dense)	(None, 10)	510	dense_2[0][0]
dense_4 (Dense)	(None, 1)	11	dense_3[0][0]

Total params: 252,219

NOT SO GOOD

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 120, 320, 3)	0
conv2d_1 (Conv2D)	(None, 58, 158, 24)	1824
conv2d_2 (Conv2D)	(None, 27, 77, 36)	21636
conv2d_3 (Conv2D)	(None, 12, 37, 48)	43248
conv2d_4 (Conv2D)	(None, 10, 35, 64)	27712
conv2d_5 (Conv2D)	(None, 8, 33, 64)	36928
dropout_1 (Dropout)	(None, 8, 33, 64)	0
flatten_1 (Flatten)	(None, 16896)	0
dense_1 (Dense)	(None, 100)	1689700
dense_2 (Dense)	(None, 50)	5050
dense_3 (Dense)	(None, 10)	510
dense_4 (Dense)	(None, 1)	11

Total params: 1,826,619
Trainable params: 1,826,619

A LITTLE BETTER

Layer (type)	Output Shape	Param #	Connected to
lambda_1 (Lambda)	(None, 75, 200, 3)	0	lambda_input_1[0][0]
convolution2d_1 (Convolution2D)	(None, 36, 98, 24)	1824	lambda_1[0][0]
convolution2d_2 (Convolution2D)	(None, 16, 47, 36)	21636	convolution2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 6, 22, 48)	43248	convolution2d_2[0][0]
convolution2d_4 (Convolution2D)	(None, 4, 20, 64)	27712	convolution2d_3[0][0]
convolution2d_5 (Convolution2D)	(None, 2, 18, 64)	36928	convolution2d_4[0][0]
dropout_1 (Dropout)	(None, 2, 18, 64)	0	convolution2d_5[0][0]
flatten_1 (Flatten)	(None, 2304)	0	dropout_1[0][0]
dense_1 (Dense)	(None, 100)	230500	flatten_1[0][0]
dense_2 (Dense)	(None, 50)	5050	dense_1[0][0]
dense_3 (Dense)	(None, 10)	510	dense_2[0][0]
dense_4 (Dense)	(None, 1)	11	dense_3[0][0]

Total params: 367,419

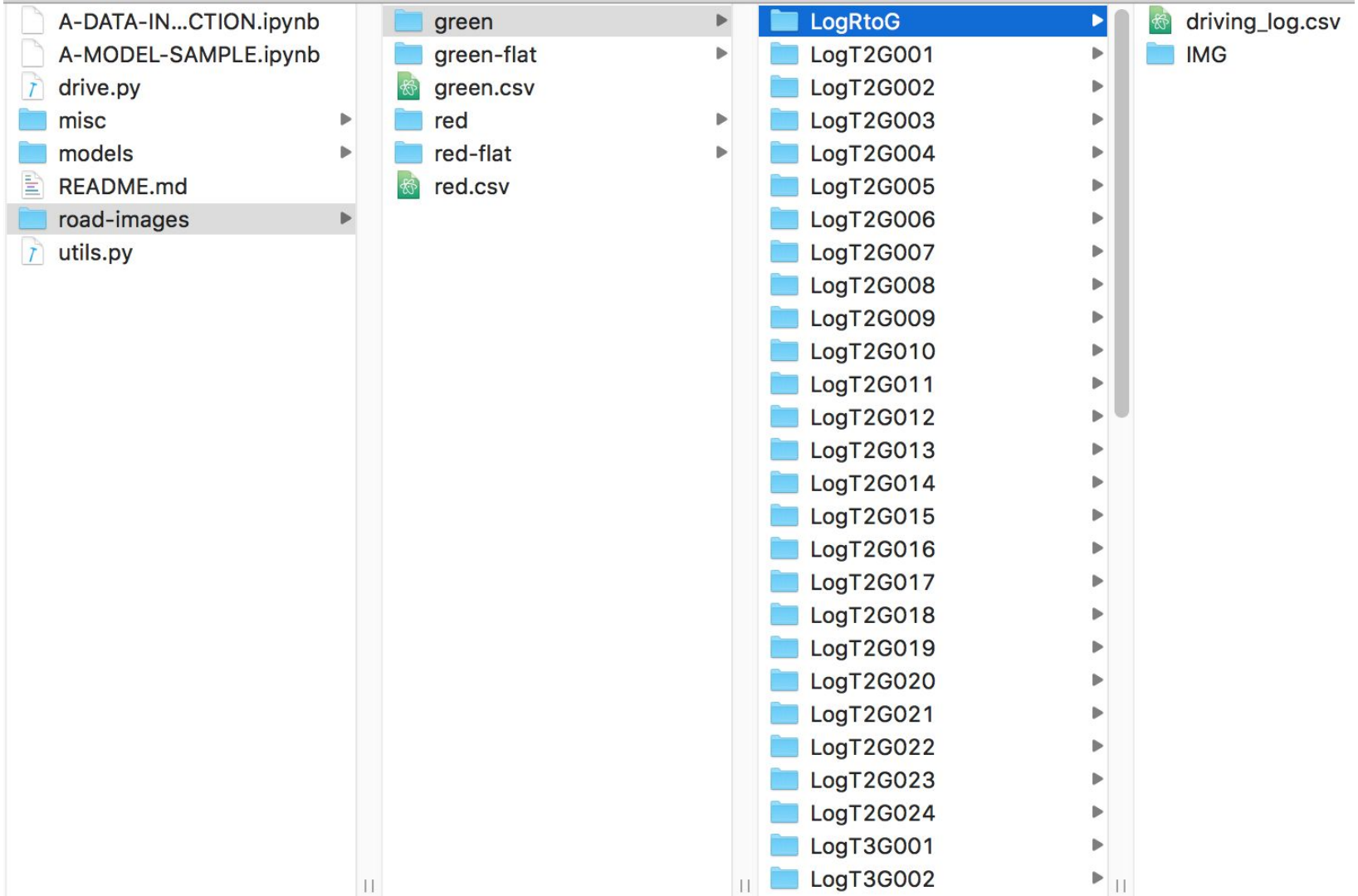
DATA SELECTION

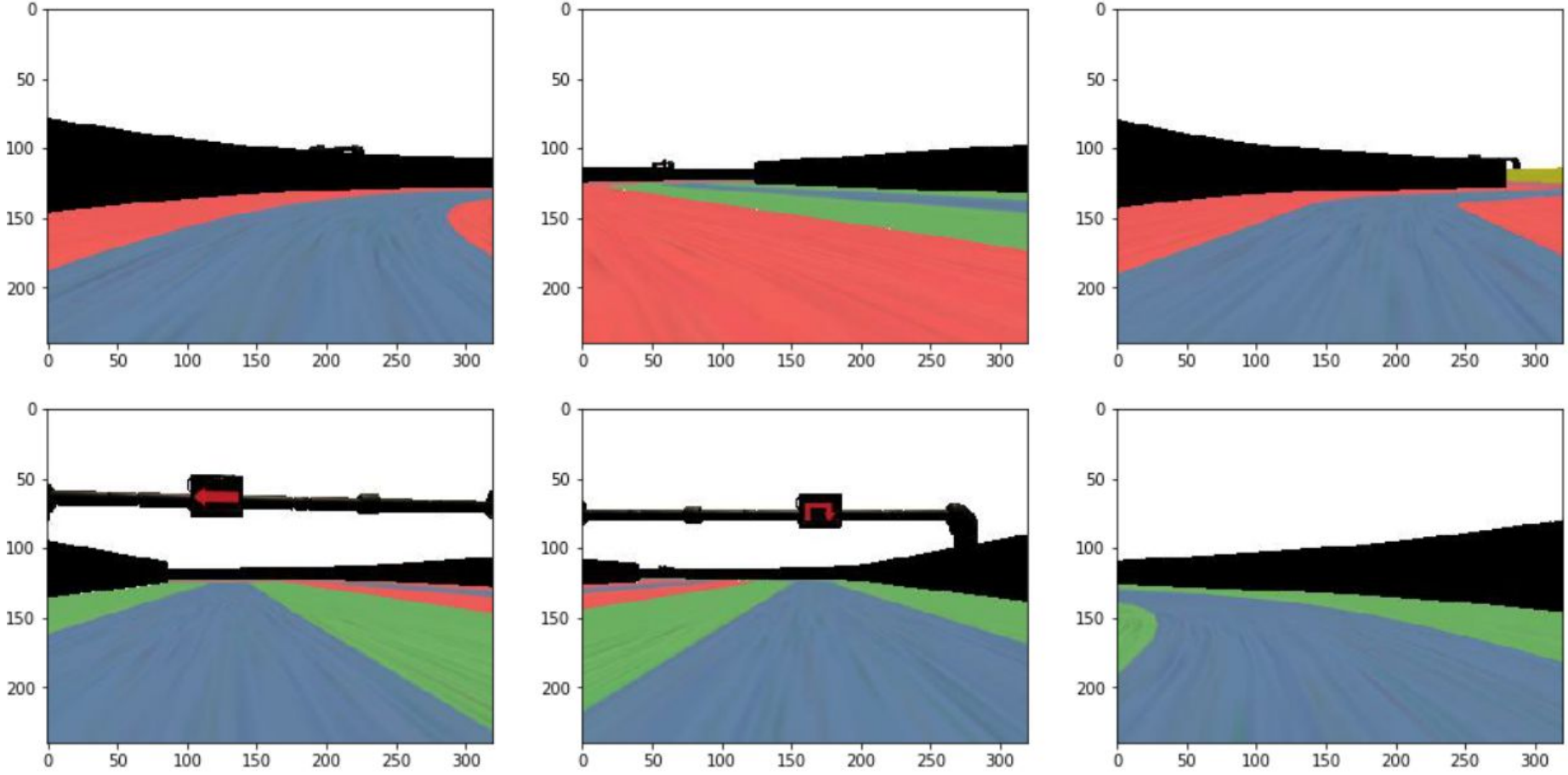
- The first step to training a neural network is selecting the frames to use.
- Sample that video at **10 FPS because a higher sampling rate would include images that are highly similar, and thus not provide much additional useful information**.

DATA SELECTION

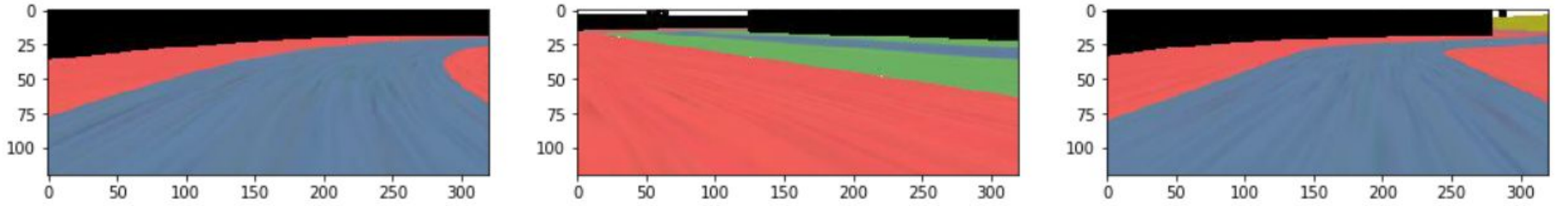
- We simply select data where the driver is staying in a lane, and discard the rest.

- To remove a bias towards driving straight the training data includes a **higher proportion of frames that represent road curves.**

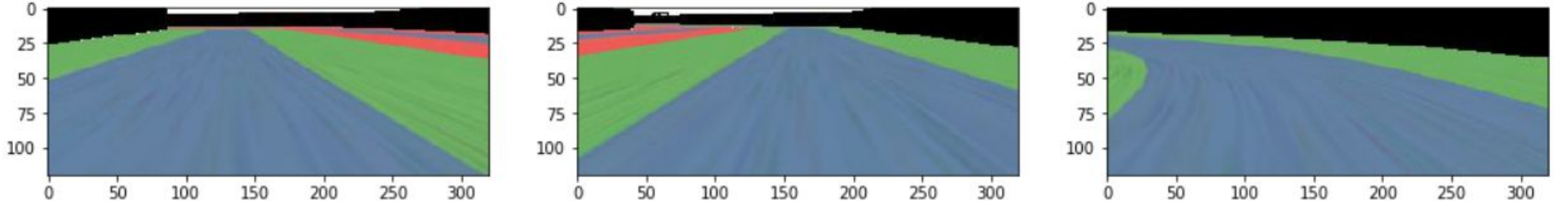

A-DATA-IN...CTION.ipynb
A-MODEL-SAMPLE.ipynb
drive.py
misc
models
README.md
road-images
utils.py
green
green-flat
green.csv
red
red-flat
red.csv
LogRtoG
LogT2G001
LogT2G002
LogT2G003
LogT2G004
LogT2G005
LogT2G006
LogT2G007
LogT2G008
LogT2G009
LogT2G010
LogT2G011
LogT2G012
LogT2G013
LogT2G014
LogT2G015
LogT2G016
LogT2G017
LogT2G018
LogT2G019
LogT2G020
LogT2G021
LogT2G022
LogT2G023
LogT2G024
LogT3G001
LogT3G002
driving_log.csv
IMG


0
50
100
150
200
0
50
100
150
200
250
300

```
x, y, z = process_img(red1), process_img(red2), process_img(red3)
plot_three_images(x, y, z, w =17, h=5)
```




```
x, y, z = process_img(green1), process_img(green2), process_img(green3)
plot_three_images(x, y, z, w =17, h=5)
```

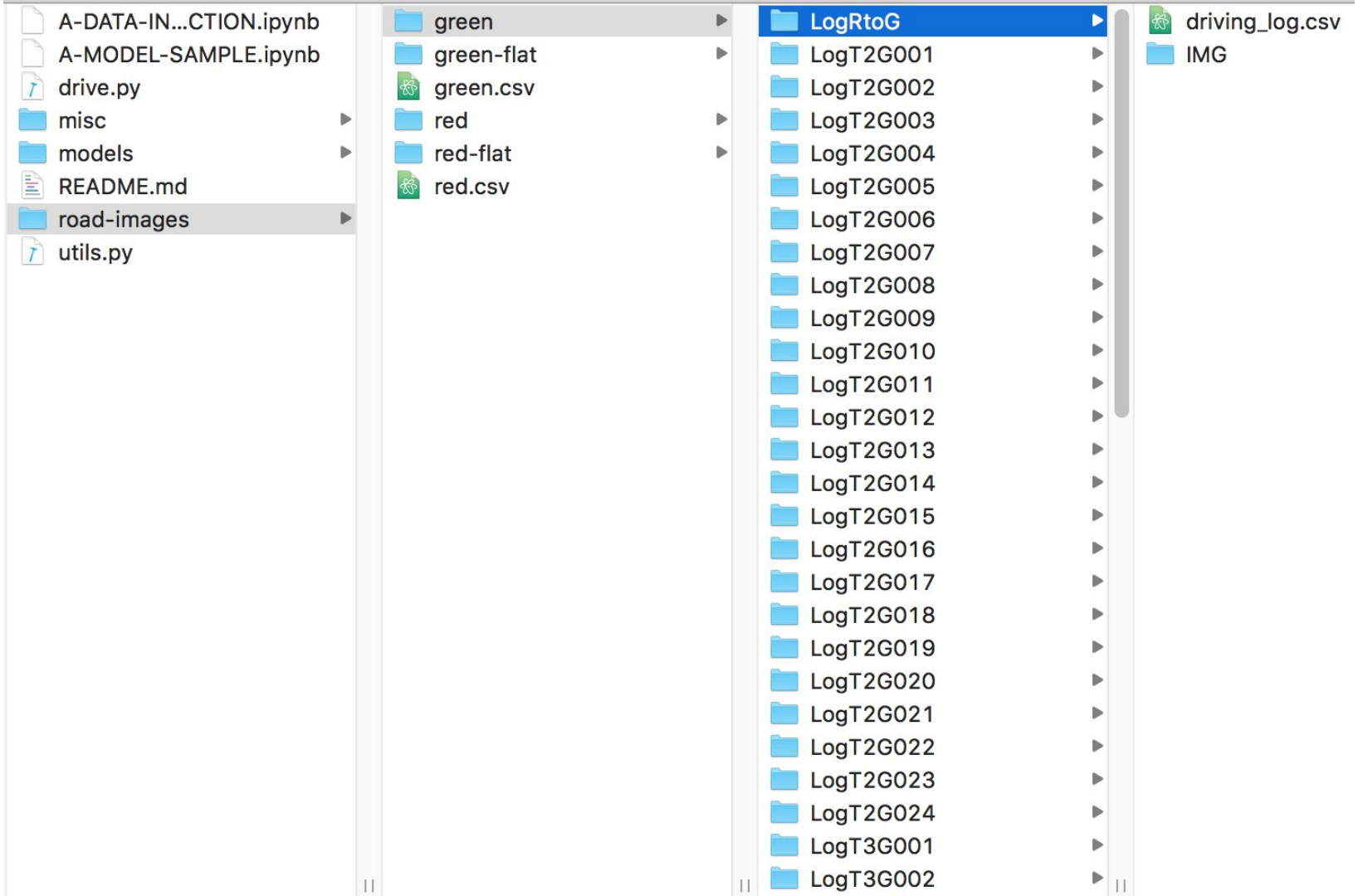


- Run to put all images from subdirectories to one directory

```
$ find ./road-images/red/ -name '*.jpg' -exec cp '{}' ./road-images/red-flat/ \;
$ find ./road-images/green/ -name '*.jpg' -exec cp '{}' ./road-images/green-flat/ \;
```

- Run script to merge all log csv files to one csv file

```
$ ruby ./misc/process-merge-csvs.rb ./PATH/TO/CSV/LOGS /PATH/TO/IMAGES /PATH/TO/FINAL/CSV/
$ ruby ./misc/process-merge-csvs.rb ./road-images/green/ ./road-images/green-flat/ ./road-images/green.csv
$ ruby ./misc/process-merge-csvs.rb ./road-images/red/ ./road-images/red-flat/ ./road-images/red.csv
```


A-DATA-IN...CTION.ipynb
A-MODEL-SAMPLE.ipynb
drive.py
misc
models
README.md
road-images
utils.py
green
green-flat
green.csv
red
red-flat
red.csv
LogRtoG
LogT2G001
LogT2G002
LogT2G003
LogT2G004
LogT2G005
LogT2G006
LogT2G007
LogT2G008
LogT2G009
LogT2G010
LogT2G011
LogT2G012
LogT2G013
LogT2G014
LogT2G015
LogT2G016
LogT2G017
LogT2G018
LogT2G019
LogT2G020
LogT2G021
LogT2G022
LogT2G023
LogT2G024
LogT3G001
LogT3G002
driving_log.csv
IMG

ving_log.csv

Open with Google Sheets

	B	C	D	E	F	G
g/Downl	0	0	0	2.11E-06	394.462	1
g/Downl	0	0.1997162	0	0.0902059	394.522	1
g/Downl	0	0.3994621	0	0.1614634	394.582	1
g/Downl	0	0.600072	0	0.2404336	394.662	1
g/Downl	0	0.8001541	0	0.3062334	394.722	1
g/Downl	0	1	0	0.3725905	394.782	1
g/Downl	0	1	0	0.4707986	394.862	1
g/Downl	0	0.800287	0	0.5354444	394.922	1
g/Downl	0	0.600481	0	0.5992693	394.982	1
g/Downl	0	0.4005671	0	0.6766577	395.061	1

```ruby
require 'csv'
require 'find'

if ARGV.size != 3
  puts "Requires PARENT_DIR, IMAGE_DIR, and OUTPUT_FILE."
  exit
end

csv_file_paths = []

# Parent directory where CSVs are stored
PARENT_DIR=ARGV[0]

# Directory where images are stored
IMG_DIR=ARGV[1]

# Full path to the output CSV file
OUTPUT_FILE=ARGV[2]

Find.find(PARENT_DIR) do |path|
  csv_file_paths << path if path =~ /.*\.csv$/
end

CSV.open(OUTPUT_FILE, "w") do |csv|

  csv << ["NAME", "STEER"]

  csv_file_paths.each do |path|
    CSV.foreach(path) do |row|
      # Requirements:
      # 1. Get first column which contains a path, and only keep filename with extension.
      # 2. Get second column as it is.
      # 3. Ignore the rest of the data.
      filenamepath = IMG_DIR + File.basename(row[0])
      steering = row[1]

      csv << [filenamepath, steering]
    end
  end
end
```

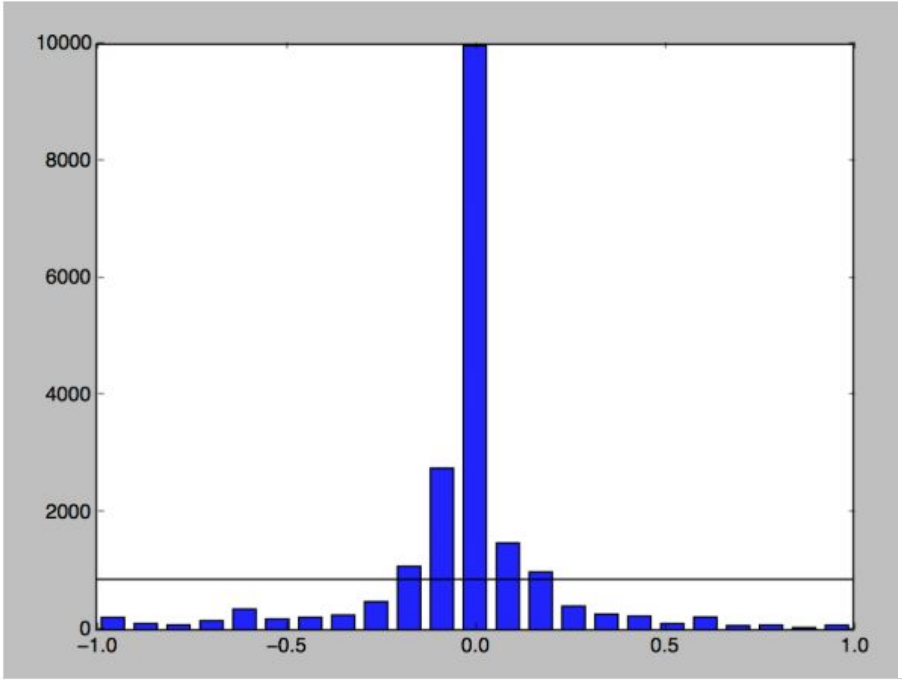
28	./road-images/green-	6
29	./road-images/green-	14
30	./road-images/green-	16.01735
31	./road-images/green-	8.005071
32	./road-images/green-	0.0003433228
33	./road-images/green-	0

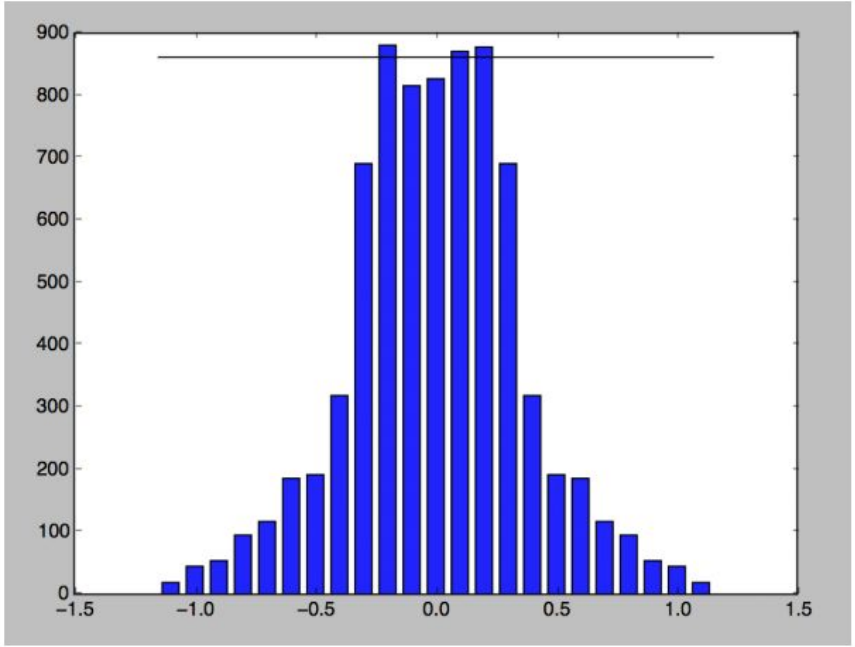
```
import pandas as pd
import matplotlib.image as mpimg
```

```
# IMPORTANT: Don't run if there's nothing wrong in your csv files
# This Deletes rows in logs that do not have existing image file in directory (IE corrupted log fi
le)

OLD_CSV_FILE = "./road-images-2/green.csv"
NEW_CSV_FILE = "./road-images-2/new-green.csv"

DATA = pd.read_csv(OLD_CSV_FILE)
print("Old csv file", OLD_CSV_FILE)
print("Number of rows from old csv file:", len(DATA))
print()

print("Show sample data:")
print(DATA.head())
print()

for i in range(0, len(DATA)):
    image_name = DATA['NAME'][i]
    image_path = image_name
    try:
        unprocessed_img = mpimg.imread(image_path)
        #print(image_name, "exists.")
        print(".", end="")
    except:
        print()
        print("delete:", i, "name:", image_name)
        DATA = DATA.drop([i])

print()
print("Cleaned data:", len(DATA))
print()

DATA.to_csv(NEW_CSV_FILE, index=False, encoding='utf8')
print("...saved to csv file", NEW_CSV_FILE)

DATA = pd.read_csv(NEW_CSV_FILE)

print()
print("Number of rows from new csv file:", len(DATA))
print()
```

https://github.com/mithi/little-miss-trendy/blob/master/A-AV2/A-DELETE-LOG-ROWS.ipynb

DO YOU REALLY NEED A BALANCED DATA SET? *normalized steering output distribution*





DO YOU REALLY NEED A BALANCED DATA SET?

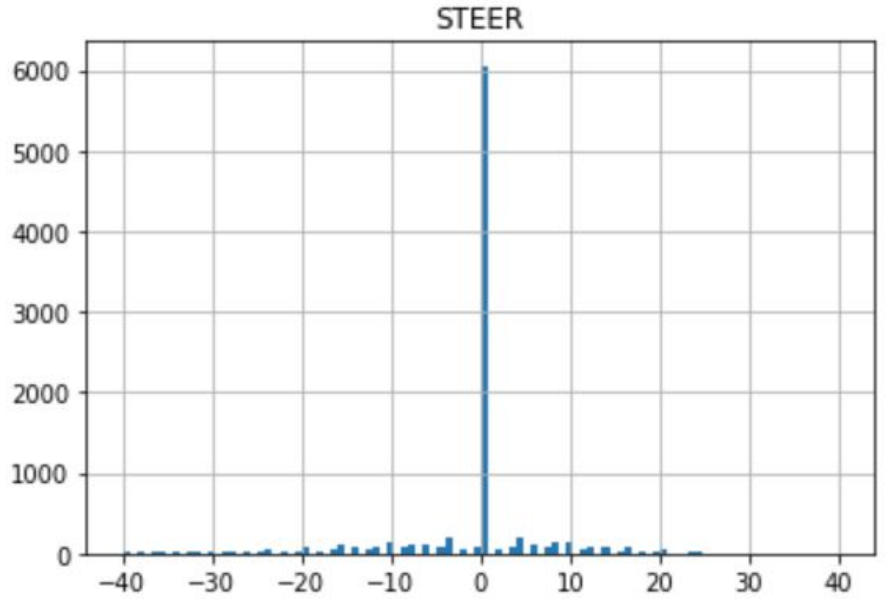
- Probably 99% of the time, but not all the time

- Do you really want your AI to think it’s alright to steer 45 degrees during curves all the time or would you rather that it learns to steer at curves more gracefully?

DO YOU REALLY NEED A BALANCED DATA SET?

However, you should still should probably delete many “straight road” images.

```
: red_frame.distribution(my_bins=100)
```



MODEL’S ABILITY TO GENERALIZE TO UNSEEN TRACKS

- We collected data only on Alpha 5 to train model
- But the model generalized on Beta 3!

Videos

- alpha4 track1
- alpha4 track2
- Data inspection video

Directory `./A-AV2/`

- Contains behavioral cloning work to make a car run autonomously on app `formula-trend-1.0.0-alpha.5`
- Contains behavioral cloning work to make a car run autonomously on app `formula-trend-1.0.0-beta.3`

Videos

- beta3 track1 red 4 laps normal
- beta3 track2 red 4 laps normal
- beta3 track3 red normal
- alpha5 track3 red 10 laps timelapse
- alpha5 track3 red 10 laps normal
- alpha5 track2 green 10 laps timelapse
- alpha5 track2 red 10 laps timelapse
- alpha5 track2 green 10 laps normal
- alpha5 track2 red 10 laps normal

VIDEOS

ALPHA 5 - FAST - GREEN - 10 LAPS

ALPHA 3 - RED - FULL LAP

THROTTLE: SIMPLE EQUATION

```
speednorm = speed/MAX_SPEED
anglenorm = angle/MAX_ANGLE

throttle =
   MAX_THROTTLE - A*speednorm² - B*anglenorm²

throttle = max(MIN_THROTTLE, throttle)
```

DATA SELECTION

- Training with data from only the human driver is not sufficient; the network must also learn **how to recover from any mistakes**

- **RECOVERY DRIVING**

RECOVERY DRIVING VIDEOS

- Beta 3 wrong (different parameters)
- Sample recovery 1
- Sample recovery 2
- Sample recovery 3

DATA SELECTION

- If you have a model that can recover then you wouldn’t have to slow down as much during curves, and you’d be able to finish laps more quickly!

MODEL’S ABILITY TO GENERALIZE TO UNSEEN TRACKS

- However, based on your goal, is a little bit of overfitting that bad?

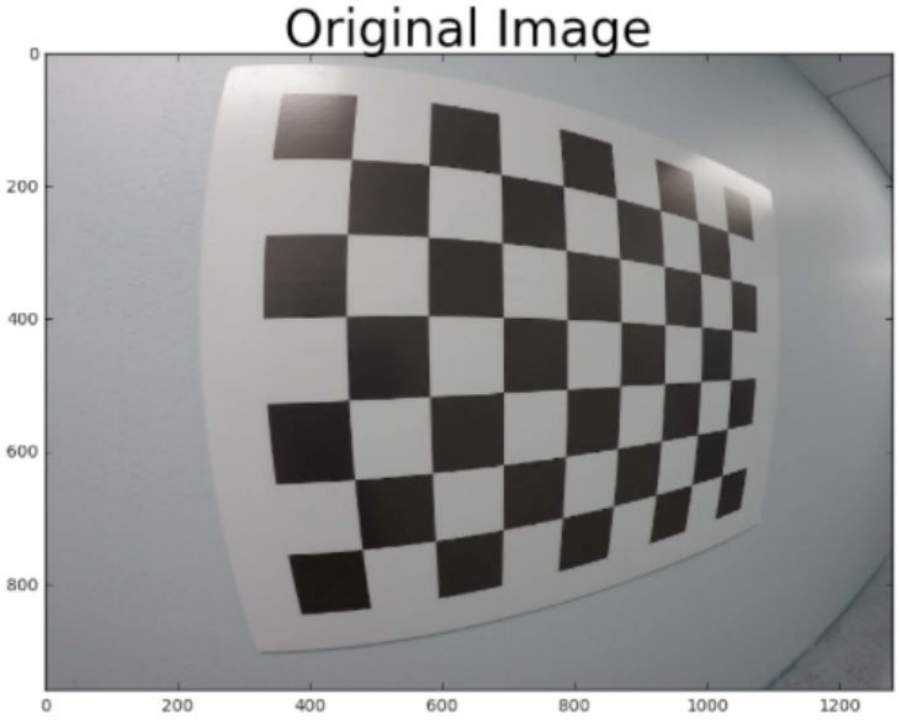
```
def switch_right():
    MODEL = MODEL_GREEN
    STATUS = RIGHT
    send_control(45, 1.0)


def switch_left():
    MODEL = MODEL_RED
    STATUS = LEFT
    send_control(-45, 1.0)
```

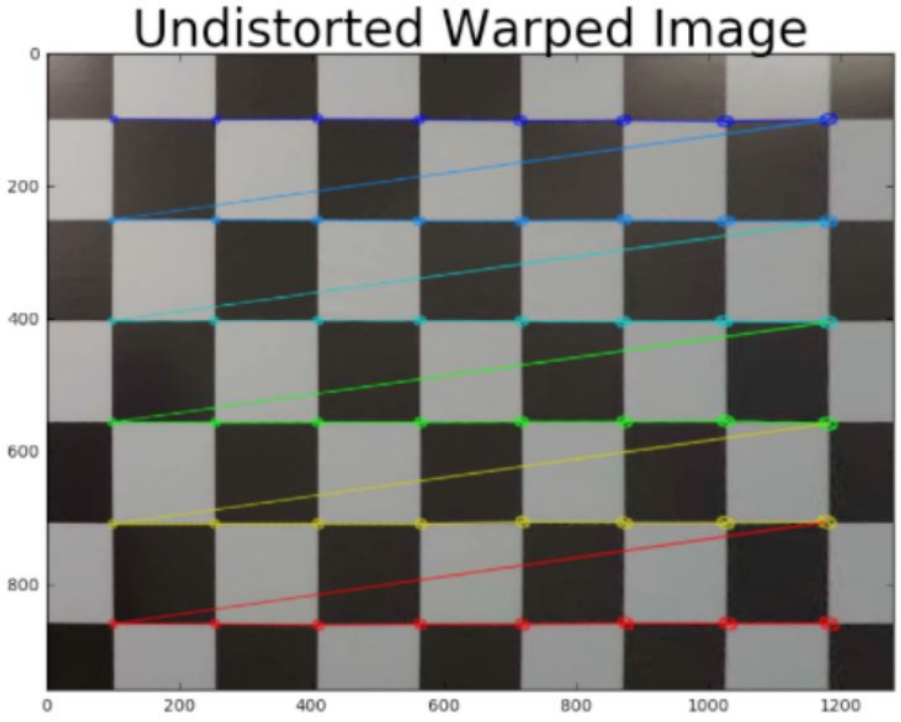
```
if obstact_detected(image):
    if STATUS == LEFT:
        switch_right()
    if STATUS == RIGHT:
        switch_left()
    return

sign = predict_sign(image)
if  STATUS == LEFT and sign == FORKGORIGHT:
    switch_right()
    return
if STATUS == RIGHT and sign == FORKGOLEFT:
    switch_left()
    return

steering_angle = predict_steer(image)
throttle = THROTTLE_MAX - C_SPEED * (speed
throttle = max(THROTTLE_MIN, throttle)
send_control(steering_angle, throttle)
```

REAL WORLD VS SIMULATION

- Correct Camera Distortion
- Data Augmentation (Shadows particularly)
- Faster OpenCV on Raspberry Pi

CORRECT CAMERA DISTORTION





Why is it important to correct for image distortion?

- Distortion can change the apparent size of an object in an image.
- Distortion can change the apparent shape of an object in an image.
- Distortion can cause an object's appearance to change depending on where it is in the field of view.
- Distortion can make objects appear closer or farther away than they actually are.

PROCESS	PURPOSE
Camera Calibration	To compute the transformation between 3D object points in the world and 2D image points.
Distortion Correction	To ensure that the geometrical shape of objects is represented consistently, no matter where they appear in an image.
Perspective Transform	To transform an image such that we are effectively viewing objects from a different angle or direction.

CORRECT CAMERA DISTORTION

OpenCV and Camera Calibration on a Raspberry Pi 3

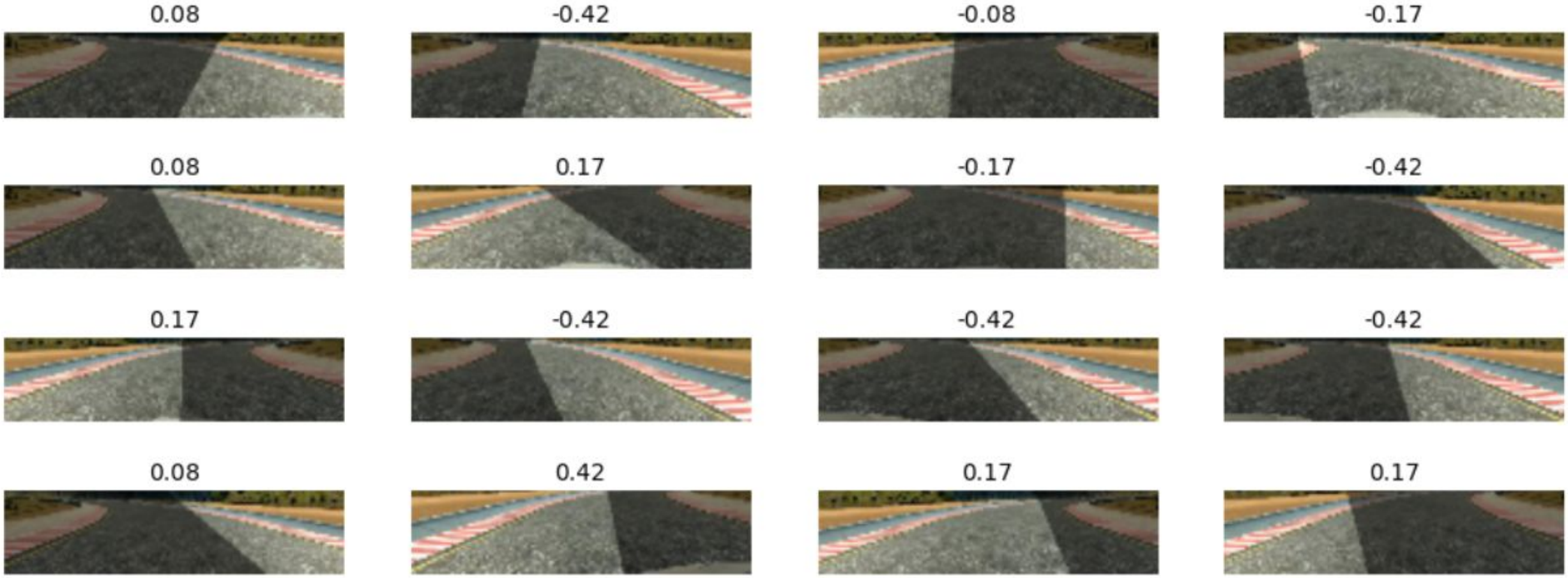
Tiziano Fiorenzani

Published on Jan 20, 2018

https://www.youtube.com/watch?v=QV1a1G4lL3U

Data Augmentation (SHADOWS)

https://github.com/navoshta/behavioral-cloning/blob/master/data.py



FASTER OPENCV ON RPI3

BY SAGI ZEEVI · PUBLISHED DEC 7, 2016 · UPDATED OCT 30, 2017

Summary:
- Don’t optimize unless you need to.
- Optimize your own code first.
- You can build a 30% faster OpenCV package for Raspberry Pi3

theimpossiblecode.com/blog/build-faster-opencv-raspberry-pi3
pyimagesearch.com/2017/10/09/optimizing-opencv-on-the-raspberry-pi

As of 1.11 you can also capture directly to numpy arrays (see Capturing to a numpy array). Due to the difference in resolution of the Y and UV components, this isn't directly useful (if you need all three components, you're better off using `PiYUVArray` as this rescales the UV components for convenience). However, if you only require the Y plane you can provide a buffer just large enough for this plane and ignore the error that occurs when writing to the buffer (picamera will deliberately write as much as it can to the buffer before raising an exception to support this use-case):

```
import time
import picamera
import picamera.array
import numpy as np

with picamera.PiCamera() as camera:
    camera.resolution = (100, 100)
    time.sleep(2)
    y_data = np.empty((112, 128), dtype=np.uint8)
    try:
        camera.capture(y_data, 'yuv')
    except IOError:
        pass
    y_data = y_data[:100, :100]
    # y_data now contains the Y-plane only
```

REAL WORLD VS SIMULATION

- Correct Camera Distortion
- Data Augmentation (Shadows particularly)
- Faster OpenCV on Raspberry Pi

```
def train_model(model, data, modelh5_name, modeljson_name):

    print('summary of model:')
    model.summary()

    print('Training model...')

    TRAINING_DATA, VALIDATION_DATA = train_test_split(data, test_size=VALIDATION_SIZE)
    TOTAL_TRAIN_DATA = len(TRAINING_DATA)
    TOTAL_VALID_DATA = len(VALIDATION_DATA)

    training_generator = generate_samples(TRAINING_DATA, batch_size=BATCH_SIZE)
    validation_generator = generate_samples(VALIDATION_DATA, batch_size=BATCH_SIZE)

    model.fit_generator(training_generator,
        samples_per_epoch=TOTAL_TRAIN_DATA,
        validation_data=validation_generator,
        nb_val_samples=TOTAL_VALID_DATA,
        nb_epoch=NUMBER_OF_EPOCHS,
        verbose=1)

    print('...Model trained.')

    print('Saving model...')

    model.save(modelh5_name)

    with open(modeljson_name, "w") as json_file:
        json_file.write(model.to_json())

    print("...Model Saved.")

    return model
```

```
left_red_model = build_modified_nvidia_model()
```

```
left_red_model = train_model(left_red_model, left_red_frame.data, "./models/left_red.h5", "./models/left_red.json")
```

```
def random_flip(image, steer):
    if np.random.random() > 0.5:
        return np.fliplr(image), -steer
    else:
        return image, steer
```

```
def get_processed_data(x, data):

    i = data.index[x]
    steer =  data['STEER'][i]
    img = mpimg.imread(data['NAME'][i])
    img = img[YSTART:YSTOP, :, :]
    img = cv2.resize(img, (INPUT_W, INPUT_H), cv2.INTER_AREA)
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)

    return random_flip(img, steer)
```

```
def generate_samples(data, batch_size):

    while True:

        SIZE = len(data)
        data.sample(frac=1)

        for start in range(0, SIZE, batch_size):
            images, steers = [], []

            for i in range(start, start + batch_size):
                if i < SIZE:
                    image, steer = get_processed_data(i, data)
                    steers.append(steer)
                    images.append(image)

            yield (np.array(images), np.array(steers))
```

```
50  # Initialize global Socket Server
51  sio = socketio.Server()
52
53
54  def process_image(img):
55      img = img[YSTART:YSTOP, :, :]
56      img = cv2.resize(img, (INPUT_W, INPUT_H), cv2.INTER_AREA)
57      img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
58      return img
59
60
61  def send_control(steering_angle, throttle):
62      sio.emit(
63          "steer",
64          data = {
65              'steering_angle': str(steering_angle),
66              'throttle': str(throttle)
67          },
68          skip_sid=True)
69
70
71  def predict_steer(img_string):
72      image = Image.open(BytesIO(base64.b64decode(img_string)))
73      image_array = process_image(np.asarray(image))
74      steering_angle = float(MODEL.predict(image_array[None, :, :, :]))
75      return steering_angle
76
```

```python
@sio.on('telemetry')
def telemetry(sid, data):

    if data:

        steering_angle = float(data["steering_angle"])
        throttle = float(data["throttle"])
        speed = float(data["speed"])
        img_string = data["image"]

        steering_angle = predict_steer(img_string)

        throttle = THROTTLE_MAX - C_SPEED * (speed / MAX_SPEED)**2 - C_STEER * (steering_angle / MAX_ANGLE)**2
        throttle = max(THROTTLE_MIN, throttle)

        send_control(steering_angle, throttle)

    else:
        sio.emit('manual', data={}, skip_sid=True)


@sio.on('connect')
def connect(sid, environ):
    print("connect ", sid)
    send_control(0.0, 0.0)

```

```
104  if __name__ == '__main__':
105
106      parser = argparse.ArgumentParser(description='Auto Driving!')
107      parser.add_argument(
108          'model',
109          type=str,
110          help='Path to model h5 file. Model should be on the same path.'
111      )
112
113      args = parser.parse_args()
114      MODEL = load_model(args.model)
115
116      app = socketio.Middleware(sio, Flask(__name__))
117      eventlet.wsgi.server(eventlet.listen(('127.0.0.1', 4567)), app)
```

Two Case Studies (Research Papers)

1. By: Georgia Tech AutoRally
2. By: NVIDIA

In the context of your specific needs

Behavioral Cloning for Lane following

- Basic tips on how to improve your models ability to work
- Simple things you can do to increase performance
- (Ex: given specs of your processor)

REAL WORLD VS SIMULATION

- Correct Camera Distortion
- Data Augmentation (Shadows particularly)
- Faster OpenCV on Raspberry Pi

QUESTIONS?

A Problem Solving Philosophy

- They’re so obvious to everyone but a lot of people don’t really keep them in mind when solving problems
- **Includes throttle adjusting methods and path-planning with CODE**

ML - What you should know by heart

- Useful to tweak parameters!
- Very basic, WILL ALWAYS COME UP
- Focus on practical, not theory or math
- **LAYER VISUALIZATION**

Two Case Studies (Research Papers)

1. By: Georgia Tech AutoRally
2. By: NVIDIA

In the context of your specific needs

Behavioral Cloning for Lane Following

- NETWORK ARCHITECTURE
- IMAGE PREPROCESSING
- DATA SELECTION
- Techniques and tips

THANK YOU!