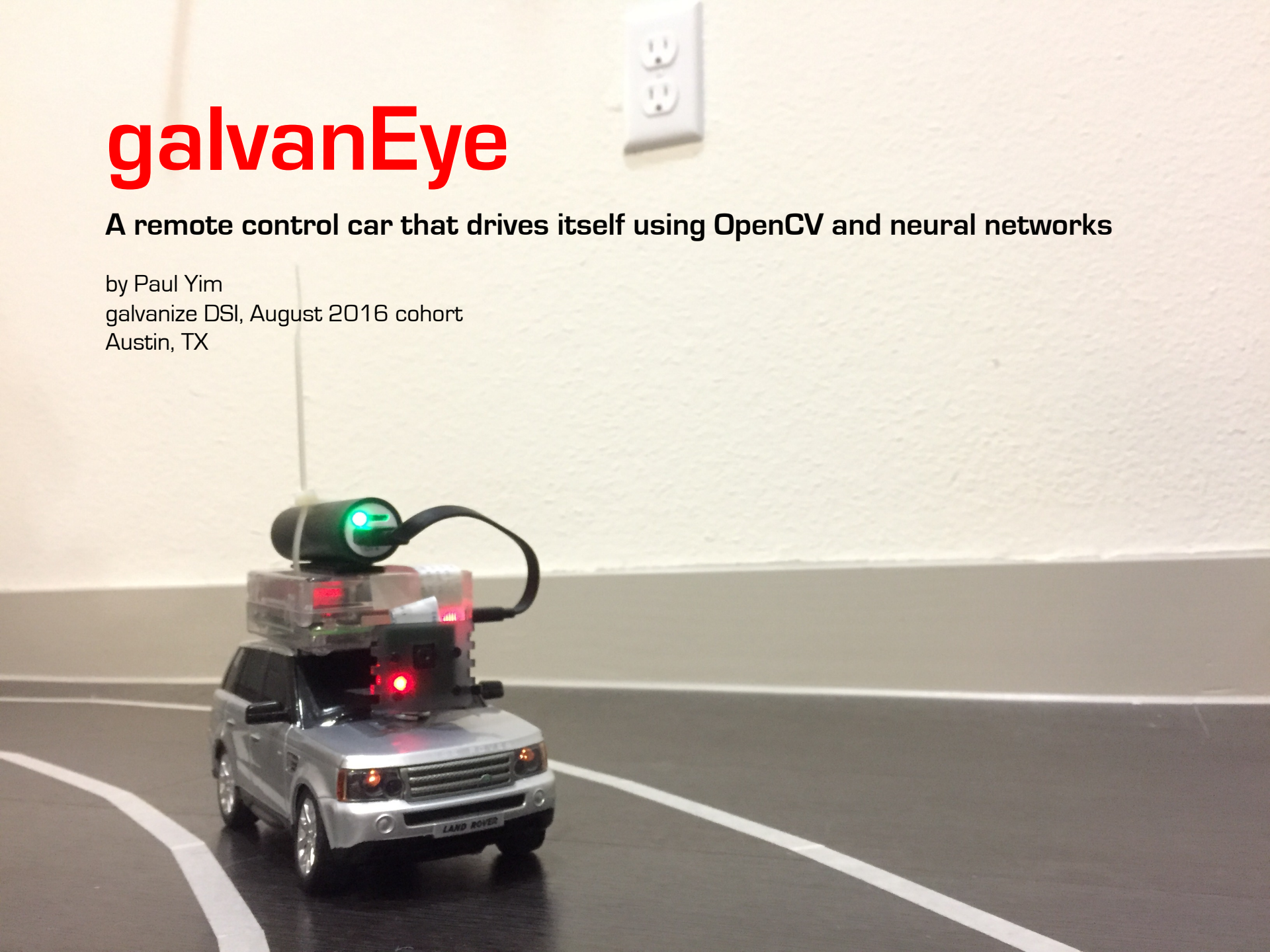
galvanEye

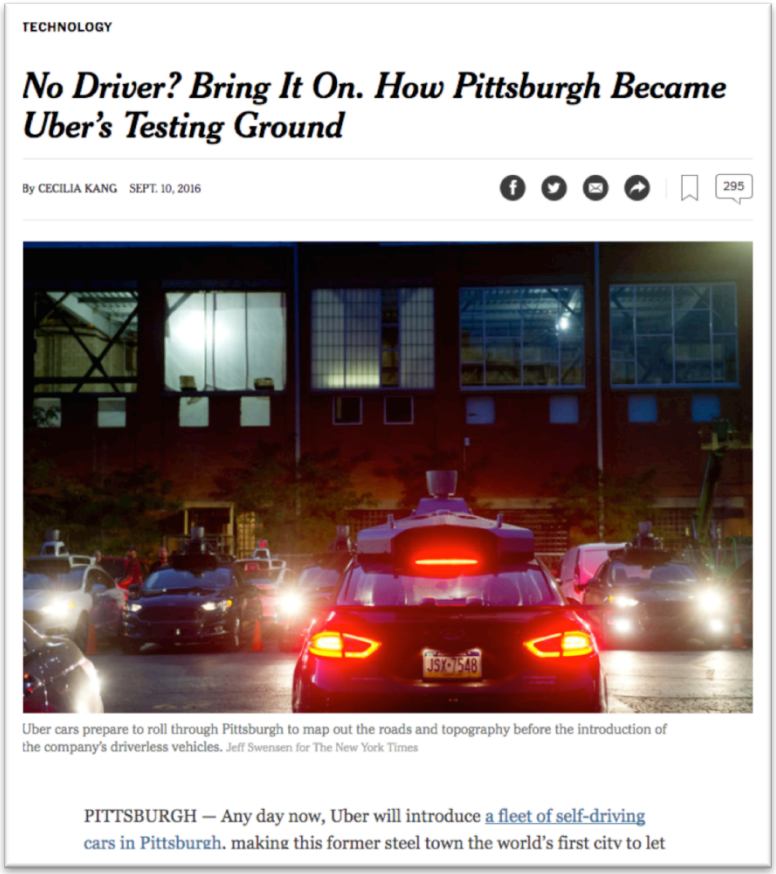
A remote control car that drives itself using OpenCV and neural networks

by Paul Yim
galvanize DSI, August 2016 cohort
Austin, TX



Motivation

The New York Times, Sep 10, 2016



TECHNOLOGY

No Driver? Bring It On. How Pittsburgh Became Uber's Testing Ground

By CECILIA KANG SEPT. 10, 2016

Uber cars prepare to roll through Pittsburgh to map out the roads and topography before the introduction of the company's driverless vehicles. Jeff Swensen for The New York Times

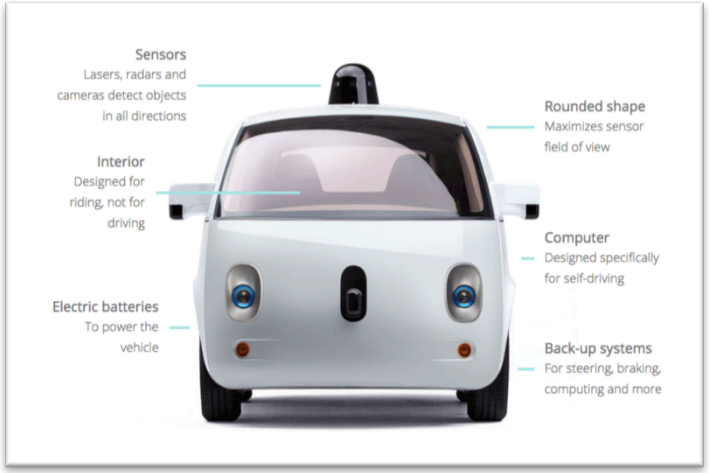
PITTSBURGH — Any day now, Uber will introduce a fleet of self-driving cars in Pittsburgh, making this former steel town the world's first city to let

- Autonomous vehicles are now on the road.
- Major implications for public policy.
- How do they work?

Motivation

Variety of systems, often working in concert:

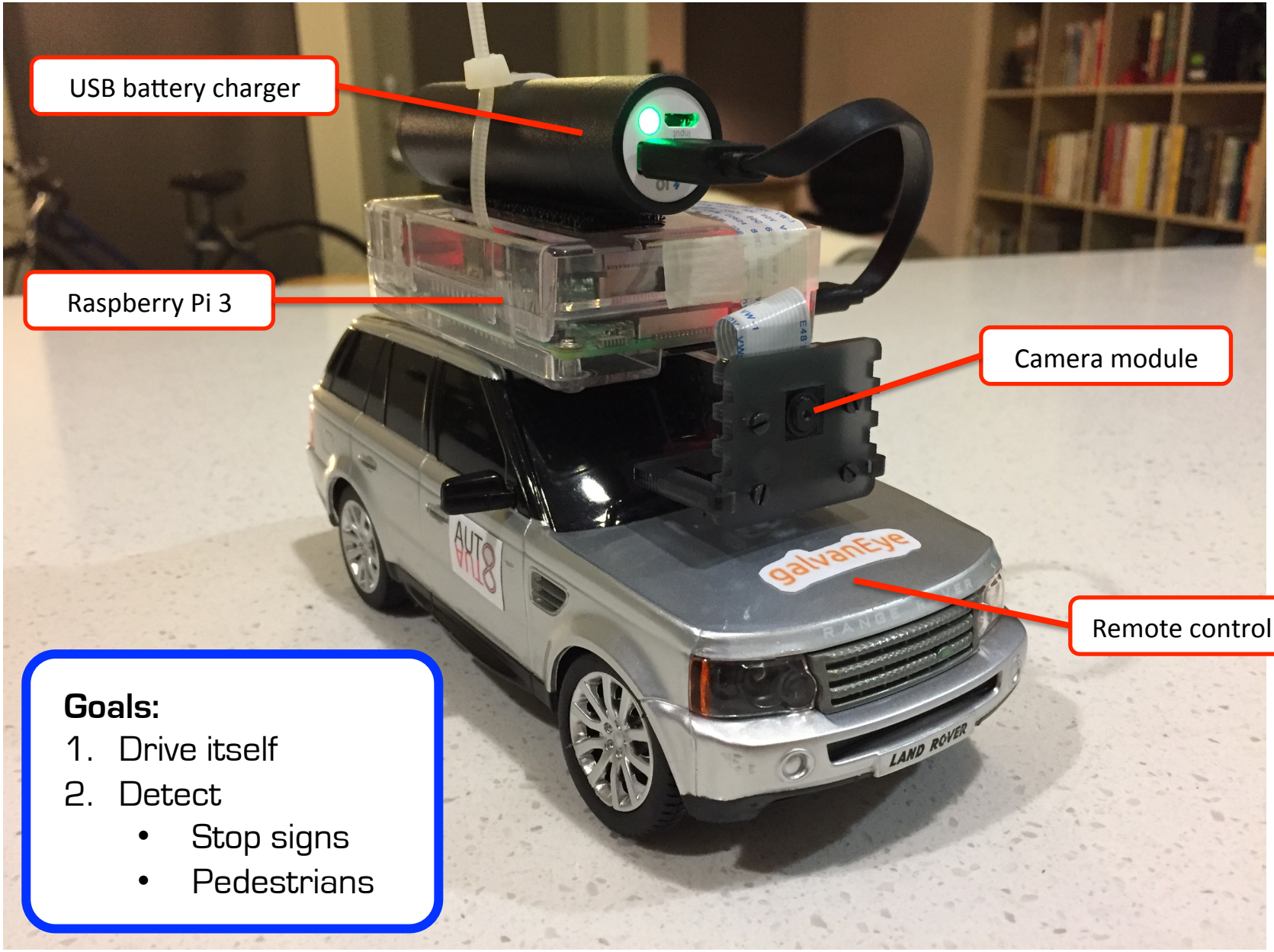
- Radar and Lidar
- Motion sensors
- Computer vision
  - **Image processing**
  - **Neural network prediction**



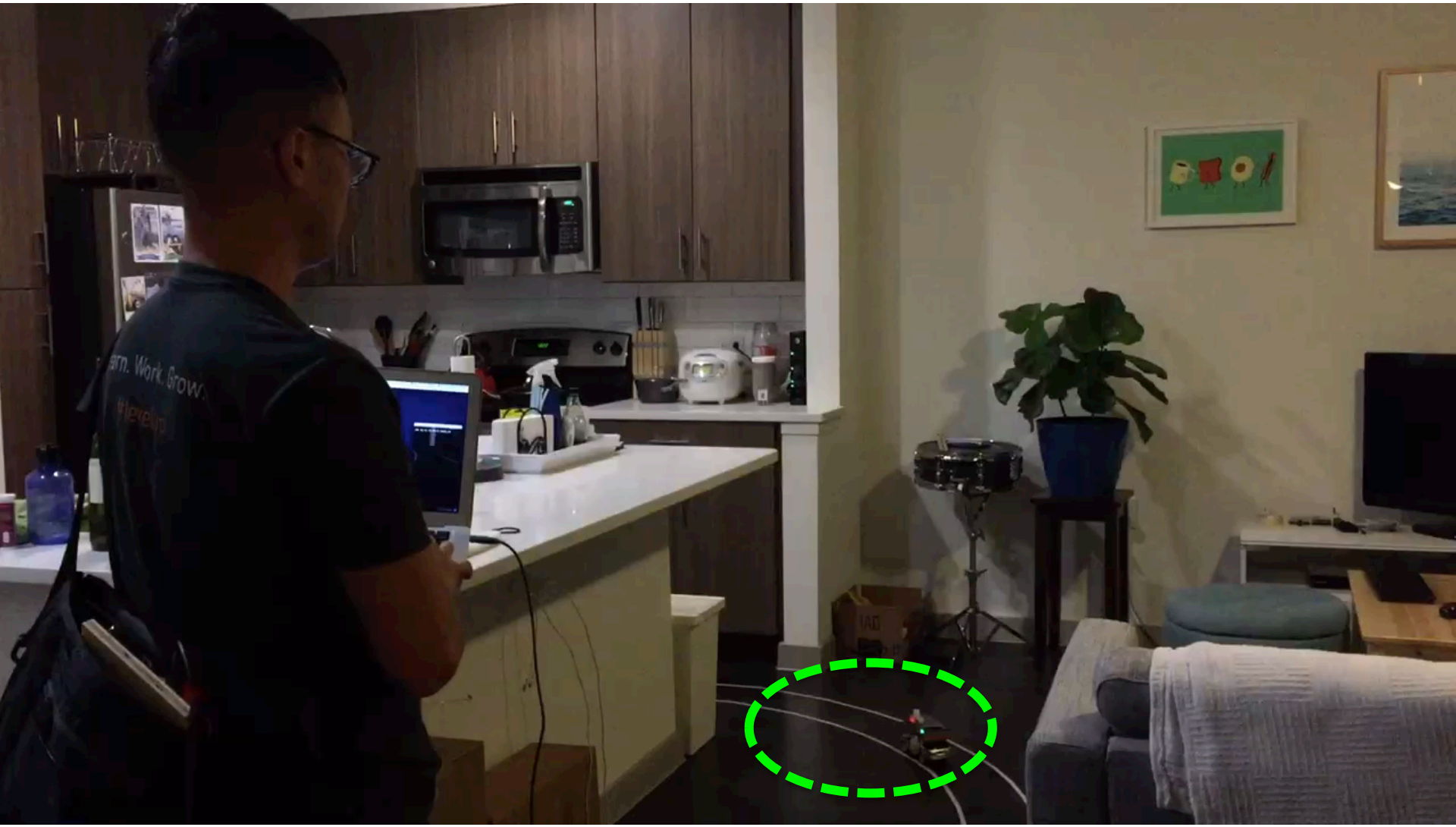
Google

Mobileye

Model

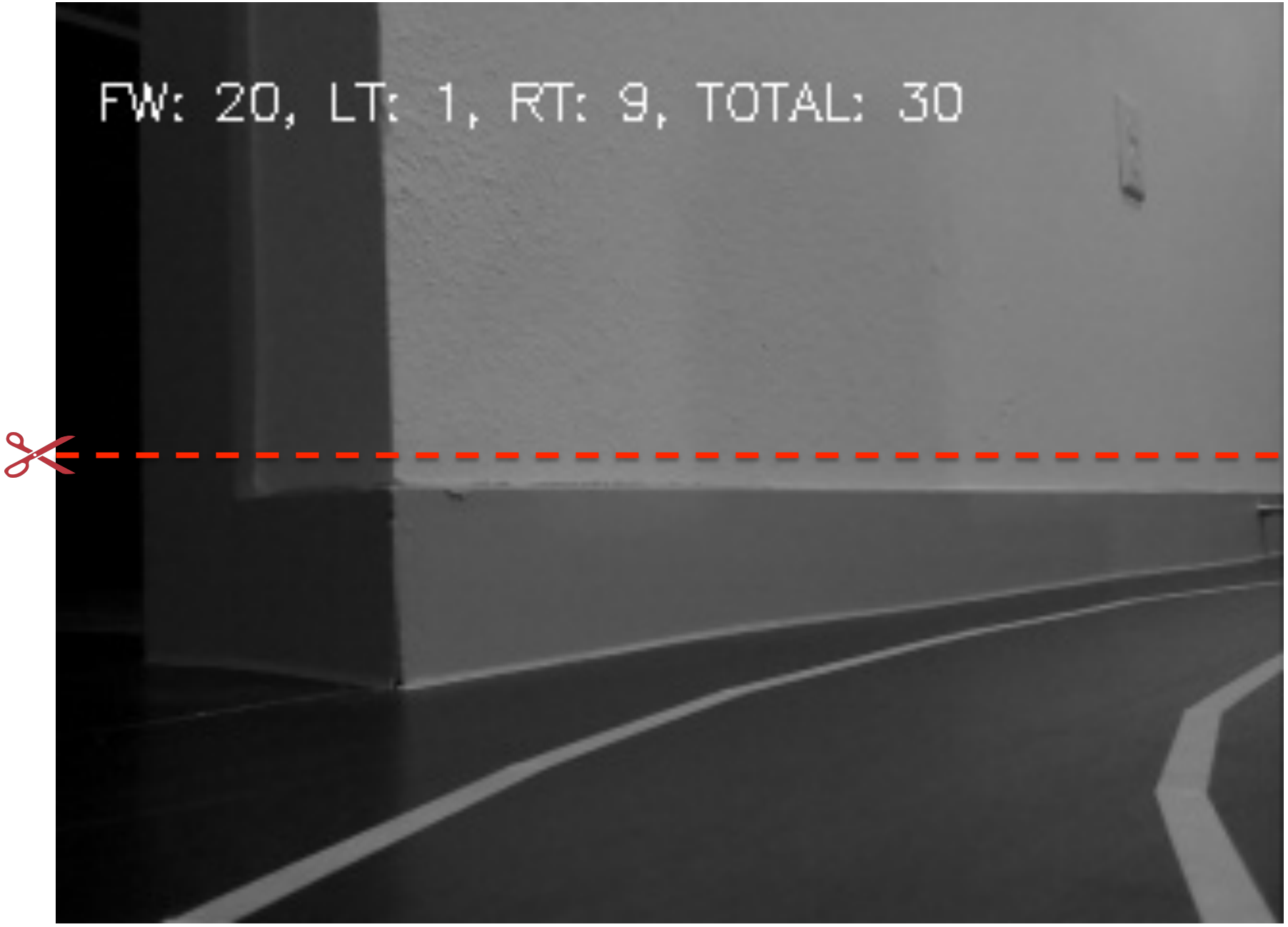

USB battery charger
Raspberry Pi 3
Camera module
Remote control car
galvanEye
LAND ROVER
Goals:
1. Drive itself
2. Detect
• Stop signs
• Pedestrians

Model: Training data



Model: Training data

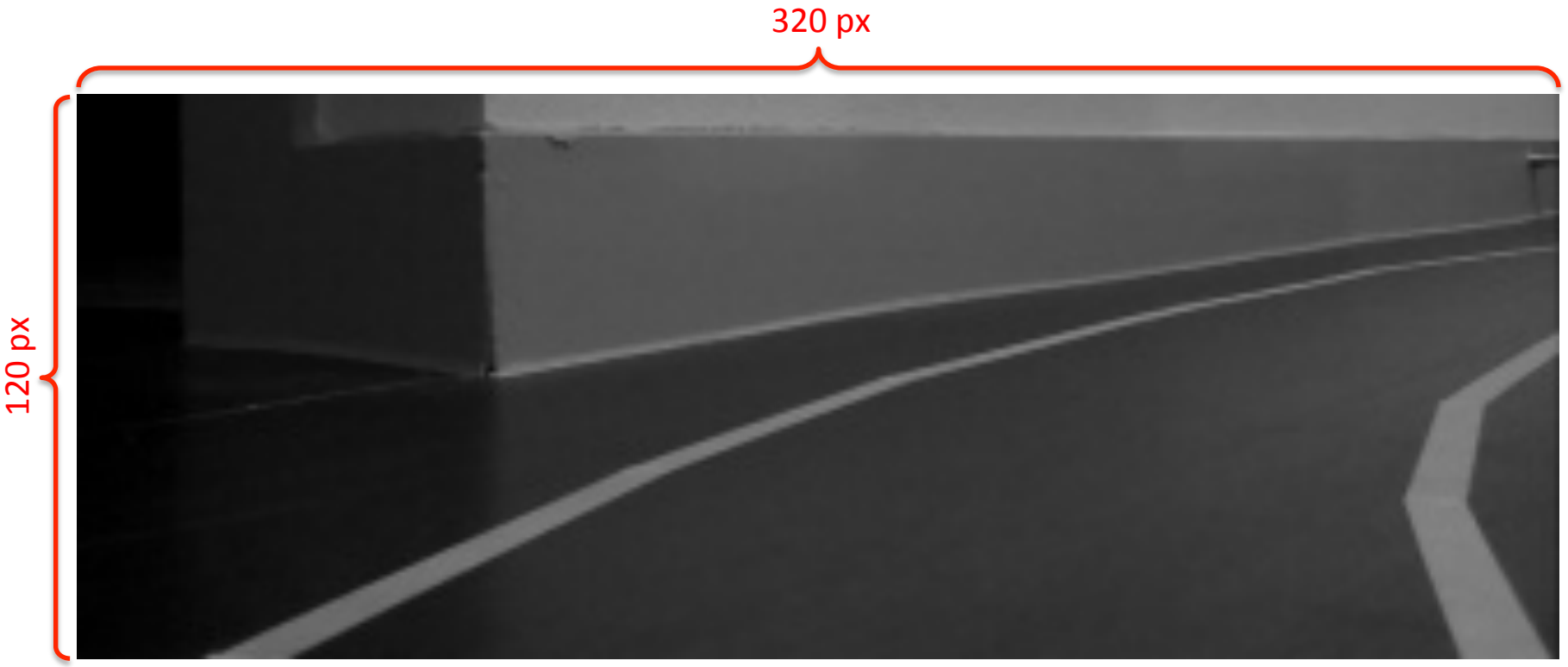
Model: Training



Model: Training

```
"LEFT"    [1, 0, 0]
"RIGHT"   [0, 1, 0]
"FORWARD" [0, 0, 1]
```

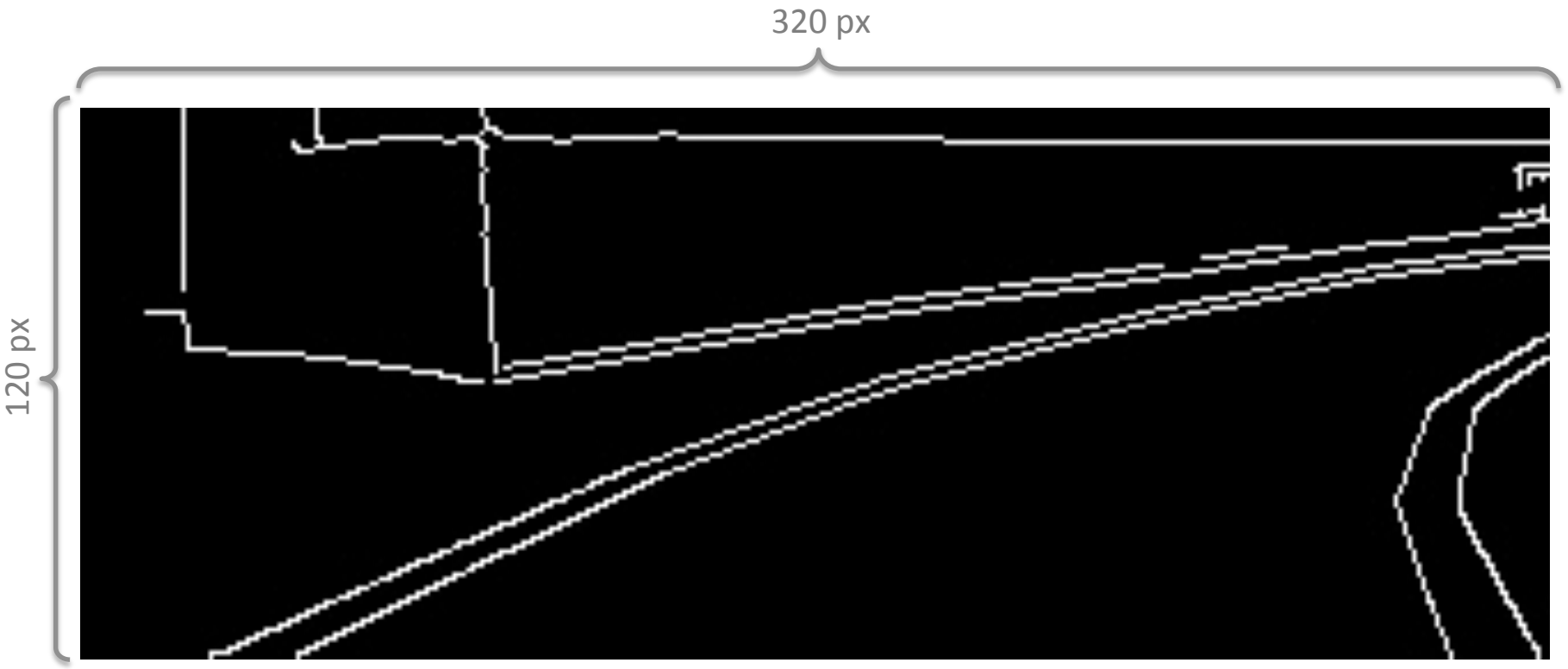
Given this image as **input**, the correct **output** is to turn "RIGHT".



Model: Training

“RIGHT” [0, 1, 0]

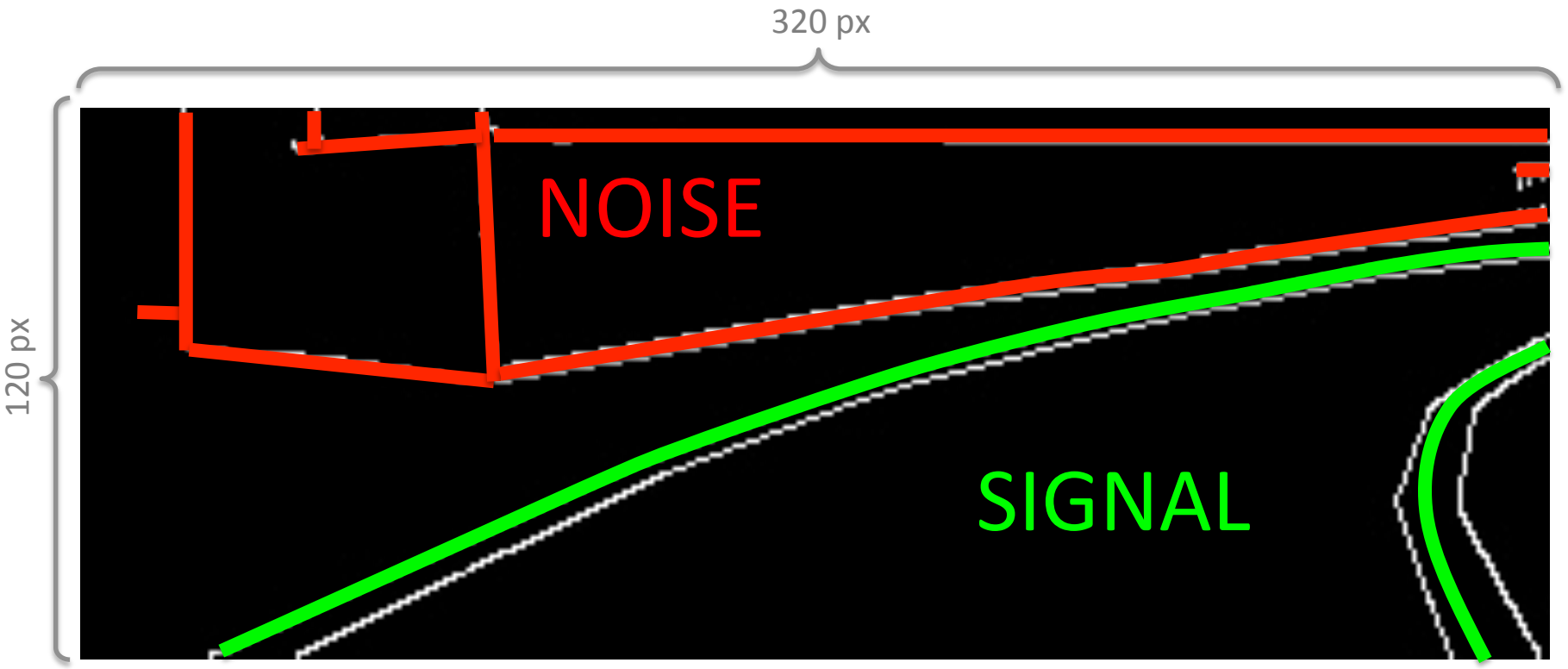
Given this image as **input,** the correct **output** is to turn “RIGHT”.



Model: Training

`"RIGHT"  [0, 1, 0]`

Given this image as **input,** the correct **output** is to turn "RIGHT".



Model: Training

“RIGHT” [0, 1, 0]

(not drawn to scale)

```
[[255,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0],
 [255,   0, 255,   0,   0,   0,   0, 255, 255, 255, 255],
 [255,   0, 255,   0, 255, 255, 255,   0,   0,   0,   0],
 [255, 255, 255, 255,   0,   0,   0,   0,   0, 255, 255],
 [  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0],
 [  0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0],
 [  0,   0,   0,   0, 255,   0,   0,   0,   0,   0, 255],
 [  0,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255],
 [  0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 255],
 [  0, 255,   0,   0,   0,   0,   0,   0,   0, 255,   0],
 [255,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0]]
```

Model: Training

“RIGHT” [0, 1, 0]

120 x 320 = 38,400

(not drawn to scale)

320 px

120 px

```
[[255,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0],
 [255,   0, 255,   0,   0,   0,   0, 255, 255, 255, 255],
 [255,   0, 255,   0, 255, 255, 255,   0,   0,   0,   0],
 [255, 255, 255, 255,   0,   0,   0,   0,   0, 255, 255],
 [  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0],
 [  0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0],
 [  0,   0,   0,   0, 255,   0,   0,   0,   0,   0, 255],
 [  0,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255],
 [  0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 255],
 [  0, 255,   0,   0,   0,   0,   0,   0,   0, 255,   0],
 [255,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0]]
```

Model: Training

“RIGHT” [0, 1, 0]

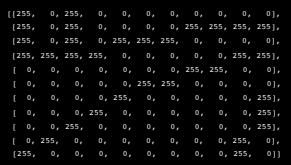
120 x 320 = 38,400

```
[[255,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0],
 [255,   0, 255,   0,   0,   0,   0, 255, 255, 255, 255],
 [255,   0, 255,   0, 255, 255, 255,   0,   0,   0,   0],
 [255, 255, 255, 255,   0,   0,   0,   0,   0, 255, 255],
 [  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0],
 [  0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0],
 [  0,   0,   0,   0, 255,   0,   0,   0,   0,   0, 255],
 [  0,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255],
 [  0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 255],
 [  0, 255,   0,   0,   0,   0,   0,   0,   0, 255,   0],
 [255,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0]]
```

Model: Training

“RIGHT” [0, 1, 0]

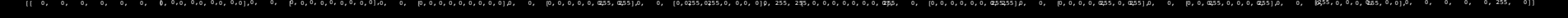
120 x 320 = 38,400



Model: Training

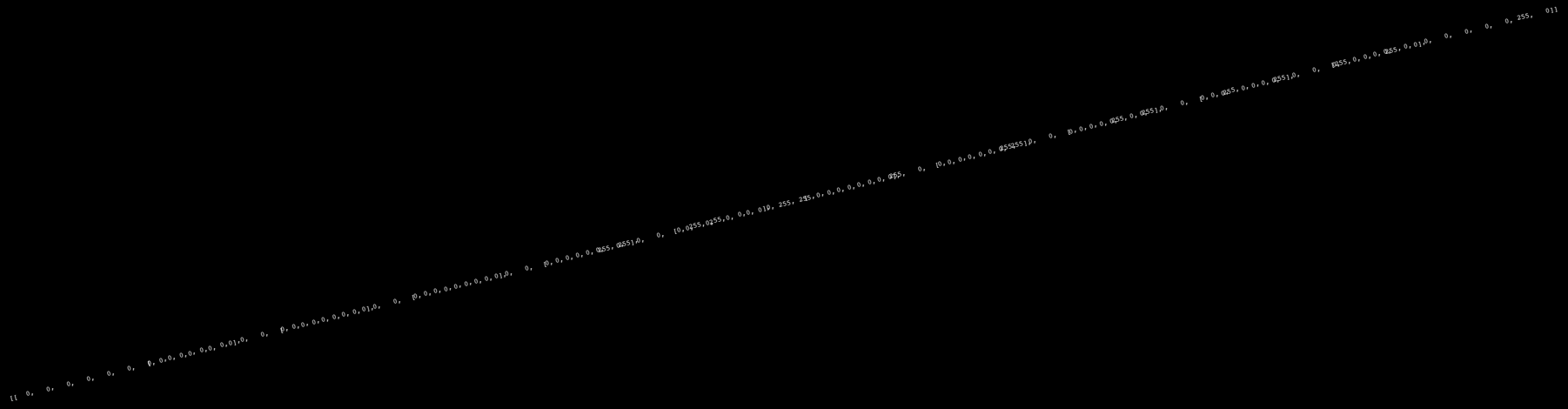
“RIGHT” [0, 1, 0]

120 x 320 = 38,400



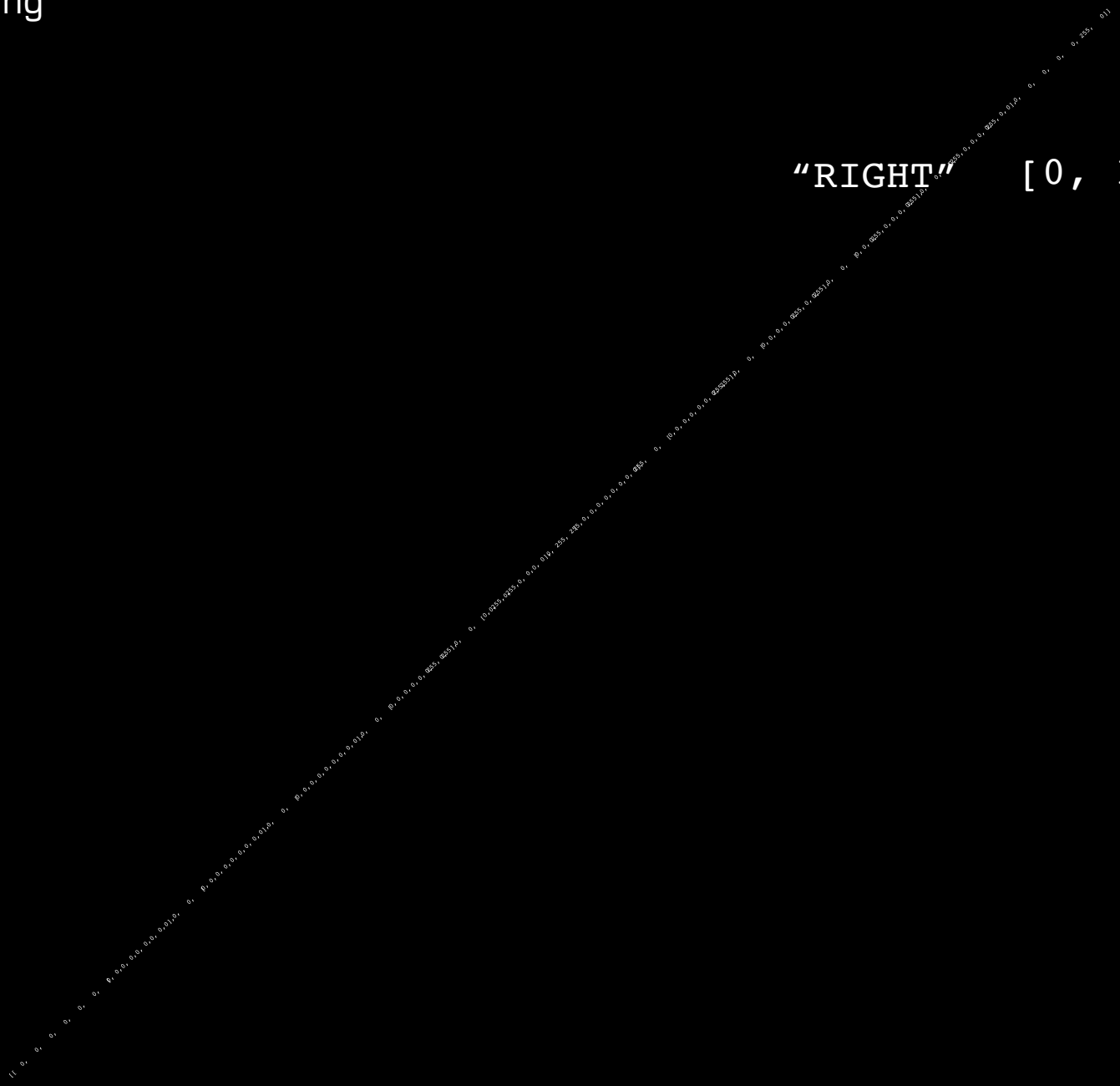
Model: Training

“RIGHT” [0, 1, 0]

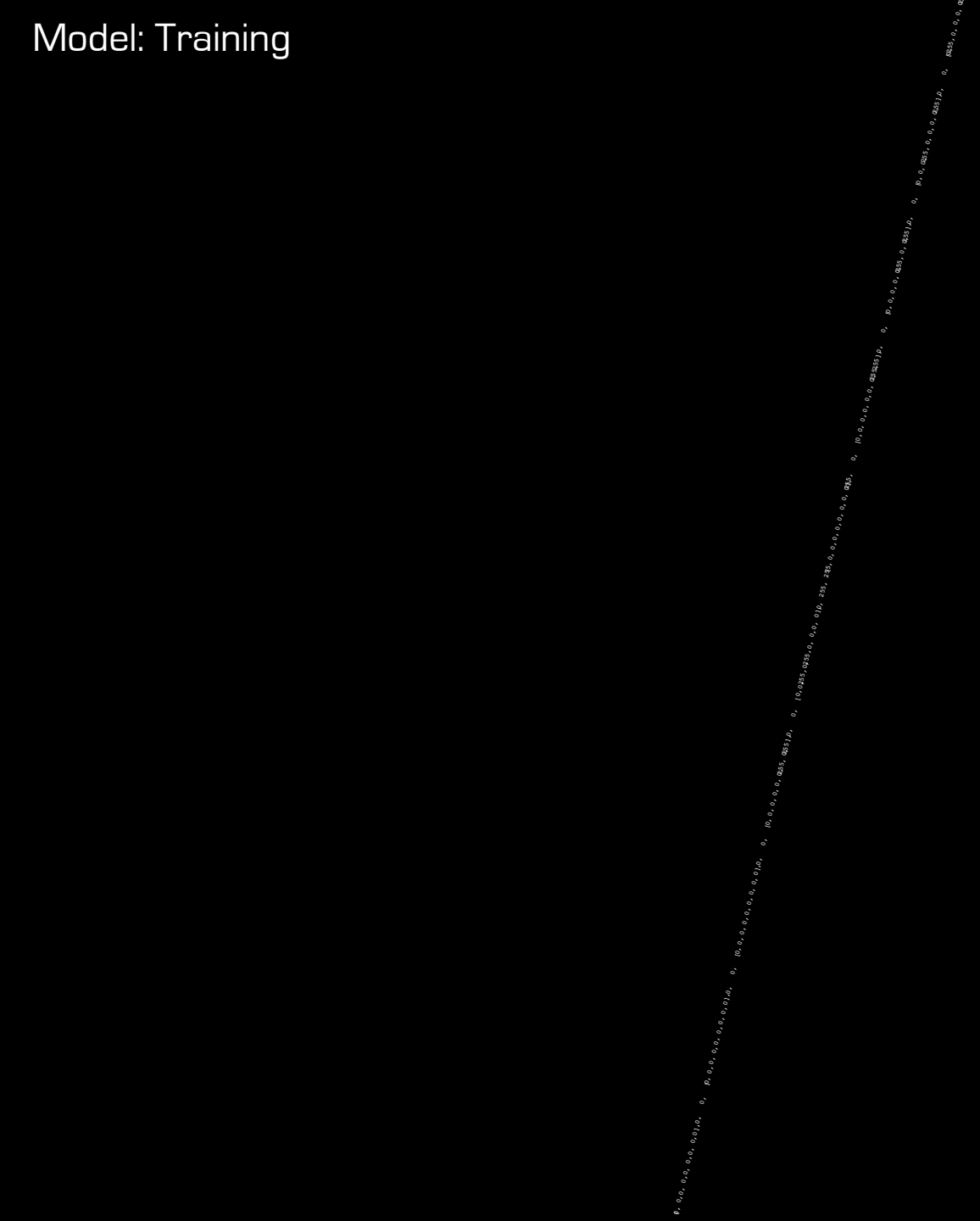


Model: Training

“RIGHT” [0, 1, 0]



Model: Training



“RIGHT” [0, 1, 0]

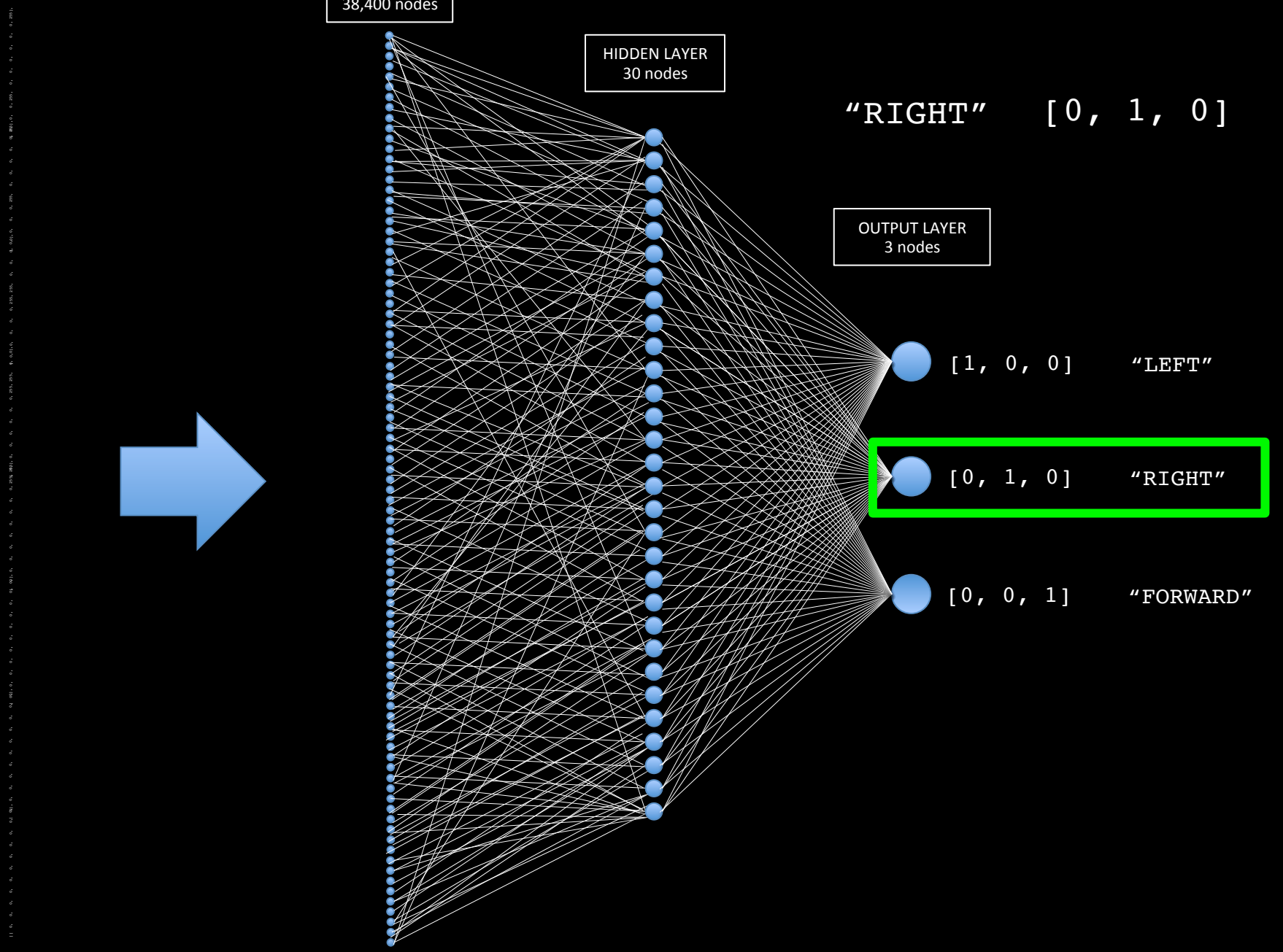
Model: Training

“RIGHT” [0, 1, 0]

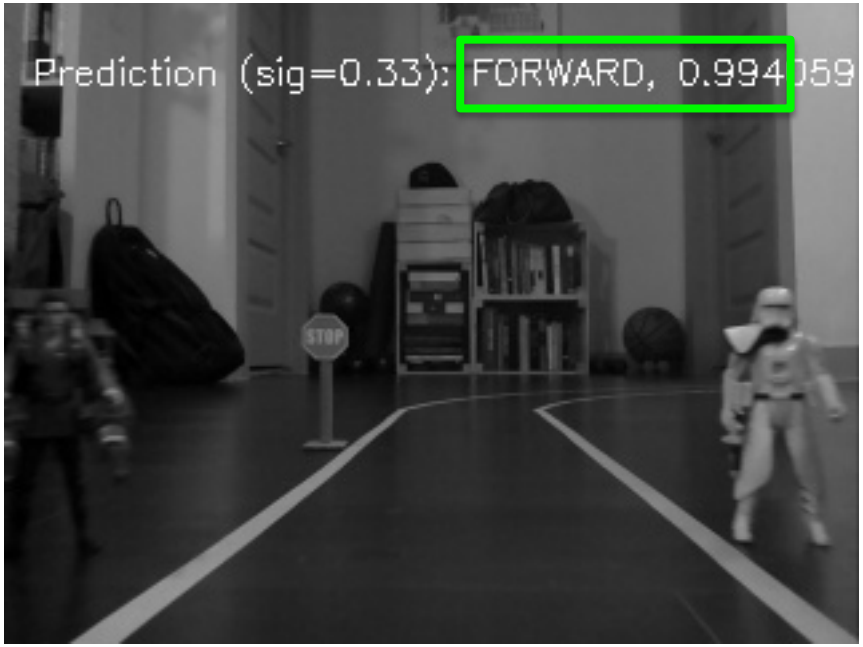
Model: Training

“RIGHT” [0, 1, 0]

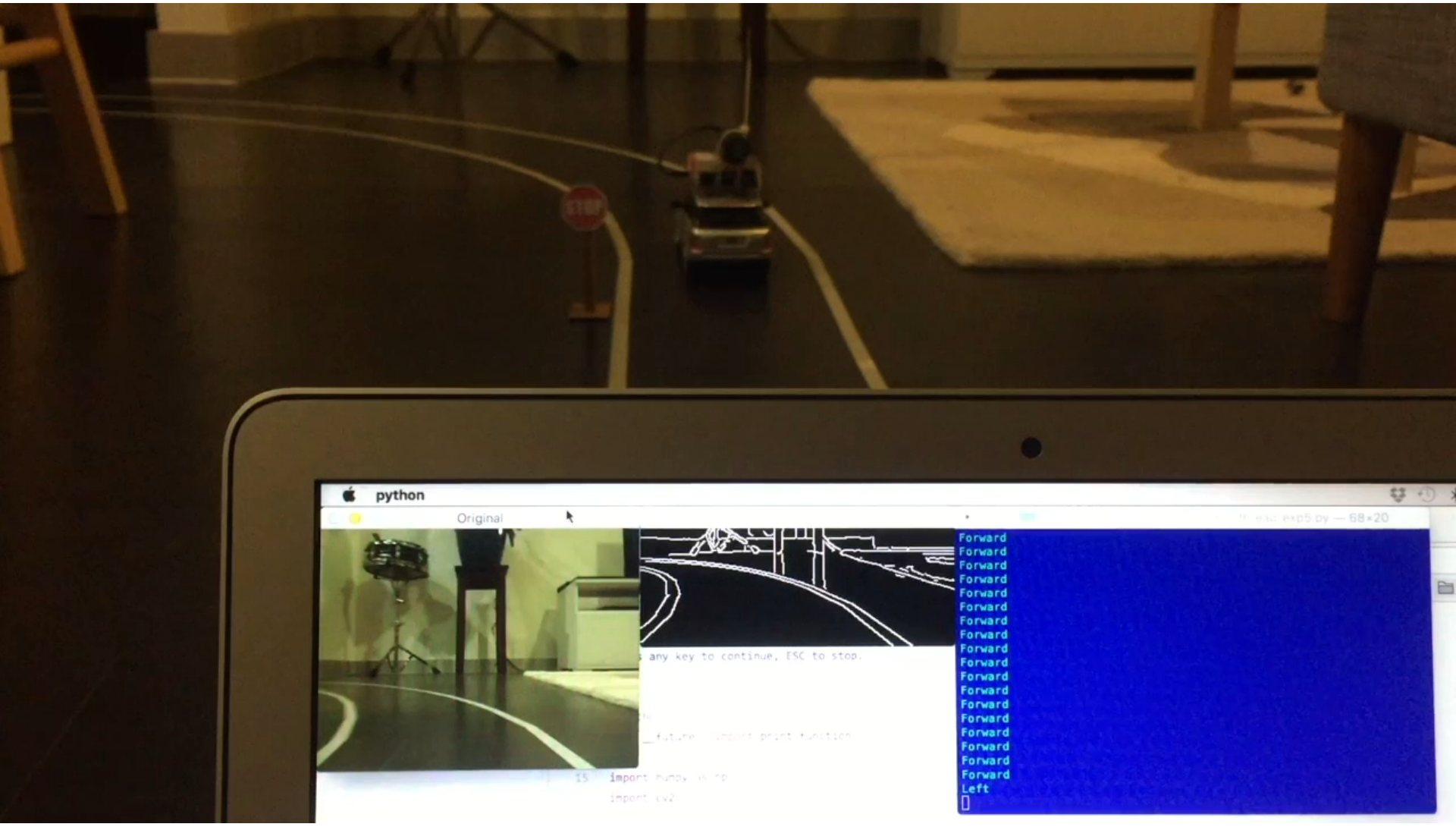
Model: Training



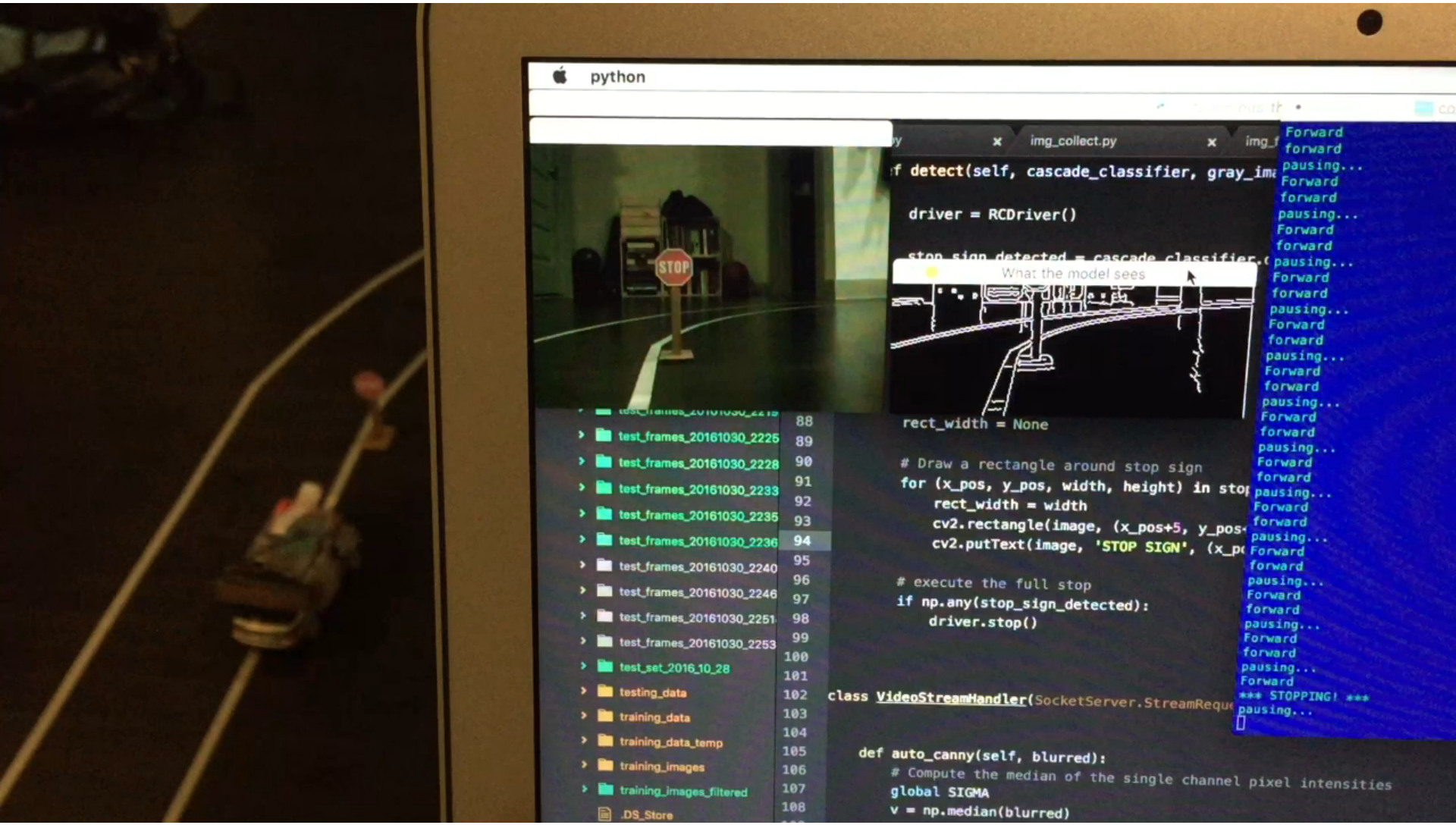
Performance: Driving



Performance: Driving



Performance: Stop sign and Pedestrian detection



Packages Used

Image Processing

- cv2 (OpenCV v3.1.0)
- imutils

Neural Network

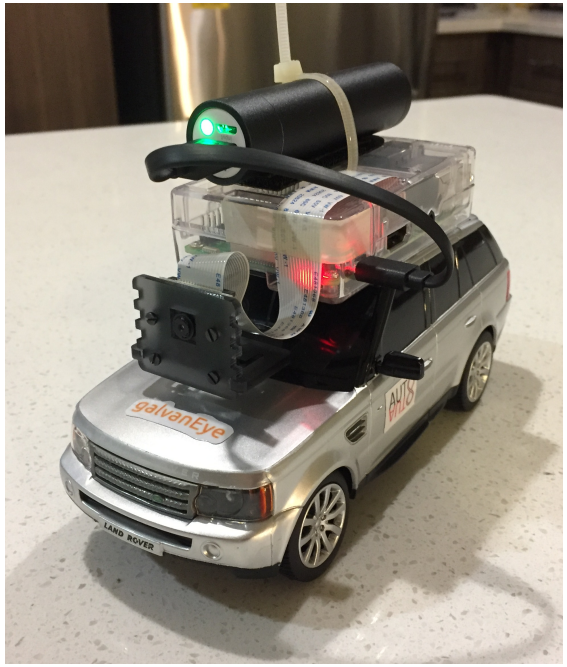
- keras (TensorFlow backend)

Other

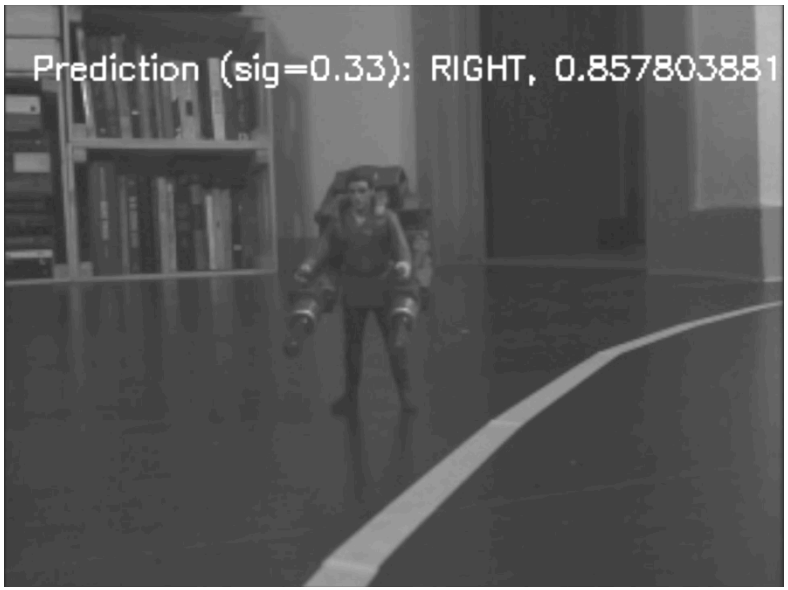
- numpy
- picamera
- pygame
- pySerial
- scikit-learn
- socket
- threading

Special thanks to

- **Grace** (my wife)
- **AutoAuto** (hackathon group)
- **Hamuchiwa** (inspiration & Github repo)



Questions?

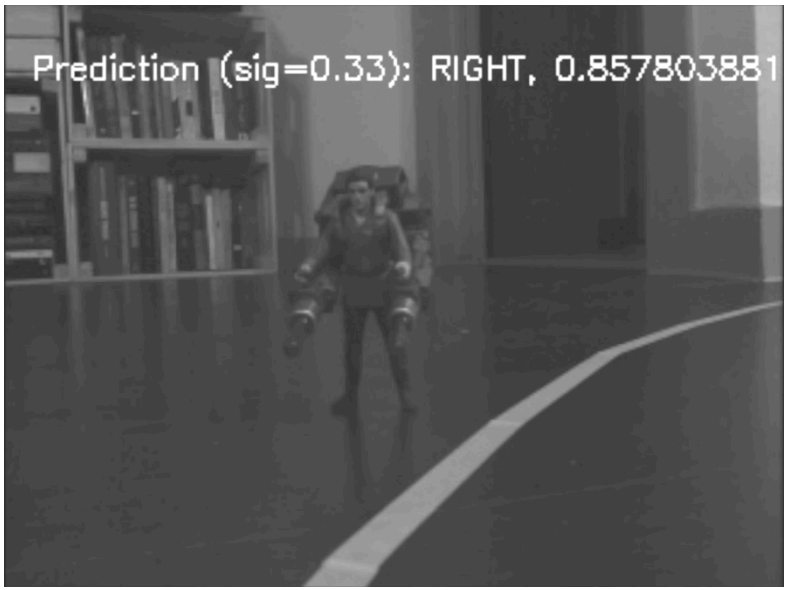


paul.j.yim@gmail.com

pauljyim

pseudoyim

Questions?

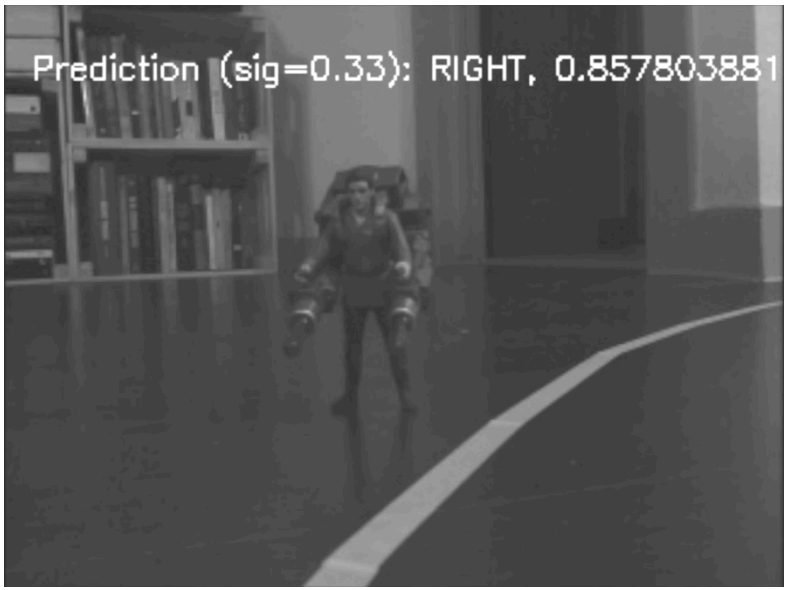


paul.j.yim@gmail.com

pauljyim

pseudoyim

Questions?

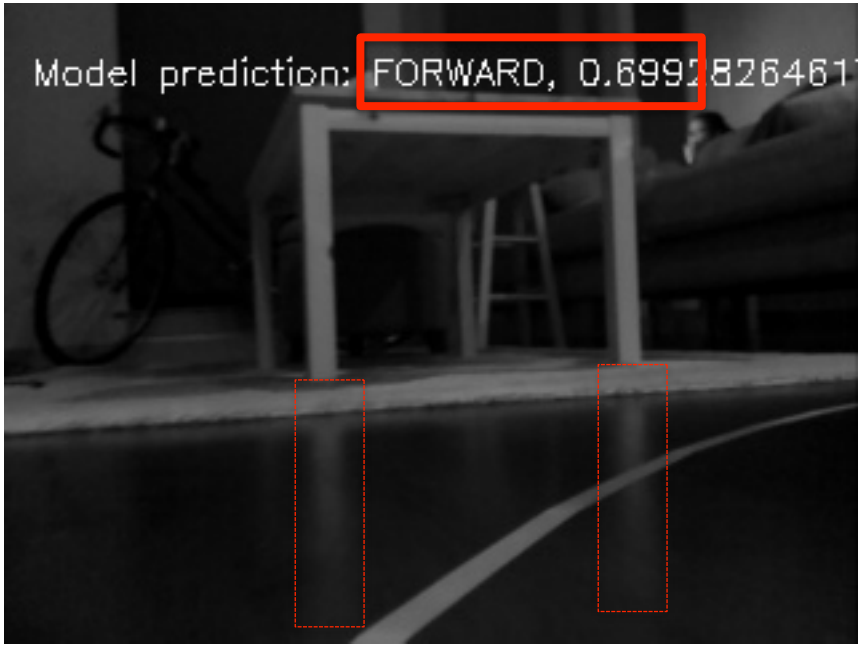


paul.j.yim@gmail.com

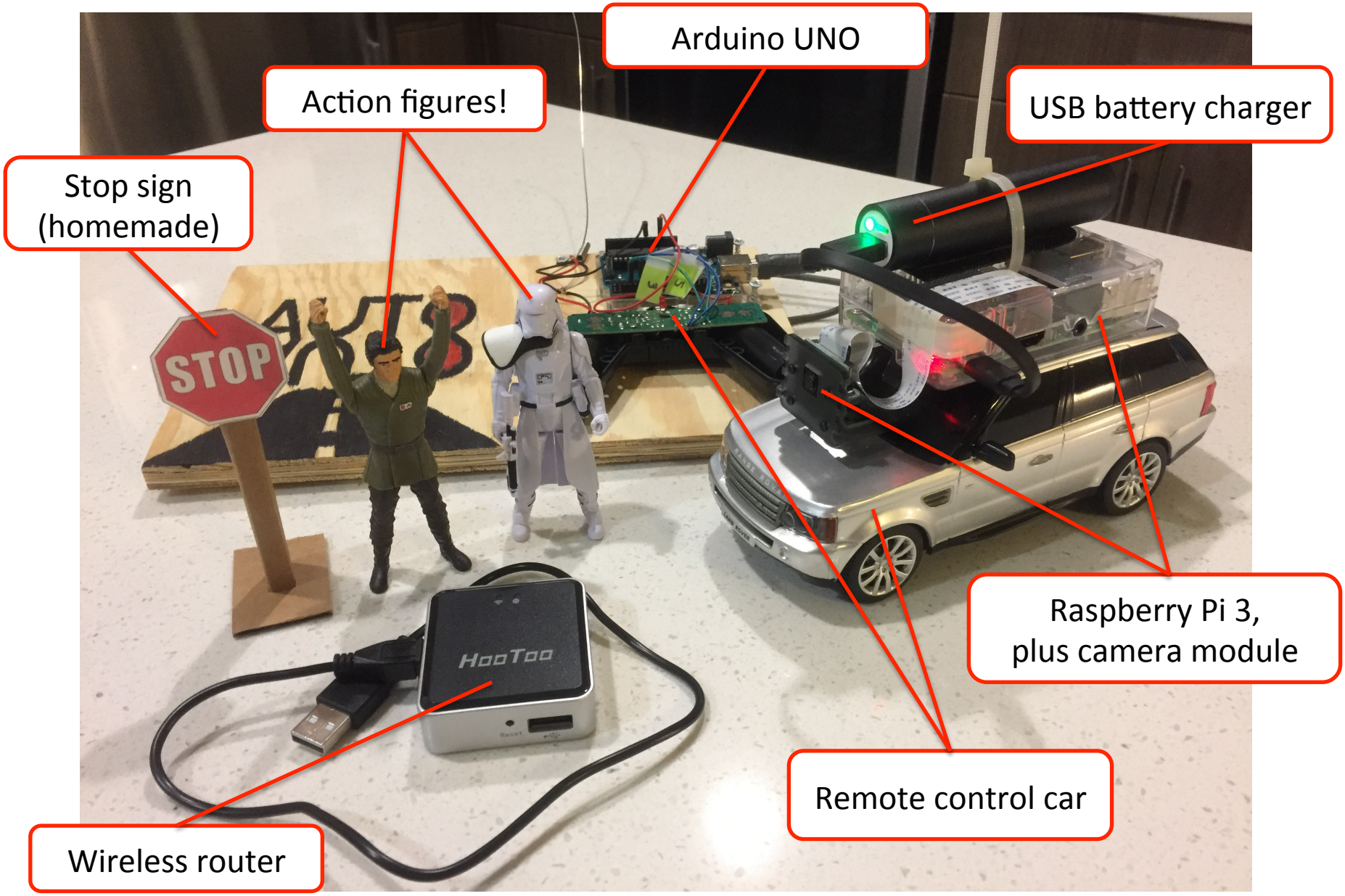
pauljyim

pseudoyim

Performance: Driving **(Not so good)**



Physical materials **(all items sold separately)**



galvanEye

A remote control car that drives itself using OpenCV and neural networks

Capstone project
galvanize Data Science Immersive
August 2016