KENTECH
Korea Institute of Energy Technology

Visionary Course – Energy AI
Week 15

June 10, 2022

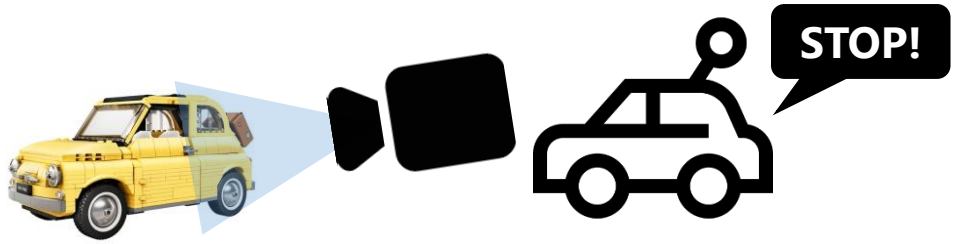
Seokju Lee

KENTECH
Korea Institute of Energy Technology

Week 15a – Driving with Object Detection (Challenge II)

Final Challenge II: Stop-and-Go

- **Goal**: Drive JetRacer under *challenging conditions*
- **Stop-and-Go mission**:
  - Obstacle items (car models) are given
  - Stop if an obstacle is detected, otherwise go!

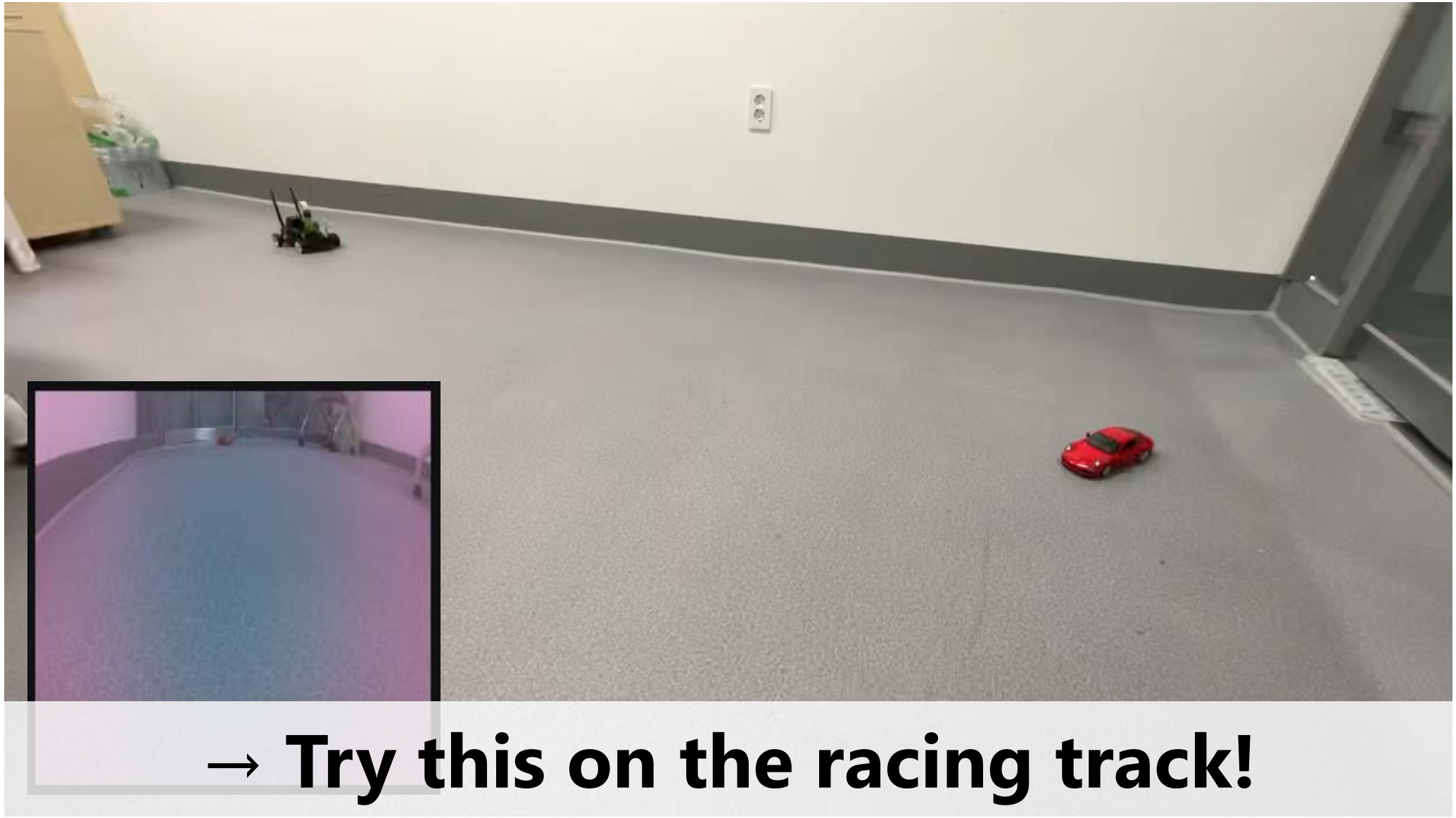


- **Schedule**
  - *June 10*: Racing with dynamic throttle mode (~1 hour), Stop-and-Go mission
  - *June 14*: Driving tests for Challenge II
  - *June 17*: Team presentation (creativity, 6~7 mins presentation + 2 mins Q&A per team)

→ Please record/upload the video presentation file!

- **Evaluation**
  - Lap time + missions clear (70%) and presentation quality (30%)
- **Awards**
  - Best award x1, excellence award x1, encouragement award x1

Demonstration



→ Try this on the racing track!

Python Code

'road_following_with_detection.ipynb'

(1)

```
import torch
import torchvision
import jetson.inference

device = torch.device('cuda')
model = torchvision.models.resnet18(pretrained=False)
model.fc = torch.nn.Linear(512, 2)
model = model.cuda().eval().half()

net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.3)
```

Road following

Detection

(1) Deploying two networks:

Road following + Object detection

(2)

```
STEERING_GAIN = 0.75
STEERING_BIAS = 0
car.throttle_gain = 0.73

const_throttle = 0.155
car.throttle = const_throttle

stop_height = 10

i = 0

while True:
    obj_height = 0
    clear_output(wait=True)
    image0 = camera.read()

    ### Steering ###
    image = preprocess(image0).half()
    output = model_trt(image).detach().cpu().numpy().flatten()
    x = float(output[0])
    y = float(output[1])
    # car.steering = x * STEERING_GAIN + STEERING_BIAS
    car.steering = 0

    ### Detection ###
    detections = net.Detect(jetson.utils.cudaFromNumpy(image0))
    for detection in detections:
        if net.GetClassDesc(detection.ClassID) == 'car':
            x1, y1 = int(detection.Left), int(detection.Top)
            x2, y2 = int(detection.Right), int(detection.Bottom)
            obj_height = y2 - y1
            '''
                -> Bounding box
                +---------------+
                | p1(x1, y1)    |
                |               |
                |               |
                +---------------+ p2(x2, y2)
            '''
    display("#{:05d}, {:s}, {:d}, {:s}, {:d}".format(i, 'Num-of-objs', len(detections), 'Height', obj_height))

    if obj_height > stop_height:
        car.throttle = 0
    else:
        car.throttle = const_throttle
    i += 1
```

Python Code

'road_following_with_detection.ipynb'

(1)

```
import torch
import torchvision
import jetson.inference

device = torch.device('cuda')
model = torchvision.models.resnet18(pretrained=False)
model.fc = torch.nn.Linear(512, 2)
model = model.cuda().eval().half()

net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.3)
```

Road following

Detection

(1) Deploying two networks:

Road following + Object detection

(2) Prediction with two models:

Road following + Object detection

(2)

```
STEERING_GAIN = 0.75
STEERING_BIAS = 0
car.throttle_gain = 0.73

const_throttle = 0.155
car.throttle = const_throttle

stop_height = 10

i = 0

while True:
    obj_height = 0
    clear_output(wait=True)
    image0 = camera.read()

    ### Steering ###
    image = preprocess(image0).half()
    output = model_trt(image).detach().cpu().numpy().flatten()
    x = float(output[0])
    y = float(output[1])
    # car.steering = x * STEERING_GAIN + STEERING_BIAS
    car.steering = 0

    ### Detection ###
    detections = net.Detect(jetson.utils.cudaFromNumpy(image0))
    for detection in detections:
        if net.GetClassDesc(detection.ClassID) == 'car':
            x1, y1 = int(detection.Left), int(detection.Top)
            x2, y2 = int(detection.Right), int(detection.Bottom)
            obj_height = y2 - y1
            '''
                -> Bounding box
                +---------------+
                | p1(x1, y1)    |
                |               |
                |               |
                +---------------+ p2(x2, y2)
            '''
    display("#{:05d}, {:s}, {:d}, {:s}, {:d}".format(i, 'Num-of-objs', len(detections), 'Height', obj_height))

    if obj_height > stop_height:
        car.throttle = 0
    else:
        car.throttle = const_throttle
    i += 1
```

→ Constant throttle mode as an example

→ Prediction on road following model (what we've covered so far)

→ Prediction on object detection model (additional model for detection)

→ Localize a bounding box with p1 and p2

→ If the size of bounding box is large, stop!

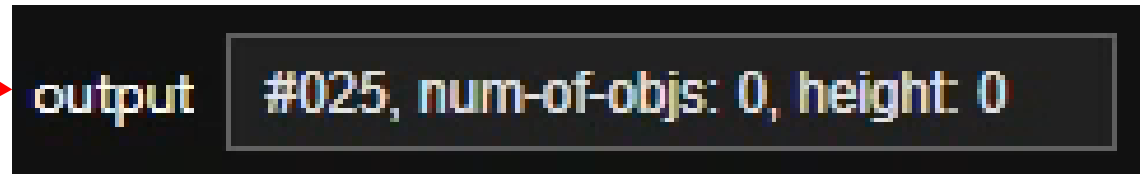
→ Otherwise, go!

→ Or, you can gradually reduce the speed depending on the obstacle's height!

Python Code with Visualization

```python
while True:
    obj_height = 0
    image0 = camera.read()

    ### Steering ###
    image = preprocess(image0).half()
    output = model_trt(image).detach().cpu().numpy().flatten()
    x = float(output[0])
    y = float(output[1])
    # car.steering = x * STEERING_GAIN + STEERING_BIAS
    car.steering = 0

    prediction = image0.copy()

    ### Detection ###
    detections = net.Detect(jetson.utils.cudaFromNumpy(image0))
    for detection in detections:
        if net.GetClassDesc(detection.ClassID) == 'car':
            x1, y1 = int(detection.Left), int(detection.Top)
            x2, y2 = int(detection.Right), int(detection.Bottom)

            prediction = cv2.rectangle(prediction, (x1, y1), (x2, y2), (0, 255, 0), 3)
            prediction = cv2.putText(prediction, str(net.GetClassDesc(detection.ClassID)), (x1, max(y1-10, 0)),
                                     cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
            obj_height = y2 - y1
            '''
                -> Bounding box
                +---------------+
                | p1(x1, y1)    |
                |               |
                |               |
                +---------------+ p2(x2, y2)
            '''
    output_text = "#{:03d}, {:s}{:d}, {:s}{:d}".format(i, 'num-of-objs: ', len(detections), 'height: ', obj_height)
    text_widget.value = output_text
    prediction_widget.value = bgr8_to_jpeg(prediction)

    if obj_height > stop_height:
        car.throttle = 0
    else:
        car.throttle = const_throttle
    i += 1
```

output #025, num-of-objs: 0, height: 0



→ You can check the number of detected objects, and their heights in real time.

Stop-and-Go Mission