

# Go To Autodrive

(GTA 게임 내 자율주행 구현)



[https://github.com/dnjstmdwo1234/AutoDrive\\_GTA5](https://github.com/dnjstmdwo1234/AutoDrive_GTA5)

기계공학과 3학년 원승재



경희대학교  
SW중심대학사업단



경희대학교 SW융합대학  
COLLEGE OF SOFTWARE KYUNG HEE UNIVERSITY

KHU LING+

# 목차

## I 제 작 배 경

- 1) GTA내 자율주행 관련 뉴스자료
- 2) 학부생의 기존의 자율주행 프로젝트에서 벗어난 새로운 시도
- 3) 들어가기 전에

## II 준 비 사 항

- 1) 사전설치
- 2) GTA 게임 내 화면 읽어오기
- 3) python 으로 GTA 게임 조작하기

## III 도로상황 인식

OpenCV 소개

- 1) 도로 색깔 추출
- 2) ROI 설정
- 3) grayscale
- 4) 도로와 비도로 구분
- 5) 신호등 인식

## IV 물체 인식

Tensorflow Object Detection API 소개

- 1) 화면 내 물체인식
- 2) 사람 인식
- 3) 차량 인식

## V 주행 알고리즘

- 1) 게임 내 자동차 제어
- 2) 다른 차가 감지되었을 때 경로추적을 기반으로 동작
- 3) 차량이 검출되지 않았을 때 도로탐색을 이용한 동작

## VI 공동 운영 repository 제안

- 1) 자율주행에 관심있는 경희대학교 학생들의 공동 운영 github repository 필요성

# I 제 작 배 경

## 1. GTA 내 자율주행 관련 뉴스자료

폭력 게임 표본이었던 GTA, 자율주행차 덕에 가치 재발굴 – 중앙일보 2017. 04. 19.

<https://news.joins.com/article/21492687>

광란의 살육파티 'GTA', 자율주행으로 이유있는 성공 – 중앙일보 2018. 11. 19.

<https://news.joins.com/article/21492687>

가상현실에서 자율주행차 주행 시험한다. – 로봇신문사 2017. 11. 2.

<https://news.joins.com/article/21492687>

최근 2년간 자율주행 시뮬레이션에 관한 관심도가 높아지고 오픈AI, 웨이모와 같은 글로벌 기업들이 자율주행 시뮬레이션 환경으로 GTA를 선택하여 테스트 해보는 것이 늘고 있다.

## 2. 학부생의 기존 자율주행 프로젝트에서 벗어난 새로운 시도

과거 학부생들이 진행한 자율주행과 관련한 여러 프로젝트를 보았을 때 아두이노나 라즈베리파이를 이용해서 모형 자동차를 만들고 모의 주행환경을 설정하여 테스트 하는 과정을 반복하였다. 그러나 현실에서 적용하기에 어려움이 있었고 따라서 현실과 매우 유사도가 높다고 알려진 GTA 라는 전 세계 범죄액션 게임 내에서 조금 더 현실과 가까운 환경에서 자율주행을 구사하여 자율주행 부문의 소프트웨어적 능력을 키우고자 제작하였다.

## 3. 들어가기 전에

GTA 내에서 차선을 인식하여 딥러닝 및 AI를 기반으로 자율주행을 시도해 보는 프로젝트 (<https://github.com/Sentdex>)는 오픈소스로 공개 되었고 google에서도 검색해보면 한국인 블로거들이 어렵지 않게 해당 코드를 이용할 수 있도록 주석처리를 해놓았다. 초기 진입장벽을 낮추기 위해 가장 기본이 되고 개발환경 세팅을 하기 위해 오픈소스로 공개된 코드를 일부 참조했고 참조시 주황색 글씨로 참조에 대한 사항을 명확히 밝혀 놓았다. 또한 오픈소스로 공개된 차선인식이 아닌 다른 차량의 주행 경로를 추적하여 해당 경로로 이동하는 알고리즘으로 자율주행을 구현하고자 했다.

# 표준비사항

## 1. python 다운로드

<https://www.python.org/downloads/release/python-359/>

### \*주의사항\*

opencv 설치를 위해 python 3.7 이하의 버전을 설치해 주세요. python 3.8 버전으로 설치 시 opencv 설치가 제한되는 경우가 생깁니다.

## 2. visual studio code 다운로드

<https://code.visualstudio.com/docs/?dv=win>

기본 환경으로

## 3. python 모듈 다운로드

### \*cmd창 입력 커맨드\*

### (1) 환경변수 설정

윈도우 -> 시스템 정보 -> 고급 시스템 설정 -> 환경변수 -> python 설치 경로를 path에 추가해 준다.

### (2) pillow 설치 및 다운로드

(cmd창 관리자 권한으로 실행)

**python -m pip install pillow**

### (3) opencv-python(python modul) 다운로드 \*3.4.7버전에서 진행

(cmd창 관리자 권한으로 실행)

**python -m pip install opencv-python==3.4.7**

### \*주의사항\*

설치가 진행되지 않을 경우 python --version을 통해 파이썬 버전을 확인해 주세요. 3.8 이상의 버전일 경우 해당 입력이 실행되지 않을 수 있습니다.

#### (4) numpy 설치 및 다운로드

(cmd창 관리자 권한으로 실행)

```
python -m pip install numpy
```

#### (5) sklearn 설치 및 다운로드

(cmd창 관리자 권한으로 실행)

```
python -m pip install sklearn
```

#### (6) tensorflow 설치 및 다운로드

(cmd창 관리자 권한으로 실행)

```
python -m pip install tensorflow
```

#### (7) matplotlib 설치 및 다운로드

(cmd창 관리자 권한으로 실행)

```
python -m pip install matplotlib
```

#### (8) tensorflow object detection api 설치 및 다운로드

[https://github.com/tensorflow/models/tree/master/research/object\\_detection](https://github.com/tensorflow/models/tree/master/research/object_detection)

github의 해당 repository에 들어가서 다운로드 받는다. 또한 object-detection api를 사용하기 위해서는 항상 폴도 경로를 object detection으로 설정해 놓고 작업을 해야한다. 가장 간단한 방법은 object-detection 안에 파일을 만들면 경로 설정에 대한 오류없이 이용할 수 있다.

## 1. GTA 게임화면 읽어오는 방법

\*주의사항\*

해당 코드는 초기 진입장벽을 낮추고자 <https://github.com/Sentdex> 코드를 참조했습니다.

GTA를 이용한 자율주행의 제일 첫 번째 순서는 게임화면을 읽어오는 것이다



위 화면은 800x600 으로 설정한 GTA 게임 실행 화면이다.

```
import numpy as np
from PIL import ImageGrab
import cv2

# 무한루프를 돌면서
while (True):
    # (0,40)부터 (800,600)좌표까지 창을 만들어서 데이터를 저장하고 screen 변수에 저장합니다
    screen = np.array(ImageGrab.grab(bbox =(0 ,40 ,800 ,600 )))

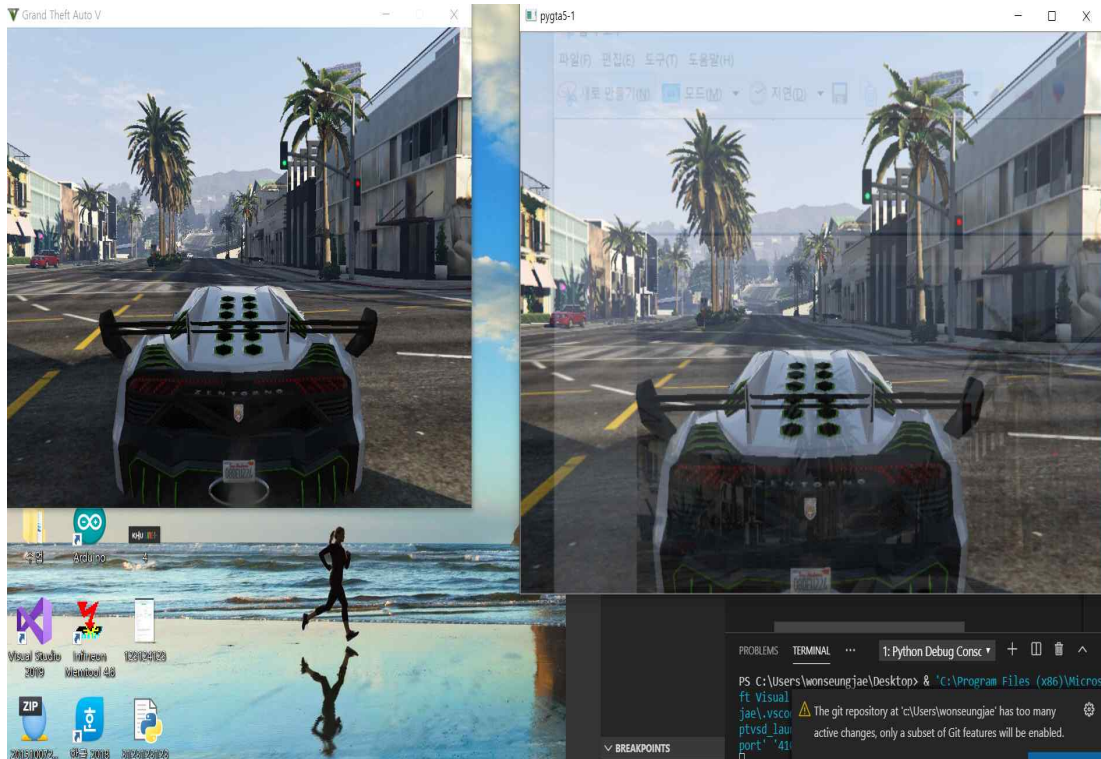
    # pygta5-1이라는 이름의 창을 생성하고 이 창에 screen 이미지를 뿌려줍니다
    cv2.imshow('pygta5-1', cv2.cvtColor(screen, cv2.COLOR_BGR2RGB))

    # 'q'키를 누르면 종료합니다
```



```
if cv2.waitKey(25) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    break
```

pillow 모듈에서 imagegrab 이라는 함수를 이용해서 이미지를 읽어온다. 위코드를 실행하게 되면



다음과 같이 가로 0~800픽셀 부분과 세로 40~600픽셀 부분을 이미지로 불러와서 창을 생성한다. 해당 이미지를 기반으로 여러 가지 OpenCV에서 제공하는 함수를 이용해 얻고자 하는 데이터를 쉽게 얻을 수 있도록 처리한다.

## 2. 파이썬 파일에서 GTA 게임 내 조작법

\*주의사항\*

해당 코드는 초기 진입장벽을 낮추고자 <https://github.com/Sentdex> 코드를 참조했습니다.

```
import ctypes
import time

SendInput = ctypes.windll.user32.SendInput
W = 0x11
A = 0x1E
S = 0x1F
D = 0x20
# C struct redefinitions
PUL = ctypes.POINTER(ctypes.c_ulong)

class KeyBdInput(ctypes.Structure):
    _fields_ = [("wVk", ctypes.c_ushort),
                ("wScan", ctypes.c_ushort),
                ("dwFlags", ctypes.c_ulong),
                ("time", ctypes.c_ulong),
                ("dwExtraInfo", PUL)]

class HardwareInput(ctypes.Structure):
    _fields_ = [("uMsg", ctypes.c_ulong),
                ("wParamL", ctypes.c_short),
                ("wParamH", ctypes.c_ushort)]

class MouseInput(ctypes.Structure):
    _fields_ = [("dx", ctypes.c_long),
                ("dy", ctypes.c_long),
                ("mouseData", ctypes.c_ulong),
                ("dwFlags", ctypes.c_ulong),
                ("time", ctypes.c_ulong),
                ("dwExtraInfo", PUL)]

class Input_I(ctypes.Union):
    _fields_ = [("ki", KeyBdInput),
                ("mi", MouseInput),
                ("hi", HardwareInput)]

class Input(ctypes.Structure):
    _fields_ = [("type", ctypes.c_ulong),
                ("ii", Input_I)]

# Actuals Functions
# 키보드를 누르는 함수
def PressKey(hexKeyCode):
    extra = ctypes.c_ulong(0)
    ii_ = Input_I()
```



```

ii_.ki = KeyBdInput(0 , hexKeyCode, 0x0008 , 0 , ctypes.pointer(extra))

x = Input(ctypes.c_ulong(1 ), ii_)
ctypes.windll.user32.SendInput(1 , ctypes.pointer(x), ctypes.sizeof(x))

# 키보드를 떼는 함수
def ReleaseKey (heyKeyCode ):
    extra = ctypes.c_ulong(0 )
    ii_ = Input_I()
    ii_.ki = KeyBdInput(0 , heyKeyCode, 0x0008 | 0x0002 , 0 , ctypes.pointer(extra))

    x = Input(ctypes.c_ulong(1 ), ii_)
    ctypes.windll.user32.SendInput(1 , ctypes.pointer(x), ctypes.sizeof(x))

if __name__ == '__main__':
    PressKey(0x11 )
    time.sleep(1 )
    ReleaseKey(0x11 )
    time.sleep(1 )

```

파이썬 파일에서 명령을 하면 GTA 게임 내 조작을 하기위한 코드이다. 사용되는 함수는 PressKey 와 ReleaseKey 함수이다. 먼저 PressKey 함수의 경우 원하는 키를 입력받아서 해당 키를 GTA 게임 내에 입력해주는 함수이다. 또한 ReleaseKey는 현재 입력되고 있는 것을 중단 시켜 주는 함수이다.

# Ⅲ 도로상황 인식

## OpenCV 소개

**\*출처\*** 나무위키

도로 인식을 위해서 먼저 opencv에 대해서 소개한다. 오픈소스 컴퓨터 비전 라이브러리 중 하나로 크로스 플랫폼과 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다. 현재 조금이라도 영상처리가 들어간다면 필수적으로 사용하게 되는 라이브러리 이다. 기능이 방대하기 때문에 OpenCV를 다 사용할 줄 아는 것만으로도 영상처리에 있어서 상당한 반열에 오를 수 있다. BSD 라이선스를 사용하고 있으므로 상업적으로 이용이 가능하다. 주요 알고리즘으로는 이진화, 노이즈제거, 외곽선 검출, ROI 설정, 이미지 변환 등이 있다.

## 1. 도로 색깔 추출

먼저 GTA 게임 내 화면에서 도로에 해당하는 영역을 캡처한다.



이미지 색상분석 프로그램을 통해 가장 밝은 색상과 어두운 색상을 추출한다.

<http://www.cssdrive.com/imagepalette/index.php>

추출된 색상은 헥사코드로 되어있는 색상이므로 해당 색상을 RGB로 변환해주는 프로그램을 이용해 색상을 RGB로 나타내준다.

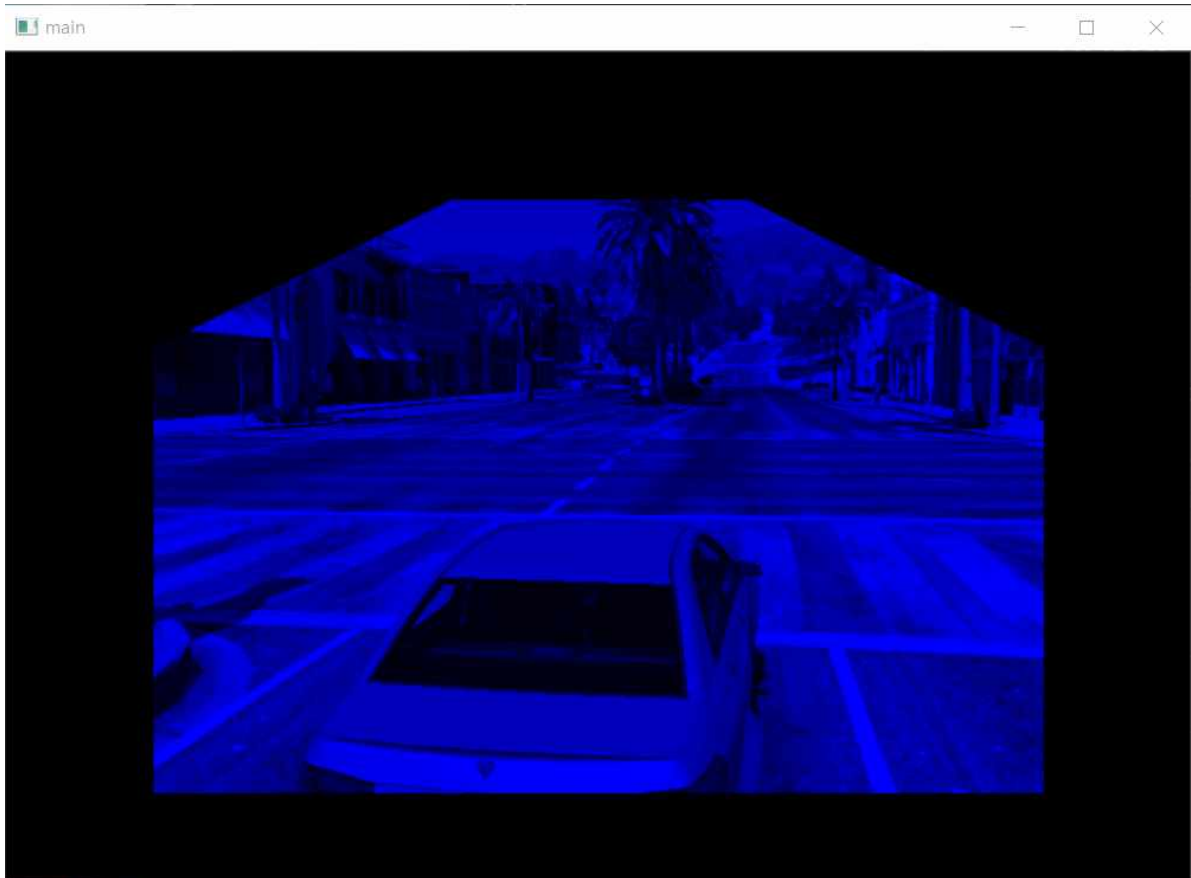
## 2. ROI 설정

opencv 내 함수 roi 사용

```
vertices = np.array([[100 ,500 ], [100 ,200 ], [300 ,100 ],[500 ,100 ],  
[700 ,200 ], [700 ,500 ]], np.int32)
```

```
# Region of Interest : 관심영역을 설정하는 함수  
def roi (img , vertices ):  
    # img 크기만큼의 영행렬을 mask 변수에 저장하고  
    mask = np.zeros_like(img)  
  
    # vertices 영역만큼의 Polygon 형상에만 255의 값을 넣습니다  
    masked = cv2.fillPoly(mask, vertices, 255 )  
  
    # img와 mask 변수를 and (비트연산) 해서 나온 값들을 masked에 넣고 반환합니  
    masked = cv2.bitwise_and(img, masked)  
    return masked
```

먼저 자르고 싶은 영역을 선택하여 꼭지점들의 좌표(픽셀값)를 입력해준다. 이후 roi 함수에 원래 이미지와 자르고 싶은 영역을 입력해주면 자르고 싶은 영역을 제외한 나머지 부분들에 있어서 검정색으로 화면이 추출된다.



### 3. RGB -> GrayScale

```
processed_img = cv2.cvtColor(mark, cv2.COLOR_BGR2GRAY)
```

BGR 화면을 OpenCV의 BGR2GRAY 라는 함수를 이용해서 grayscale 화면으로 바꿔준다.



### 4. 도로와 비도로 구분

```
blue_threshold = 160
green_threshold = 160
red_threshold = 160
bgr_threshold = [blue_threshold, green_threshold, red_threshold]
thresholds = (image[:, :, 0] > bgr_threshold[0]) & \
              (image[:, :, 1] > bgr_threshold[1]) & \
              (image[:, :, 2] > bgr_threshold[2])
mark[thresholds] = [255, 255, 255]
processed_img = cv2.cvtColor(mark, cv2.COLOR_BGR2GRAY)

blue_threshold = 50
green_threshold = 50
red_threshold = 50
bgr_threshold = [blue_threshold, green_threshold, red_threshold]
thresholds = (image[:, :, 0] < bgr_threshold[0]) & \
```

```

| (image[:,1 ] < bgr_threshold[1 ]) &
| (image[:,2 ] < bgr_threshold[2 ])
mark[thresholds] = [255 ,255 ,255 ]

```

1번에서 진행했던 과정에서 도로의 가장 밝은 부분의 픽셀과 가장 어두운 부분의 픽셀을 알 수 있었다. 추출결과 RGB 값이 160보다 어둡고 50보다 밝을 때 도로로 추정한다는 것이라 정했다. 따라서 이미지상의 픽셀들을 비교해 가면서 160보다 크거나 50보다 작은 값들을 모두 흰색으로 처리하여 도로를 구분했다.



다음과 같이 최종 이미지 변환 결과를 볼 수 있는데 차선이 매우 뚜렷하게 보이는 것을 알 수 있고 뿐만 아니라 도로에 그려진 횡단보도와 직진, 우회전, 좌회전 화살표 또한 매우 뚜렷하게 보이는 것을 확인할 수 있다.

## 5. 신호등 인식

```
screen_ = np.array(ImageGrab.grab(bbox=(250,100,450,300)))
screen_ = cv2.cvtColor(screen_, cv2.COLOR_BGR2RGB)
traffic_light_red(screen_)
```

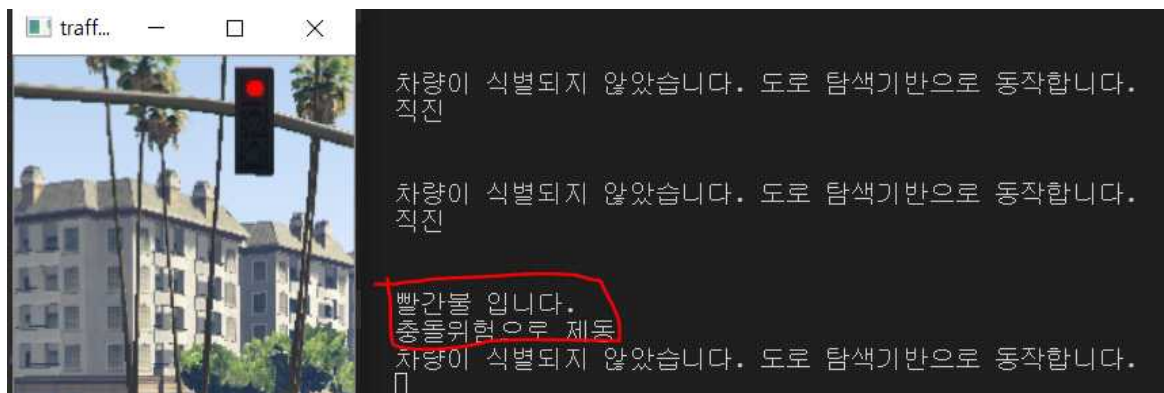
게임 화면에서 신호등이 존재할 것이라고 예상되는 부분을 200x100 픽셀로 설정하여 해당 부분에 대해서만 이미지 처리를 해주었다. OpenCV는 기본적으로 BGR 이미지를 송출하기 때문에 BGR 이미지를 우리가 평소에 자주 볼 수 있는 RGB 이미지로 변환해 주었다.

```
def traffic_light_red (screen):
    num = 0
    b, g, r = screen[:, :, 0], screen[:, :, 1], screen[:, :, 2]
    shape = b.shape
    shape1 = shape[0]*shape[1]
    b = b.reshape(shape1)
    shape = r.shape
    shape1 = shape[0]*shape[1]
    r = r.reshape(shape1)
    shape = g.shape
    shape1 = shape[0]*shape[1]
    g = g.reshape(shape1)

    for i in range(0,100):
        if (b[i] > 180 & g[i] < 40 & r[i] < 40):
            num += 1

    if num >= 10:
        print("빨간불 입니다.")
        control_car(5)
```

이미지 변환 후에 각각 픽셀에 해당하는 RGB 값을 하나의 배열로 만들기 위한 shape 작업을 진행했다. 이후 200 개의 픽셀 중 특정 개수 이상의 픽셀이 빨간색으로 인식되면 "빨간불입니다." 라는 메시지를 출력하고 차를 정지시킨다.



# IV 물체 인식 하는법

## object-detection 소개

**\*출처\*** 위키디피아

컴퓨터 비전 및 이미지 처리와 관련된 컴퓨터 기술로 디지털 이미지 및 비디오에서 특정 클래스의 의미 객체 탐지를 처리하는 것을 말한다. 보통 머신러닝 방식과 딥러닝 방식의 차이가 있고 일반적으로 CNN을 기반으로 한다.

## tensorflow소개

**\*출처\*** 나무위키

딥러닝과 기계학습 분야를 일반인들도 사용하기 쉽도록 다양한 기능을 제공하는 구글의 오픈소스이다. 데이터 플로우 그래프를 통한 풍부한 표현력, 아이디어 테스트에서 서비스 단계까지 이용 가능, 계산 구조와 목표 함수만을 정의하면 자동으로 미분 계산을 처리 하는 등의 특징을 가진다.

## 1. 화면 내 물체 인식

**\*주의사항\***

해당 코드는 초기 진입장벽을 낮추고자 tensorflow object detection api 예제코드를 참조했습니다.

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys

import tarfile
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
#from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
from PIL import Image
from grabscreen import grab_screen
#from IPython import get_ipython
import cv2
# This is needed to display the images.
#get_ipython().magic('matplotlib inline')

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("../")
```



```

# ## Object detection imports
# Here are the imports from the object detection module.

from utils import label_map_util
from utils import visualization_utils as vis_util

# # Model preparation
# What model to download.
MODEL_NAME = 'ssd_mobilenet_v1_coco_2017_11_17'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for the
object detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90

# 하나의 그래프(노드&엣지로 이루어진 하나의 시스템)를 생성합니다
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    # CKPT (저장된 가중치) 파일을 불러온 후 모델을 복원하는 코드
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

# 라벨, 카테고리 데이터를 불러오는 코드
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes = NUM_CLASSES, use_display_name = True )
category_index = label_map_util.create_category_index(categories)

# image vector를 numpy array로 변경하는 함수
def load_image_into_numpy_array (image ):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape((im.height, im_width, 3 )).astype(np.uint8)

# 위에서 복원한 모델이 있는 그래프에서
with detection_graph.as_default():
    # 세션을 하나 실행한다
    with tf.Session(graph =detection_graph) as sess:
        while True :

```

```

# 아래 2줄이 Object Detection 예제코드에서 수정된 부분이다
# grab_screen을 사용해서 해당 윈도우를 캡처하는 코드
screen = cv2.resize(grab_screen(region =(0 ,40 ,800 ,600 )), (800 ,600 ))
image_np = cv2.cvtColor(screen, cv2.COLOR_BGR2RGB)

# 이미지를 인식해서 box, score, classes를 그려주는 코드
image_np_expanded = np.expand_dims(image_np, axis =0 )
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

scores = detection_graph.get_tensor_by_name('detection_scores:0')
classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

(boxes, scores, classes, num_detections) = sess.run([boxes, scores, classes,
num_detections], feed_dict ={image_tensor : image_np_expanded})

# 실제로 이 코드가 box와 label을 visualize해주는 코드인듯하다
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates =True ,
    line_thickness =8 )

# 콘솔창을 띄운다
cv2.imshow('pygta-p17', image_np)

# q키를 누르면 종료한다
if cv2.waitKey(25 ) & 0xff == ord ('q'):
    cv2.destroyAllWindows()
    break

```



위와 같이 예제 코드가 제공하는 90개의 class로 등록된 물체에 대해서는 이미지로 인식했을 때 해당 물체와 일치정도를 판단하는 %값과 이미지 추정 box를 통해 표시되는 것을 알 수 있다.

## 2. 사람 인식

```
for i,b in enumerate (boxes[0 ]):
    if classes[0 ][i] == 1 :
        apx_distance = round (((1 - (boxes[0 ][i][3 ] - boxes[0 ][i][1 ]))**4 ),1 )
        if apx_distance <= 0.1 :
            print ("사람이 감지 되었습니다. 정지하세요.")
            control_car(5 )
```

위에서 설명한 tensorflow object-detection api 의 기본 예제에서 사람을 추출하는 코드이다. 사람을 나타내는 class 번호가 1이므로 classes[0][i] 가 1이고 추정거리가 매우 가까울 때 충돌위험이 있다고 판단하고 제동하는 동작을 하도록 했다.



1 대 차가 식별됨, 그 중 대표 차 1대 추출  
 차 추정값 : 56.0 % 일치  
 중심편차 : 0.6069774627685547  
 내 차와의 거리(추정) : 0.8  
 충돌위험으로 제동  
 사람이 감지 되었습니다. 정지하세요.  
 충돌위험으로 제동  
 사람이 감지 되었습니다. 정지하세요.  
 충돌위험으로 제동  
 사람이 감지 되었습니다. 정지하세요.  
 충돌위험으로 제동  
 사람이 감지 되었습니다. 정지하세요.  
 충돌위험으로 제동  
 사람이 감지 되었습니다. 정지하세요.  
 충돌위험으로 제동  
 사람이 감지 되었습니다. 정지하세요.

위와 같이 주행 중에 일정 거리 내의 사람을 발견하게 된다면 모든 명령을 취소하고 차가 정지한다.

### 3. 차량 인식

```
a = []
x = []
y = []
min_i = 0
for i,b in enumerate (boxes[0 ]):
    #          car          bus          truck
    if classes[0 ][i] == 3 or classes[0 ][i] == 6 or classes[0 ][i] ==
8 :
        if scores[0 ][i] >= 0.5 :
            apx_distance = round (((1 - (boxes[0 ][i][3 ] - boxes[0
][i][1 ]))*4 ),1 )
            mid_x_ = (boxes[0 ][i][1 ]+boxes[0 ][i][3 ])/2
            mid_y_ = (boxes[0 ][i][0 ]+boxes[0 ][i][2 ])/2
            a.append(apx_distance)
            x.append(mid_x_)
            y.append(mid_y_)
            num_1 = 1

for i in range (0 ,len (a)):
    if (i ==0 ):
        min_a = abs ((b[1 ] - a[0 ])/b[1 ])*100
        min_x = abs ((b[0 ] - x[0 ])/b[0 ])*100
        min_y = abs ((b[2 ] - y[0 ])/b[2 ])*100
        min = min_x +min_y
        min_i = i

    else :
        min_a = abs ((b[1 ] - a[i])/b[1 ])*100
        min_x = abs ((b[0 ] - x[i])/b[0 ])*100
        min_y = abs ((b[2 ] - y[i])/b[2 ])*100
        ch = min_x +min_y
        if (min > ch):
            min = ch
            min_i = i

if (num_1 == 0 ):

    x.append(0 )

if x[0 ] != 0 :
    min_x = abs ((b[0 ] - x[min_i])/b[0 ])*100
    min_y = abs ((b[2 ] - y[min_i])/b[2 ])*100
    min_a = abs ((b[1 ] - a[min_i])/b[1 ])*100
```

```

print (len (a)+1 ,"대 차가 식별됨, 그 중 대표 차 1대 추출")
print ("이전 object와의 오차율(거리) : ", min_a,"% 오차율")
print ("이전 object와의 오차율(x좌표) : ", min_x,"% 오차율")
print ("이전 object와의 오차율(y좌표) : ", min_y,"% 오차율")
print ("차 추정값 : ", round (scores[0 ][min_i]*100 ), "% 일치")
mid_x = (boxes[0 ][min_i][1 ]+boxes[0 ][min_i][3 ])/2
print ("중심편차 : ", mid_x)
apx_distance = round (((1 - (boxes[0 ][min_i][3 ] - boxes[0
][min_i][1 ]))*4 ),1 )
a.append(apx_distance)

print ("내 차와의 거리(추정) : ", apx_distance)

```

차에 해당하는 class 가 3번 버스와 트럭에 해당하는 class 번호가 각각 6번과 8번이다. 그런데 여러대의 차를 인식할 경우 주행 경로 추정에 혼란이 있을 수 있다. 이를 위해 이전 이미지와 비교하여 가장 적합한 차를 정해서 해당 차의 주행 경로를 따라가는 것으로 정했다. 먼저 a,x,y 배열은 각각 내 차와의 거리, 해당 차의 x 좌표, y 좌표를 의미한다. tensorflow object detection api class에등록된 모든 차종들 ex) car, bus truck의 내 차와의 거리, x좌표, y좌표를 추출해서 배열에 추가한다. 이때 50% 이상의 확률로 해당 차종이라고 인식하였을 때만 위와 같이 동작한다. 배열 b[0,1,2] 는 각각 이전 x좌표, 거리, y좌표를 의미한다. 두 번째 for문은 추출한 차들 중 이전 이미지와 비교하였을 때 주행 경로를 추적하기 가장 좋은 차를 찾는 과정이다. 이전 이미지의 거리, x좌표, y좌표를 바탕으로 현재 이미지에서 추출한 모든 차들의 x,y,z 좌표의 오차율을 각각 계산해서 오차율이 가장 작은 것을 선택하여 해당 차종을 추적하게 하도록 구현했다. 만약 현재 이미지에서 차를 인식하지 못하였으면 해당 과정을 생략하고 도로 추정으로 다시 넘어간다.

# V 주행 알고리즘

## 1. 자동차 제어 코드

```
def control_car (num ):
    if (num ==1 ):
        PressKey(W)
        time.sleep(1 )
        ReleaseKey(W)
        print ("직진")

    elif (num ==2 ):
        PressKey(A)
        PressKey(W)
        time.sleep(1 )
        ReleaseKey(A)
        PressKey(W)
        time.sleep(1 )
        ReleaseKey(W)
        print ("좌회전")

    elif (num ==3 ):
        PressKey(D)
        PressKey(W)
        time.sleep(1 )
        ReleaseKey(D)
        PressKey(W)
        time.sleep(1 )
        ReleaseKey(W)
        print ("우회전")

    elif (num ==5 ):
        ReleaseKey(D)
        ReleaseKey(W)
        ReleaseKey(A)
        ReleaseKey(S)
        time.sleep(1 )
        print ("충돌위험으로 제동")

    else :
        PressKey(S)
        time.sleep(3 )
        ReleaseKey(S)
        PressKey(W)
        time.sleep(1 )
        ReleaseKey(W)
```



```
print ("후진")
```

control\_car 이라는 함수를 만들어서 원하는 차량 구동에 맞추어 숫자를 입력받는다. 1번은 직진, 2번은 좌회전, 3번은 우회전, 5번은 사람이나 다른 차가 너무 가까이 감지되었을 경우 긴급제동을 할 수 있게 하는 제동 코드, 그 밖의 숫자가 입력되면 후진을 한다. 준비단계에서 GTA내 조작을 위해 만들었던 PressKey 함수와 ReleaseKey 함수를 적절히 이용하여 구현한다.

## 2. 다른 차가 감지되었을 때 경로 추적을 기반으로 동작

```
if b[1] < a[min_i]:
    if mid_x > 0.1 and mid_x < 0.9 :
        control_car(1)
    elif mid_x <= 0.1 :
        control_car(2)
    elif mid_x >= 0.9 :
        control_car(3)
    else :
        print ("a")
else :
    if mid_x > 0.2 and mid_x < 0.8 :
        control_car(5)
```

직전의 이미지와 비교하였을 때 감지한 차의 거리가 늘어날 경우 감지한 차가 이동한다고 생각하여 경로를 추적한다. 중심으로부터 멀리떨어지지 않았으면 직진을 한다. 또한 중심으로부터 왼쪽으로 떨어지면 좌회전, 오른쪽으로 떨어지면 우회전을 한다. 만약 이전 이미지에 감지했을 때 보다 현재 이미지가 더 가깝고 중심으로부터 이미지의 편차가 있지 않다면 차가 멈춰있는데 점점 가까워지는 것이라 판단하고 충돌위험이 있다고 판단한다.

object\_detection



www

2 대 차가 식별됨, 그 중 대표 차 1대 추출  
이전 object와의 오차율(거리) : 191.48377787787007 % 오차율  
이전 object와의 오차율(x좌표) : 77.67238809188531 % 오차율  
이전 object와의 오차율(y좌표) : 21.363698191449704 % 오차율  
차 추정값 : 81.0 % 일치  
중심편차 : 0.141950786113739  
내 차와의 거리(추정) : 0.3

직진

ITS

### 3. 다른 차량이 감지되지 않았을 경우 일반적인 도로탐색을 이용한 동작

```
else :
    print ("차량이 식별되지 않았습니다. 도로 탐색기반으로 동작합니다.")
    if (num ==0 ):
        count = 0
        control_num = compare_1_2_3(image1_color,image2_color,image3_color)
        control_car(control_num)
        control_last_num = control_num

        num += 1
    else :
        control_num = compare_1_2_3(image1_color,image2_color,image3_color)

        if (control_num == control_last_num):
            count += 1

        else :
            count = 0
            if (count ==3 & control_last_num != 1 ):
                print ("같은 입력이 3번 반복되었습니다. 차가 부딪힌 것으로 추정합니
다.")
                control_car(4 )
                count = 0
            else :
                control_car(control_num)
                control_last_num = control_num
```

차량이 인식되지 않았을 경우 현재 도로를 바탕으로 주행을 하게 했다. 이때 직진을 제외한 좌회전, 우회전 신호가 3번 이상 반복되면 차가 충돌하거나 주행 경로를 벗어난 것이라고 판단하고 차를 일정 시간 후진시킨 후에 다시 도로 상황을 인식하고 주행하도록 했다.

```
if cv2.waitKey(25 ) & 0xFF == ord ('r'):
    print ("차가 부딪혔습니다. 후진하겠습니다.")
    control_car(4 )
    break
```

만약 부딪혔을 경우 사용자의 입력을 통해 후진하고 경로를 재 탐색할 수 있다.

main



www

차량이 식별되지 않았습니다. 도로 탐색기만으로 동작합니다.  
직진

차량이 식별되지 않았습니다. 도로 탐색기만으로 동작합니다.  
직진

차량이 식별되지 않았습니다. 도로 탐색기만으로 동작합니다.  
직진

TS

Exceptions

ht Exceptio...

차량이 식별되지 않았습니다. 도로 탐색기만으로 동작합니다.  
□

# VI 공동운영 repository 제안

## 1. 자율주행에 관심있는 경희대학교 학생들의 공동 운영 github repository 필요성

완벽한 자율주행을 구현하기 위해서는 다양한 기능들이 필요하다. 따라서 자율주행에 관심이 있는 경희대학교 학생들이 GTA 내 자율주행 시뮬레이션을 주제로 하나의 github repository를 만들면 좋겠다고 생각했다. 본 프로젝트에서 진행한 구현들에 대해 실습해 본다면 앞으로 다양한 학생들이 자율주행에 필요하다고 생각하는 기능들에 대해서 스스로 구현하고 repository에 업로드할 수 있다. 다른 사람이 업로드한 코드를 학습하고 또 다른 기능을 구현할 수 있는 선순환 구조일 뿐만 아니라 GTA 내 자율주행 시뮬레이션이라는 프로젝트의 완성도가 점점 더 높아질 것이라고 생각한다.

