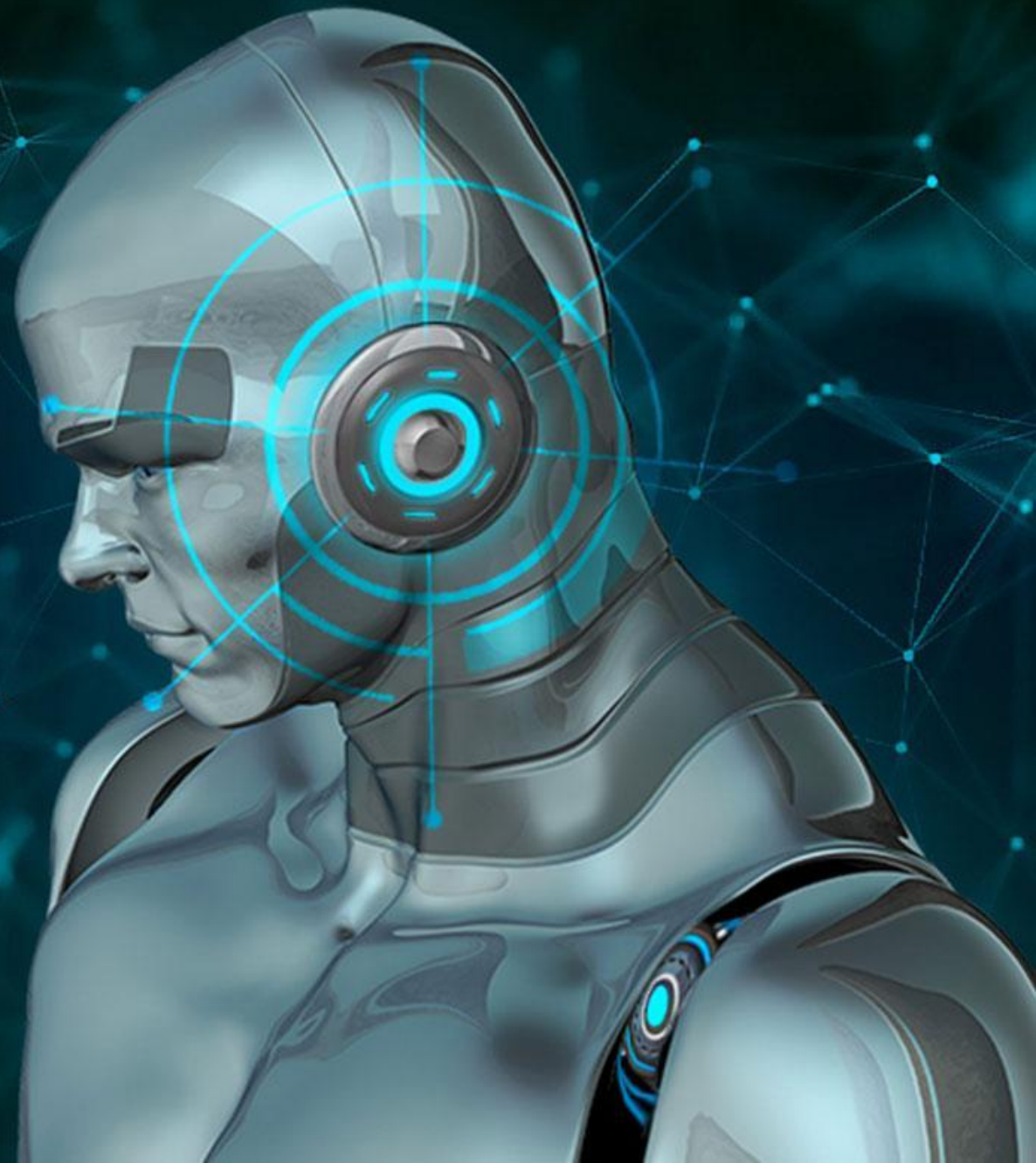
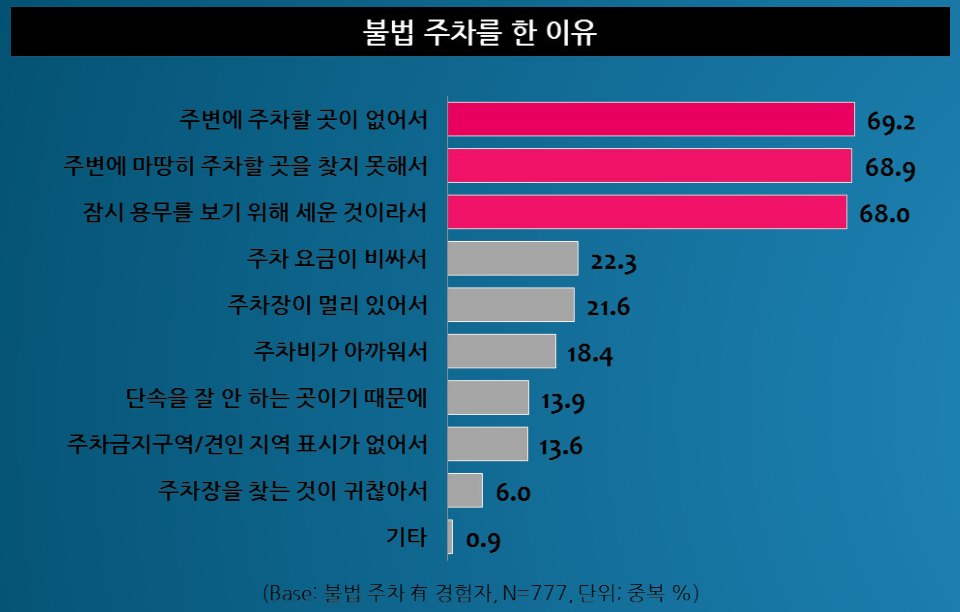
스마트 주차장 관리

차량 출입 관리 + 객체를 인식하는 cv

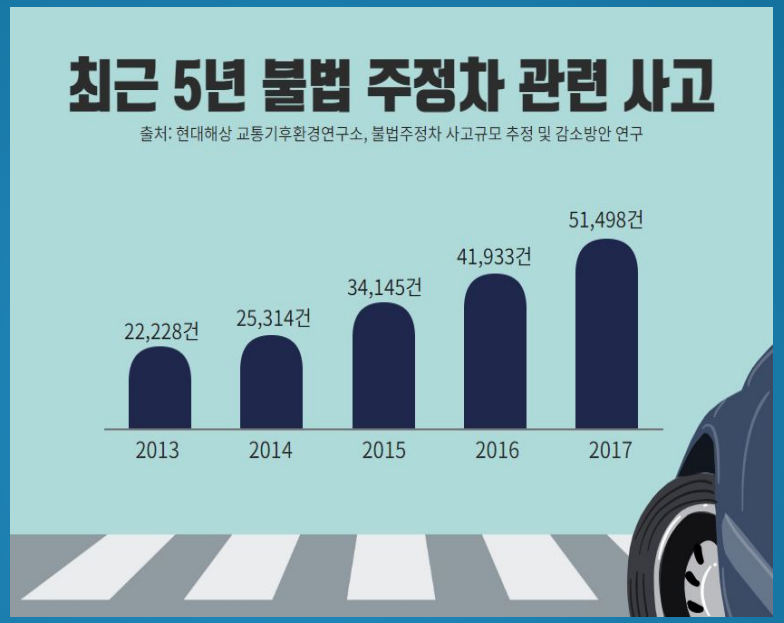


Contents

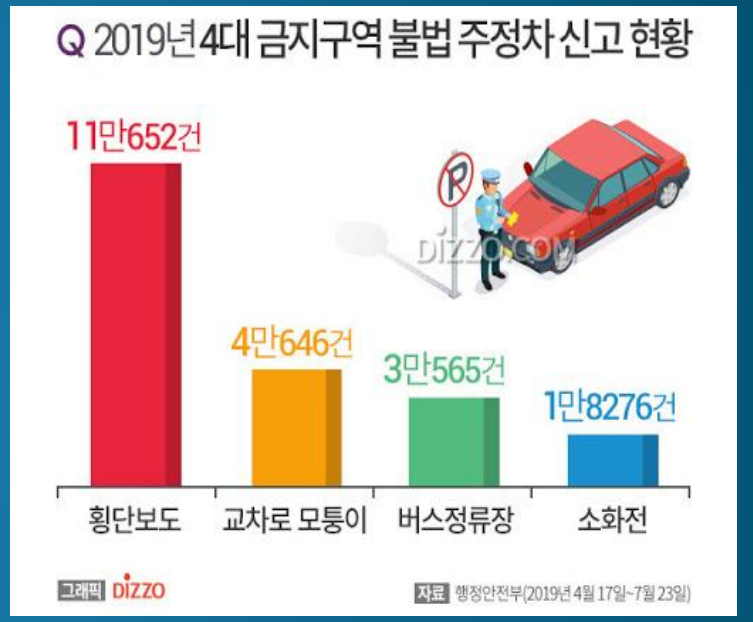
01. 분석 배경



불법주차의 이유



불법 주정차 사고



불법 주정차 신고 현황

전기車 아닌데 주차…충전구역은 무법지역

- **일반차량 주차시 과태료 내지만 100명 이상 대형 주차장만 적용, 공용주택에선 단속 못해**
- 서울시에 따르면 서울에 설치된 전체 전기차 전용 충전구역 4219곳 중 시가 단속할 수 있는 구역은 43곳(1%)에 불과하다.
- 15일 국토교통부에 따르면 지난달 기준 국내에 등록된 전기차는 13만5391대로 집계됐다.

한국경제 2021.02.15.

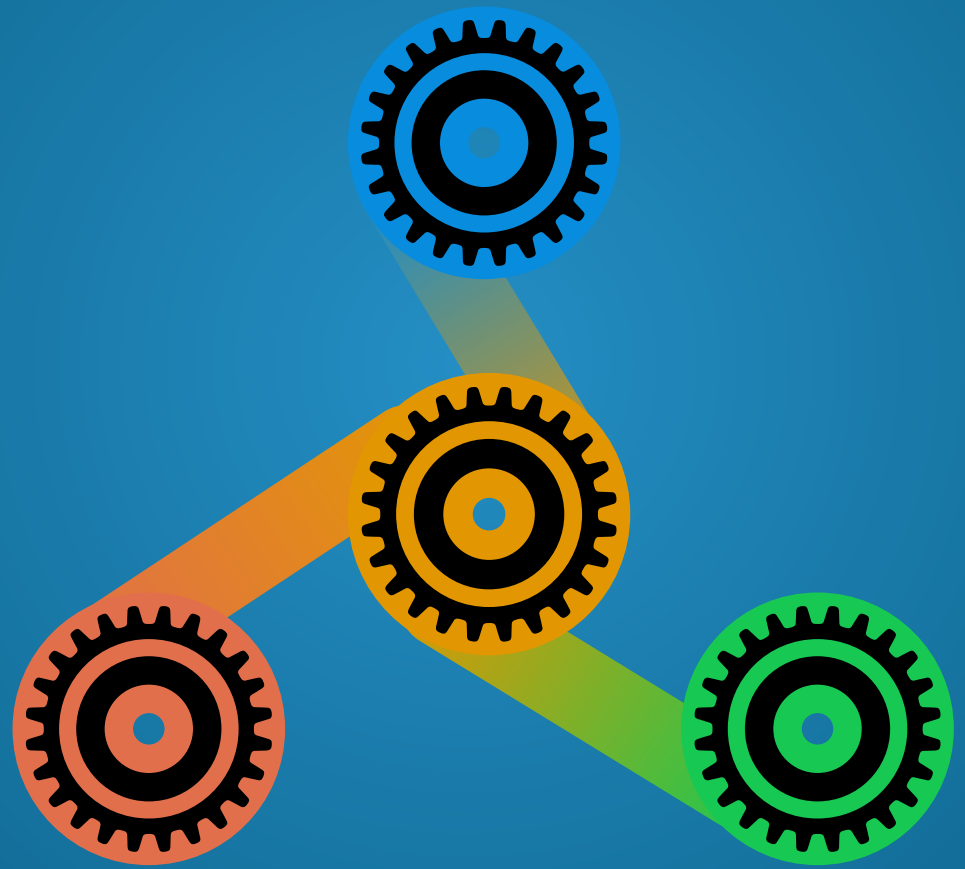
주차 불편 순천향대서울병원, 코로나19로 더 심각

- **내원객 자차 이용 늘면서 혼잡 심화, 병원 "대중교통 이용 등 협조" 요청**
- “코로나19 걱정 때문에 자차 이용 비율은 몇 배 이상 증가한 것으로 파악하고 있다”
- “병원 홈페이지에 표시된 곳(P1~P5) 외 발렛 서비스를 제공하는 몇 군대를 합치면 총 320여대 정도이며 피크 타임 기준 하루 평균 이용 고객들은 700~800여대 정도 된다”고 설명했다.

데일리메디, 2020.10.30

주차난을 개선하면 얻는것

교통체증 해소



교통사고 방지

긴급상황시
통행 원활

01. 분석 목적

추가적인 장비없이
단순 영상 감지만으로
저비용 고효율의 효과

경제적 측면

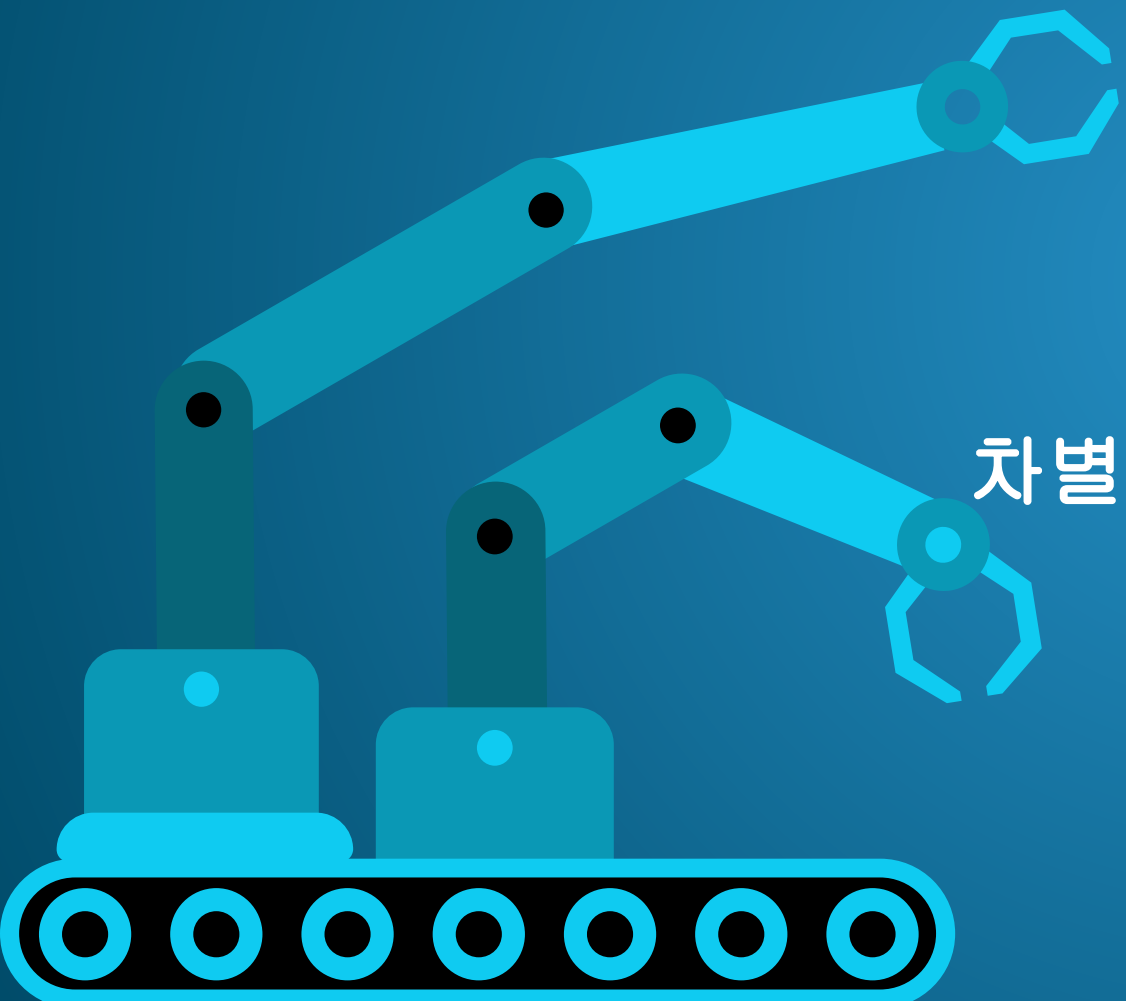
법적제재로 부터 원활,
특정 위치에서 측정하여
사람이 아닌 차량만 감지

법, 제도적 측면

기존 CCTV의 사용으로
실내외에서 측정하여
비대면과 함께 안전성 보장

사회적 측면

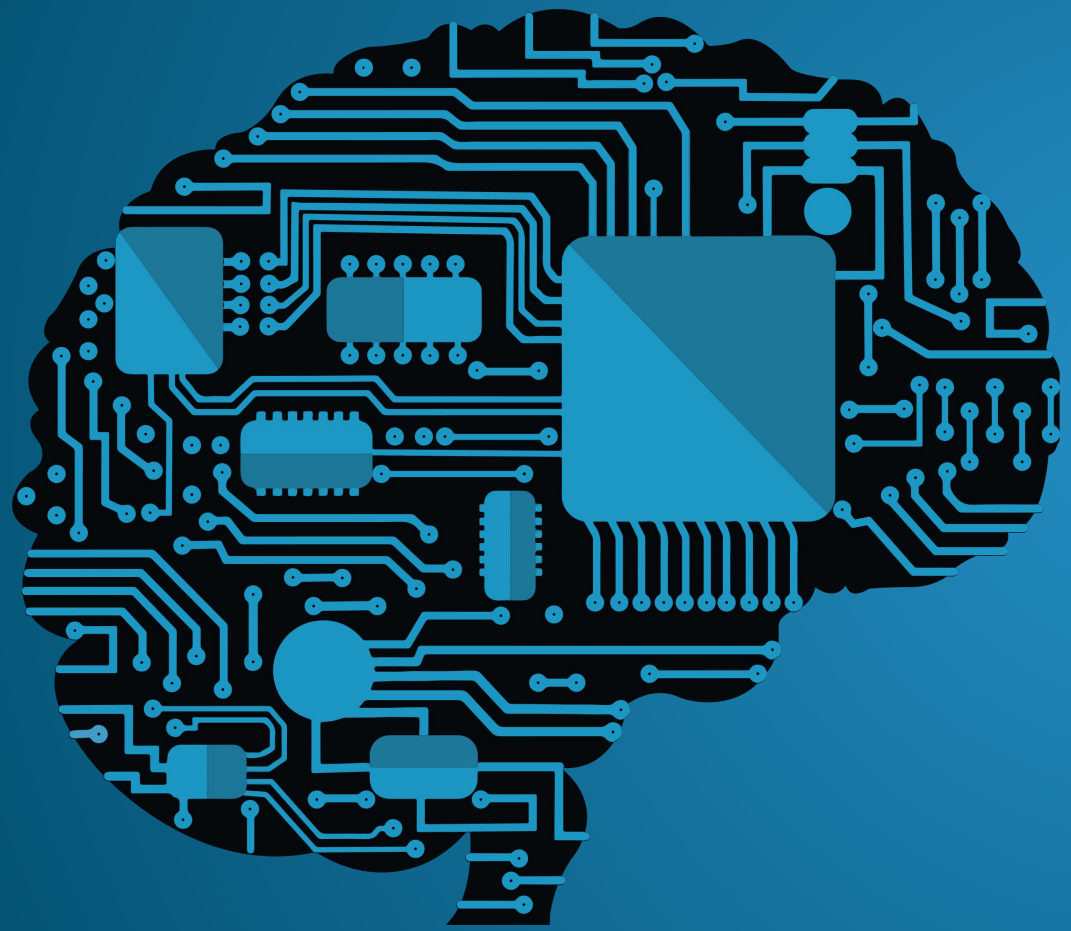
02. 사업성 검토



시장성 CCTV가 있는 거의 모든 주차장에 사용 가능

경쟁력 별도의 센서 설치 없이 소프트웨어 만으로 사용 가능한 경제성

차별성과 전략 기존의 센서 시스템과 다른 방식



발전하려는 기술의 접목

최근에 급격한 발전을 이룬
CV(Computer Vision) 기술을 응용하여 개발.

사용한 프로그램과 데이터 정리

CVAT

Visual Studio Code

python

roboflow

YOLOv5를 사용하여 학습시킨 **Detection** 모델과 **Tracking** 모델 활용

활용 방법

Object Detection

활용 방법

Object Tracking

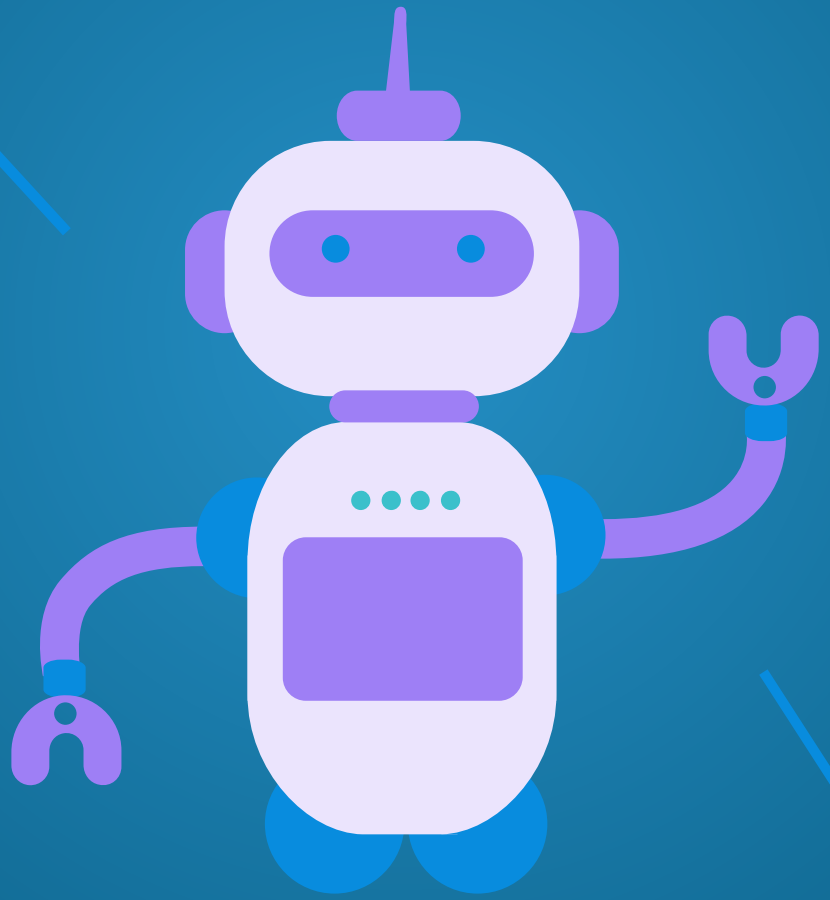
활용 방법

Object Tracking


기술과 분석

OpenCV

Open Source Computer Vision Library의 약자로, 영상 처리와 컴퓨터 비전 관련 오픈 소스 라이브러리입니다.



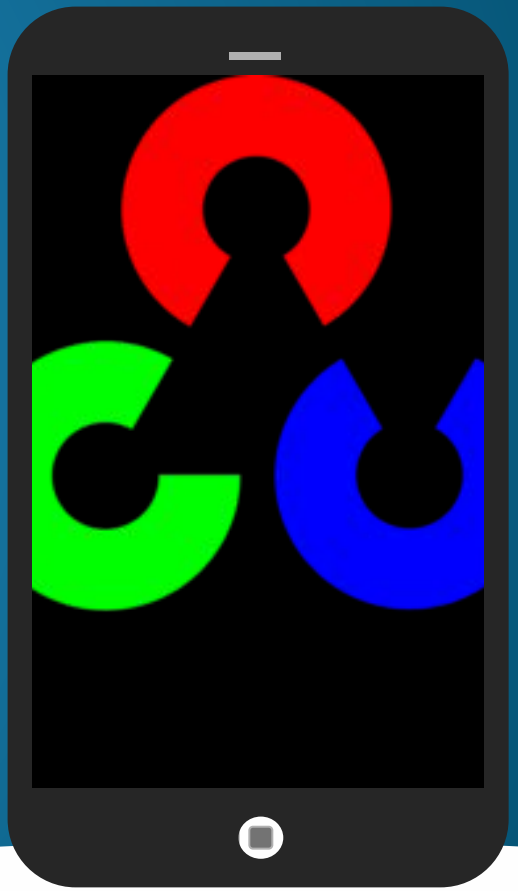
2,500개가 넘는 알고리즘으로 구성되어 있습니다. 해당 구성은 다음과 같습니다.

OpenCV란

영상 처리, 컴퓨터 비전, 기계 학습과 관련된 전통적인 알고리즘

얼굴 검출과 인식, 객체 인식, 객체 3D 모델 추출, 스테레오 카메라에서 3D 좌표 생성

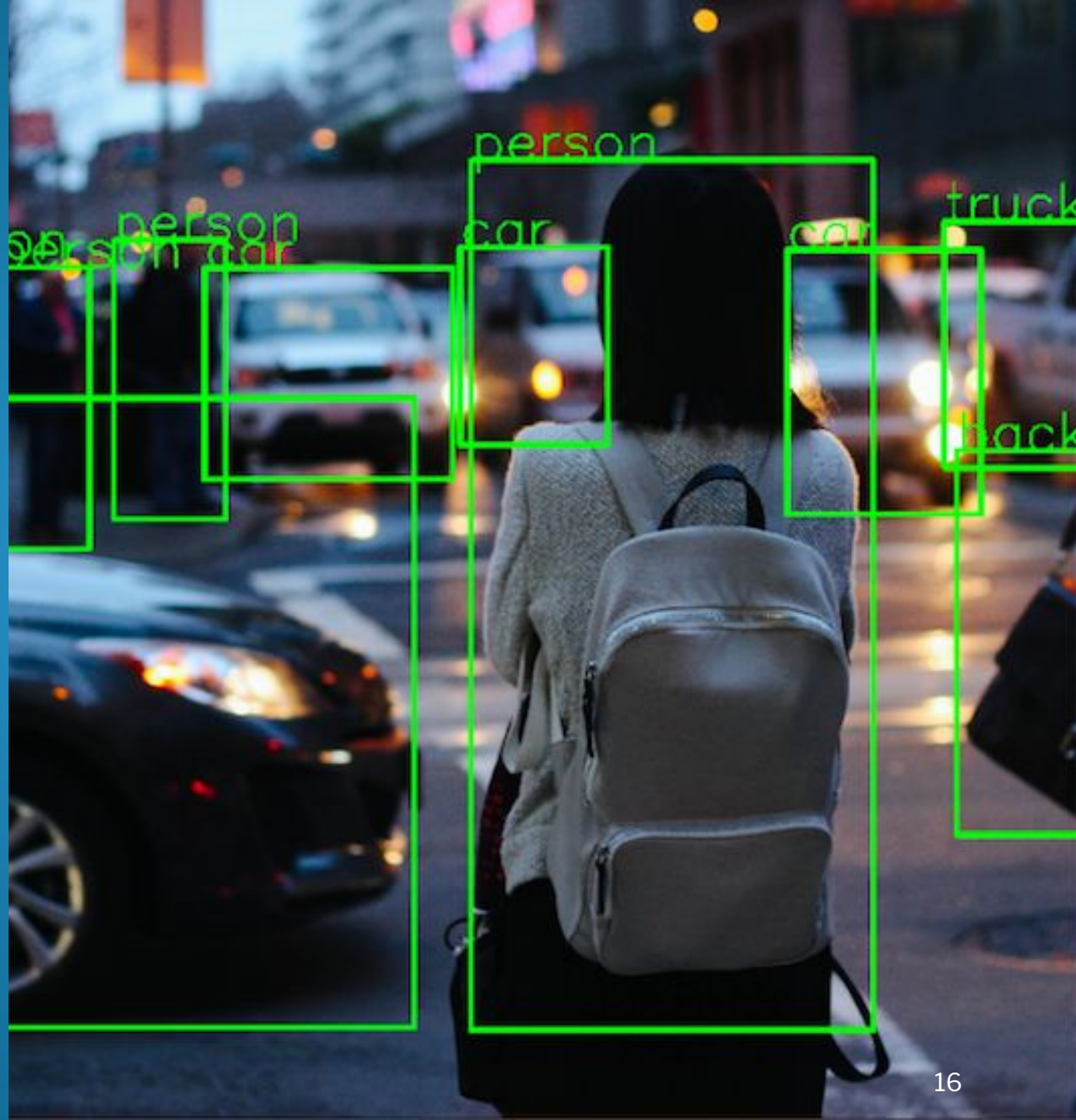
고해상도 영상 생성을 위한 이미지 스티칭, 영상 검색, 적목 현상 제거, 안구 운동추적



Object Detection

객체 탐지(Object Detection)는
무엇일까요?

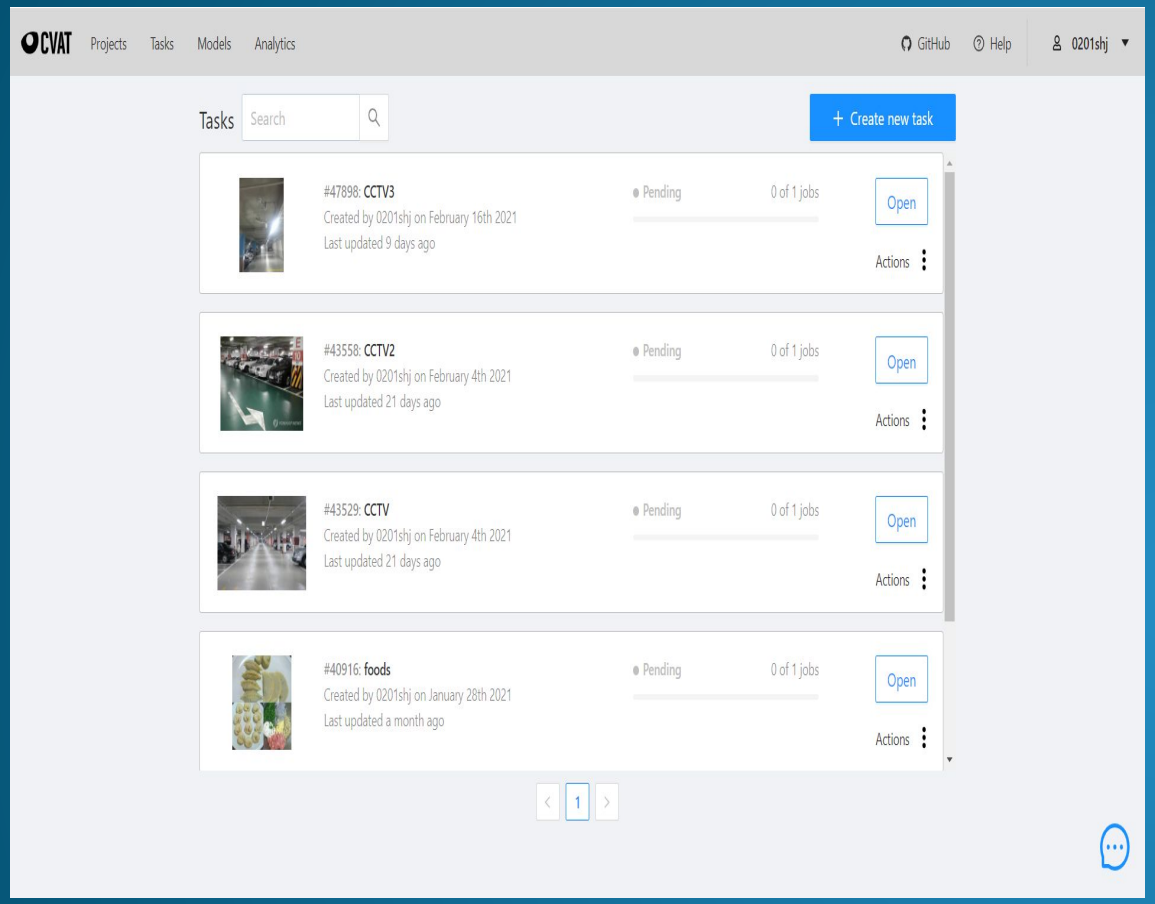
객체 탐지는 이미지에서 목표인 객체를 배경과 구분해 식별하는 **자동화 기법**으로,
컴퓨터 비전(Computer Vision) 기술의 하위 집합이기도 합니다.

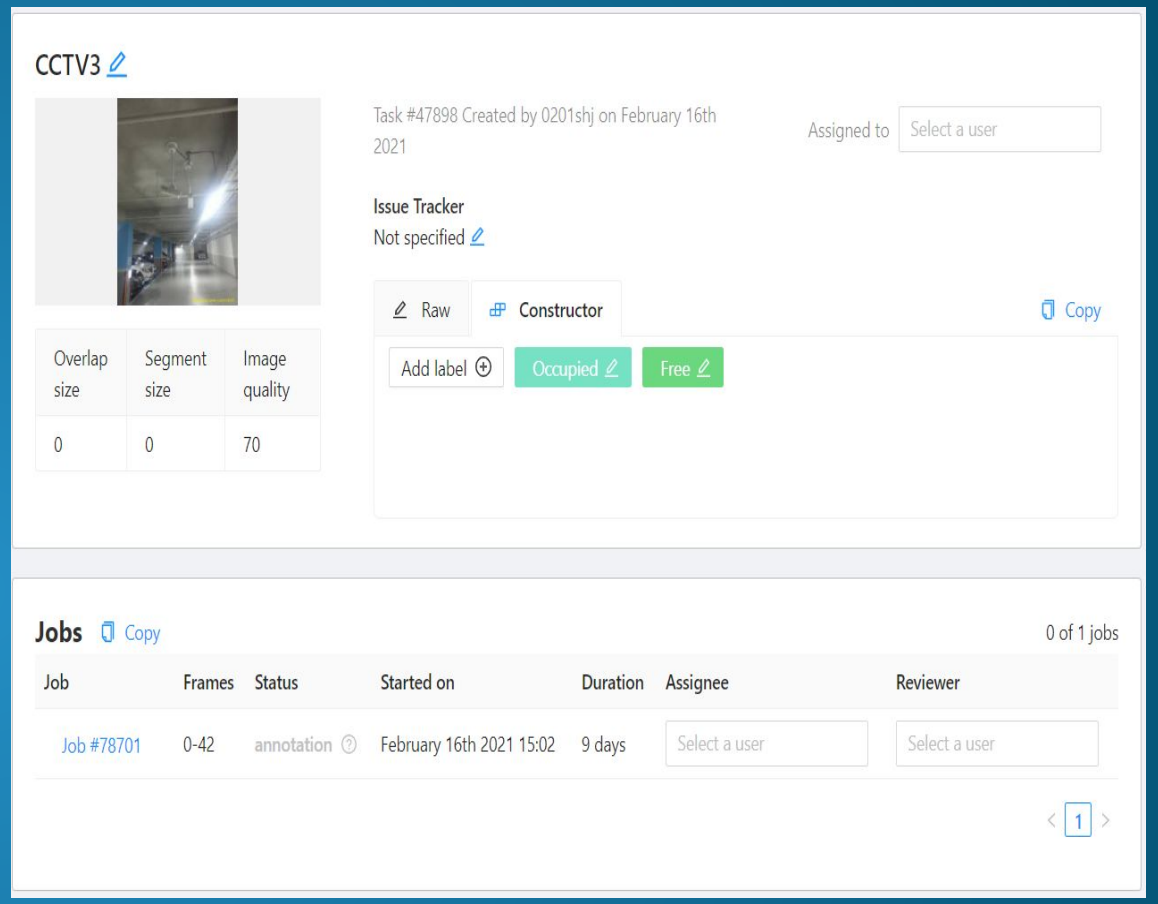


CVAT

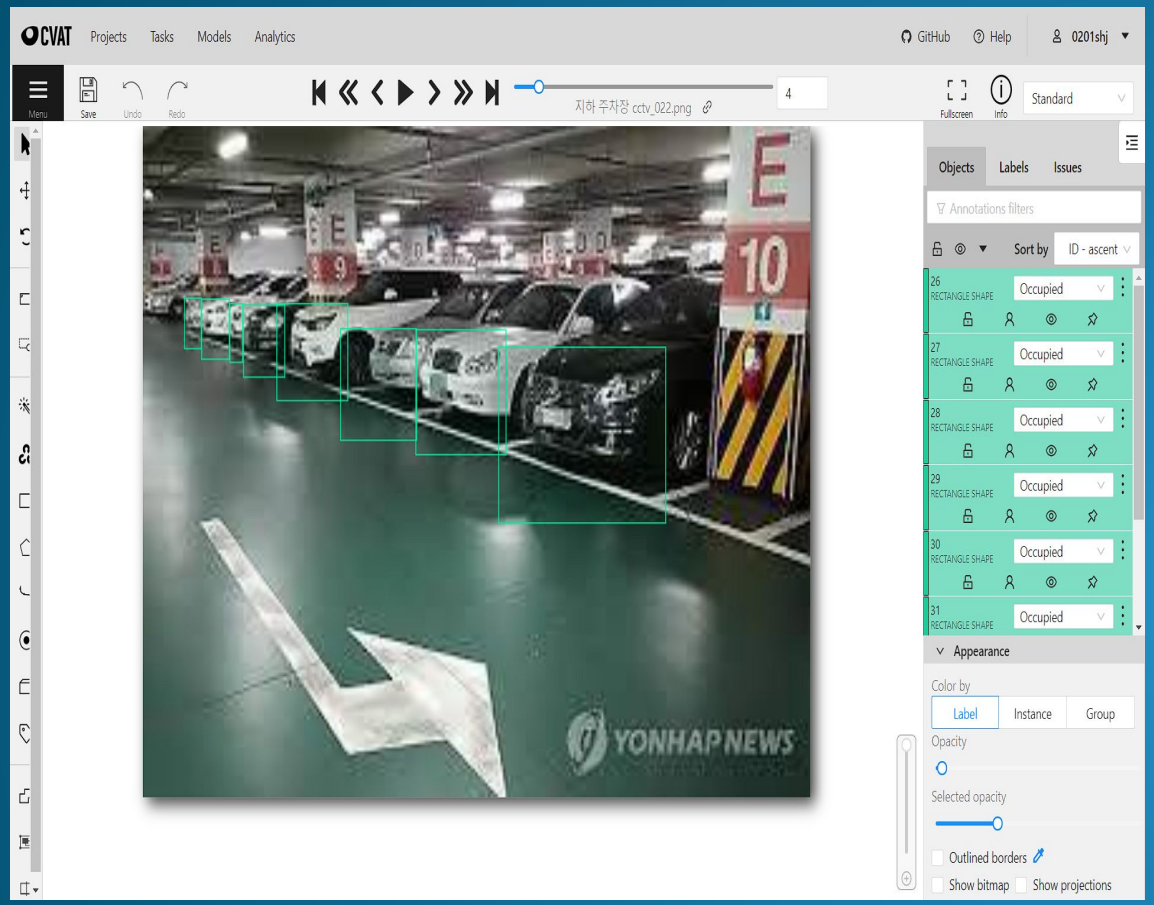
Computer Vision Annotation Tool

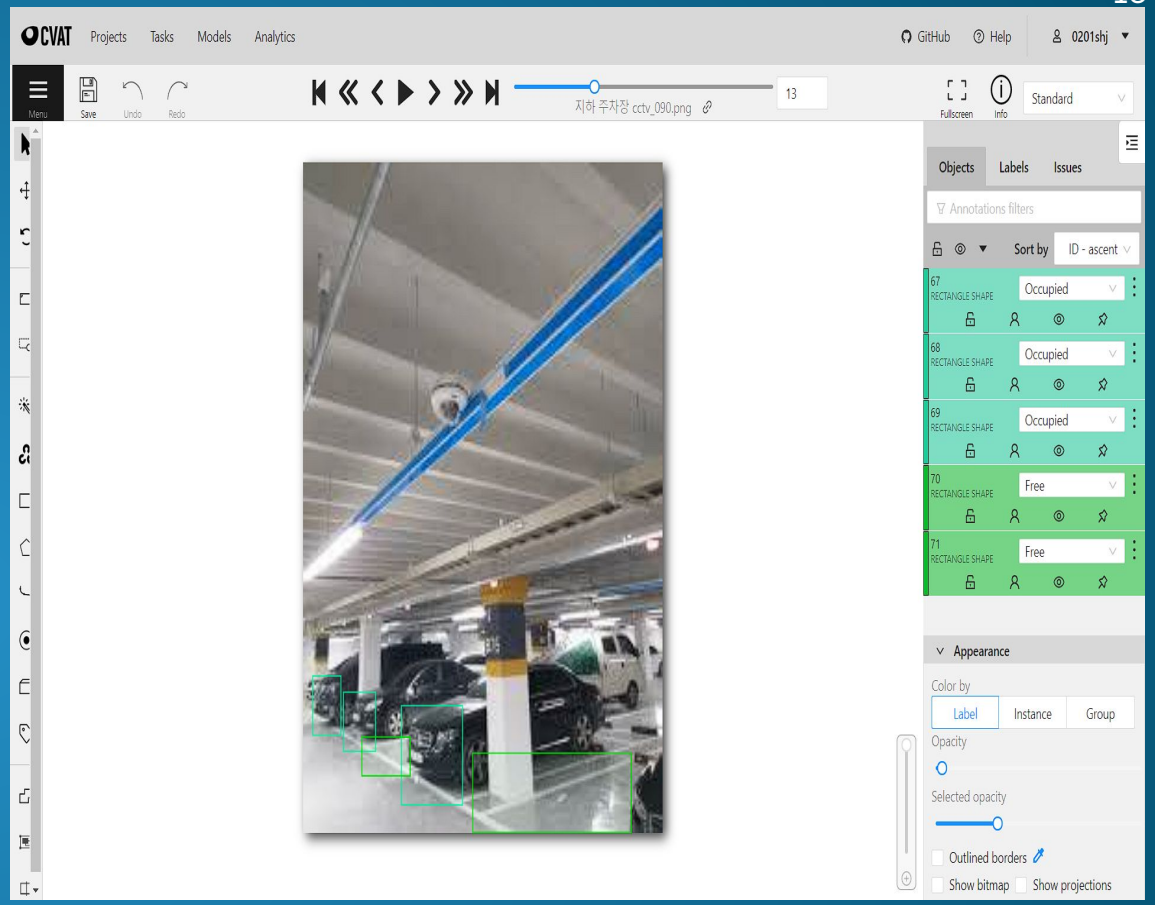
CVAT
Object Detection





CVAT
Object Detection

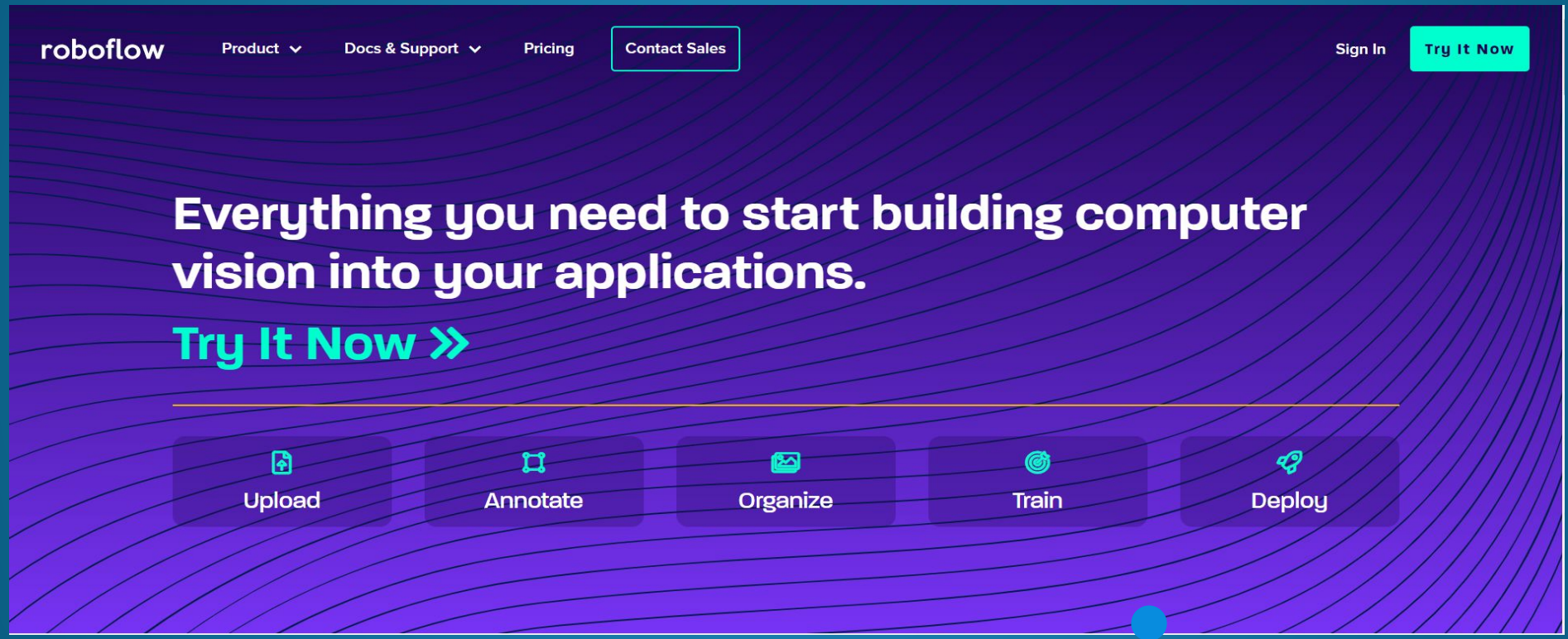




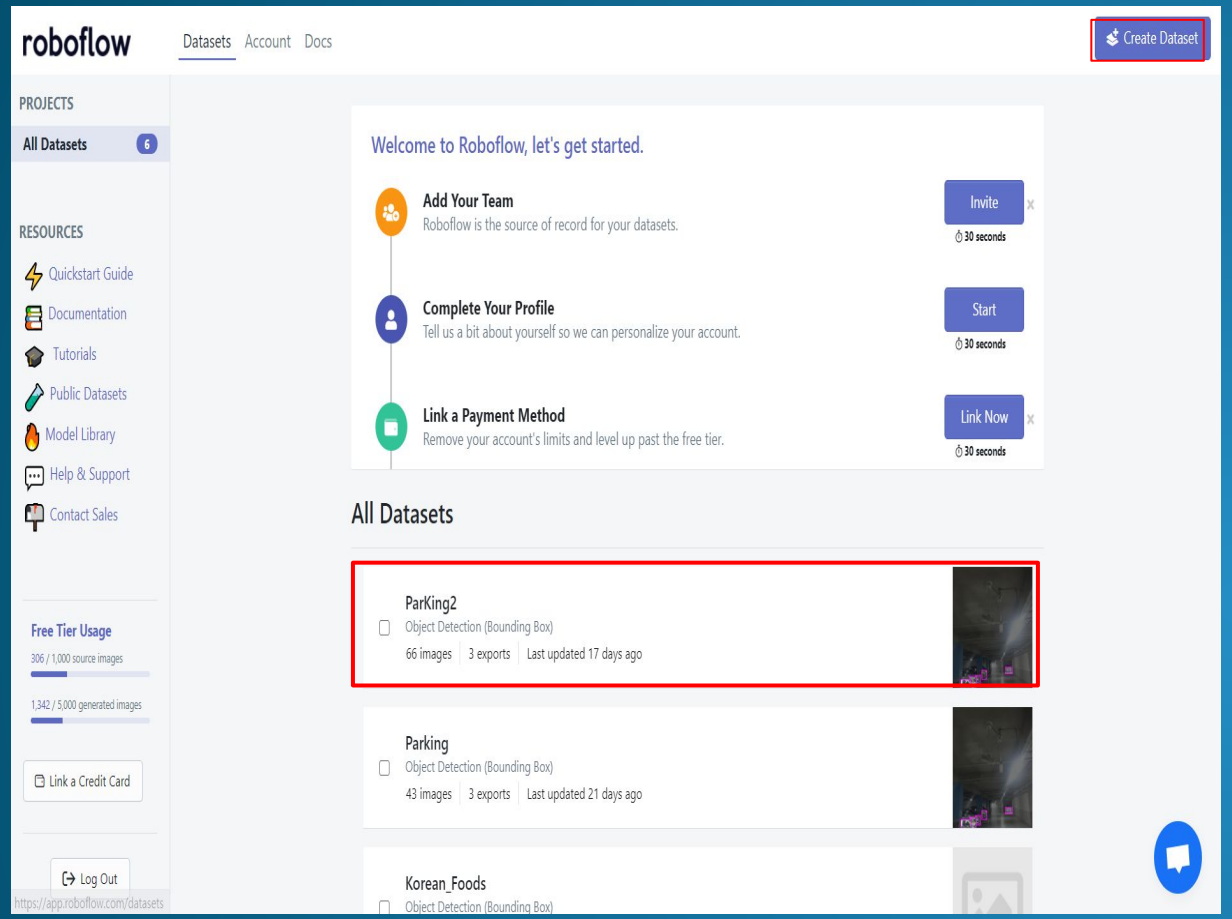
CVAT
Object Detection

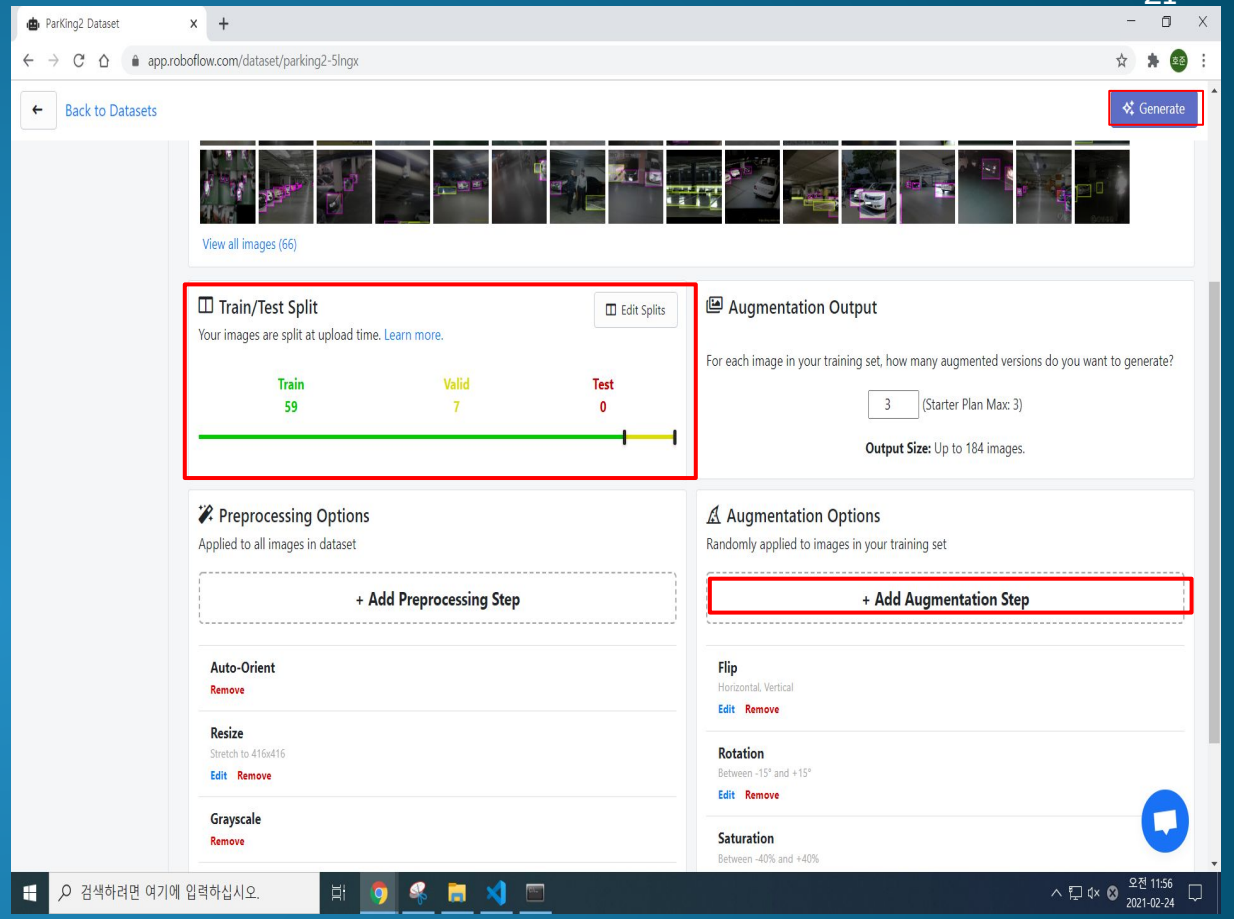
roboflow

Roboflow

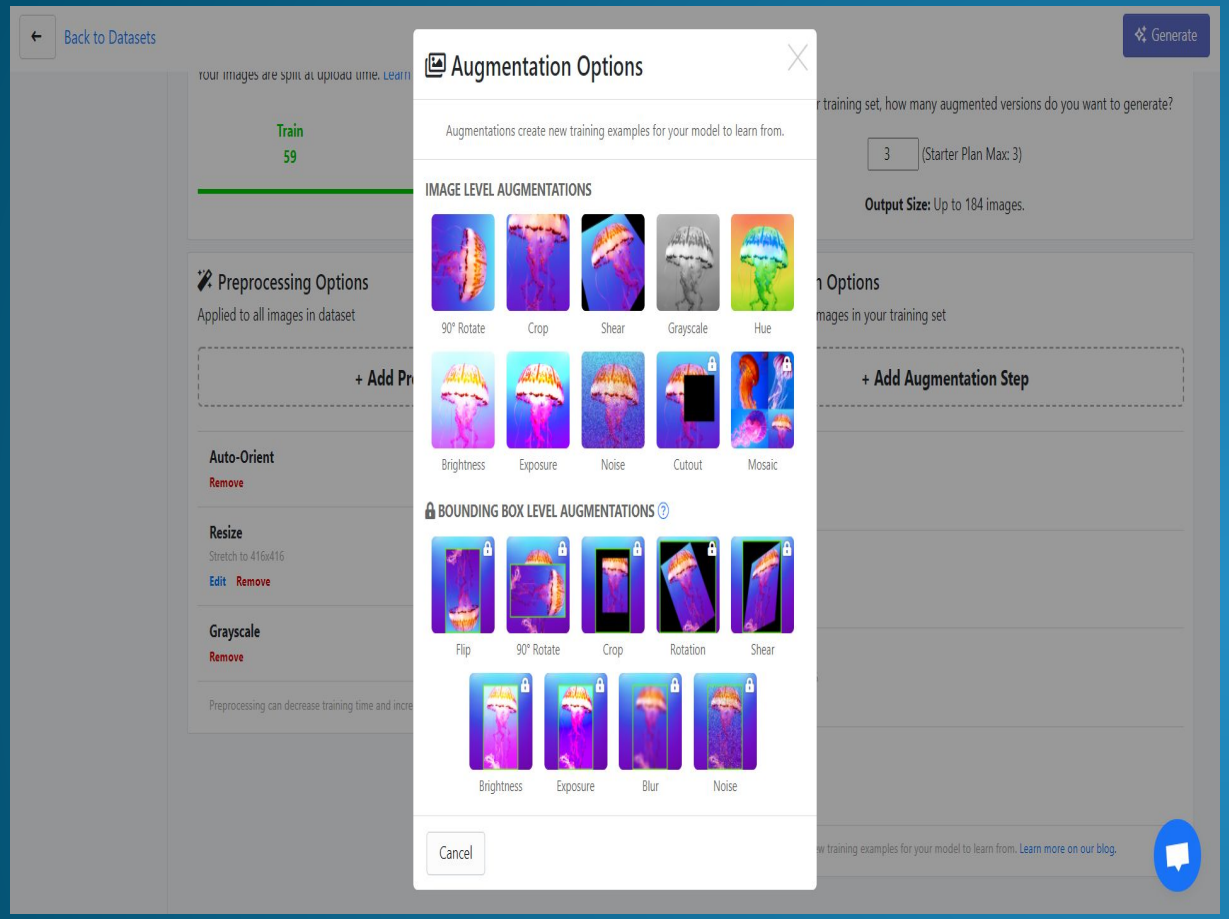


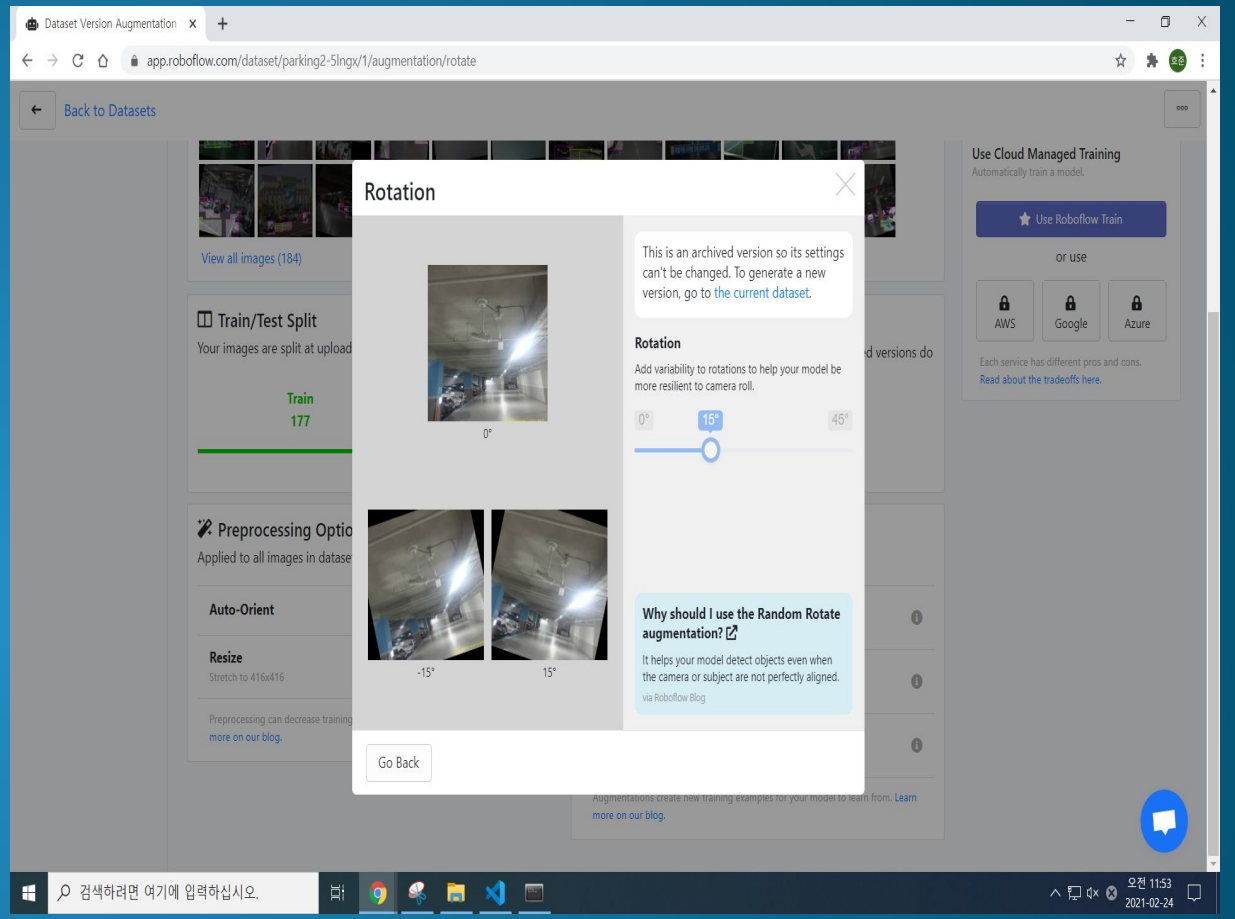
Roboflow
Object Detection



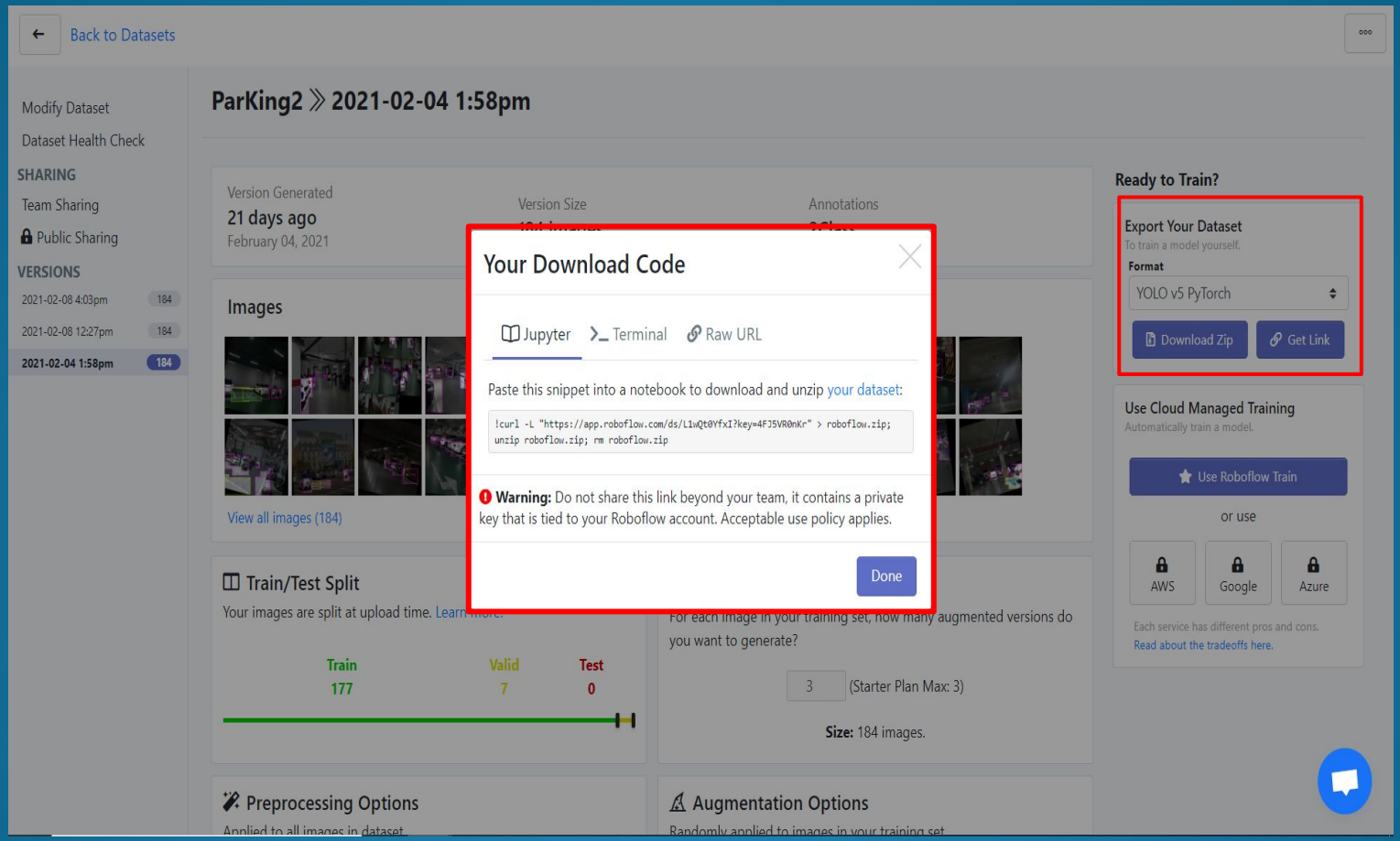


Roboflow
Object Detection





Roboflow
Object Detection



Roboflow
Object Detection

YOLOv5

참고자료 / 깃허브 YOLOv5
https://github.com/0201shj/Object_Project/blob/main/Project_Yolov5.ipynb

Yolov5
Object Detection

Roboflow Image Training Data set

```
[1]  !curl -L "https://app.roboflow.com/ds/LiwQt0Yfxl?key=4FJ5YROnKr" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
     extracting: train/labels/          cctv_1023_jpg.rf.7b30f4a67e9cac79d00d23275bfaec64.txt
     extracting: train/labels/          cctv_1023_jpg.rf.a355a42cd533e1720e18b4efd25fc5b5.txt
     extracting: train/labels/          cctv_1023_jpg.rf.bf750cb73e87c2f6123ac175686fdd48.txt
     extracting: train/labels/          cctv_1024_jpg.rf.8763b30058bcd9c5b803ed91d8c92b20.txt
     extracting: train/labels/          cctv_1024_jpg.rf.8af42e6bad421fe9d7135ff873cf729f.txt
     extracting: train/labels/          cctv_1024_jpg.rf.b306099e227574235eabc250e5ed5f04.txt
     extracting: train/labels/          cctv_1025_jpg.rf.4569b68e7b03876180c7ff24c2aed767.txt
     extracting: train/labels/          cctv_1025_jpg.rf.7504f43bf3c924738048faa284c13d4f.txt
     extracting: train/labels/          cctv_1025_jpg.rf.776307a309e3bb630b844506c1255b69.txt
     extracting: train/labels/          cctv_1026_jpg.rf.089d2fc457ed309eef6554566d1aea61.txt
     extracting: train/labels/          cctv_1026_jpg.rf.952a49438c6a9e13c0f7092106f0ffd7.txt
     extracting: train/labels/          cctv_1026_jpg.rf.ef26df15478e4042a2969999601174b3.txt
     extracting: train/labels/          cctv_110_png.rf.450b59e22bf777f06f56e888bc54e6f4.txt
     extracting: train/labels/          cctv_110_png.rf.8a690a2a9d558b4c1fd5cc6ea6c334c7.txt
     extracting: train/labels/          cctv_110_png.rf.c71f0952cbc0101a1e6622d6c6207f61.txt
     extracting: train/labels/          cctv_125_png.rf.4e90e20d204823b9bb16227c32a4e836.txt
     extracting: train/labels/          cctv_125_png.rf.872c8aec5b8c6fcc2573c86f78ae4c76.txt
     extracting: train/labels/          cctv_125_png.rf.bc047b3a7569e9ae84bd7a7cd18a6cfd.txt
     extracting: train/labels/          cctv_134_png.rf.b8097fbba5fbc690082f39c1c9e04743.txt
     extracting: train/labels/          cctv_134_png.rf.5b7ee05b4c5b6e9c949d4cb96456cd8a.txt
     extracting: train/labels/          cctv_134_png.rf.bc7a0bbbf86321c4b57ab6d409a90bd9.txt
     extracting: train/labels/          cctv_149_png.rf.2cfd77d39b104f0f58f8520d560fa213.txt
     extracting: train/labels/          cctv_149_png.rf.44f60c002b857b95a5eebef0c0f1fa46.txt
     extracting: train/labels/          cctv_149_png.rf.c2f40d2dd91dc987d94ae22eb18c8b51.txt
     extracting: train/labels/          cctv_152_png.rf.20945f60c03dd28e8509d6107902c5f5.txt
```

Yolov5

Object Detection

Yolov5 link download

```
[2]  # clone YOLOv5 and reset to a specific git checkpoint that has been verified working
     !git clone https://github.com/ultralytics/yolov5  # clone repo
     %cd yolov5
     !git reset --hard 68211f72c99915a15855f7b99bf5d93f5631330f
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 5004 (delta 8), reused 11 (delta 6), pack-reused 4987
Receiving objects: 100% (5004/5004), 7.93 MiB | 11.36 MiB/s, done.
Resolving deltas: 100% (3399/3399), done.
/content/yolov5
HEAD is now at 68211f7 FROM nvcr.io/nvidia/pytorch:20.10-py3 (#1553)
```

Yolov5
Object Detection

Pytorch download & requirements.txt download

```
[3]  # install dependencies as necessary
     !pip install -qr requirements.txt  # install dependencies (ignore errors)
     import torch

     from IPython.display import Image, clear_output  # to display images
     from utils.google_utils import gdrive_download  # to download models/datasets

     # clear_output()
     print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else

          |████████████████████████████████| 645kB 7.8MB/s
     Setup complete. Using torch 1.7.0+cu101 _CudaDeviceProperties(name='Tesla T4', major=7, minor=5, total_memory=15109MB, multi_processor
```

Yolov5
Object Detection

Data.yaml check

```
[6]  # this is the YAML file Roboflow wrote for us that we're loading into this notebook with our data
     %cat data.yaml
```

```
train: ../train/images
val: ../valid/images

nc: 2
names: ['Free', 'Occupied']
```

Yolov5
Object Detection

```
#this is the model configuration we will use for our tutorial
%cat /content/yolov5/models/yolov5m.yaml
```

```
# parameters
nc: 80  # number of classes
depth_multiple: 0.67  # model depth multiple
width_multiple: 0.75  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, BottleneckCSP, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
```

```
[22]  %%writetemplate /content/yolov5/models/custom_yolov5m.yaml

# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
```

```python
# Freeze
freeze = []  # parameter names to freeze (full or partial)
for k, v in model.named_parameters():
    v.requires_grad = True  # train all layers
    if any(x in k for x in freeze):
        print('freezing %s' % k)
        v.requires_grad = False
```

```python
# Freeze
freeze = ['model.%s.' % x for x in range(24)]  # parameter names to freeze (full or partial)
for k, v in model.named_parameters():
    v.requires_grad = True  # train all layers
    if any(x in k for x in freeze):
        print('freezing %s' % k)
        v.requires_grad = False
```

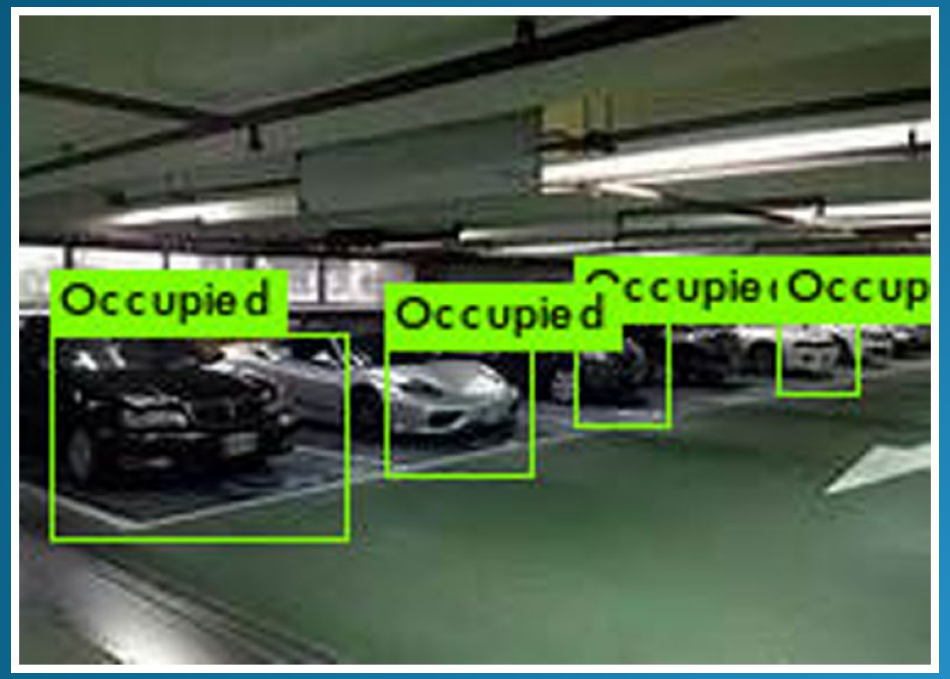
Yolov5
Object Detection

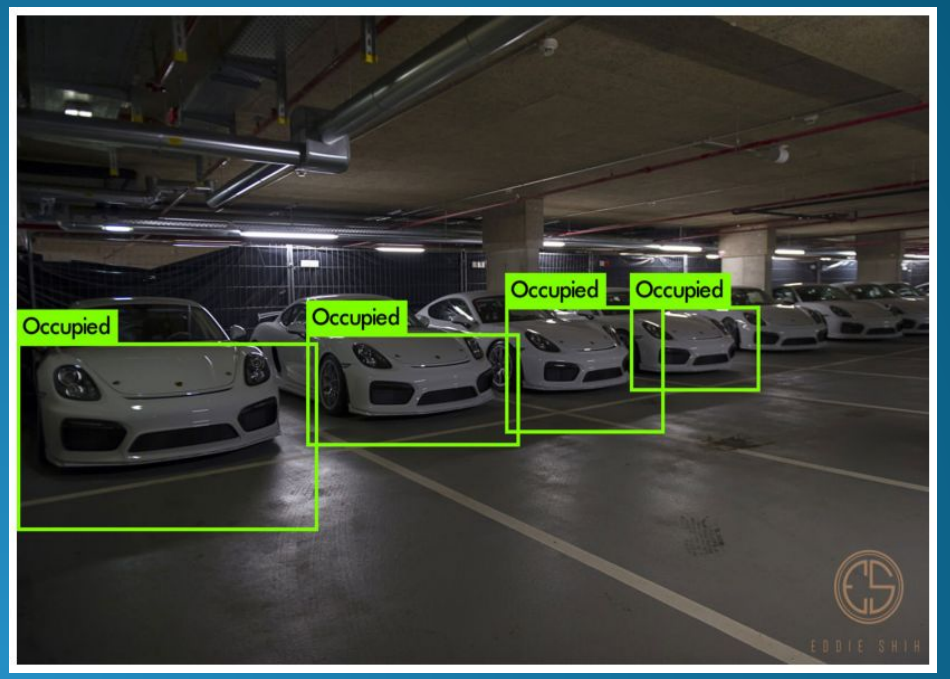
```
[25]  # train yolov5m on custom data for 150 epochs
      # time its performance
      %%time
      %cd /content/yolov5/
      !python train.py --img 416 --batch 16 --epochs 150 --data '../data.yaml' --cfg ./models/custom_yolov5m.yaml --weights '' --name yolov5m_results  --cache
```

```
                 all          7         24      0.0427       0.238       0.092      0.0312

       Epoch   gpu_mem       box       obj       cls     total   targets  img_size
      63/149     1.86G   0.08243    0.1374   0.01334    0.2331        15       416: 100% 12/12 [00:01<00:00,  7.37it/s]
                 Class      Images     Targets         P         R      mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 13.13it/s]
                   all          7         24      0.0295       0.262       0.147      0.0383

       Epoch   gpu_mem       box       obj       cls     total   targets  img_size
      64/149     1.86G   0.07646    0.1199   0.01047    0.2068         1       416: 100% 12/12 [00:01<00:00,  7.36it/s]
                 Class      Images     Targets         P         R      mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 11.52it/s]
                   all          7         24      0.0419       0.119        0.04     0.00718

       Epoch   gpu_mem       box       obj       cls     total   targets  img_size
      65/149     1.86G   0.07776    0.1233   0.01207    0.2131         4       416: 100% 12/12 [00:01<00:00,  7.33it/s]
                 Class      Images     Targets         P         R      mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 13.08it/s]
                   all          7         24      0.0411        0.31       0.138      0.0295

       Epoch   gpu_mem       box       obj       cls     total   targets  img_size
      66/149     1.86G   0.07832    0.1231   0.01257     0.214         3       416: 100% 12/12 [00:01<00:00,  6.90it/s]
                 Class      Images     Targets         P         R      mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 14.09it/s]
                   all          7         24      0.0244        0.31      0.0558      0.0102

       Epoch   gpu_mem       box       obj       cls     total   targets  img_size
      67/149     1.86G   0.07825    0.1261   0.01143    0.2157         7       416: 100% 12/12 [00:01<00:00,  7.27it/s]
                 Class      Images     Targets         P         R      mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 13.08it/s]
                   all          7         24      0.0214       0.405       0.178      0.0477

       Epoch   gpu_mem       box       obj       cls     total   targets  img_size
      68/149     1.86G   0.07679    0.121   0.01049    0.2183        13       416: 100% 12/12 [00:01<00:00,  7.10it/s]
```

Yolov5
Object Detection

Yolov5
Object Detection

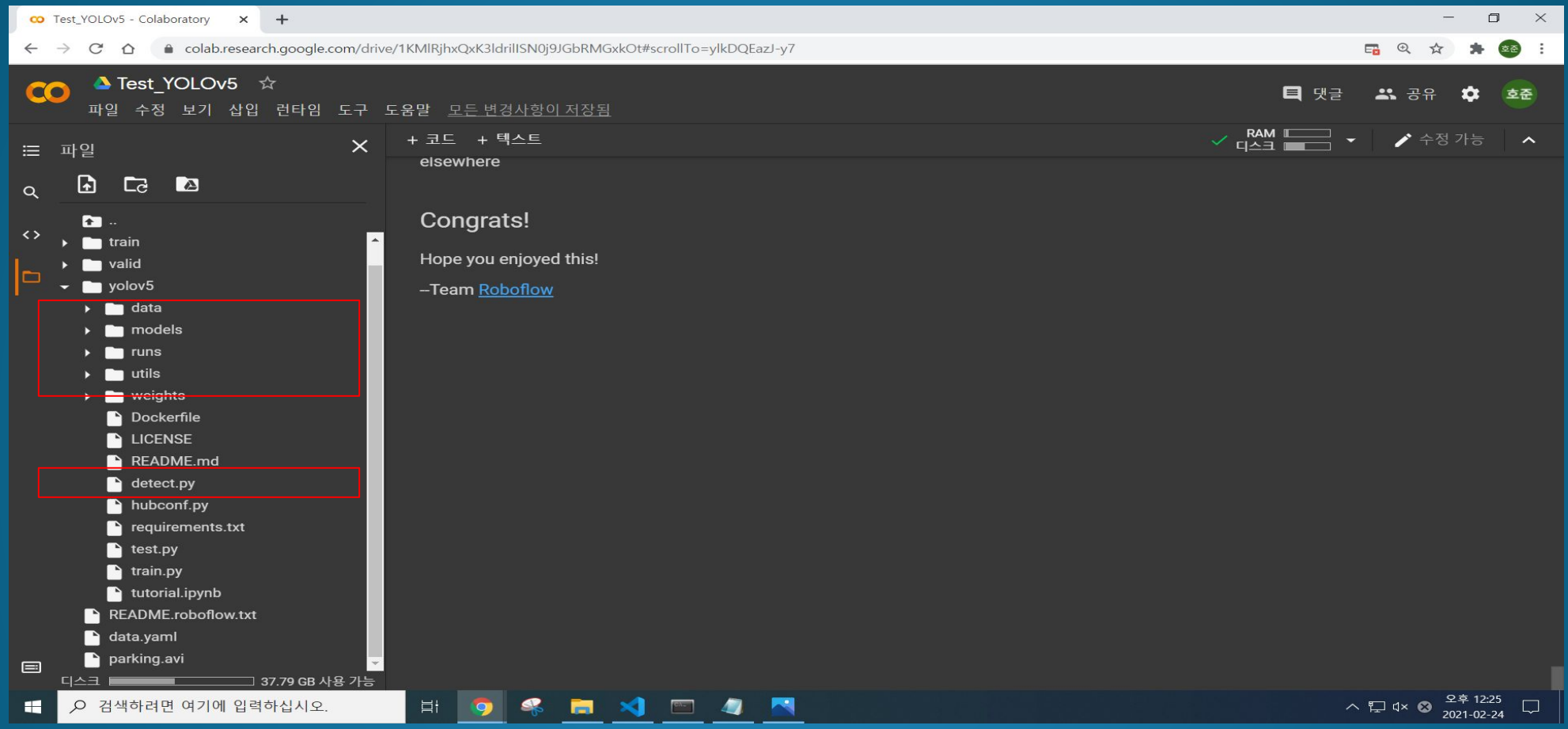




checking model - best.pt

```
[ ]  !python detect.py --weights runs/train/yolov5m_results/weights/best.pt  --save-conf --conf 0.15 --source  ../parking.avi
```

Yolov5
Object Detection



Yolov5 Object Detection

Visual Studio Code - Python

참고 자료 / 깃허브 OBject_Project

https://github.com/0201shj/Object_Project

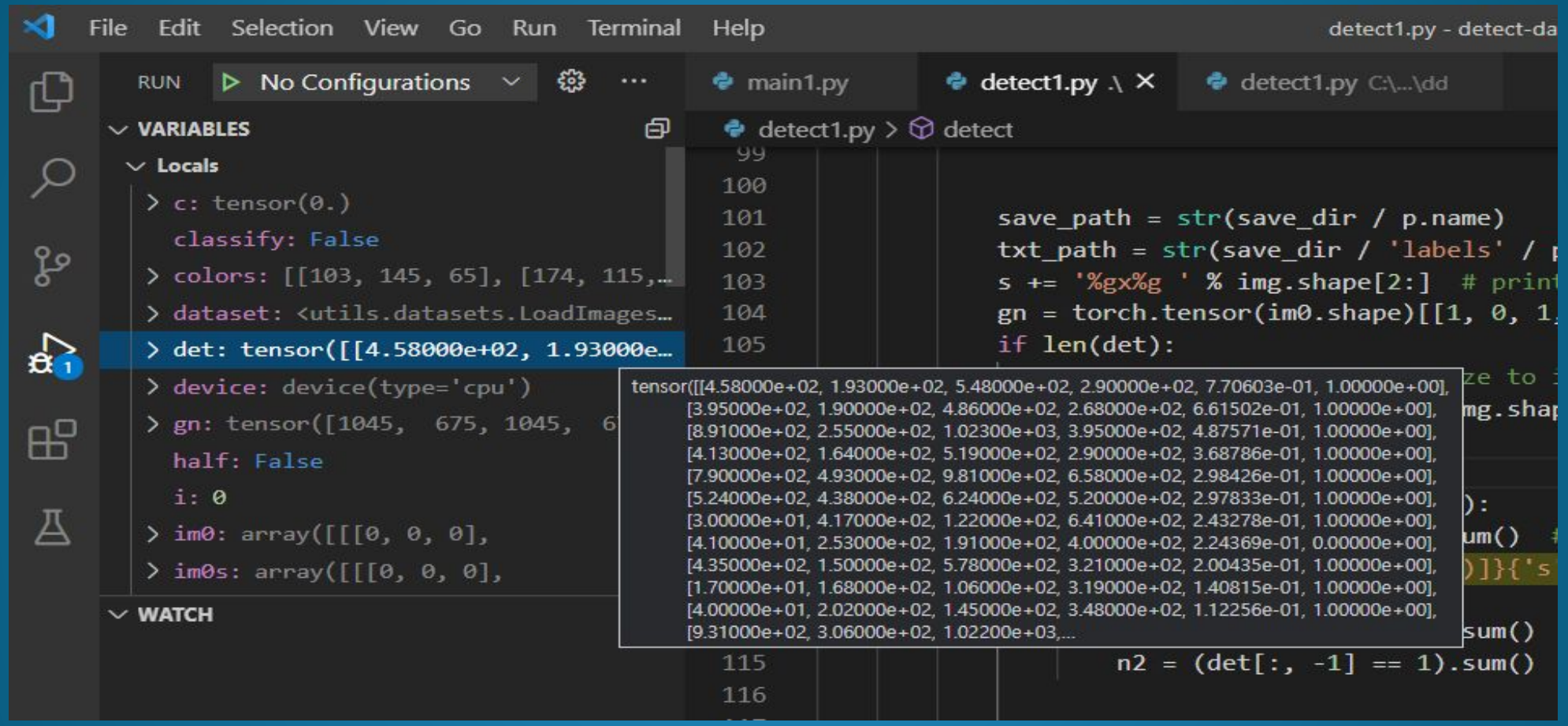
VS code

Object Detection

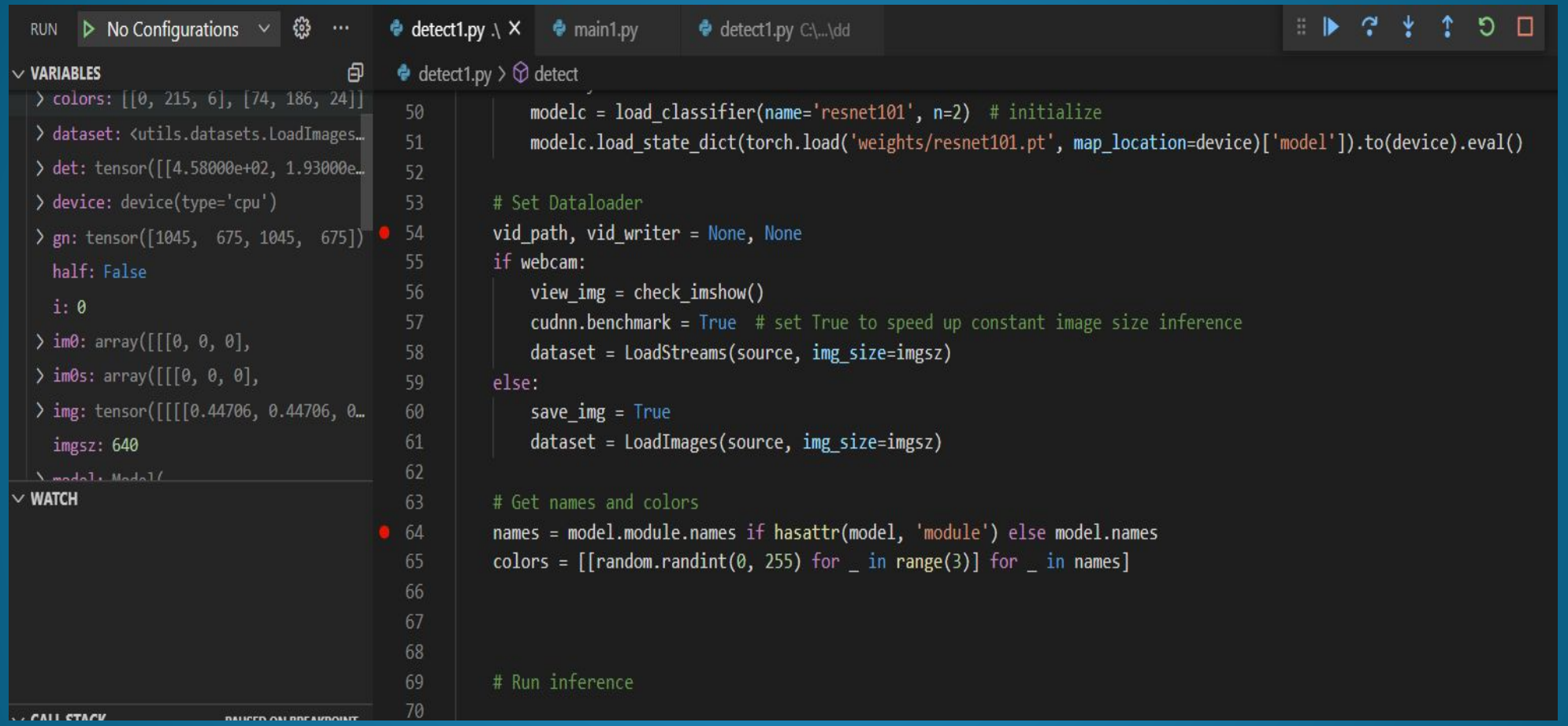
```
(base) PS D:₩testtesst> python detect.py --weights best.pt --save-conf --conf 0.15 --source parking.jpg
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.15, device='', exist_ok=False, img_size=640, iou_thres=0
.45, name='exp', project='runs/detect', save_conf=True, save_txt=False, source='parking.jpg', update=False, view_img=False, weig
hts=['best.pt'])
YOLOv5  torch 1.7.1 CPU

Fusing layers...
Model Summary: 232 layers, 7249215 parameters, 0 gradients
image 1/1 D:₩testtesst₩parking.jpg: 320x640 17 Occupieds, Done. (0.153s)
Results saved to runs₩detect₩exp3
Done. (0.184s)
```

VS code

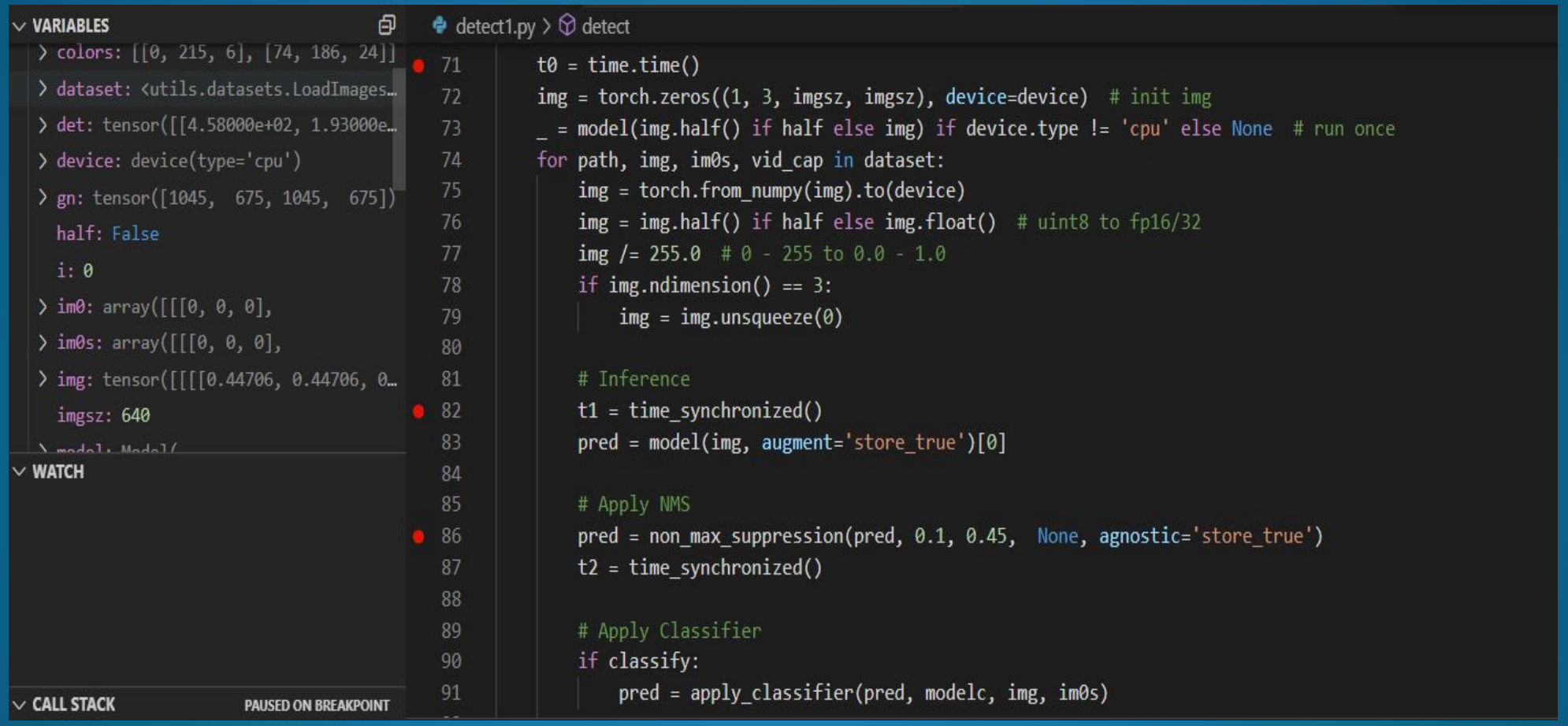
Object Detection



VS code
Object Detection



VS code
Object Detection



VS code
Object Detection

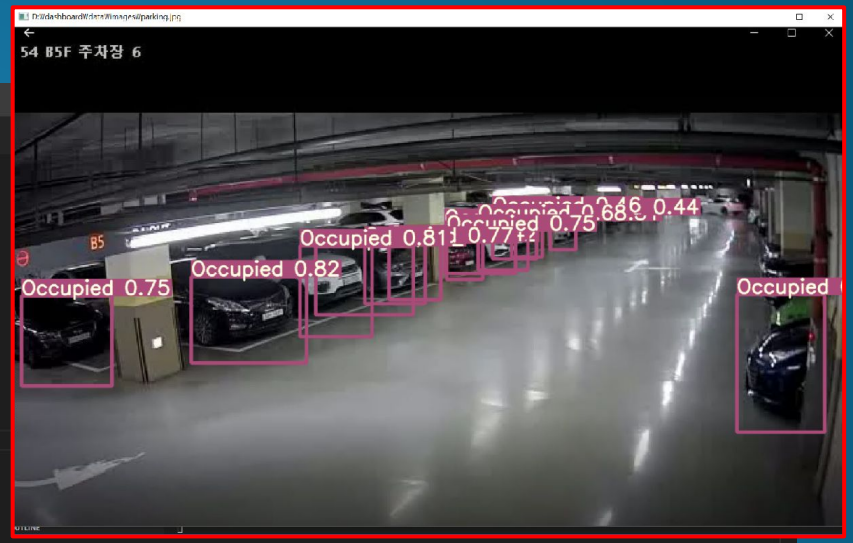
detect.py > detect

```
91          gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
92          if len(det):
93              # Rescale boxes from img_size to im0 size
94              det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()
95
96              # Print results
97              for c in det[:, -1].unique():
98                  n = (det[:, -1] == c).sum()  # detections per class
99                  s += '%g %ss, ' % (n, names[int(c)])  # add to string
100
```

```
95
96              # Print results
97              for c in det[:, -1].unique():
98                  n = (det[:, -1] == c).sum()  # detections per class
99                  #s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string
100
101                 #print(def[:, -1] ==c).sum())
102
103
104                 n1 = (det[:, -1] == 0).sum()
105                 n2 = (det[:, -1] == 1).sum()
106
107
108             #Data = {'Empty': [n1.tolist()],'Occupied': [n2.tolist()]}
109             #print(Data)
110
111             Empty = n1.tolist()
112             Occupied = n2.tolist()
113
```

```
(base) PS D:₩detect> python detect1.py
0 13
```

detect.py
Object Detection



File Edit Selection View Go Run Terminal Help — main.py - detect - Visual Studio Code

EXPLORER

DETECT
- __pycache__
- data
- models
- runs
- utils
- best.pt
- detect.py
- detect1.py
- Empty.py
- main.py
- parking.JPG
- path_ch.txt
- requirements.txt

OUTLINE

main.py

```
1  import Empty
2
3  Empty.detect()
4
5  val1, val2 = Empty.detect()
6
7  a = format(val1, '03')
8  b = format(val2, '03')
9  print('Empty: ', a)
10 print('Occupied: ', b)
11
12
13
```

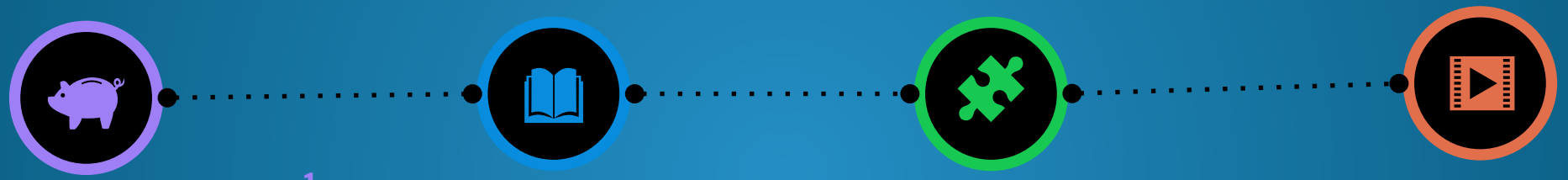
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL — 1: Python

```
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\detect>D:/anaconda3/Scripts/activate.bat

(base) D:\detect>D:/anaconda3/python.exe d:/detect/main.py
Empty:  001
Occupied:  008

(base) D:\detect>
```

Python 3.8.5 64-bit (conda) ⊗ 0 ⚠ 0 Analyzing in background, 3527 items left... Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python

main.py
Object Detection

Object Tracking



1

Tracking은 움직이는 장면에서 사진상 물체의 path 근사치를 구하는 것

2

영상속 움직이는 물체의 path가 얼마나 그 전 프레임과 유사한가

3

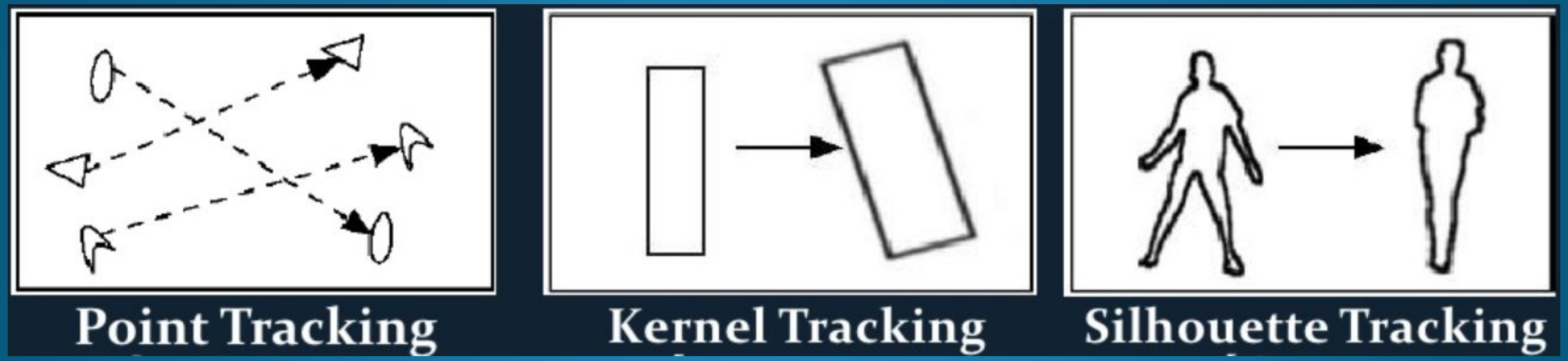
동일 객체라고 인식하면, 그 물체를 계속 Tracking 하는 것

4

Object Tracking 하는 방법에는
Point Tracking,
Kernel Tracking,
Silhouette Tracking

Path: 통로
Object Tracking: 객체 추적
Kernel : 핵심
Silhouette: 실루엣

Point, Kernel, Silhouette Tracking



참고자료 / Object Tracking survey (2010)

https://www.slideshare.net/khanhat/object-tracking-survey

Object Tracking

- Object Traking(객체 추적)은 영상속 필요한 영역의 물체를 찾아 추적
- 즉, 추적하는 영상속 이미지의 프레임의 앞과 뒤의 움직임을 찾고 유사한 객체라 인식되면 그 물체를 계속 Tracking(추적) 하는 것
- 최종적으로 추적된 값을 확인하고 출력

순서

1. Object Detection (객체 탐지)
2. Object Classification (객체 분류)
3. Object Tracking (객체 추적)

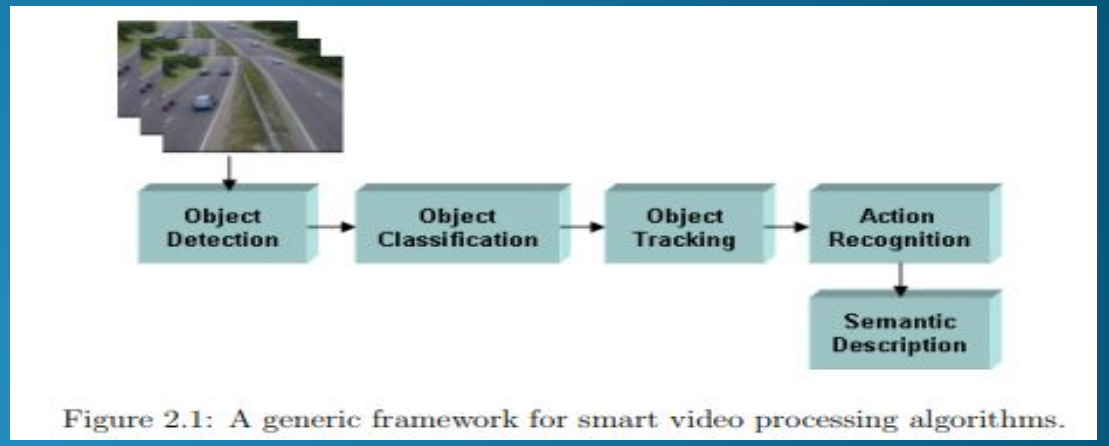


Figure 2.1: A generic framework for smart video processing algorithms.

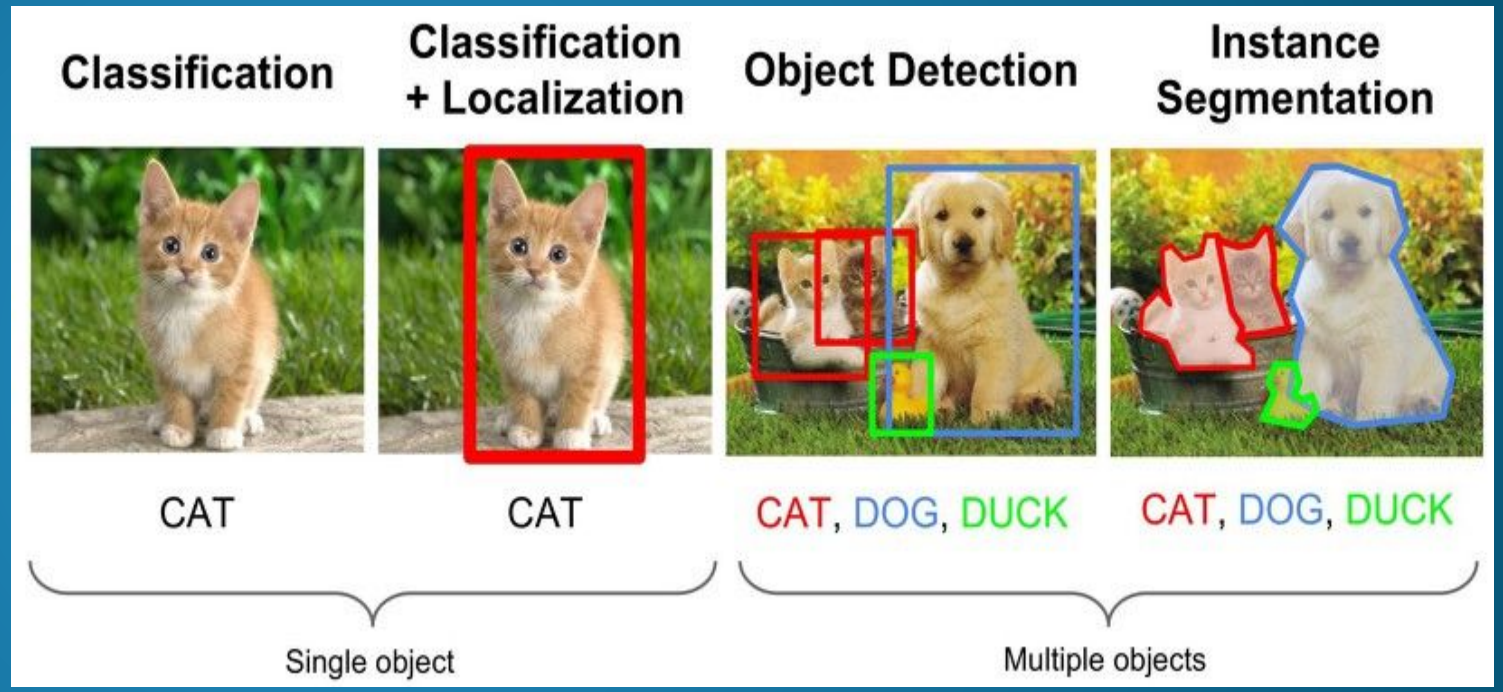
참고자료 / Bilkent Üniversitesi (2004)

http://repository.bilkent.edu.tr/handle/11693/29543

Object Classification:: 객체 분류
Instance Segmentation: 분할영역

Object Classification

- 객체 분류(Object Classification)은 입력된 이미지 CAT을 출력
- 이후 Localization으로 좌표 출력
- Instance Segmentation을 사용하여 좀더 정확한 윤곽을 그리게 되는 방식



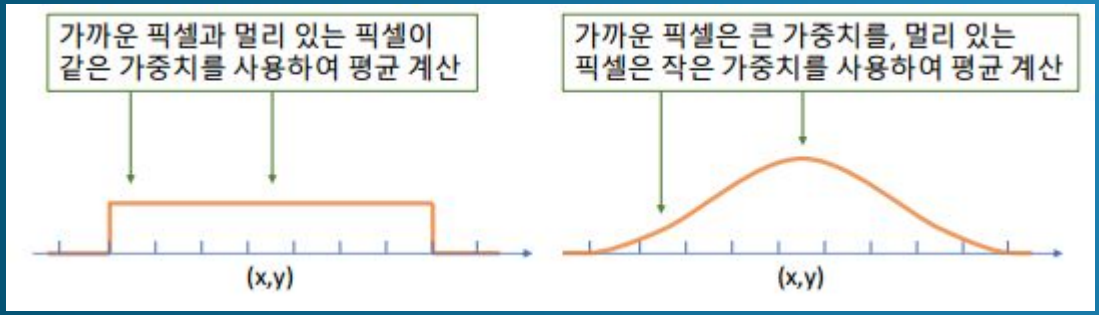
Object Classification:: 객체 분류
Localization: 위지정보 출력
Instance Segmentation: 분할영역

참고자료 / 딥러닝 객체 검출 (2018)

https://light-tree.tistory.com/75

Gaussian Blur

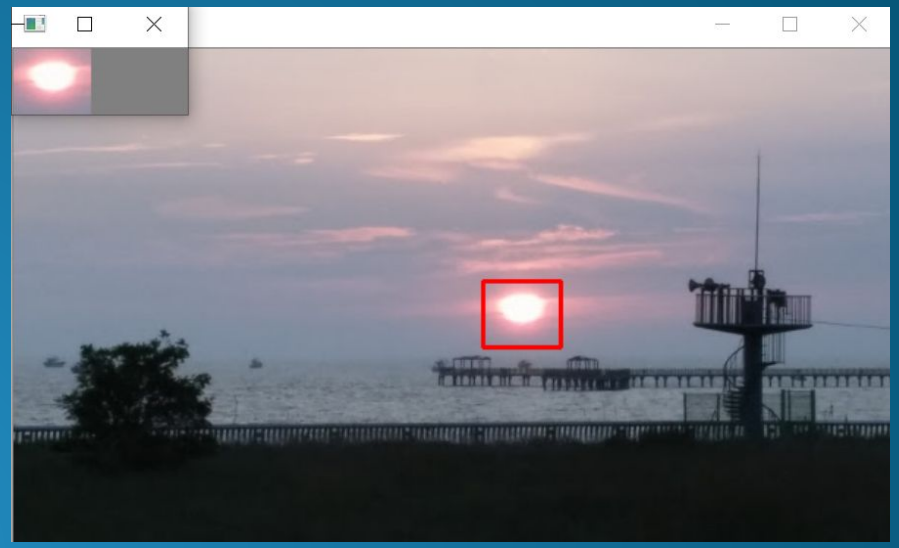
- Gaussian Blur => 모자이크, 영상처리
- 여기선 노이즈를 줄여 얼룩제거에 사용
- 특정 물체에 대해 강조하여 사용
- 이미지와 영상처리에 주로 사용

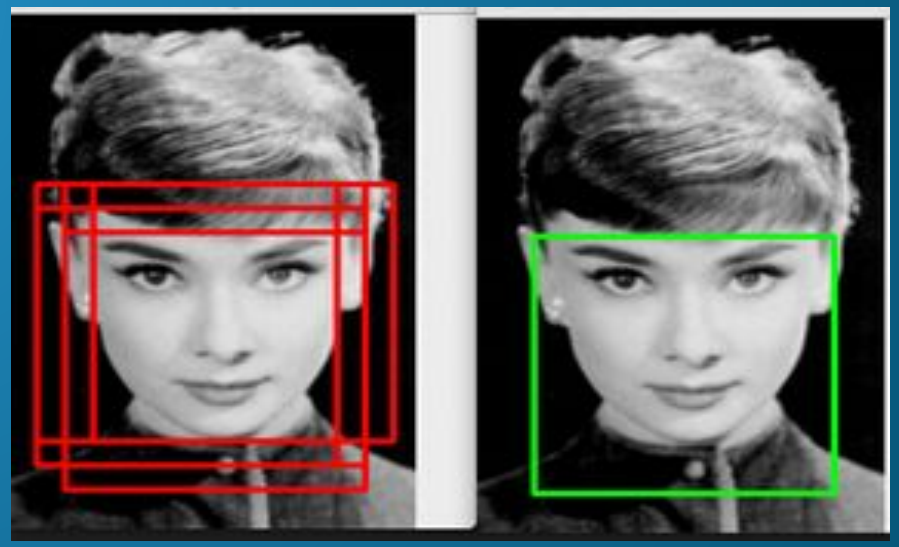


참고자료 / 위키미디어 (2008), OpenCV 가우시안 필터 (2020)
https://commons.wikimedia.org/wiki/File:Sediul_Uniunii_Arhitectilor_din_Romania.jpg
https://deep-learning-study.tistory.com/144

ROI(Region of Interest)

- ROI는 원하는 영역을 사용하는 것
- 노이즈를 줄여 얼룩제거에 사용
- 동일한 크기로 만들거나 복제에 사용
- 여러 중복된 마스크를 하나로 합치는 효과





참고자료 / Faster R-CNN (2018)
http://incredible.ai/deep-learning/2018/03/17/Faster-R-CNN/

ROI: 관심 영역

코드 설명
Object Tracking

```
import cv2
#from tracker import *

# Create tracker object
tracker = EuclideanDistTracker()

cap = cv2.VideoCapture("B3-Parking.avi")
#cap = cv2.VideoCapture("pklot.mp4")
# Object detection from Stable camera#object_detector = cv2.createBackgroundSubtractorKNN(his

object_detector = cv2.createBackgroundSubtractorMOG2(history=2000, varThreshold = 500, detectShadows=False)

while True:
    ret, frame = cap.read()
    height, width, _ = frame.shape

    # Extract Region of interest
    #roi = frame[100: 200, 50: 150] # Y, X  28 93 79 50 [175: 300, 375: 500]
    roi = frame[105: 200, 375: 500] # Y, X
    #roi = frame[10: 400, 10: 600]
```

ROI: 탐지 영역
Object Detector: 객체 탐지

코드 설명
Object Tracking

```
# 1. Object Detection
mask = object_detector.apply(roi)
blur = cv2.GaussianBlur(mask,(121,121),0)
#median = cv2.medianBlur(mask,5)
_, mask = cv2.threshold(blur, 200, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)# THRESH_BINARY_INV
#mask = cv2.adaptiveThreshold(mask,240,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,15,2)
#mask = cv2.adaptiveThreshold(mask,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)
# https://opencv-python.readthedocs.io/en/latest/doc/09.imageThresholding/imageThresholding.html


contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
detections = []
for cnt in contours:
    # Calculate area and remove small elements
    area = cv2.contourArea(cnt)
    if area > 3350:
        cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
        x, y, w, h = cv2.boundingRect(cnt)


        detections.append([x, y, w, h])
```

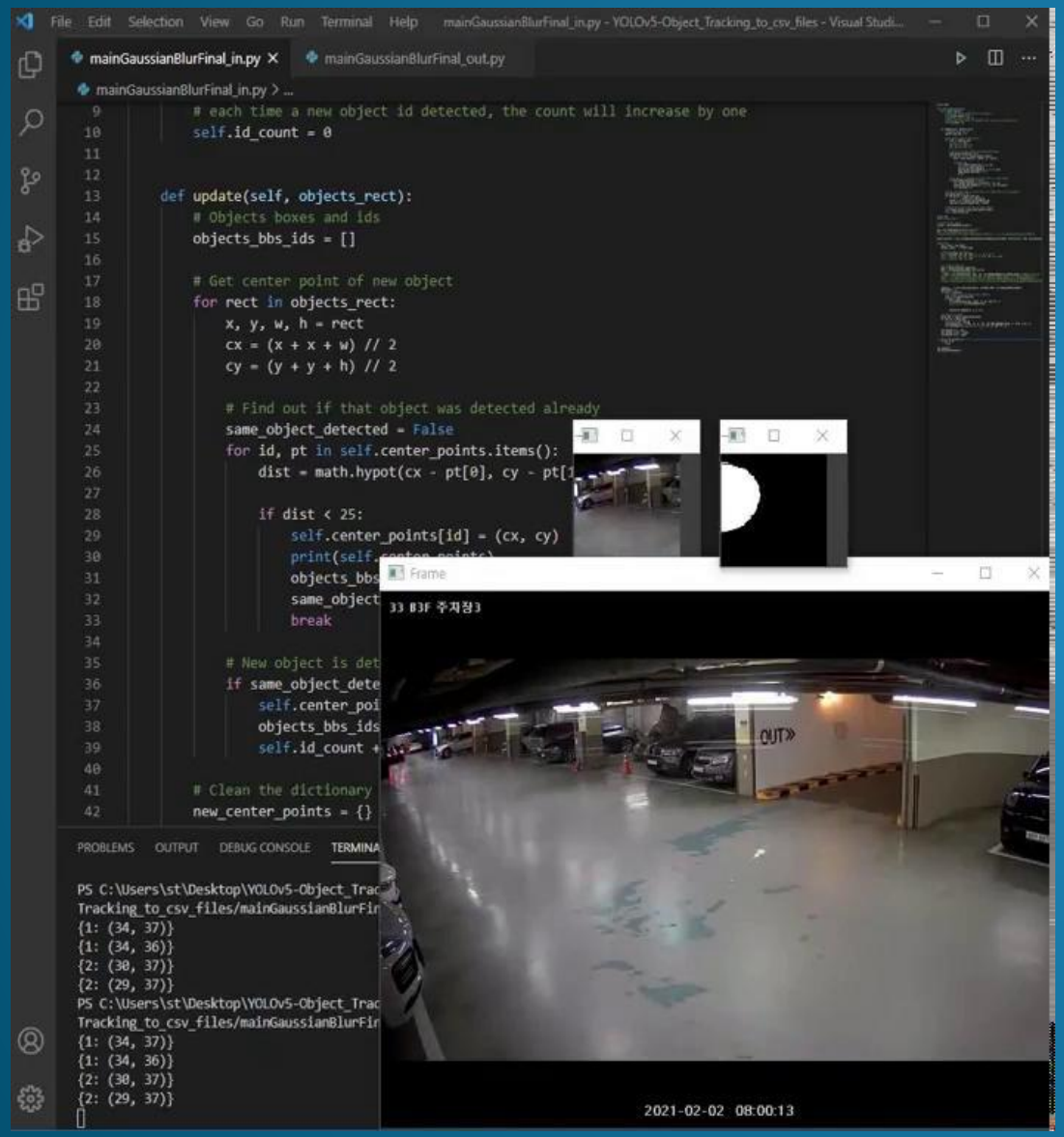
ROI(Region of interest): 탐지 영역
Object Detector: 객체 탐지
Contours: 윤곽
Gaussioan Blur: 노이즈를 줄여 흐릿하게 처리

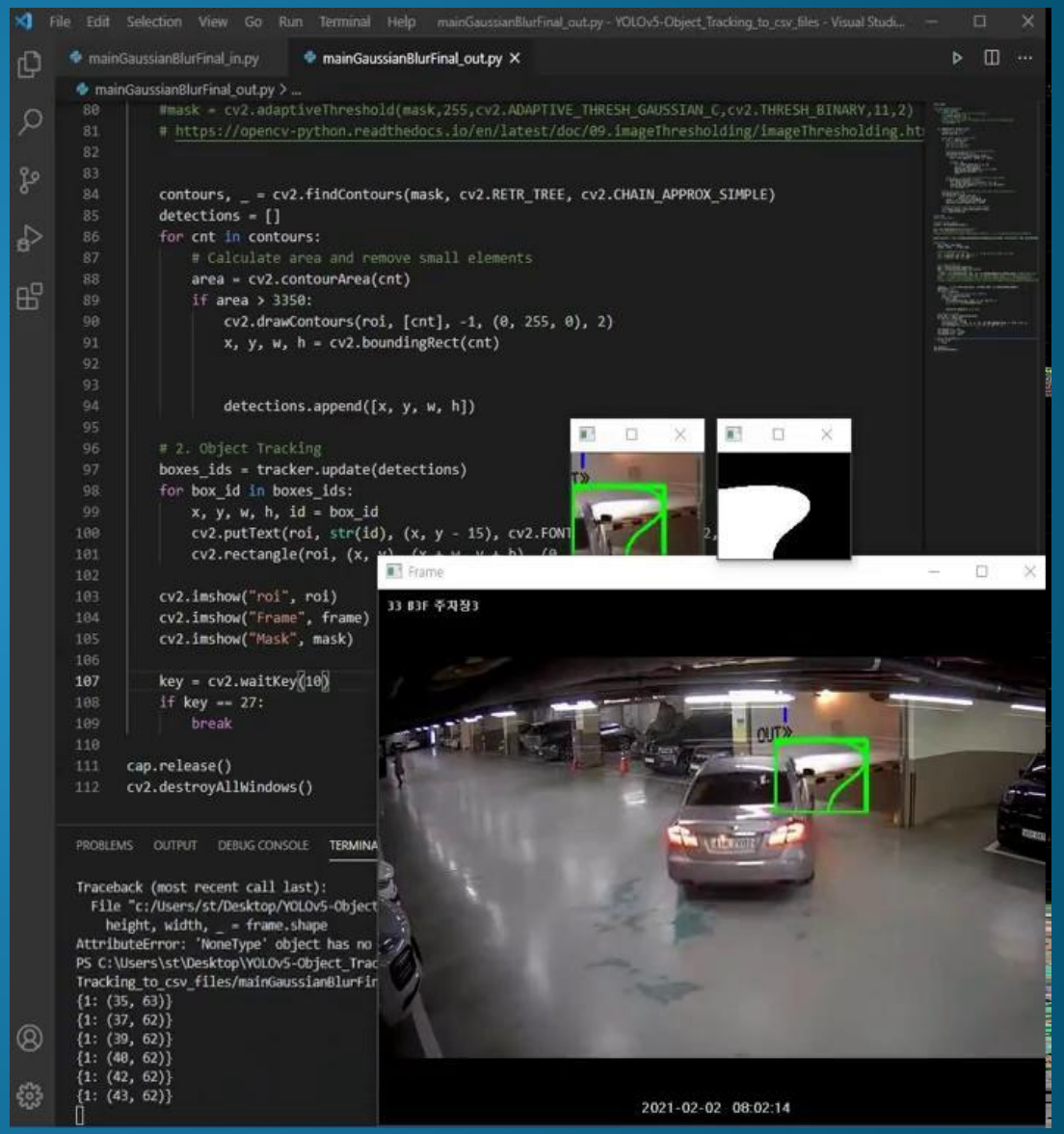
코드 설명
Object Tracking

```
    # 2. Object Tracking
    boxes_ids = tracker.update(detections)
    for box_id in boxes_ids:
        x, y, w, h, id = box_id
        cv2.putText(roi, str(id), (x, y - 15), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)
        cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)

    cv2.imshow("roi", roi)
    cv2.imshow("Frame", frame)
    cv2.imshow("Mask", mask)

    key = cv2.waitKey(10)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

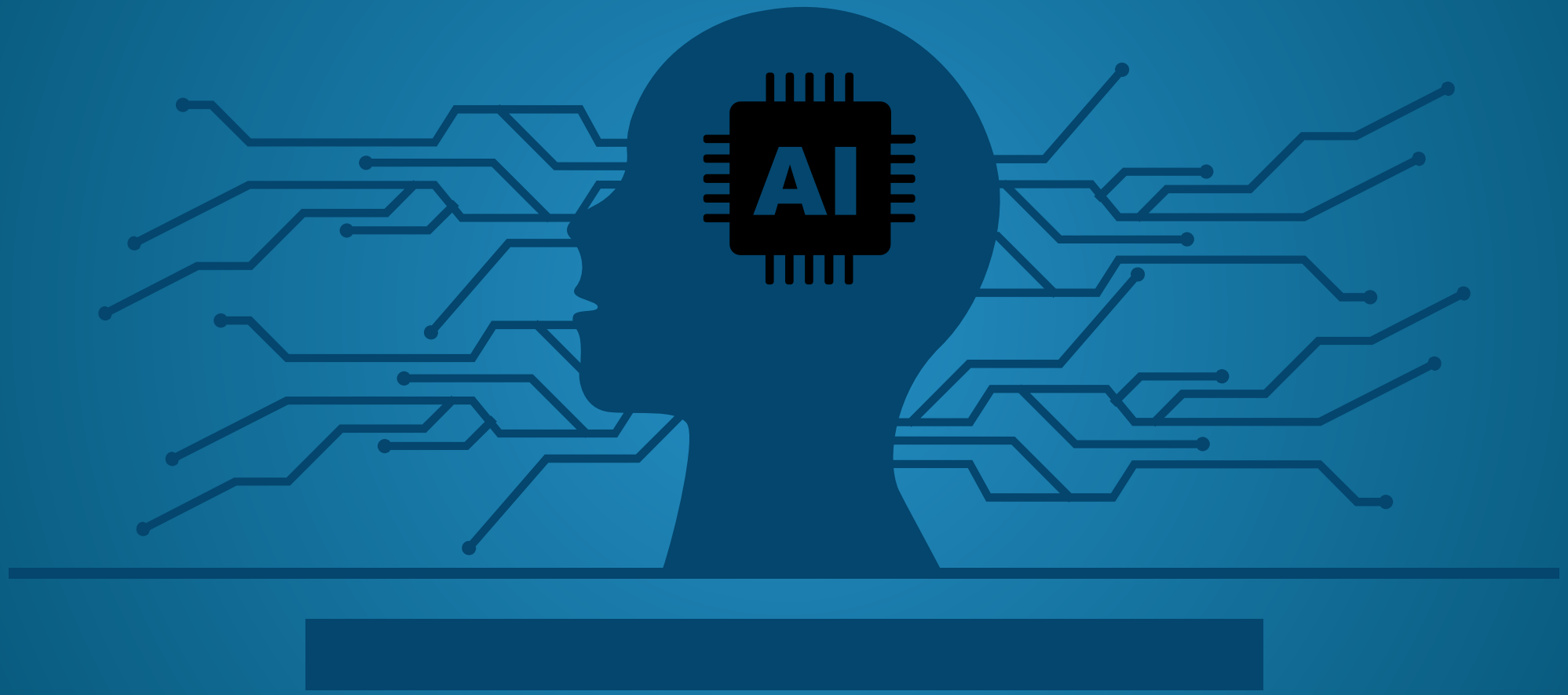
실행 파일
https://github.com/qwerlarlgus/YOLO_Project/tree/main/YOLOv5-Object_Tracking_to_csv_files

실행 영상


mainGaussianBlurFinal_in.py
mainGaussianBlurFinal_out.py
# each time a new object id detected, the count will increase by one
self.id_count = 0
def update(self, objects_rect):
# Objects boxes and ids
objects_bbs_ids = []
# Get center point of new object
for rect in objects_rect:
x, y, w, h = rect
cx = (x + x + w) // 2
cy = (y + y + h) // 2
# Find out if that object was detected already
same_object_detected = False
for id, pt in self.center_points.items():
dist = math.hypot(cx - pt[0], cy - pt[1
if dist < 25:
self.center_points[id] = (cx, cy)
print(self.center_points)
objects_bbs
same_object
break
New object is det
if same_object_dete
self.center_poi
objects_bbs_ids
self.id_count +
# Clean the dictionary
new_center_points = {}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINA
PS C:\Users\st\Desktop\YOLOv5-Object_Trac
Tracking_to_csv_files/mainGaussianBlurFin
{1: (34, 37)}
{1: (34, 36)}
{2: (30, 37)}
{2: (29, 37)}
PS C:\Users\st\Desktop\YOLOv5-Object_Trac
Tracking_to_csv_files/mainGaussianBlurFin
{1: (34, 37)}
{1: (34, 36)}
{2: (30, 37)}
{2: (29, 37)}
Frame
33 B3F 주차장3
OUT
2021-02-02 08:00:13


mainGaussianBlurFinal_in.py
mainGaussianBlurFinal_out.py
#mask = cv2.adaptiveThreshold(mask,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)
https://opencv-python.readthedocs.io/en/latest/doc/09.imageThresholding/imageThresholding.ht
contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
detections = []
for cnt in contours:
# Calculate area and remove small elements
area = cv2.contourArea(cnt)
if area > 3350:
cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
x, y, w, h = cv2.boundingRect(cnt)
detections.append([x, y, w, h])
# 2. Object Tracking
boxes_ids = tracker.update(detections)
for box_id in boxes_ids:
x, y, w, h, id = box_id
cv2.putText(roi, str(id), (x, y - 15), cv2.FONT
cv2.rectangle(roi, (x,
cv2.imshow("roi", roi)
cv2.imshow("Frame", frame)
cv2.imshow("Mask", mask)
key = cv2.waitKey(10)
if key == 27:
break
cap.release()
cv2.destroyAllWindows()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINA
Traceback (most recent call last):
File "c:/Users/st/Desktop/YOLOv5-Object
height, width, _ = frame.shape
AttributeError: 'NoneType' object has no
PS C:\Users\st\Desktop\YOLOv5-Object_Trac
Tracking_to_csv_files/mainGaussianBlurFin
{1: (35, 63)}
{1: (37, 62)}
{1: (39, 62)}
{1: (40, 62)}
{1: (42, 62)}
{1: (43, 62)}
Frame
33 B3F 주차장3
OUT
2021-02-02 08:02:14

결과



대시보드
Object Tracking

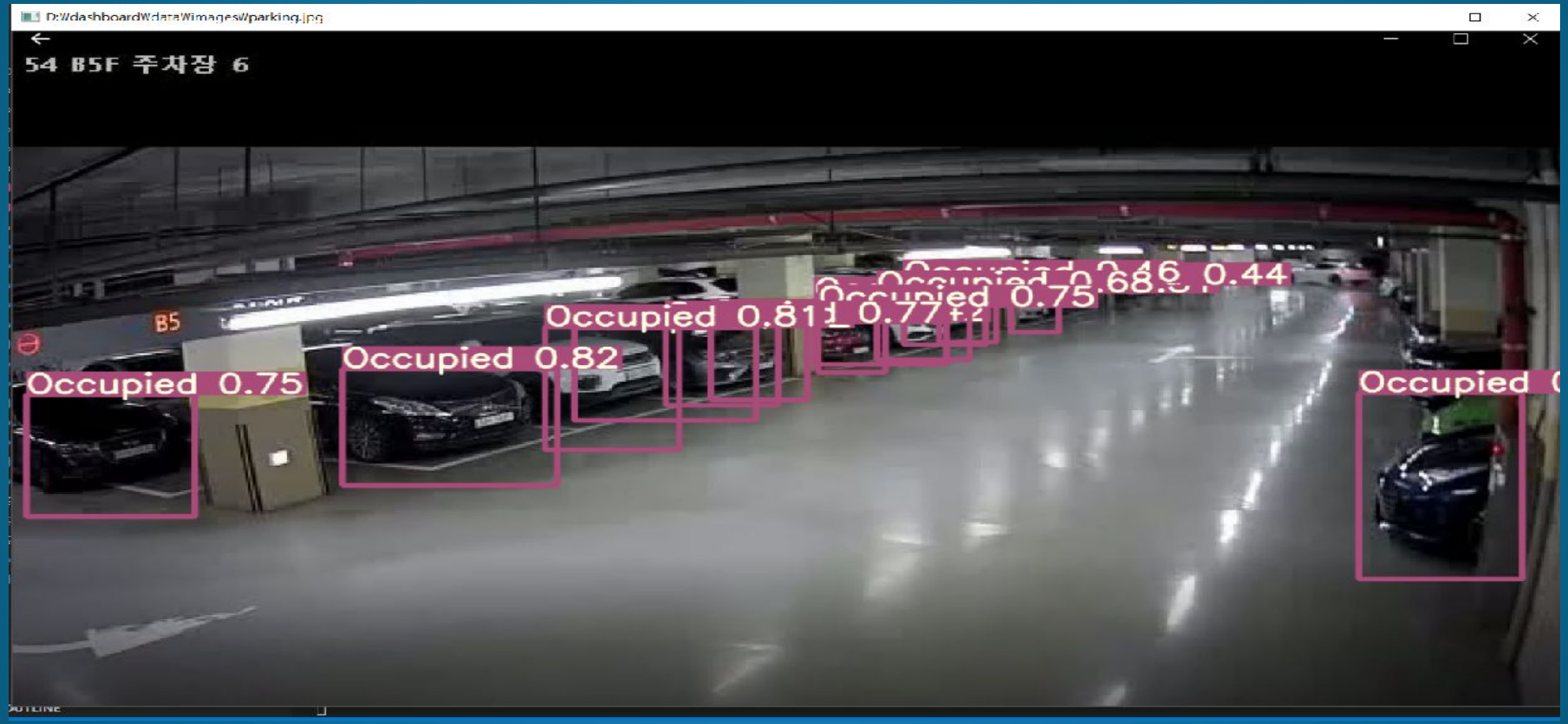
dashboard.py

```python
import sys
from mainGaussianBlurentry import *
from mainGaussianBlurexit  import *
import Empty

if __name__ == '__main__':
    argument = sys.argv
    del argument[0]

a, b = Empty.detect()


print('Empty    : {:03d}'.format(a))
print('Occupied : {:03d}'.format(b))

print('# of Entry : {:03d}'.format(entry))
print('# of Exit  : {:03d}'.format(exit))

```

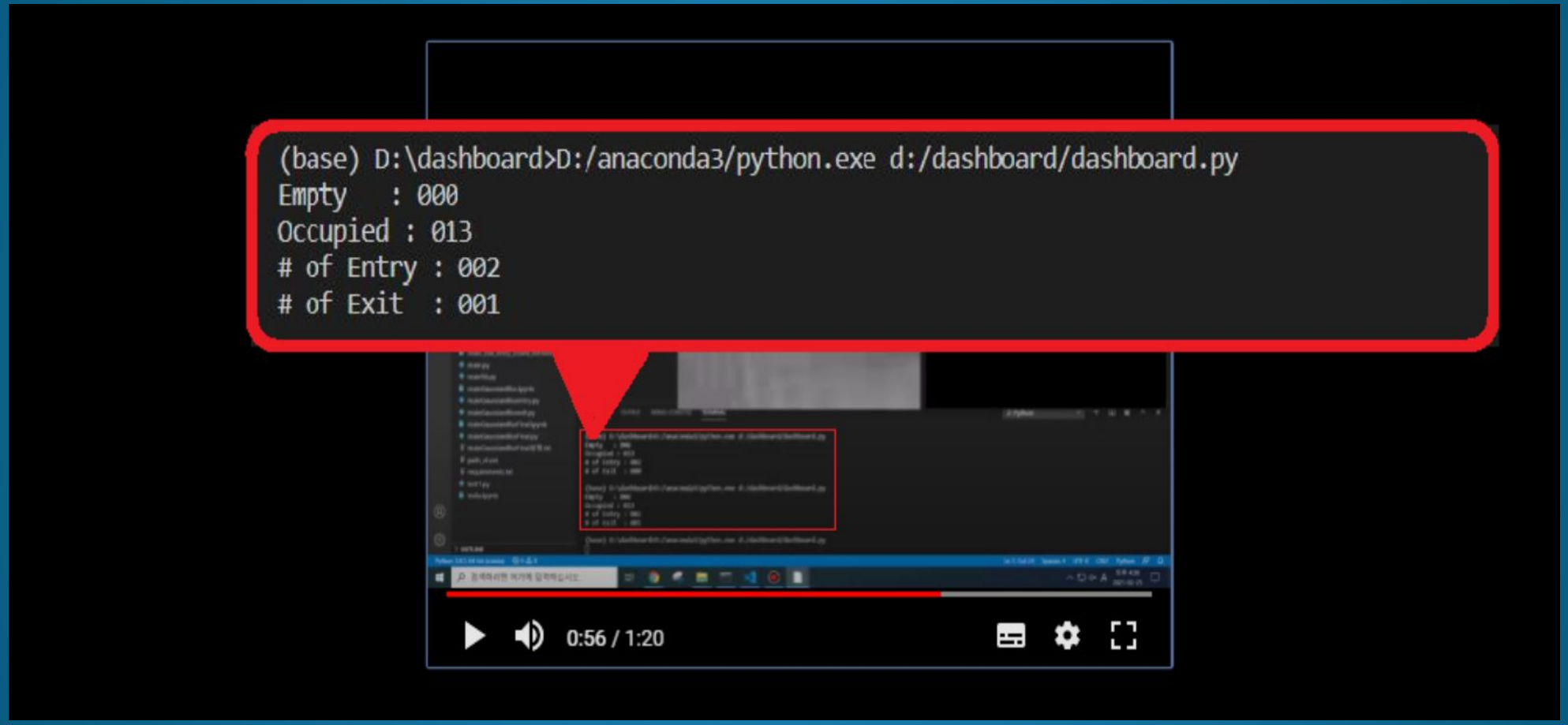
Empty: 빈 공간
Occupied: 차있는 공안
Entry: 들어온 자동차 대수
Exit: 나간 자동차 대수

최종결과

최종결과


D:\dashboard\data\images\parking.jpg
54 B5F 주차장 6
B5
Occupied 0.75
Occupied 0.82
Occupied 0.81
Occupied 0.75
Occupied 0.46 0.44
Occupied

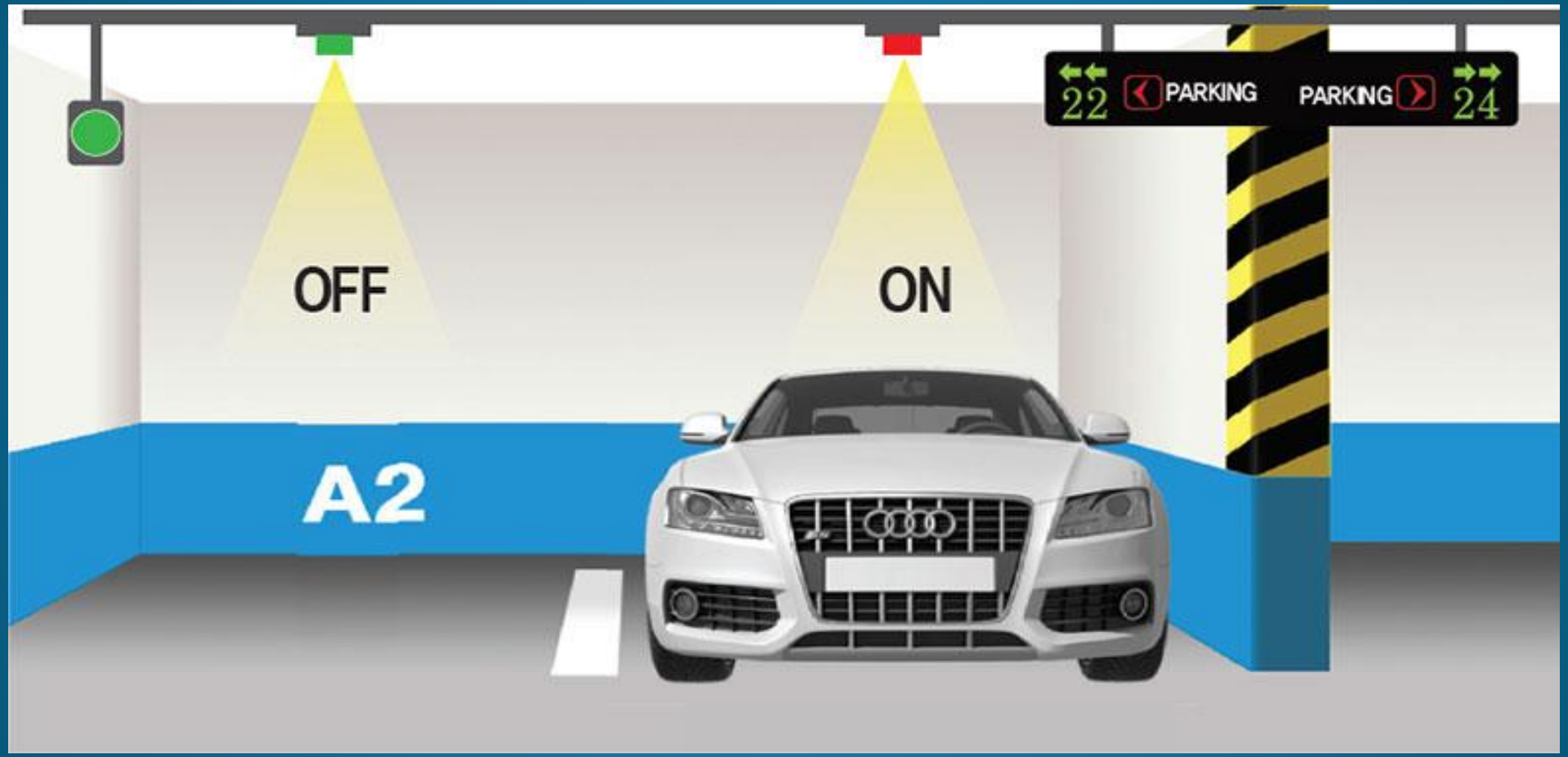
최종결과


(base) D:\dashboard>D:/anaconda3/python.exe d:/dashboard/dashboard.py
Empty : 000
Occupied : 013
# of Entry : 002
# of Exit : 001
0:56 / 1:20

대시보드

- 대시보드로 주차장의 실시간 확인가능
- 센서 설치 없이 영상만으로 사용 가능
- 이미 적용된 시스템에 폭넓게 적용가능
- 실내 뿐만 아닌 실외에서도 사용 가능

차이점



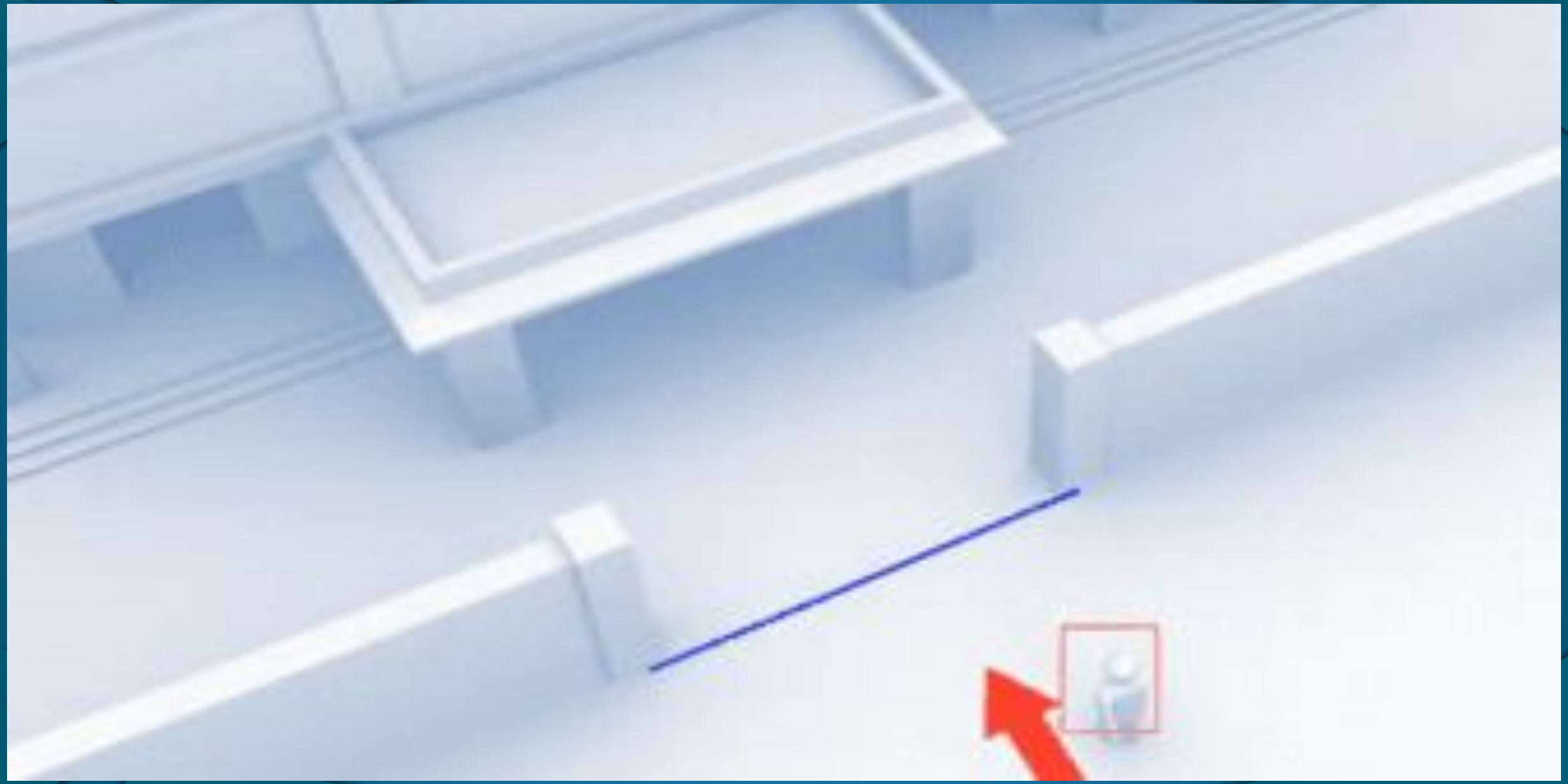
사진출처 : http://smarcom.biz/ko/parking-guidance-system/

차이점

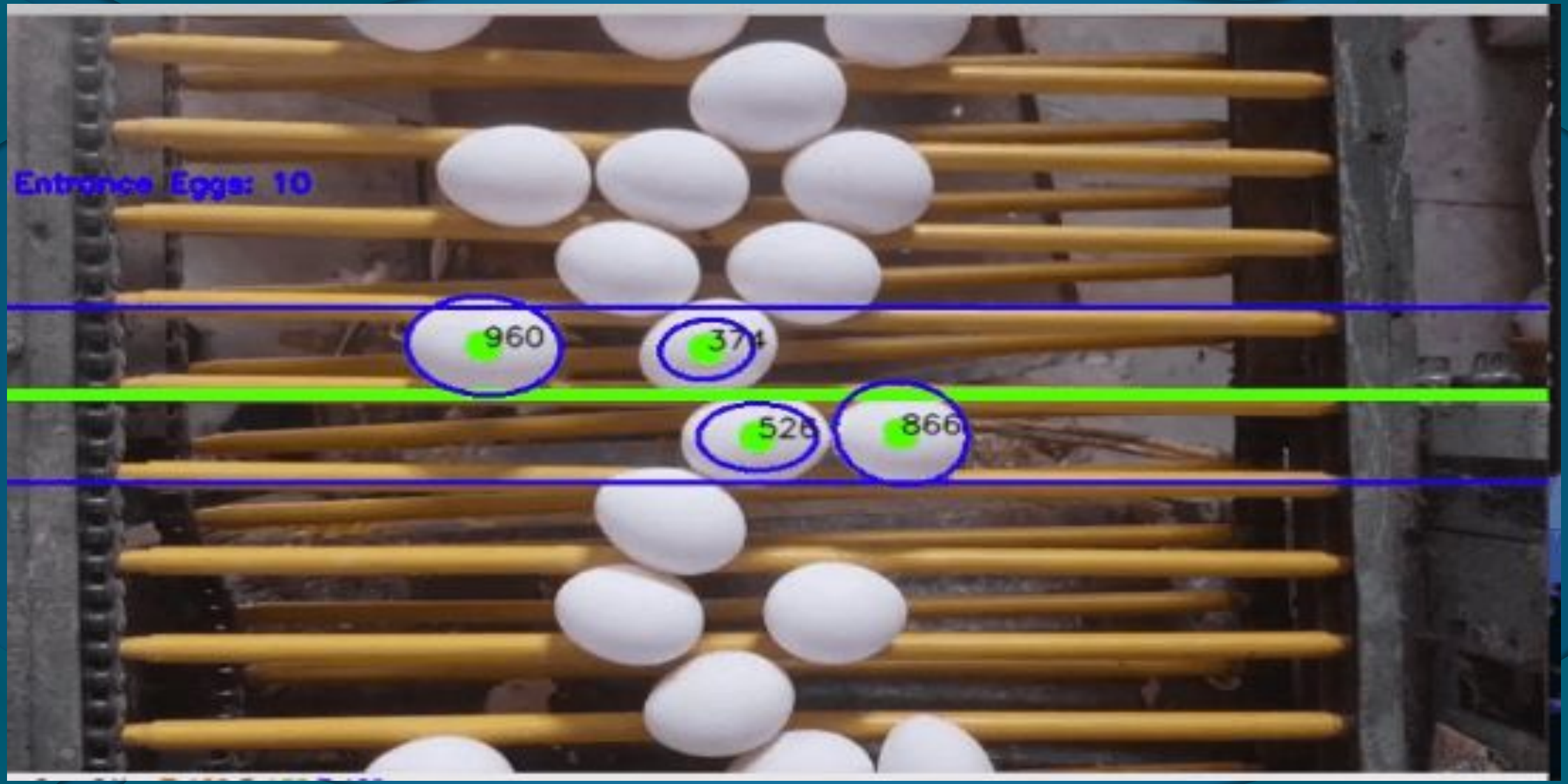
사진출처 : http://www.hmobility.co.kr/press/4312

https://github.com/0201shj

활용 방안



출처 : https://www.youtube.com/watch?v=vDV43kZB0yM



출처 : https://www.youtube.com/watch?v=XKEeAPnmWB0

출처 : https://www.youtube.com/watch?v=eeIEH2wjvhg

출처 : https://www.youtube.com/watch?v=bMA1YPJ5ifo

출처 : https://youtu.be/eABA3Dw5ADQ

향후 과제

- 학습 데이터 추가 필요
- 카메라/디스플레이 보정
- 오류 수정

질의응답

THANK YOU

컴퓨터 비전을 이용한 주차장 관리 시스템