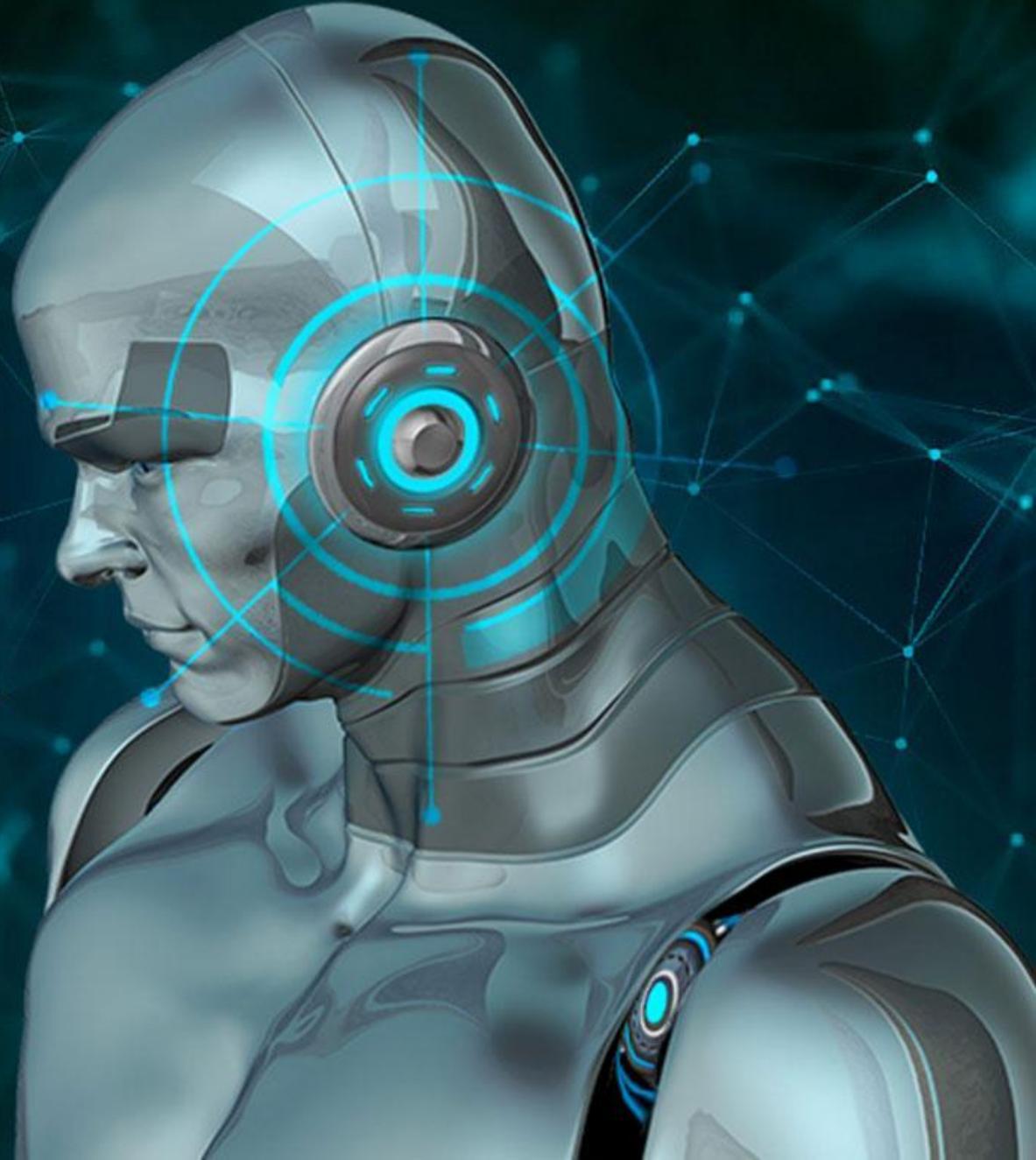


스마트 주차장 관리

차량 출입 관리 + 객체를 인식하는 CV





Contents_s

01 배경 및 목적

02 사업성 검토

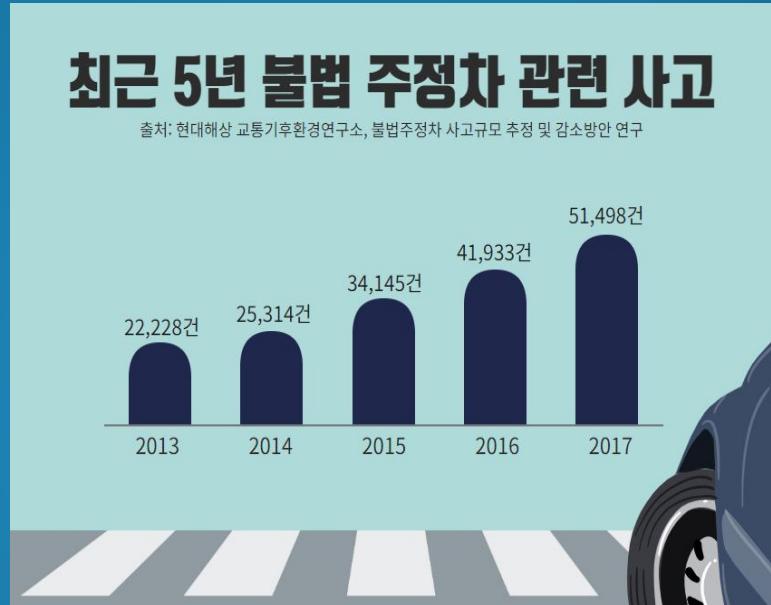
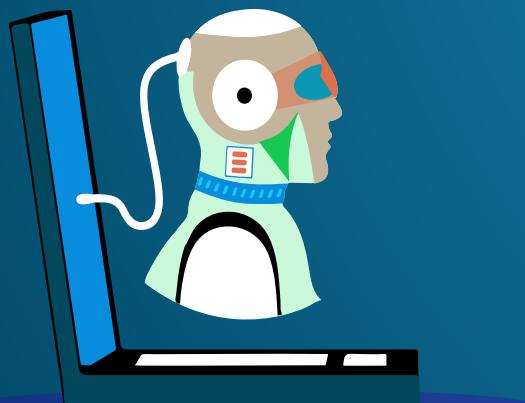
03 분석 결과

04 활용방안

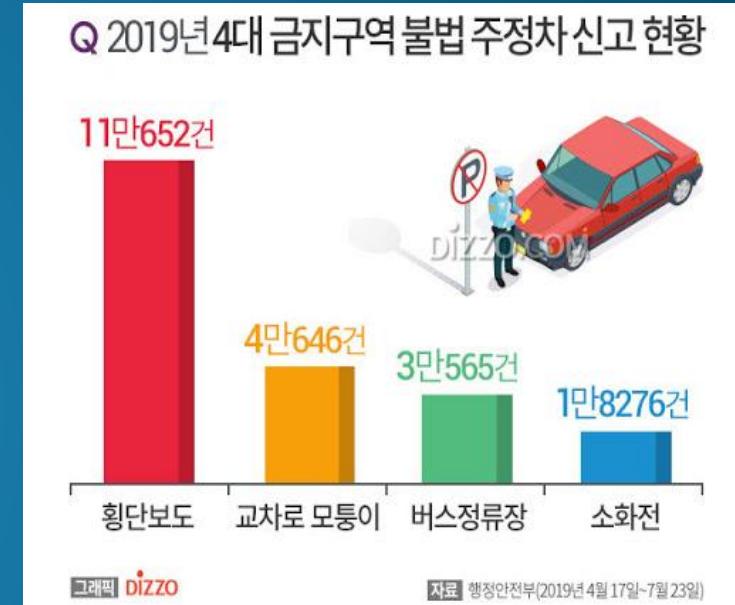
01. 분석 배경



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

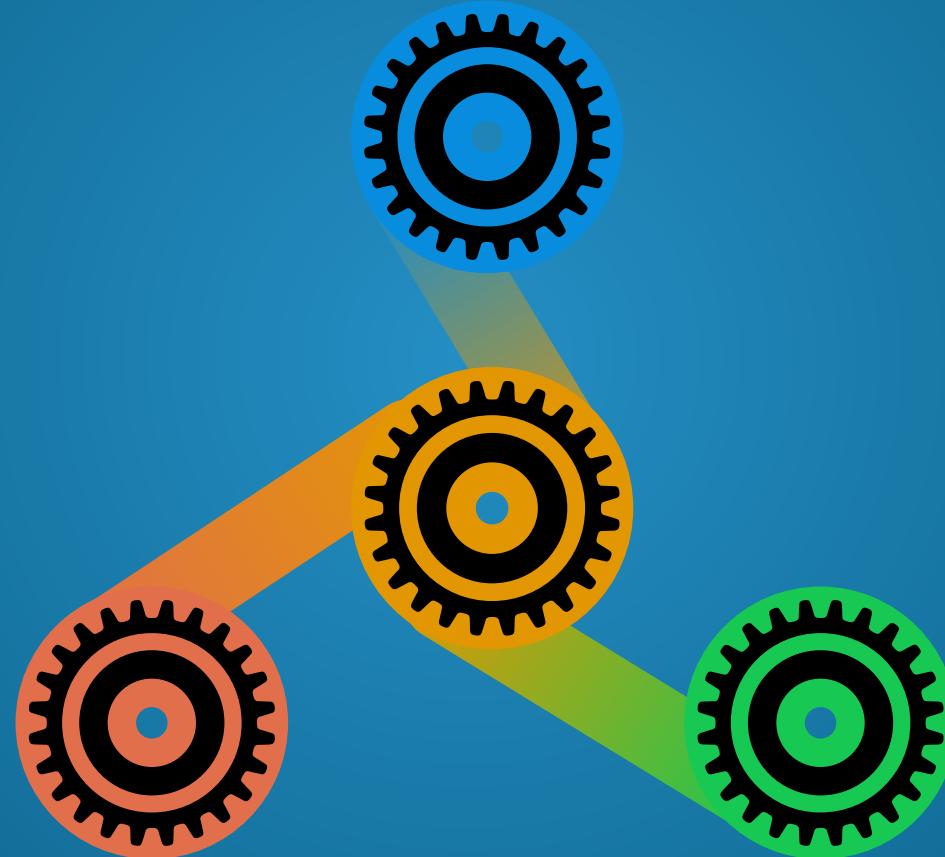
- 내원객 자차 이용 늘면서 혼잡 심화, 병원 "대중교통 이용 등 협조" 요청
- "코로나19 걱정 때문에 자차 이용 비율은 몇 배 이상 증가한 것으로 파악하고 있다"
- "병원 홈페이지에 표시된 곳(P1~P5) 외 발렛 서비스를 제공하는 몇 군대를 합치면 총 320여대 정도이며 피크 타임 기준 하루 평균 이용 고객들은 700~800여대 정도 된다"고 설명했다.

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



법적제재로 부터 원활,
특정 위치에서 측정하여
사람이 아닌 차량만 감지



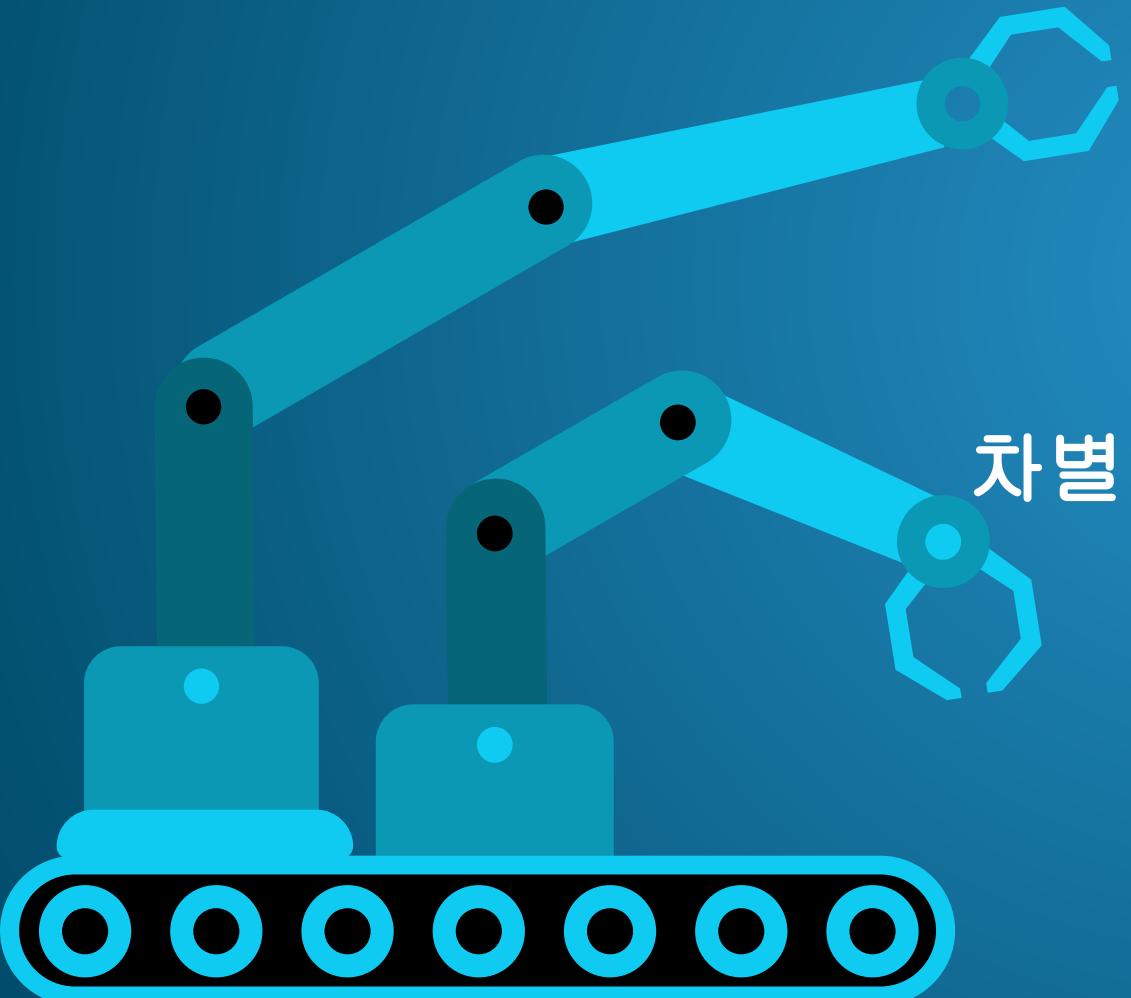
기존 CCTV의 사용으로
실내외에서 측정하여
비대면과 함께 안전성 보장

경제적 측면

법, 제도적 측면

사회적 측면

02. 사업성 검토



시장성

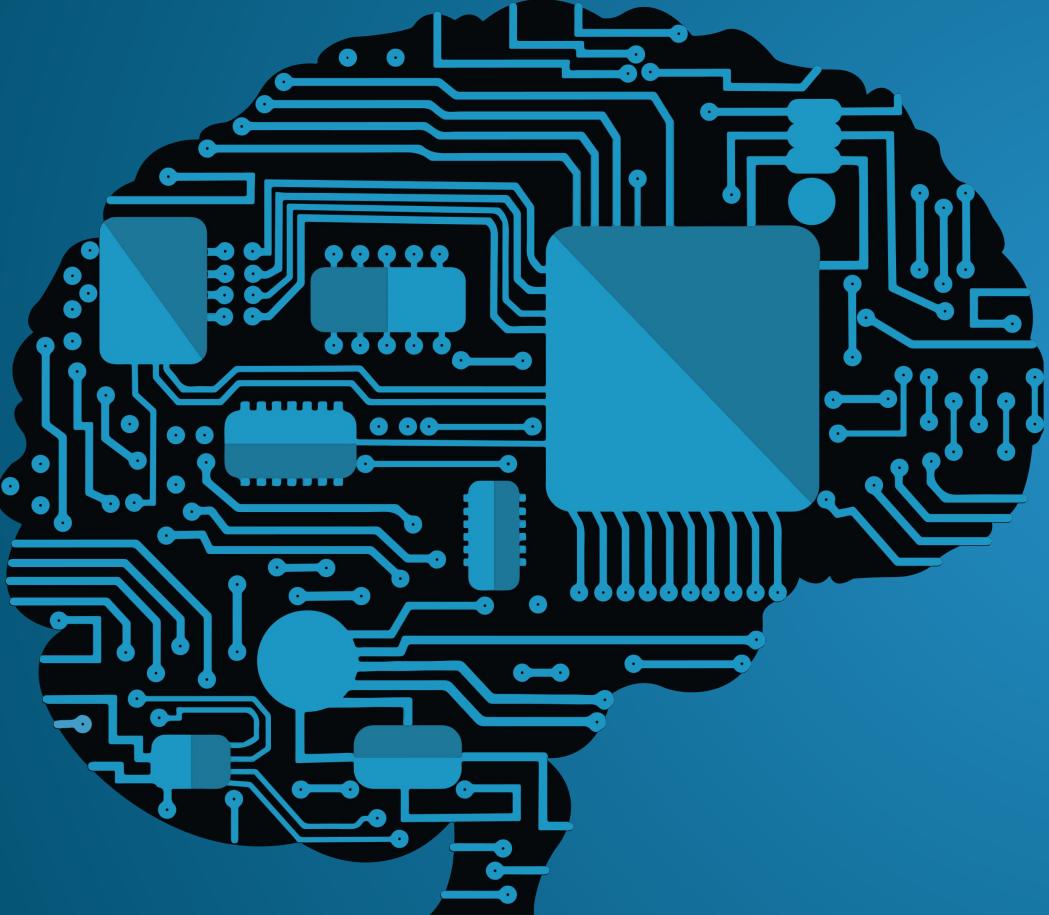
CCTV가 있는 거의 모든
주차장에 사용 가능

경쟁력

별도의 센서 설치 없이 소프트웨어
만으로 사용 가능한 경제성

차별성과 전략

기존의 센서 시스템과 다른 방식



발전하려는 기술의 접목

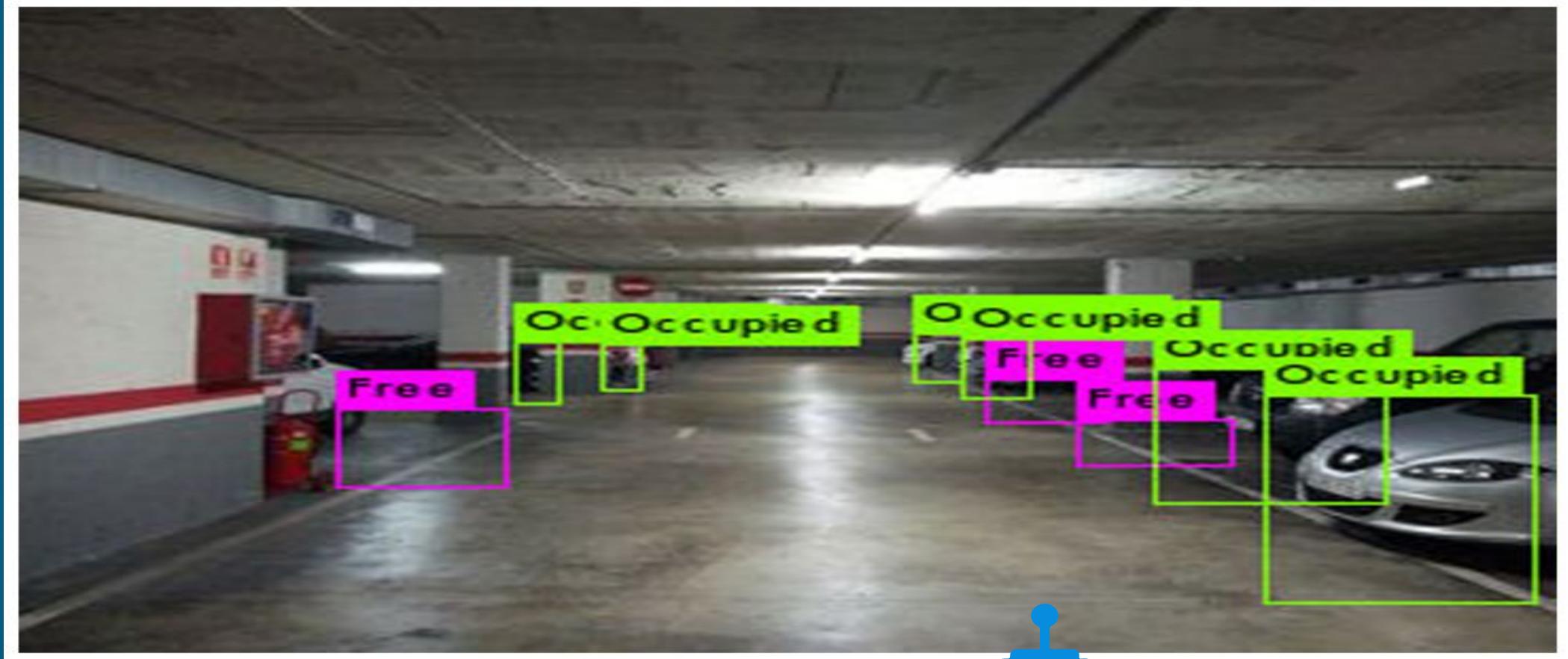
최근에 급격한 발전을 이룬
CV(Computer Vision) 기술을 응용하여 개발.

사용한 프로그램과 데이터 정리



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

활용 방법



Object Detection

활용 방법



Object Tracking

활용 방법



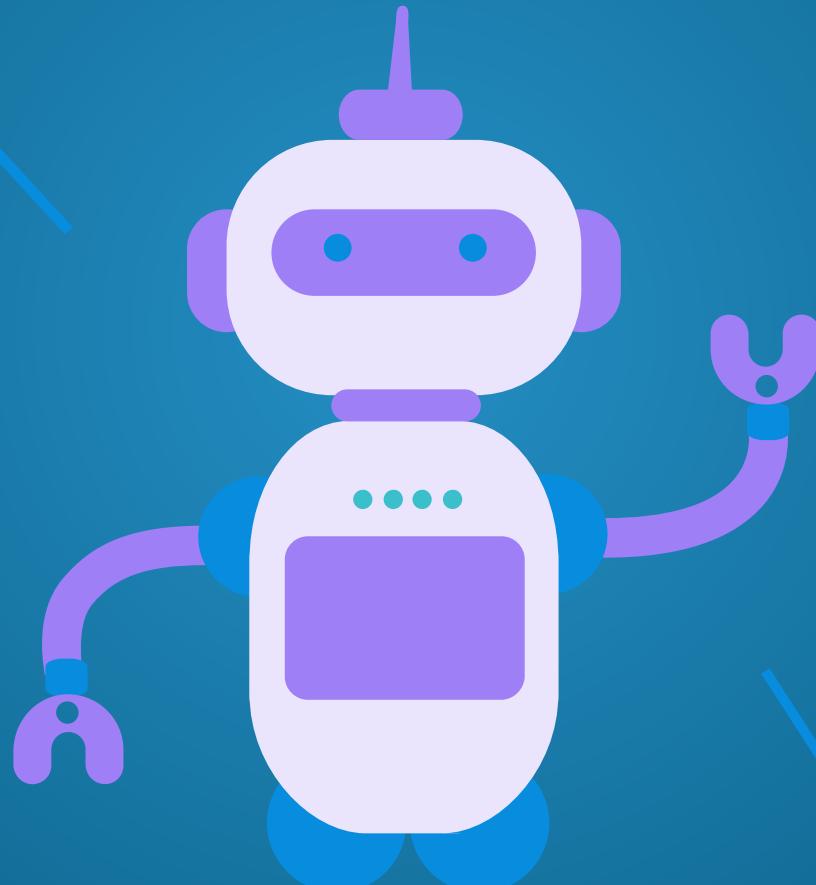
Object Tracking



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



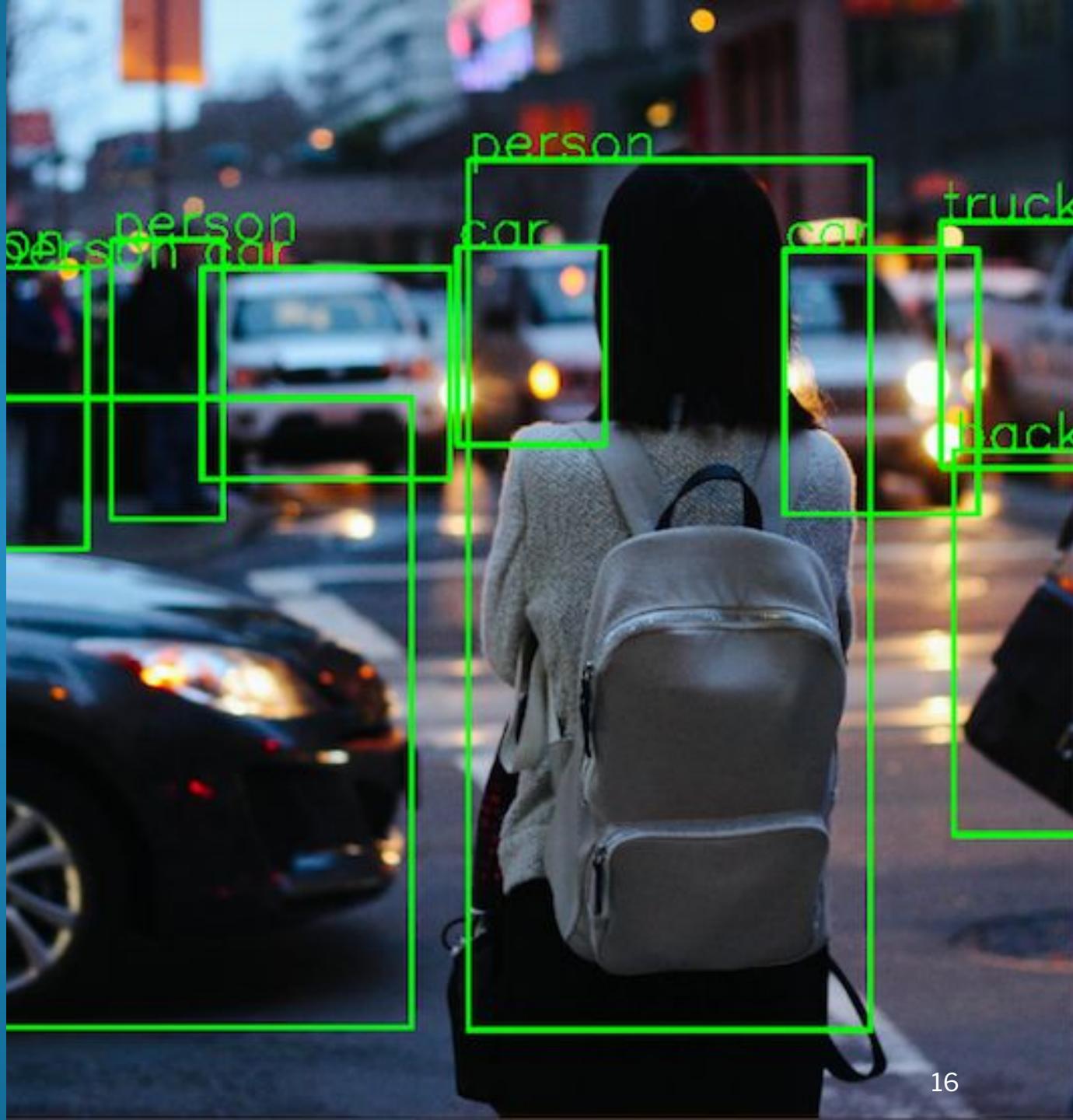
고해상도 영상 생성을 위한 이미지
스티칭, 영상 검색, 적목 현상 제거, 안구
운동추적



Object Detection

객체 탐지(**Object Detection**)는 무엇일까요?

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





Computer Vision Annotation Tool



CVAT
Object Detection

CVAT Projects Tasks Models Analytics GitHub Help 0201shj ▾

Tasks Search 

+ Create new task

Task ID	Task Name	Status	Jobs	Actions
#47898	CCTV3	Pending	0 of 1 jobs	Open
#43558	CCTV2	Pending	0 of 1 jobs	Open
#43529	CCTV	Pending	0 of 1 jobs	Open
#40916	foods	Pending	0 of 1 jobs	Open

< 1 >

CCTV3 

Task #47898 Created by 0201shj on February 16th 2021
Assigned to

Issue Tracker
Not specified 



 Raw  Constructor  Copy

Add label  Occupied  Free 

Jobs  0 of 1 jobs

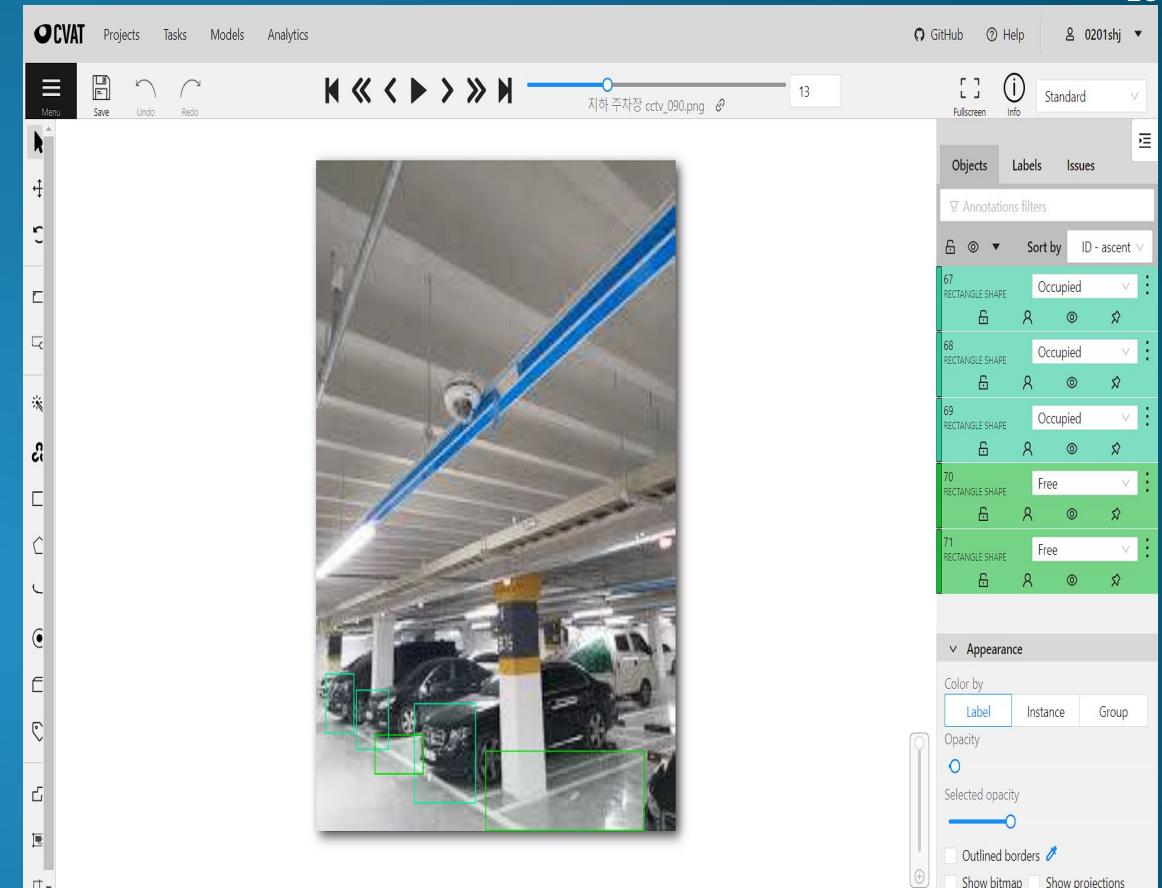
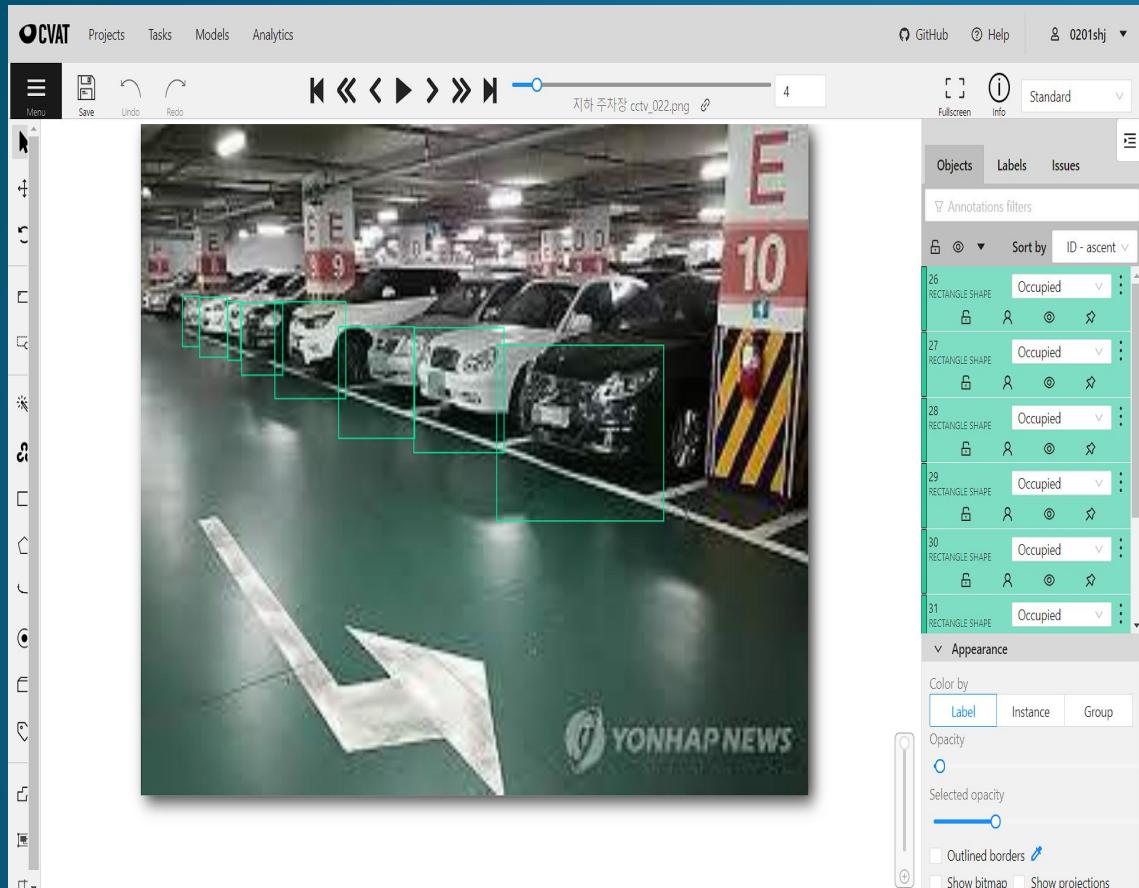
Job	Frames	Status	Started on	Duration	Assignee	Reviewer
Job #78701	0-42	annotation 	February 16th 2021 15:02	9 days	<input type="text" value="Select a user"/>	<input type="text" value="Select a user"/>

< 1 >



CVAT

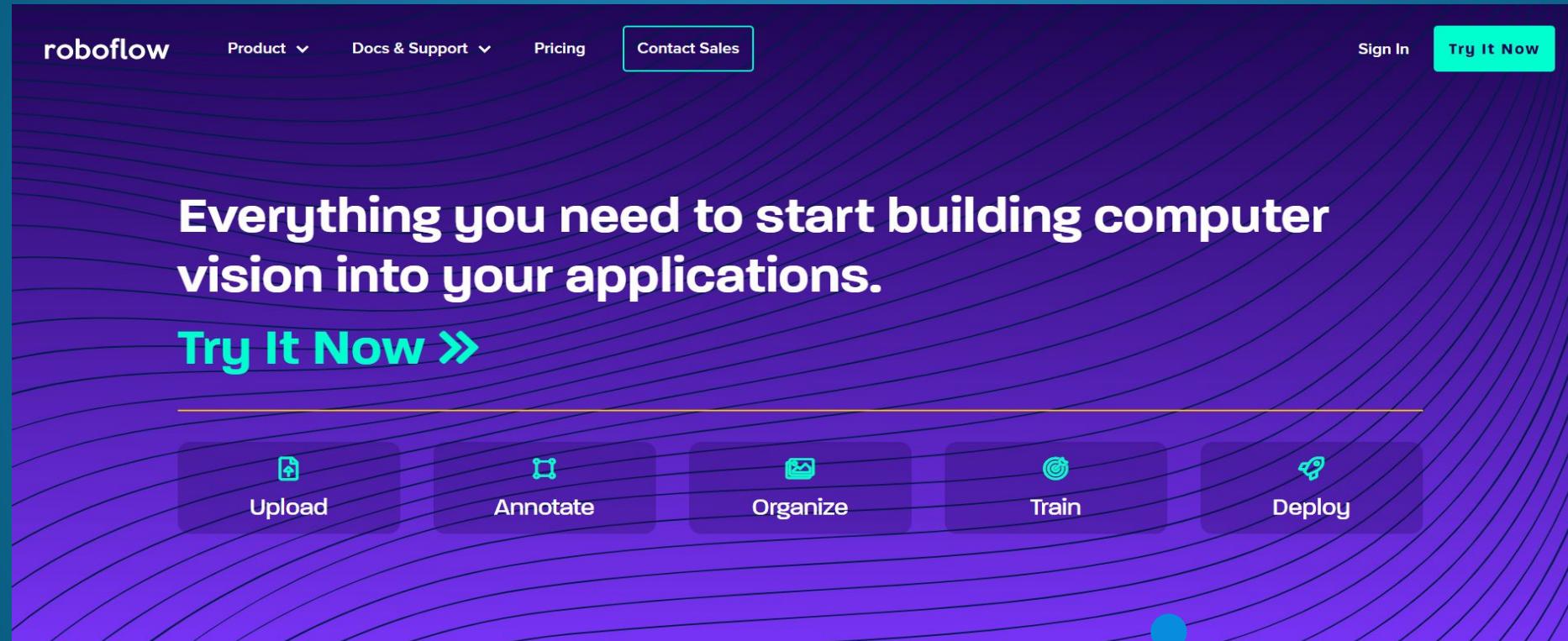
Object Detection



CVAT Object Detection

roboflow

Roboflow



Roboflow
Object Detection

roboflow

Datasets Account Docs [Create Dataset](#)

PROJECTS All Datasets **6**

RESOURCES Quickstart Guide Documentation Tutorials Public Datasets Model Library Help & Support Contact Sales

Free Tier Usage 306 / 1,000 source images 1,342 / 5,000 generated images

[Link a Credit Card](#) [Log Out](#)

Welcome to Roboflow, let's get started.

- Add Your Team** Roboflow is the source of record for your datasets.
- Complete Your Profile** Tell us a bit about yourself so we can personalize your account.
- Link a Payment Method** Remove your account's limits and level up past the free tier.

[Invite](#) 30 seconds [Start](#) 30 seconds [Link Now](#) 30 seconds

All Datasets

Dataset Name	Type	Images	Exports	Last Updated
ParkKing2	Object Detection (Bounding Box)	66 images	3 exports	17 days ago
Parking	Object Detection (Bounding Box)	43 images	3 exports	21 days ago
Korean_Foods	Object Detection (Bounding Box)			

[View all images \(66\)](#)

ParkKing2 Dataset [+ Generate](#)

[Back to Datasets](#)

Train/Test Split Your images are split at upload time. [Learn more](#).

Train	Valid	Test
59	7	0

[Edit Splits](#)

Augmentation Output For each image in your training set, how many augmented versions do you want to generate?

3 (Starter Plan Max: 3)

Output Size: Up to 184 images.

Preprocessing Options Applied to all images in dataset

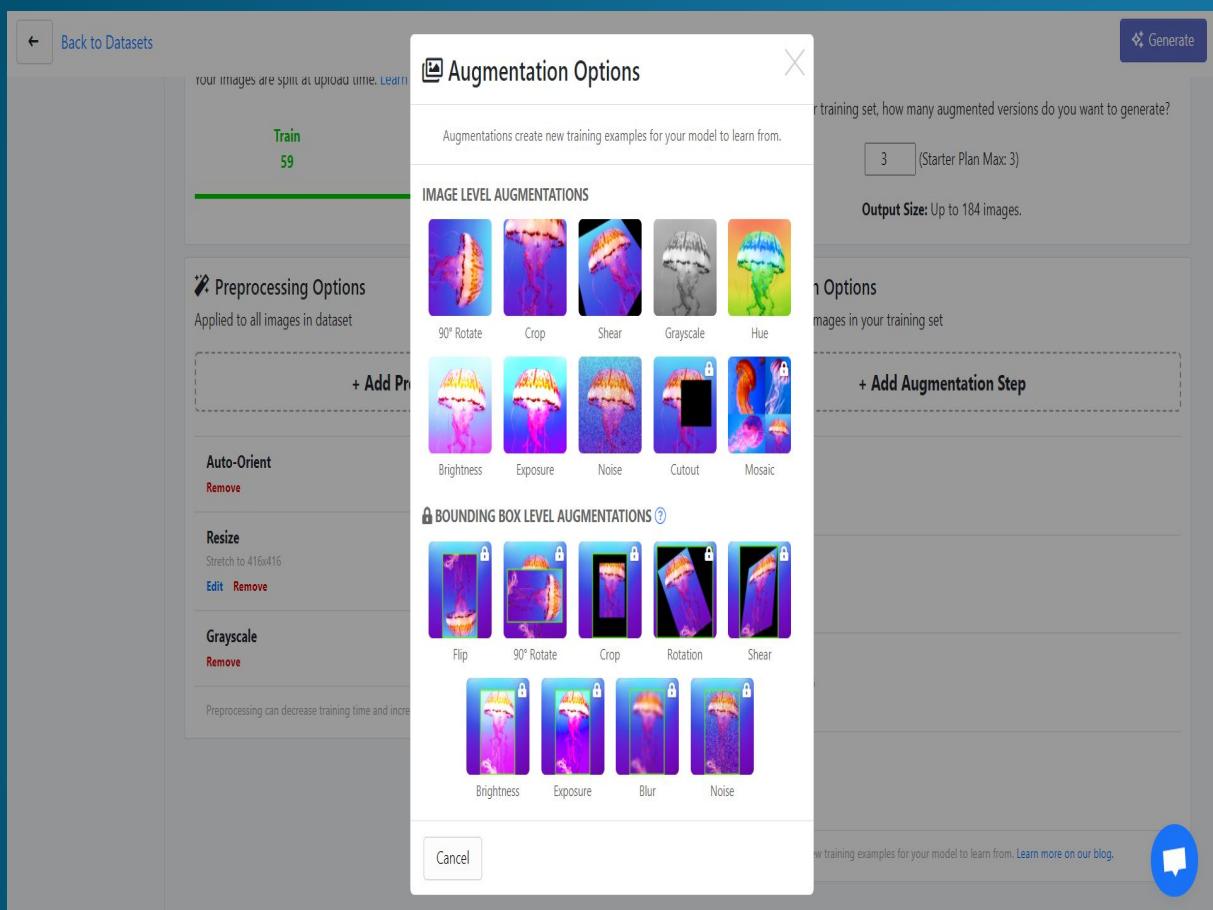
- + Add Preprocessing Step
- Auto-Orient [Edit](#) [Remove](#)
- Resize Stretch to 416x416 [Edit](#) [Remove](#)
- Grayscale [Edit](#) [Remove](#)

Augmentation Options Randomly applied to images in your training set

- + Add Augmentation Step
- Flip Horizontal, Vertical [Edit](#) [Remove](#)
- Rotation Between -15° and +15° [Edit](#) [Remove](#)
- Saturation Between -40% and +40% [Edit](#) [Remove](#)



Roboflow Object Detection



Dataset Version Augmentation

Back to Datasets

View all images (184)

Train 177

Preprocessing Options
Applied to all images in dataset

Auto-Orient

Resize
Stretch to 416x416

Preprocessing can decrease training time and increase accuracy.

Rotation

This is an archived version so its settings can't be changed. To generate a new version, go to the current dataset.

Add variability to rotations to help your model be more resilient to camera roll.

0° 15° 45°

Why should I use the Random Rotate augmentation? ⓘ

It helps your model detect objects even when the camera or subject are not perfectly aligned.
via Roboflow Blog

Go Back

Augmentations create new training examples for your model to learn from. Learn more on our blog.

Windows taskbar icons: Search, Task View, File Explorer, Edge, Task Manager, Start.

Bottom right: 오전 11:53, 2021-02-24



Roboflow

Object Detection

The screenshot shows the Roboflow dataset interface for a dataset named "ParKing2" last updated on February 4, 2021, at 1:58pm. A modal window titled "Your Download Code" is displayed, containing three tabs: Jupyter, Terminal, and Raw URL. The Terminal tab is selected, showing a command-line snippet to download and unzip the dataset:

```
curl -L "https://app.roboflow.com/ds/L1wQt0YfxI?key=4FJ5VR0nKr" > roboflow.zip;
unzip roboflow.zip; rm roboflow.zip
```

A warning message below the code states: "Warning: Do not share this link beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies." A "Done" button is at the bottom right of the modal.

To the right of the modal, a section titled "Ready to Train?" is highlighted with a red box. It includes a dropdown menu set to "YOLO v5 PyTorch", a "Download Zip" button, and a "Get Link" button.

Below this, sections for "Train/Test Split" (Train: 177, Valid: 7, Test: 0), "Preprocessing Options" (Applied to all images in dataset), and "Augmentation Options" (Randomly applied to images in your training set) are visible.



Roboflow Object Detection



YOLOv5



Yolov5
Object Detection

참고자료 / 깃허브 YOLOv5

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

Roboflow Image Training Data set

```
[1] icurl -L "https://app.roboflow.com/ds/L1wQb0YfxI?key=4FJ5YBOnKc" > roboflow.zip | unzip roboflow.zip; rm roboflow.zip
```

```
extracting: train/labels/cctv_1023.jpg.rf.7b30f4a67e9cac79400d23275bf1aec64.txt
extracting: train/labels/cctv_1023.jpg.rf.a3b5a42cc539fe1720e18b4ef02efc5b5.txt
extracting: train/labels/cctv_1023.jpg.rf.bf750cb73e87c2f6123ac175685fd149.txt
extracting: train/labels/cctv_1024.jpg.rf.8763b300588cd9c5b803ed91d8c92a20.txt
extracting: train/labels/cctv_1024.jpg.rf.8af42e6bad421fe9d7135ff873cf729f.txt
extracting: train/labels/cctv_1024.jpg.rf.b306099e227574236esbc250e5ed6104.txt
extracting: train/labels/cctv_1025.jpg.rf.4b69fb0e7b038676180c7ff24c2seit767.txt
extracting: train/labels/cctv_1025.jpg.rf.7504f43bf3b924738048faa284c13d41.txt
extracting: train/labels/cctv_1025.jpg.rf.776307a908e0bb530b844506c1256a69.txt
extracting: train/labels/cctv_1026.jpg.rf.089d2fc4b7e1309ee16554588d1ara81.txt
extracting: train/labels/cctv_1026.jpg.rf.952a49493c6a9e13c0117092106f0ff47.txt
extracting: train/labels/cctv_1026.jpg.rf.ef26df15478e4042a2963999601174b3.txt
extracting: train/labels/cctv_110.png.rf.450b59e220f777106f56e3888bc54ef14.txt
extracting: train/labels/cctv_110.png.rf.8a690a2a9c5859b4c1fd5cc6eabca34c7.txt
extracting: train/labels/cctv_110.png.rf.e71f0952cc0101a1e8622d6c6207f81.txt
extracting: train/labels/cctv_125.png.rf.4e90e20d204829b3bb16227c92a4e836.txt
extracting: train/labels/cctv_125.png.rf.872c8aec5c8c91cc2573c86f78aa4c76.txt
extracting: train/labels/cctv_125.png.rf.b047b3a7969e9aa84bd7a7c18a6c1d.txt
extracting: train/labels/cctv_134.png.rf.580911bba27bb690062139c1cbe04/43.txt
extracting: train/labels/cctv_134.png.rf.5b7ee05b4c5b7e9c949d4cb99456cd8a.txt
extracting: train/labels/cctv_134.png.rf.b073a0bbb186321c4b57ab6d409a20bd9.txt
extracting: train/labels/cctv_149.png.rf.2cd77d39c104f0f5a18520d560fa213.txt
extracting: train/labels/cctv_149.png.rf.44160c002c857b95a5eebe10c011fa46.txt
extracting: train/labels/cctv_149.png.rf.c2149d1dd1dc987d14ae22e18c8b51.txt
extracting: train/labels/cctv_152.png.rf.20945f80c03dd28e850906107902c5f5.txt
```



Yolov5
Object Detection

Yolov5 link download

```
[2] # clone YOL0v5 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
                                              'CPU'))
|██████████| 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

# YOL0v5 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
]

# YOL0v5 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]],
 [-1, 1, nn.Upsample, [None, 2, 'nearest']]
```



```
[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

# YOL0v5 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
]

# YOL0v5 head
head:
[[-1, 1, Conv, [512, 1, 1]],
 [-1, 1, nn.Upsample, [None, 2, 'nearest']],
 [[-1, 6], 1, Concat, [1]], # cat backbone P4
```

```
# 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
```



```
# 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
```



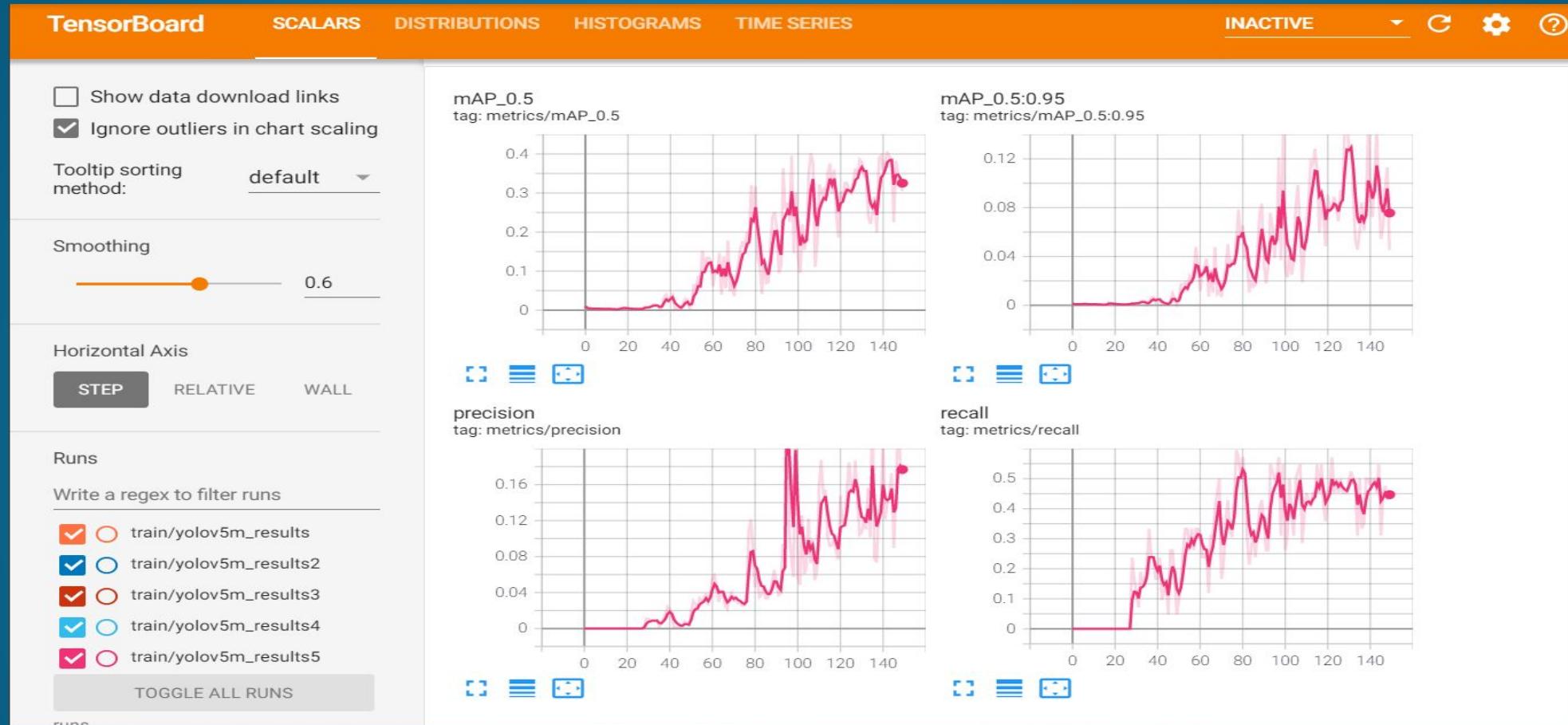
Yolov5
Object Detection

```
[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: .95: 100% 1/1 [00:00<00:00, 13.13it/s]
	all		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: .95: 100% 1/1 [00:00<00:00, 11.52it/s]
	all		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: .95: 100% 1/1 [00:00<00:00, 13.08it/s]
	all		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: .95: 100% 1/1 [00:00<00:00, 14.09it/s]
	all		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: .95: 100% 1/1 [00:00<00:00, 13.08it/s]
	all		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	0.2182	12	416: 100% 12/12 [00:01<00:00, 7.10it/s]



**Yolov5
Object Detection**



Yolov5
Object Detection

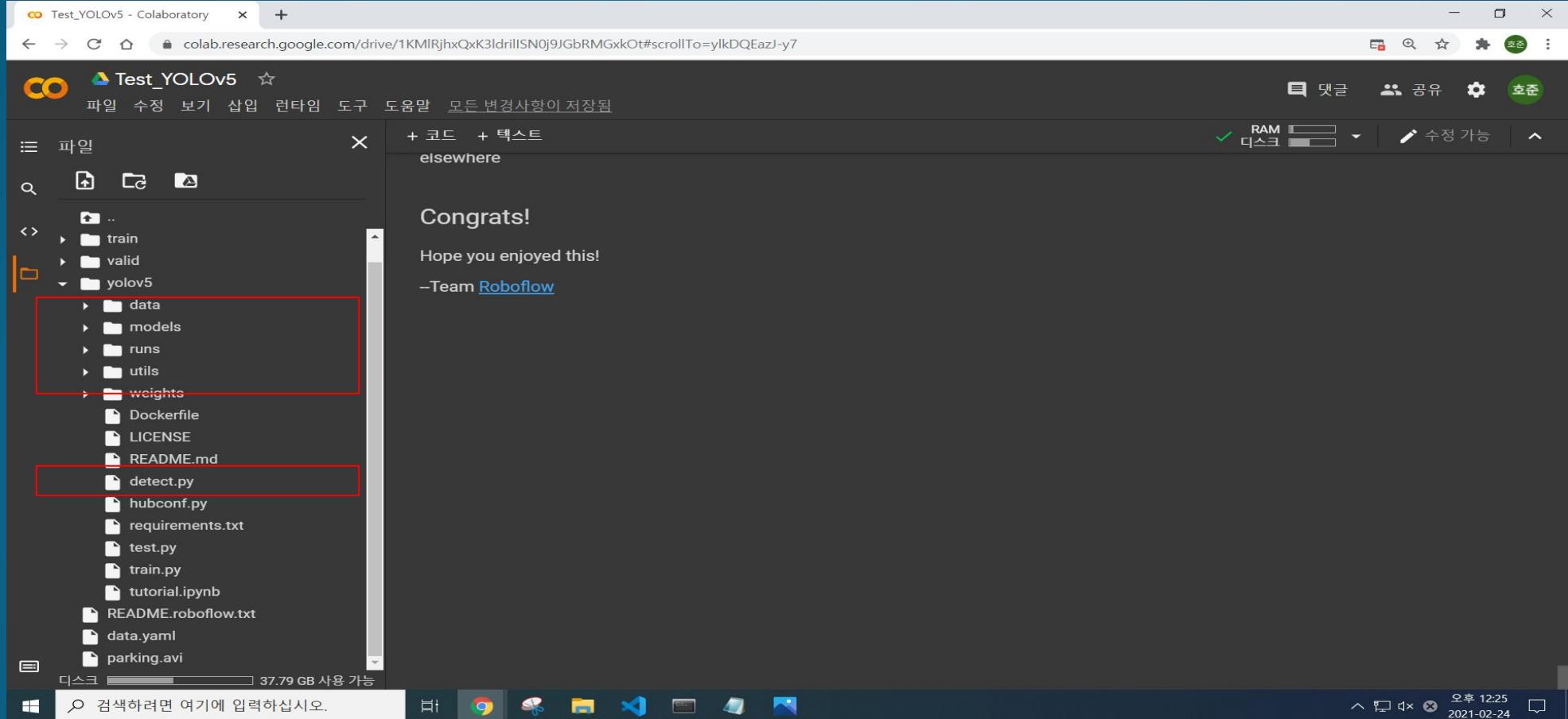


checking model - best.pt

```
[ ] !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

The screenshot shows the Visual Studio Code (VS Code) interface with a Python debugger session. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar indicates "detect1.py - detect-data". The left sidebar features icons for RUN, No Configurations, Variables, Locals, Watch, and others. The main area displays three tabs: "main1.py", "detect1.py", and "detect1.py C:\...\.dd". The "Locals" tab is active, showing variables and their values:

- > c: tensor(0.)
- > classify: False
- > colors: [[103, 145, 65], [174, 115,...]
- > dataset: <utils.datasets.LoadImages...
- > det: tensor([[4.58000e+02, 1.93000e...
- > device: device(type='cpu')
- > gn: tensor([1045, 675, 1045, 6
- > half: False
- > i: 0
- > im0: array([[0, 0, 0],
- > im0s: array([[[0, 0, 0],

The "det" variable is expanded in a tooltip, showing its tensor structure:

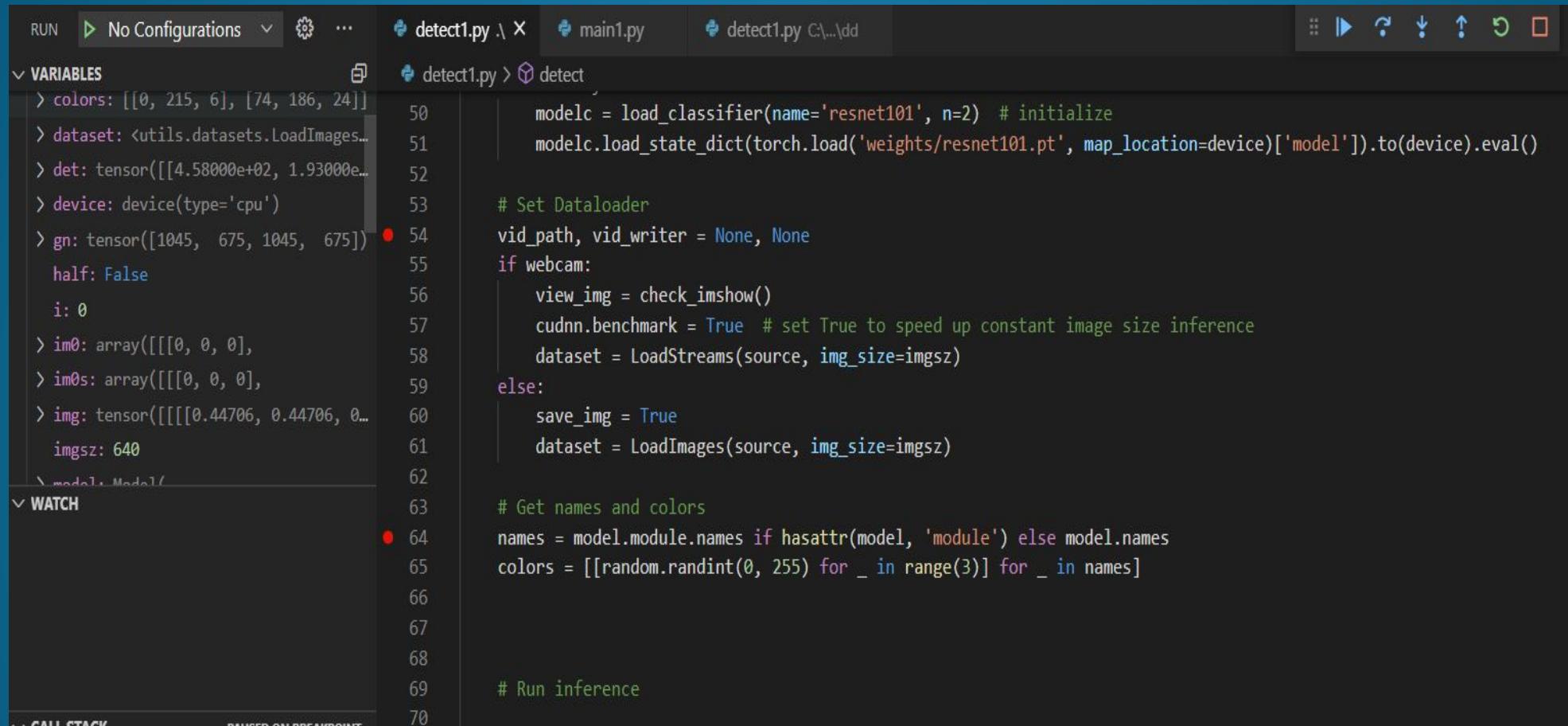
```
tensor([[4.58000e+02, 1.93000e+02, 5.48000e+02, 2.90000e+02, 7.70603e-01, 1.00000e+00],  
[3.95000e+02, 1.90000e+02, 4.86000e+02, 2.68000e+02, 6.61502e-01, 1.00000e+00],  
[8.91000e+02, 2.55000e+02, 1.02300e+03, 3.95000e+02, 4.87571e-01, 1.00000e+00],  
[4.13000e+02, 1.64000e+02, 5.19000e+02, 2.90000e+02, 3.68786e-01, 1.00000e+00],  
[7.90000e+02, 4.93000e+02, 9.81000e+02, 6.58000e+02, 2.98426e-01, 1.00000e+00],  
[5.24000e+02, 4.38000e+02, 6.24000e+02, 5.20000e+02, 2.97833e-01, 1.00000e+00],  
[3.00000e+01, 4.17000e+02, 1.22000e+02, 6.41000e+02, 2.43278e-01, 1.00000e+00],  
[4.10000e+01, 2.53000e+02, 1.91000e+02, 4.00000e+02, 2.24369e-01, 0.00000e+00],  
[4.35000e+02, 1.50000e+02, 5.78000e+02, 3.21000e+02, 2.00435e-01, 1.00000e+00],  
[1.70000e+01, 1.68000e+02, 1.06000e+02, 3.19000e+02, 1.40815e-01, 1.00000e+00],  
[4.00000e+01, 2.02000e+02, 1.45000e+02, 3.48000e+02, 1.12256e-01, 1.00000e+00],  
[9.31000e+02, 3.06000e+02, 1.02200e+03,...])
```

The code editor shows a portion of the "detect" function:

```
save_path = str(save_dir / p.name)  
txt_path = str(save_dir / 'labels' / p.name)  
s += '%gx%g ' % img.shape[2:] # print  
gn = torch.tensor(im0.shape)[[1, 0, 1]]  
if len(det):
```

The bottom right corner features a blue robot head icon and the text "VS code Object Detection".

VS code
Object Detection

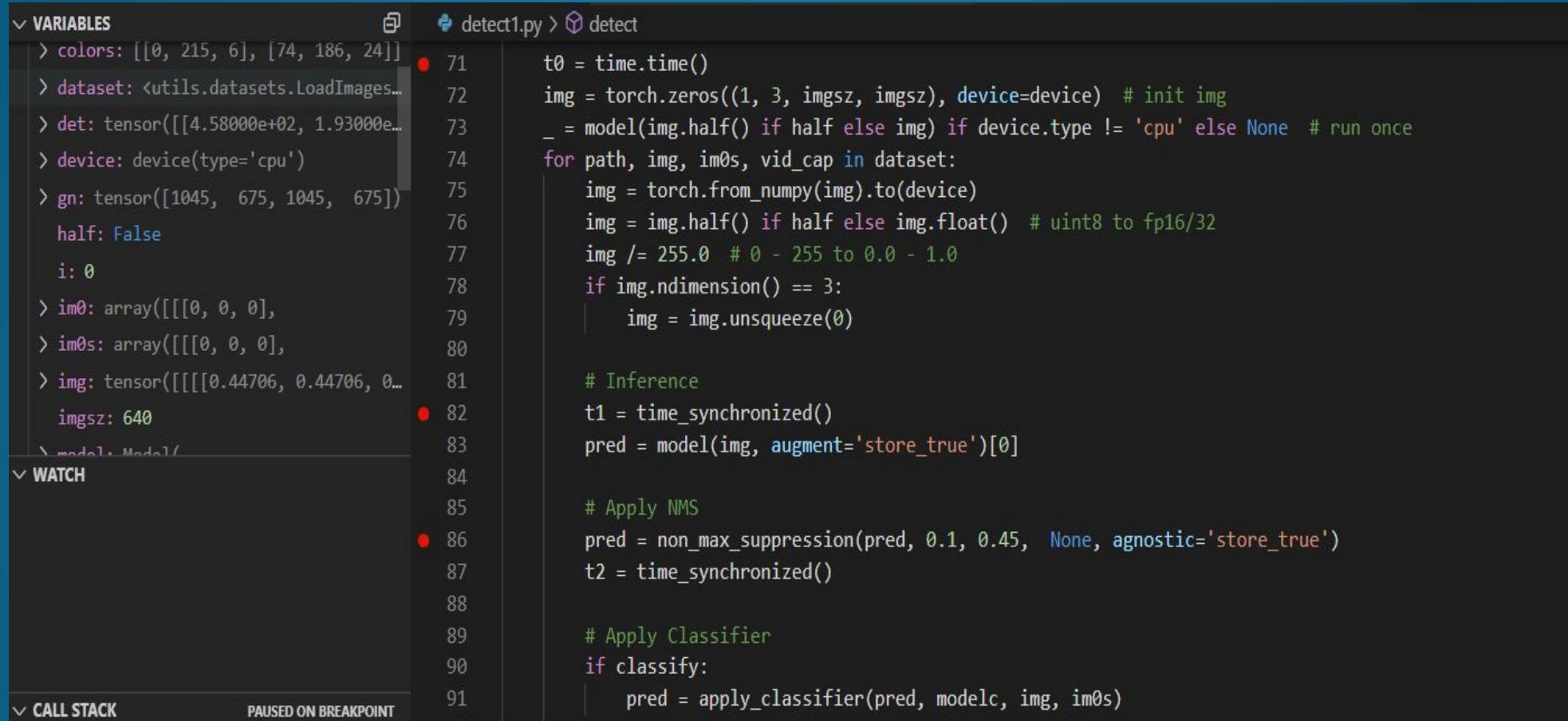


The screenshot shows the Visual Studio Code (VS Code) interface with the following details:

- Top Bar:** RUN, No Configurations, ... (with icons for file operations), detect1.py (active tab), main1.py, detect1.py (another tab), and several other icons.
- VARIABLES View:** Shows local variables. A red dot marks a breakpoint at line 54. Variables listed include colors, dataset, det, device, gn, half, i, im0, im0s, img, and imgszt.
- detect1.py Content:** The code uses PyTorch for object detection. It initializes a ResNet101 model, sets up a dataloader, and handles both webcam and image inputs. It then gets names and colors from the model and runs inference.
- Breakpoints:** Two breakpoints are set: one at line 54 and another at line 64.
- Bottom Status Bar:** PAUSED ON BREAKPOINT.



VS code
Object Detection



The screenshot shows the VS Code interface with a Python script named `detect1.py` open. The code is for object detection using PyTorch and OpenCV. It includes imports for `cv2`, `torch`, and `torchvision`. The script initializes a model, processes images from a dataset, performs inference, applies NMS, and applies a classifier. A breakpoint is set at line 82. The left sidebar shows the `VARIABLES` and `WATCH` panes, and the bottom shows the `CALL STACK` and `PAUSED ON BREAKPOINT` status.

```
    > colors: [[0, 215, 6], [74, 186, 24]] 71     t0 = time.time()
    > dataset: <utils.datasets.LoadImages... 72     img = torch.zeros((1, 3, imgsz, imgsz), device=device) # init img
    > det: tensor([[4.58000e+02, 1.93000e... 73     _ = model(img.half() if half else img) if device.type != 'cpu' else None # run once
    > device: device(type='cpu') 74     for path, img, im0s, vid_cap in dataset:
    > gn: tensor([1045, 675, 1045, 675]) 75         img = torch.from_numpy(img).to(device)
    half: False 76         img = img.half() if half else img.float() # uint8 to fp16/32
    i: 0 77         img /= 255.0 # 0 - 255 to 0.0 - 1.0
    > im0: array([[[0, 0, 0], 78         if img.ndim == 3:
    > im0s: array([[[0, 0, 0], 79             img = img.unsqueeze(0)
    > img: tensor([[[[0.44706, 0.44706, 0... 80
    imgsz: 640 81         # Inference
    > model: Model 82         t1 = time_synchronized()
    > modelc: Model 83         pred = model(img, augment='store_true')[0]
    > modelc: Model 84
    > modelc: Model 85         # Apply NMS
    > modelc: Model 86         pred = non_max_suppression(pred, 0.1, 0.45, None, agnostic='store_true')
    > modelc: Model 87         t2 = time_synchronized()
    > modelc: Model 88
    > modelc: Model 89         # Apply Classifier
    > modelc: Model 90         if classify:
    > modelc: Model 91             pred = apply_classifier(pred, modelc, img, im0s)
```



VS code
Object Detection

```
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
97     # Print results
98     for c in det[:, -1].unique():
99         n = (det[:, -1] == c).sum() # detections per class
100        #s += f"{n} {names[int(c)]}{('s' * (n > 1))}, " # add to string
101
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

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER**: Shows a folder structure under **DETECT** containing files like `Empty.py`, `detect.py`, `detect1.py`, `Empty.py`, `main.py`, `parkingJPG`, `path_ch.txt`, and `requirements.txt`.
- main.py - detect - Visual Studio Code**: The code editor window displays the following Python script:

```
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)
```
- TERMINAL**: The terminal window shows the output of running the script:

```
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

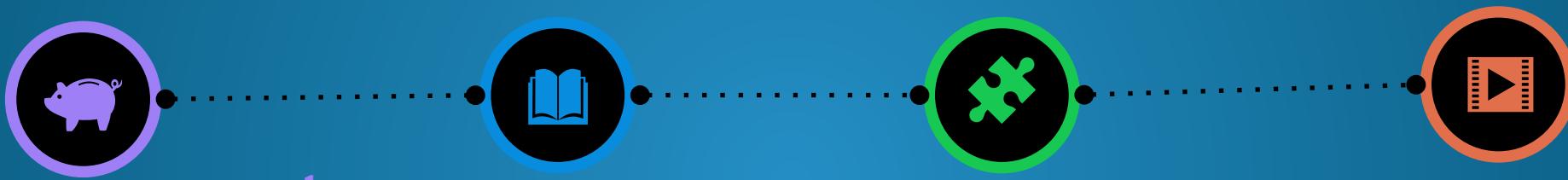
(base) D:\detect>
```
- OUTPUT**: Shows "Analyzing in background, 3527 items left..."
- PROBLEMS**: Shows 0 errors and 0 warnings.
- DEBUG CONSOLE**: Shows 0 errors and 0 warnings.
- OUTLINE**: Shows the outline of the current file.
- STATUS BAR**: Shows "Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python" and icons for file operations.

A red box highlights the image processing results in the terminal window, which shows a parking garage scene with several cars detected and labeled as "Occupied" with confidence scores (e.g., 0.811, 0.772, 0.75).



main.py
Object Detection

Object Tracking



Tracking은 움직이는
장면에서 사진상 물체의
path 근사치를 구하는 것

1

영상 속 움직이는 물체의
path가 얼마나 그 전
프레임과 유사한가

3

2

동일 객체라고 인식하면,
그 물체를 계속 Tracking
하는 것

4

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

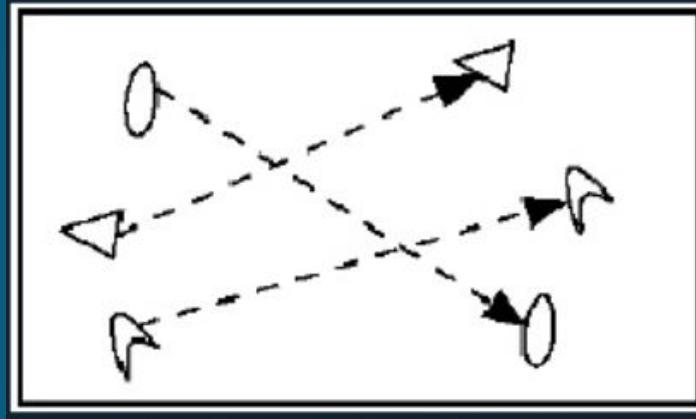
Path: 통로

Object Tracking: 객체 추적

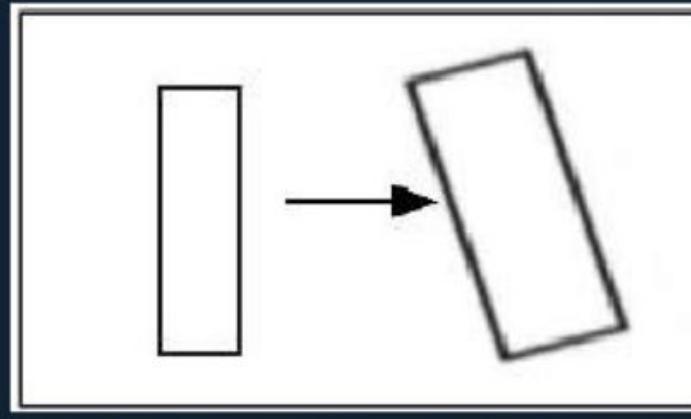
Kernel : 핵심

Silhouette: 실루엣

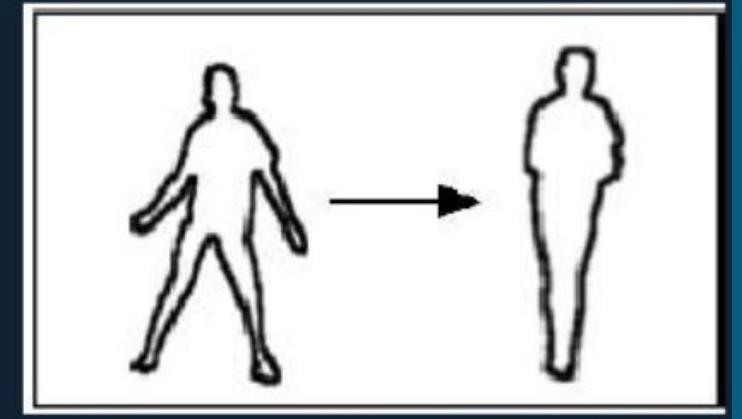
Point, Kernel, Silhouette Tracking



Point Tracking



Kernel Tracking



Silhouette Tracking

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)

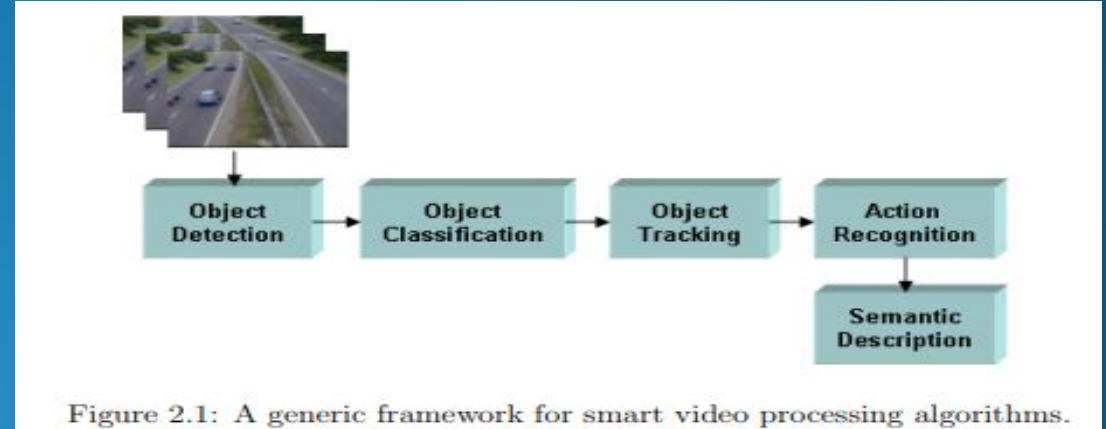
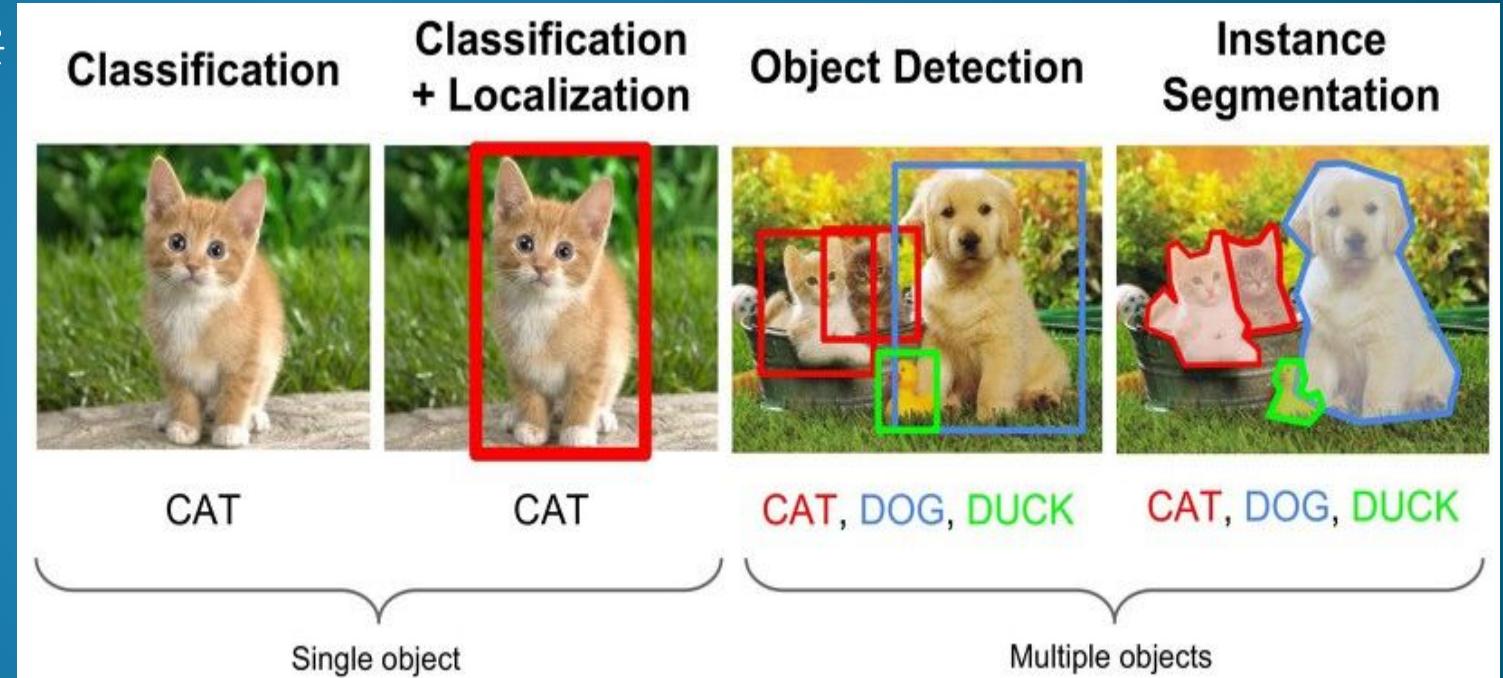


Figure 2.1: A generic framework for smart video processing algorithms.



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: 위치정보 출력

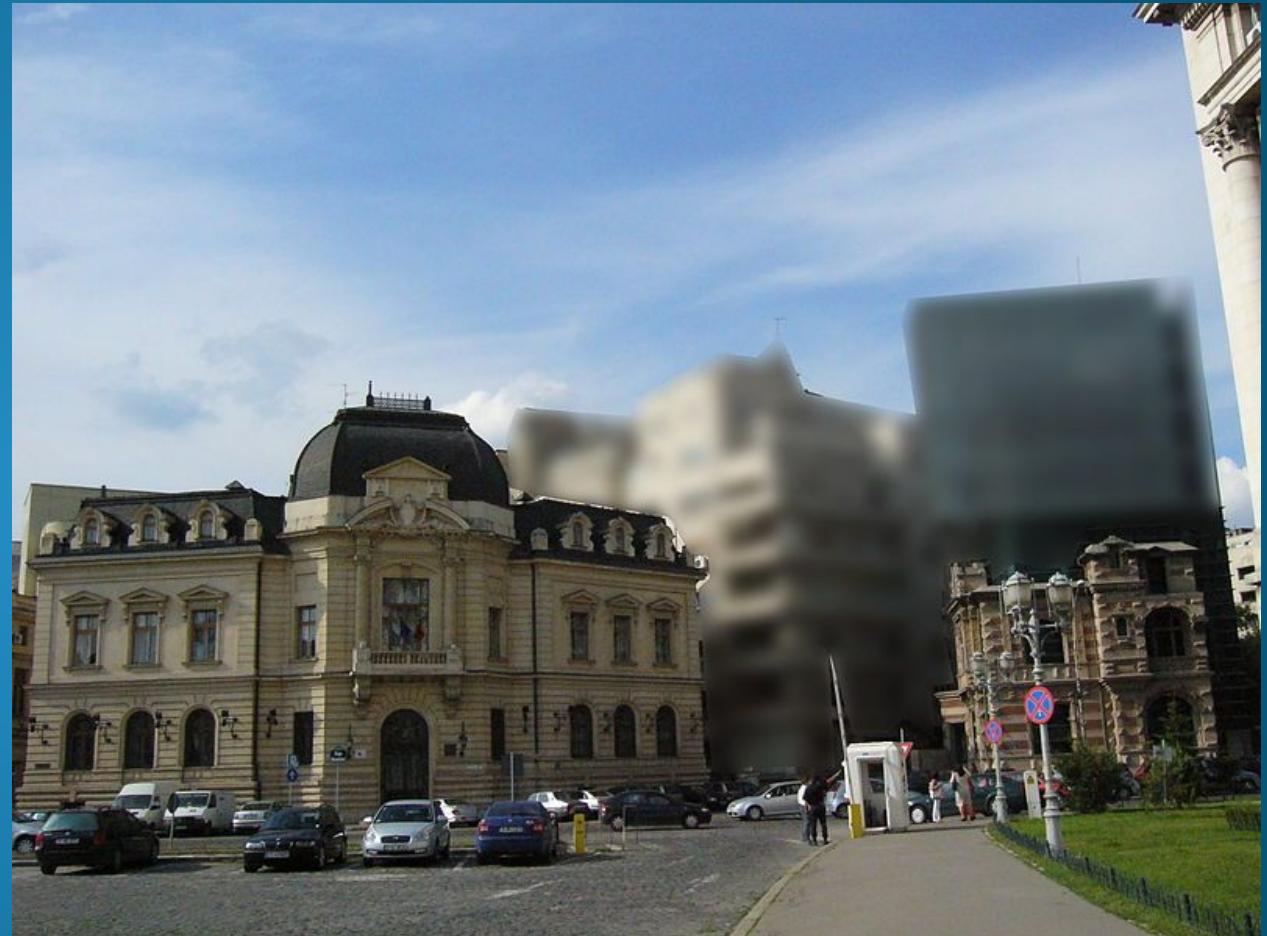
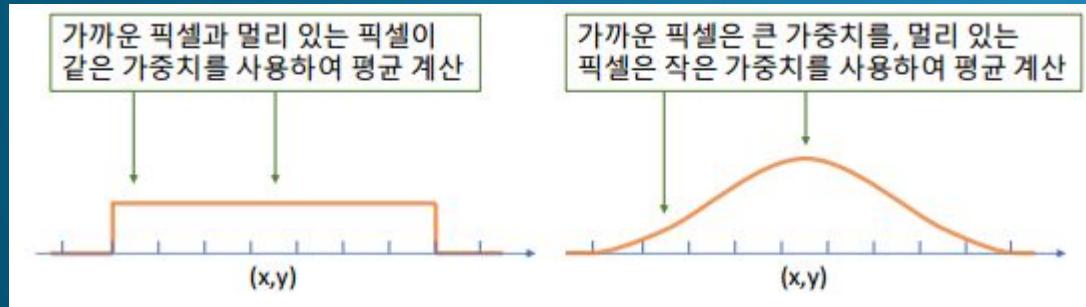
Instance Segmentation: 분할영역

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

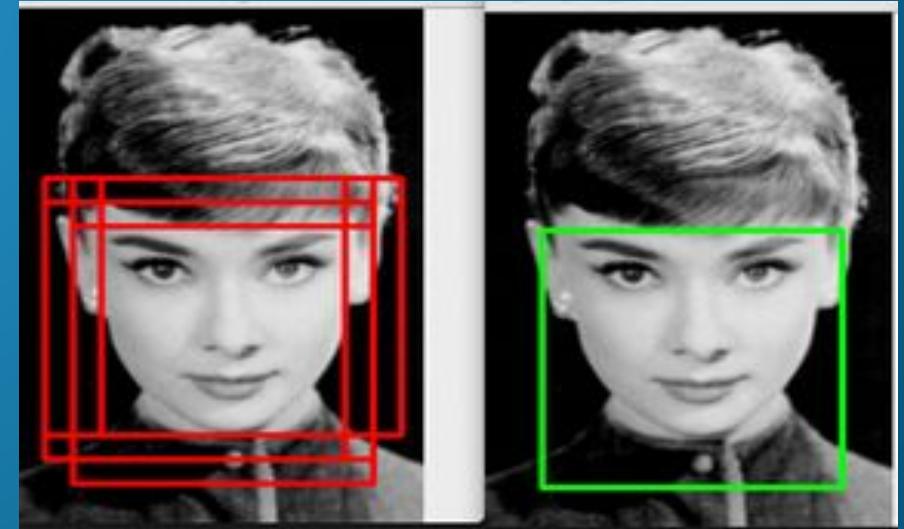
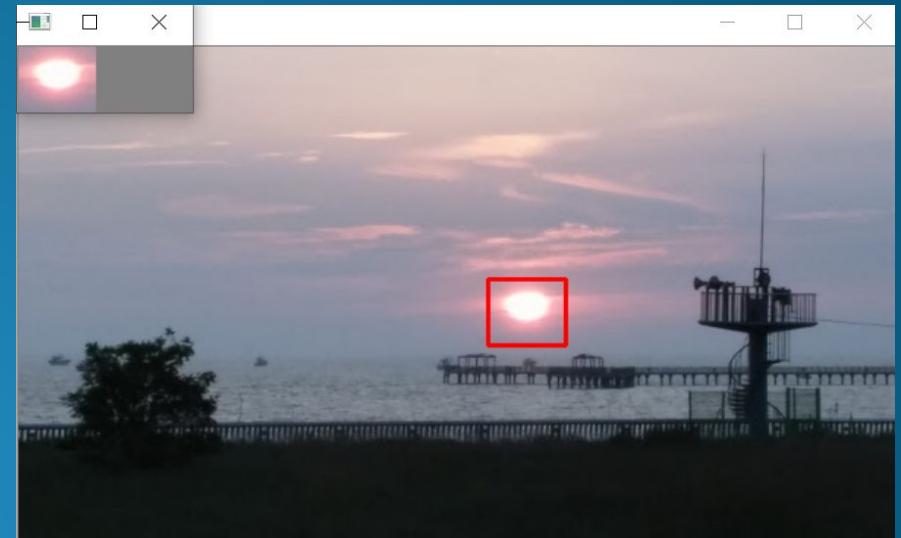


참고자료 / 위키미디어 (2008), OpenCV 가우시안 필터 (2020)

https://commons.wikimedia.org/wiki/File:Sediul_Unionii_Arхitectilor_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/>



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(history=1000, detectShadows=True)

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

Gaussian 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 영상

File Edit Selection View Go Run Terminal Help mainGaussianBlurFinal_in.py - YOLOv5-Object_Tracking_to_csv_files - Visual Studio Code

```
mainGaussianBlurFinal_in.py X mainGaussianBlurFinal_out.py
```

```
mainGaussianBlurFinal_in.py > ...
```

```
9     # each time a new object id detected, the count will increase by one
10    self.id_count = 0
11
12
13    def update(self, objects_rect):
14        # Objects boxes and ids
15        objects_bbs_ids = []
16
17        # Get center point of new object
18        for rect in objects_rect:
19            x, y, w, h = rect
20            cx = (x + x + w) // 2
21            cy = (y + y + h) // 2
22
23            # Find out if that object was detected already
24            same_object_detected = False
25            for id, pt in self.center_points.items():
26                dist = math.hypot(cx - pt[0], cy - pt[1])
27
28                if dist < 25:
29                    self.center_points[id] = (cx, cy)
30                    print(self.center_points)
31                    objects_bbs_ids.append([rect, id])
32                    same_object_detected = True
33                    break
34
35            # New object is detected
36            if not same_object_detected:
37                self.center_point[objects_bbs_ids[-1][1]] = (cx, cy)
38                objects_bbs_ids[-1].append(objects_bbs_ids[-1][1])
39                self.id_count += 1
40
41        # Clean the dictionary
42        new_center_points = {}
43
44        PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

```
PS C:\Users\st\Desktop\YOLOv5-Object_Tracking_to_csv_files\mainGaussianBlurFinal_in.py
{1: (34, 37)}
{1: (34, 36)}
{2: (38, 37)}
{2: (29, 37)}
PS C:\Users\st\Desktop\YOLOv5-Object_Tracking_to_csv_files\mainGaussianBlurFinal_in.py
{1: (34, 37)}
{1: (34, 36)}
{2: (38, 37)}
{2: (29, 37)}
```

2021-02-02 08:00:13

File Edit Selection View Go Run Terminal Help mainGaussianBlurFinal_out.py - YOLOv5-Object_Tracking_to_csv_files - Visual Studio Code

```
mainGaussianBlurFinal_in.py X mainGaussianBlurFinal_out.py
```

```
mainGaussianBlurFinal_in.py > ...
```

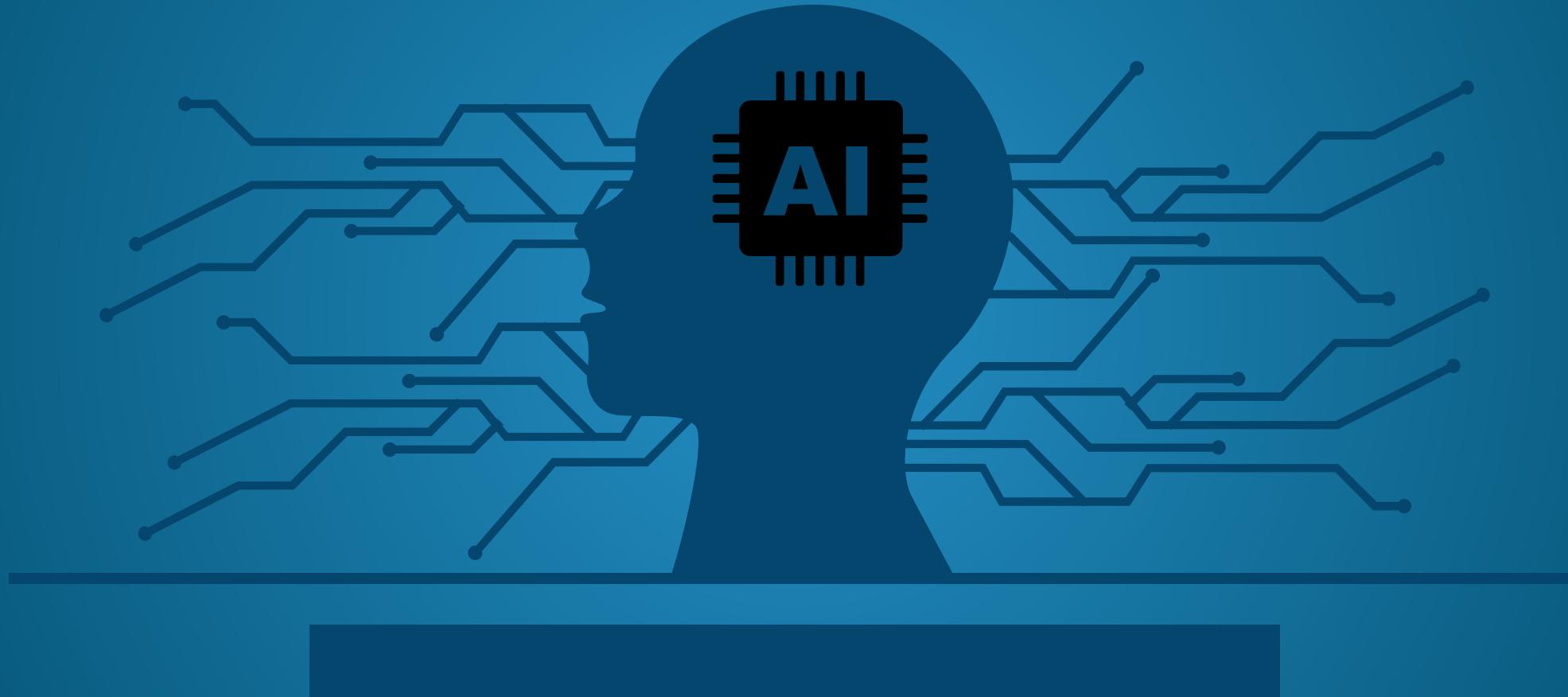
```
80     mask = cv2.adaptiveThreshold(mask, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
81     # https://opencv-python.readthedocs.io/en/latest/doc/09.imageThresholding/imageThresholding.html
82
83
84     contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
85     detections = []
86     for cnt in contours:
87         # Calculate area and remove small elements
88         area = cv2.contourArea(cnt)
89         if area > 3350:
90             cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
91             x, y, w, h = cv2.boundingRect(cnt)
92
93             detections.append([x, y, w, h])
94
95     # 2. Object Tracking
96     boxes_ids = tracker.update(detections)
97
98     for box_id in boxes_ids:
99         x, y, w, h, id = box_id
100        cv2.putText(roi, str(id), (x - 15, y - 15), cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0))
101        cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 2)
102
103        cv2.imshow("roi", roi)
104        cv2.imshow("Frame", frame)
105        cv2.imshow("Mask", mask)
106
107        key = cv2.waitKey(10)
108        if key == 27:
109            break
110
111    cap.release()
112    cv2.destroyAllWindows()
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

```
Traceback (most recent call last):
  File "C:/Users/st/Desktop/YOLOv5-Object_Tracking_to_csv_files/mainGaussianBlurFinal_out.py", line 107, in <module>
    key = cv2.waitKey(10)
AttributeError: 'NoneType' object has no attribute 'waitKey'
```

2021-02-02 08:02:14

결과





대시보드

Object Tracking

dashboard.py ●

dashboard.py > ...

```
1 import sys
2 from mainGaussianBlurentry import *
3 from mainGaussianBlurexit import *
4 import Empty
5
6 if __name__ == '__main__':
7     argument = sys.argv
8     del argument[0]
9
10 a, b = Empty.detect()
11
12
13 print('Empty : {:03d}'.format(a))
14 print('Occupied : {:03d}'.format(b))
15
16 print('# of Entry : {:03d}'.format(entry))
17 print('# of Exit : {:03d}'.format(exit))
18
19
```

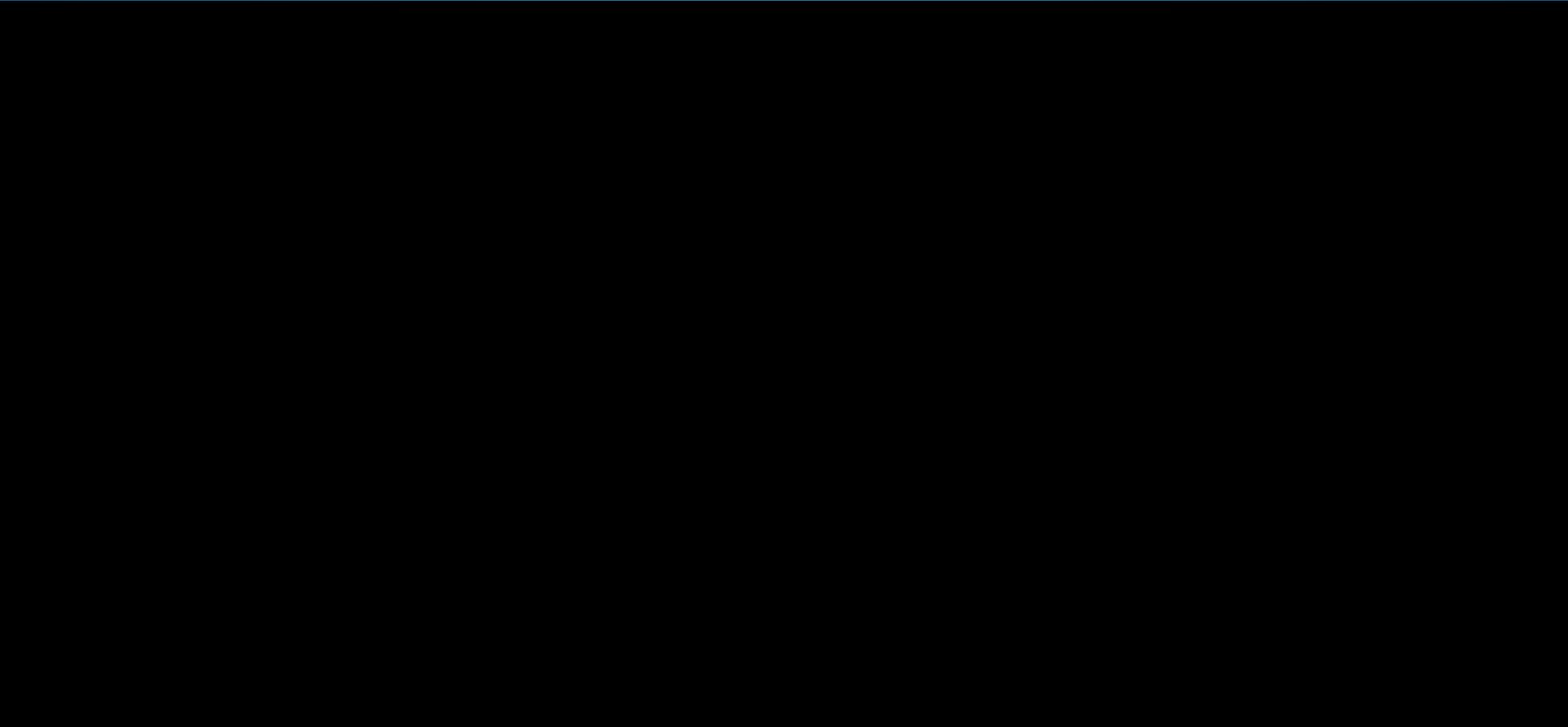
Empty: 빈 공간

Occupied: 차있는 공간

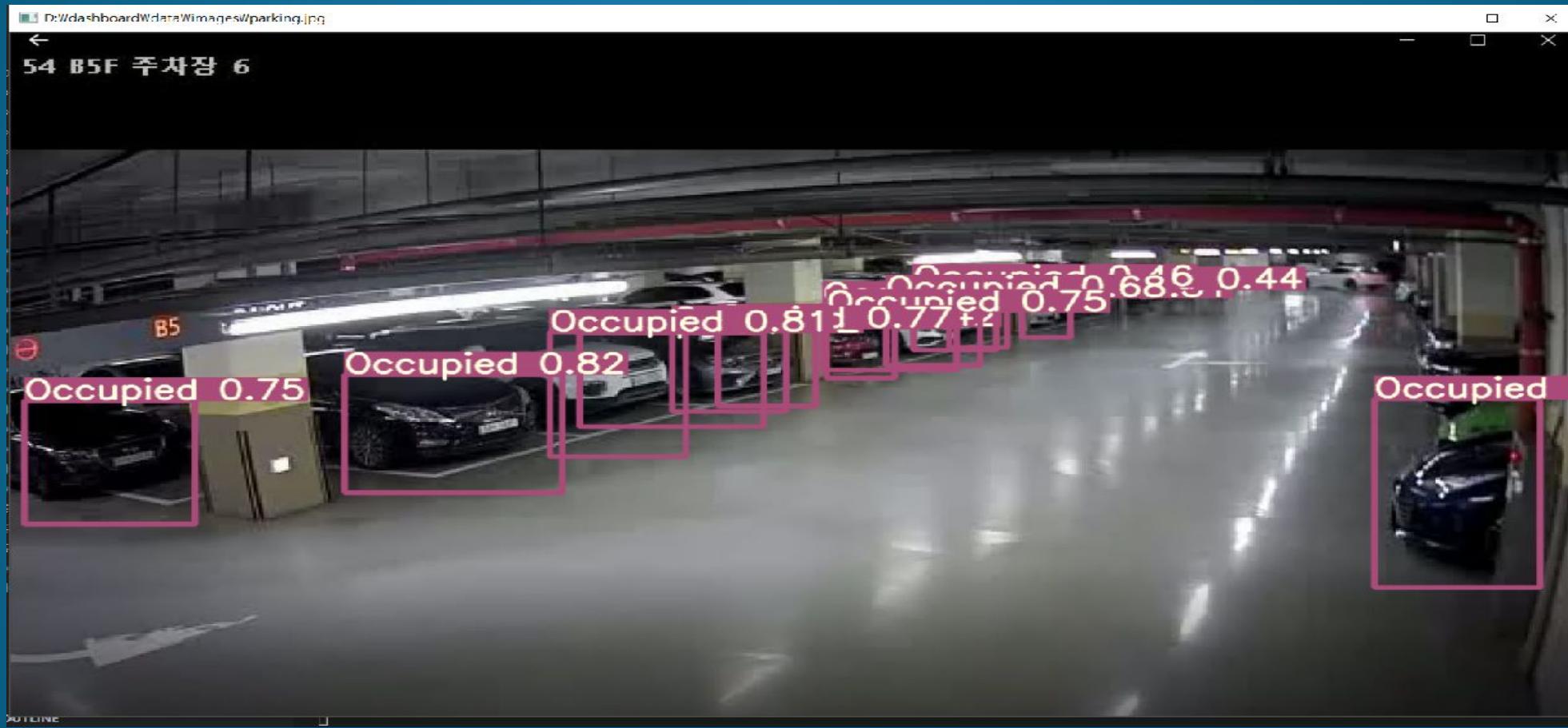
Entry: 들어온 자동차 대수

Exit: 나간 자동차 대수

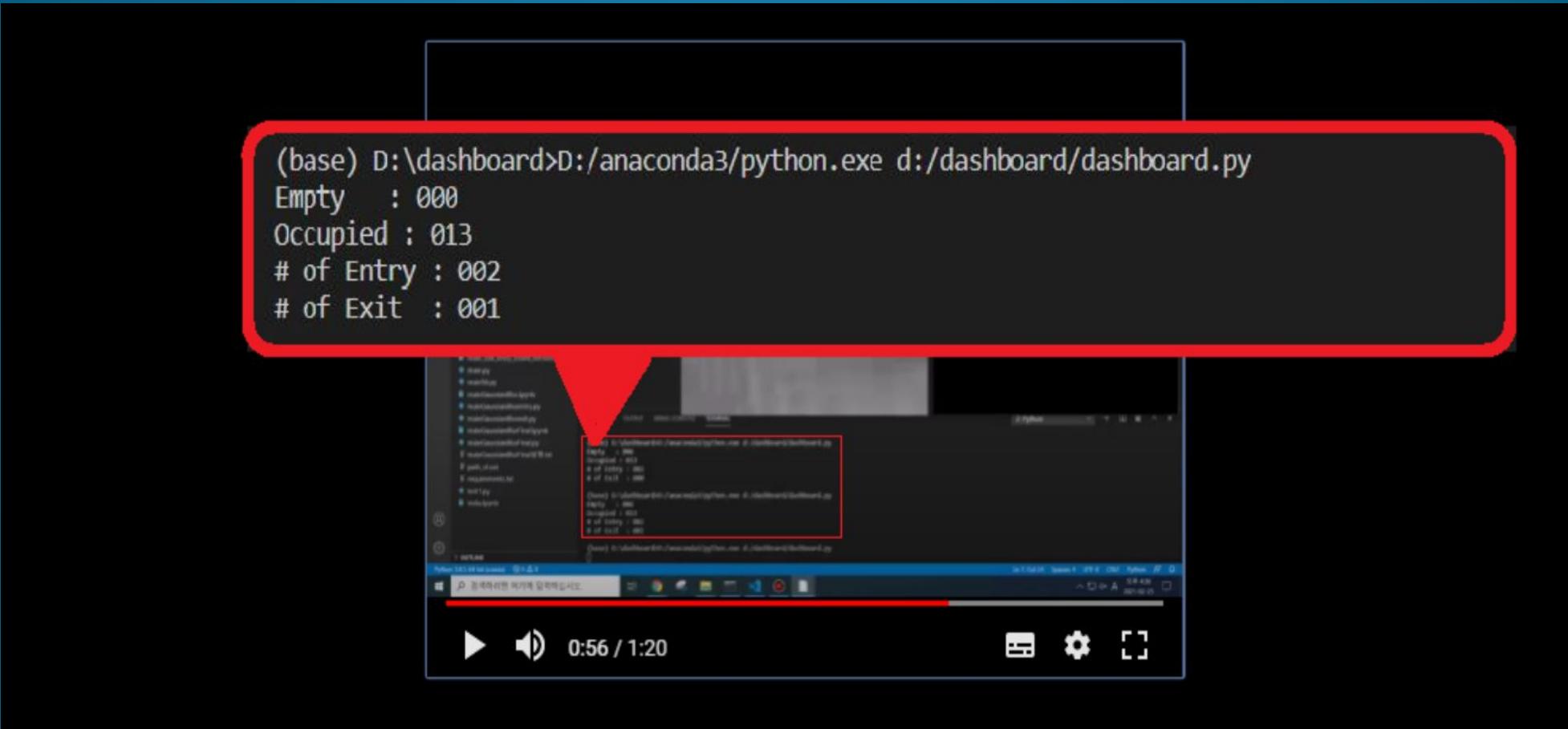
최종결과



최종결과



최종결과



The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a red border and displays the following output:

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

The terminal window is positioned over a video player interface, which is displaying a video titled "Python 3.6.4 (64-bit) Python 3.6.4 (64-bit) Python 3.6.4 (64-bit)". The video player controls at the bottom include a play button, volume control, and a progress bar showing 0:56 / 1:20.

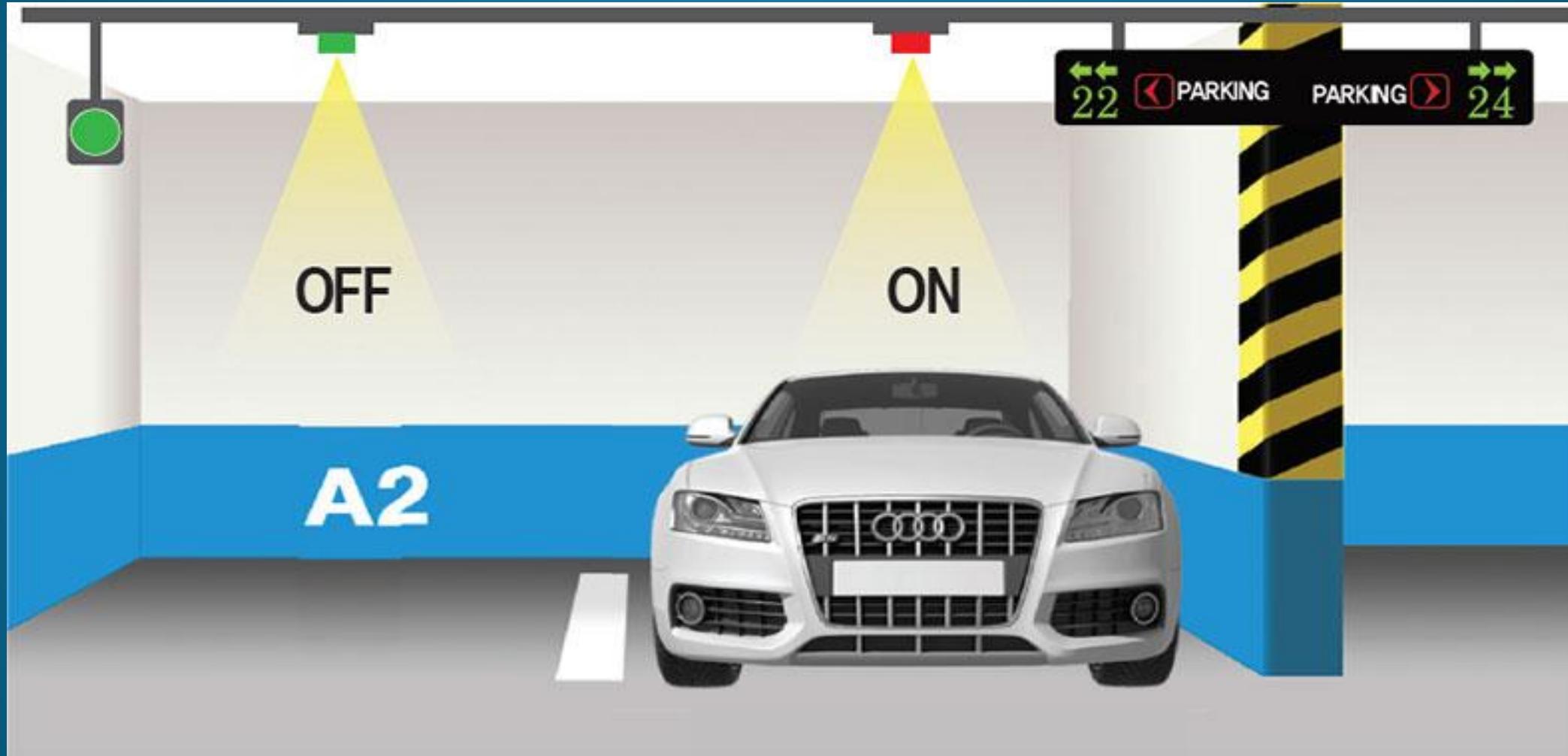


대시보드

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



차이점

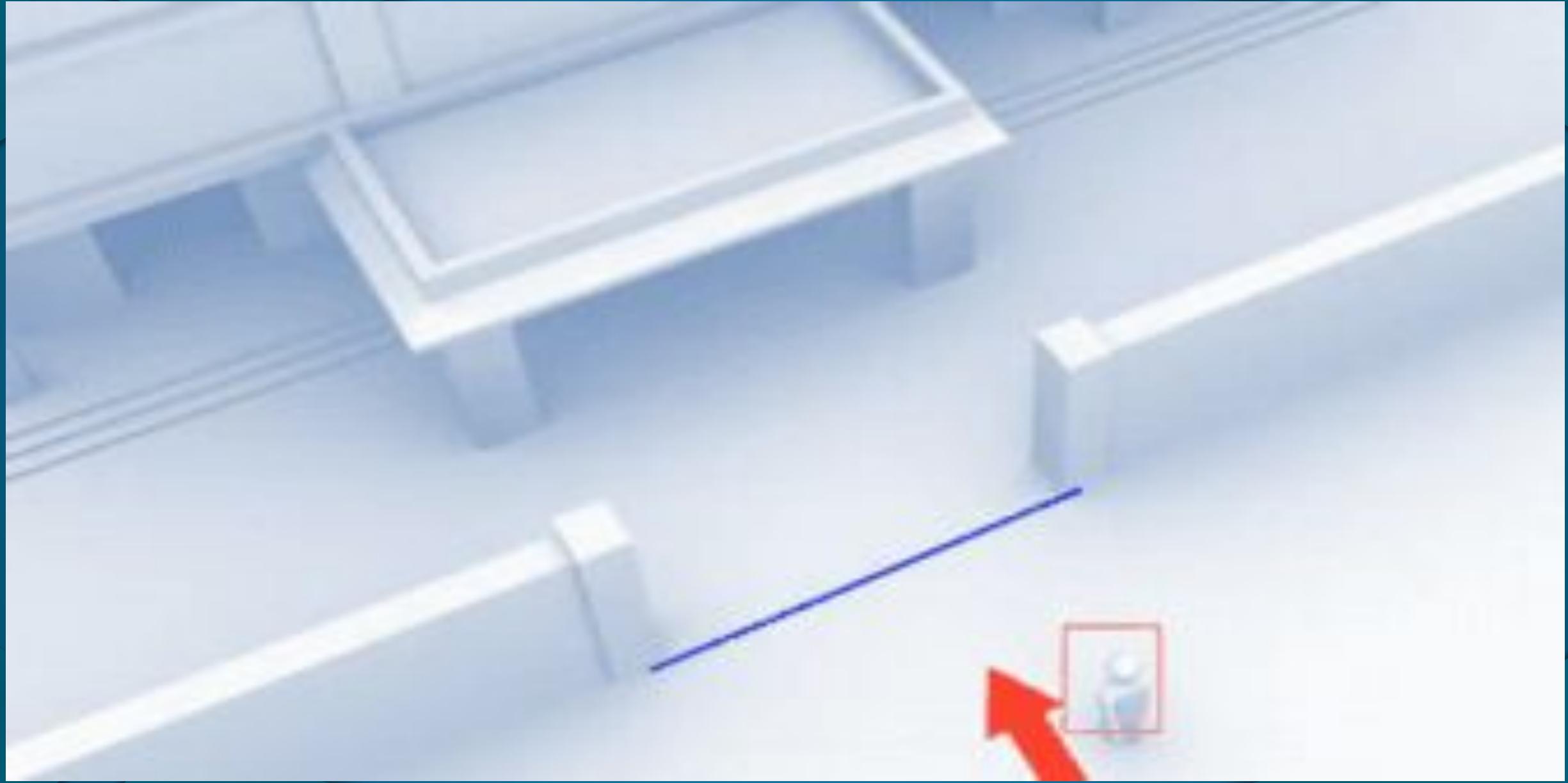


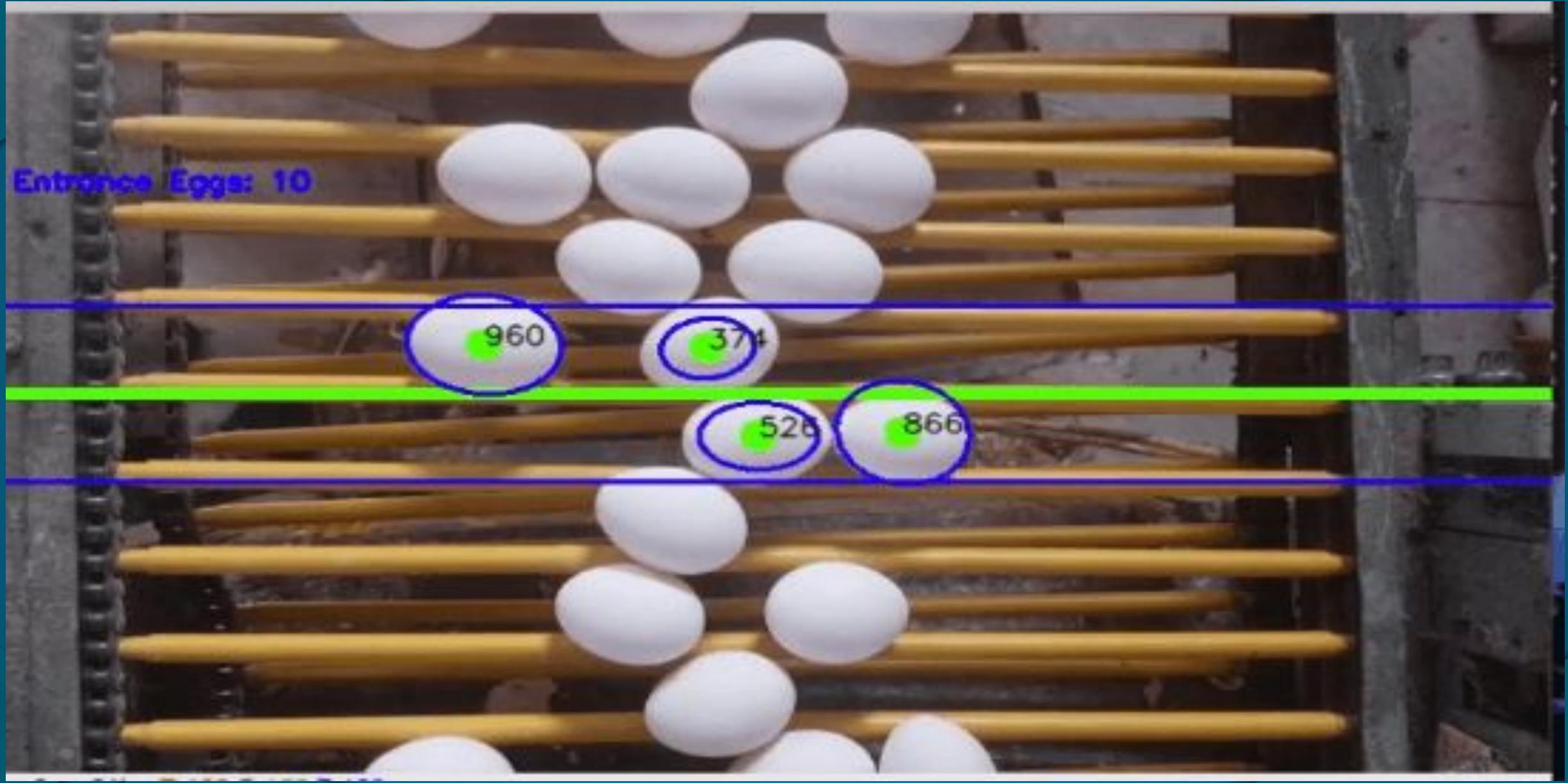
차이점



<https://github.com/0201shj>

활용방안





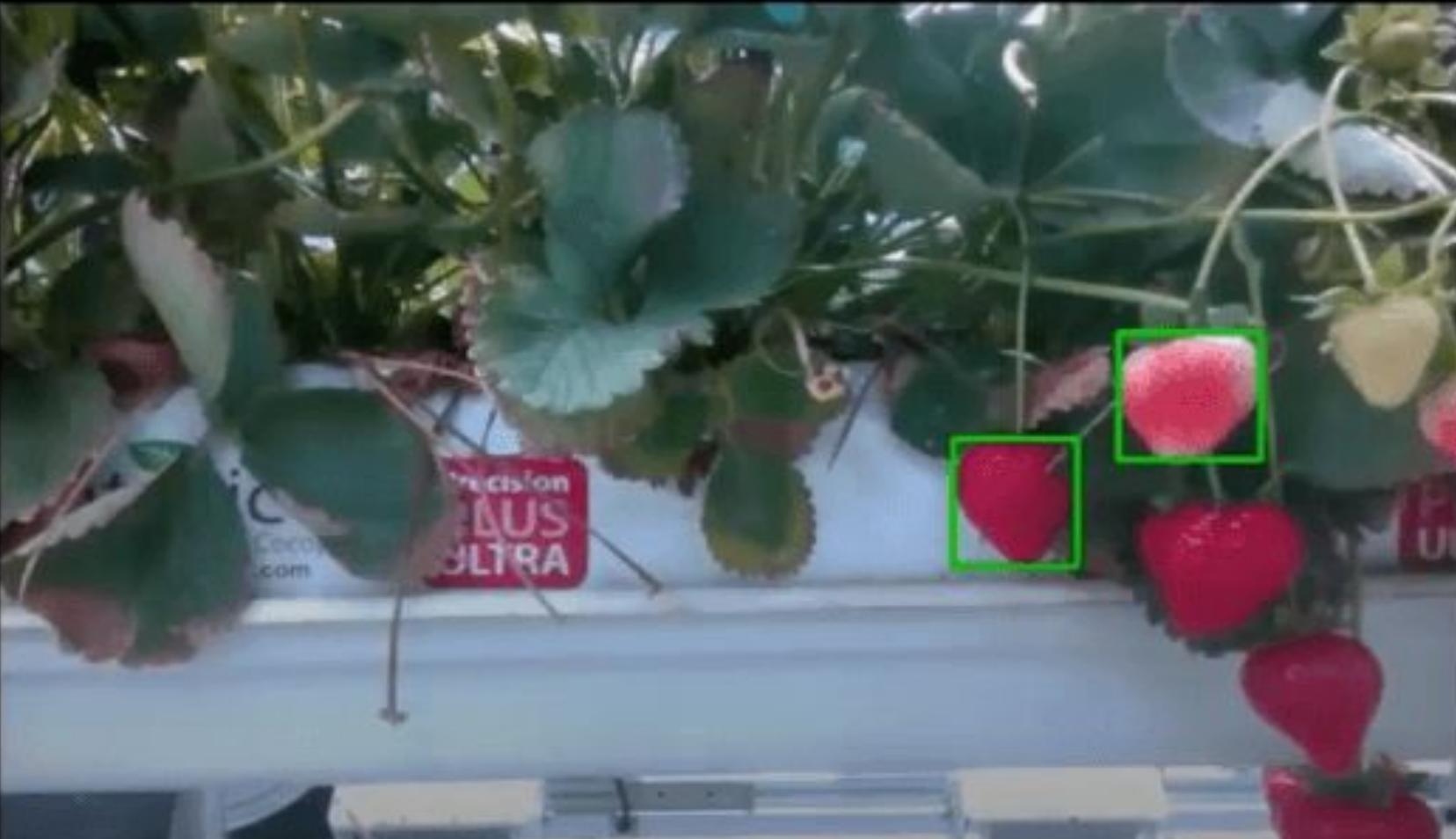




person 0.91

70%

sports ball 0.89



먼저 로봇은 3D카메라와 딥러닝 알고리즘으로
딸기의 익은 정도를 파악하게 되는데요,

향후 과제

- 학습 데이터 추가 필요
- 카메라/디스플레이 보정
- 오류 수정

질의응답



THANK YOU

컴퓨터 비전을 이용한 주차장 관리 시스템