

비식별화 (DE-IDENTIFICATION)

(주)베스텔라랩 박영준 오정석

목차

1. 실행방법
2. 주요 코드 설명
3. 문제점
4. 해결방안
5. 이후 추가 할 내용

1.

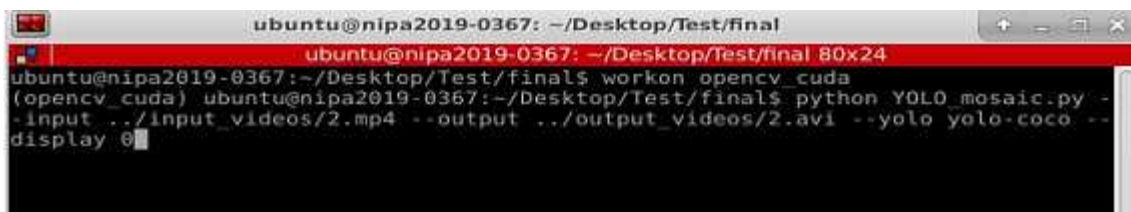
실행 방법

1. [input_videos], [output_videos], [원하는 이름의 폴더] 3개의 폴더를 만든다.
2. [원하는 이름의 폴더]안에 YOLO.mosaic.py, coco.names, yolov3.cfg, yolov3.weights 다운로드 받은 4개의 파일을 넣어준다.
3. [input_videos]에 원하는 동영상 파일을 넣는다.
4. [원하는 이름의 폴더]에서 usage를 cmd에 입력한다.
5. [output_videos]에 avi파일이 생성되는지 확인한다.

```
1 # python YOLO_mosaic.py --input ../input_videos/2.mp4 --output ../output_videos/2.avi --yolo yolo-coco --display 0
2
3 import cv2
4 import numpy as np
5 import argparse
```

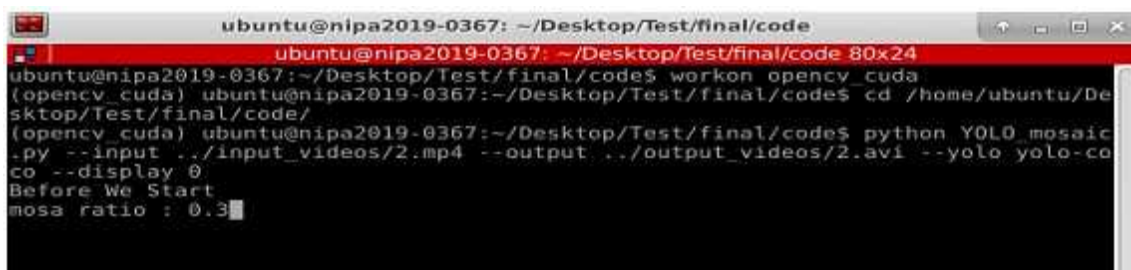
- 첫 번째 줄은 cmd창에 들어갈 명령어

-> ([원하는 이름의 폴더]안에 들어있는 비식별 처리 할 동영상)



```
ubuntu@nipa2019-0367: ~/Desktop/Test/final
ubuntu@nipa2019-0367: ~/Desktop/Test/final 80x24
ubuntu@nipa2019-0367:~/Desktop/Test/final$ workon opencv_cuda
(opencv_cuda) ubuntu@nipa2019-0367:~/Desktop/Test/final$ python YOLO_mosaic.py --input ../input_videos/2.mp4 --output ../output_videos/2.avi --yolo yolo-coco --display 0
```

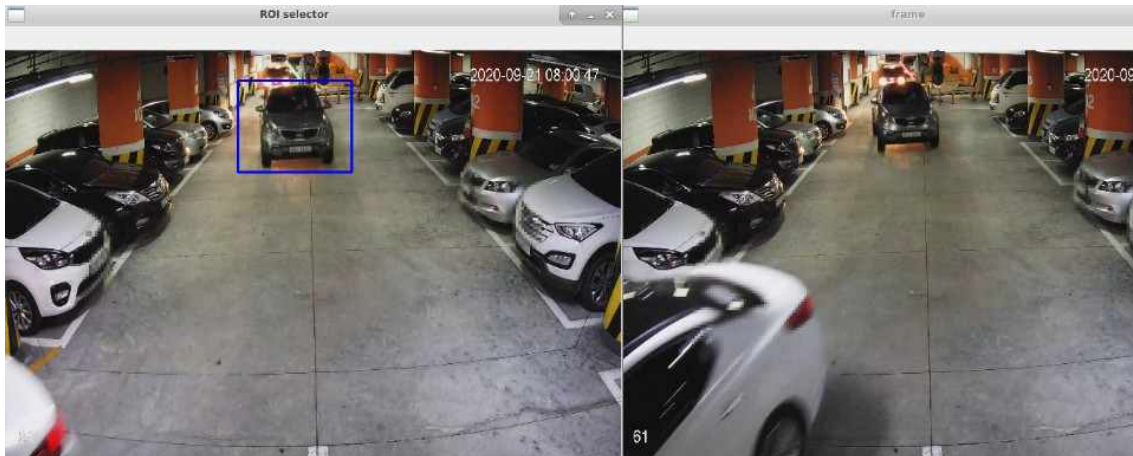
- opencv cuda를 활성화 시켜준 후 첫 번째 줄의 경로를 Import



```
ubuntu@nipa2019-0367: ~/Desktop/Test/final/code
ubuntu@nipa2019-0367: ~/Desktop/Test/final/code 80x24
ubuntu@nipa2019-0367:~/Desktop/Test/final/code$ workon opencv_cuda
(opencv_cuda) ubuntu@nipa2019-0367:~/Desktop/Test/final/code$ cd /home/ubuntu/Desktop/Test/final/code/
(opencv_cuda) ubuntu@nipa2019-0367:~/Desktop/Test/final/code$ python YOLO_mosaic.py --input ../input_videos/2.mp4 --output ../output_videos/2.avi --yolo yolo-coco --display 0
Before We Start
mosa ratio : 0.3
```

- 임의로 추가 해 줄 영역의 모자이크 강도를 입력함 위의 경우 0.3만큼 Resize

6. 임의로 ROI를 선택하려면 Q버튼을 입력 후 마우스로 드래그 후 스페이스바를 입력한다.



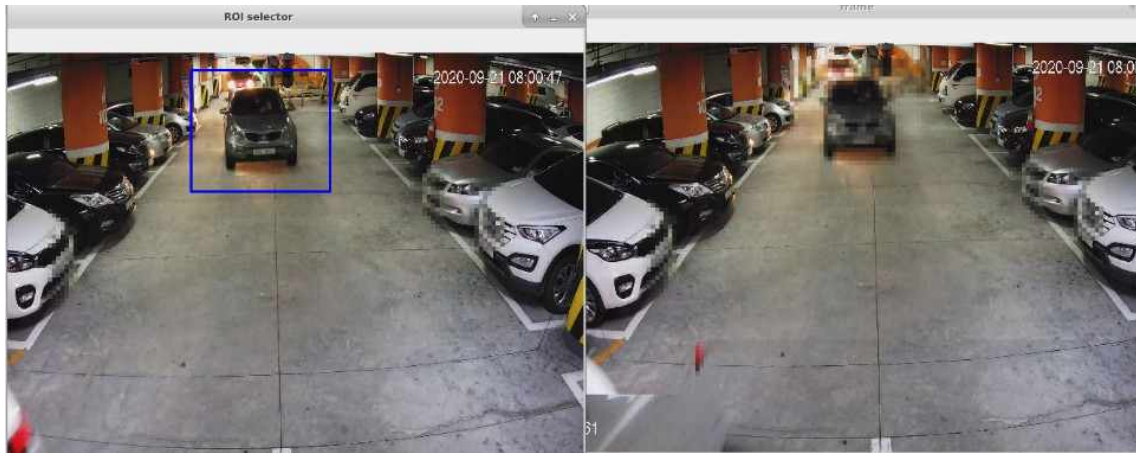
- Center 영역에서의 모자이크가 풀려 임의로 ROI selector에서 Bounding Box 그렸을 때의 사진

결과

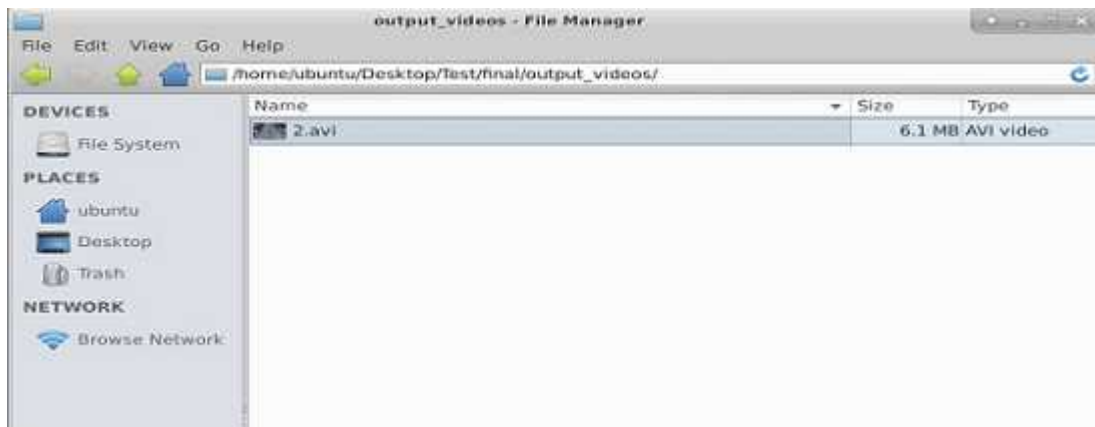
```
Before We Start
mosa ratio : 0.2
[INFO] loading YOLO from disk...
[INFO] accessing video stream...
[INFO] Saving Video...
qt.qpa.xcb: failed to initialize XRandr
qt.qpa.xcb: XKeyboard extension not present on the X server
OpenCV(4.4.0) /tmp/pip-req-build-sw 3pm 8/opencv/modules/imgproc/src/resize.cpp:
3929: error: (-215:Assertion failed) !ssize.empty() in function 'resize'

Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
(opencv_cuda) ubuntu@nipa2019-0367:~/Desktop/Test/final/code$
```

- 위의 명령창을 실행 시키면 input 영상이 불러오게 되고 임의의 Bounding Box를 그릴 수 있음



- 오른쪽 그림에서 확인할 수 있듯이 모자이크 효과가 추가됨



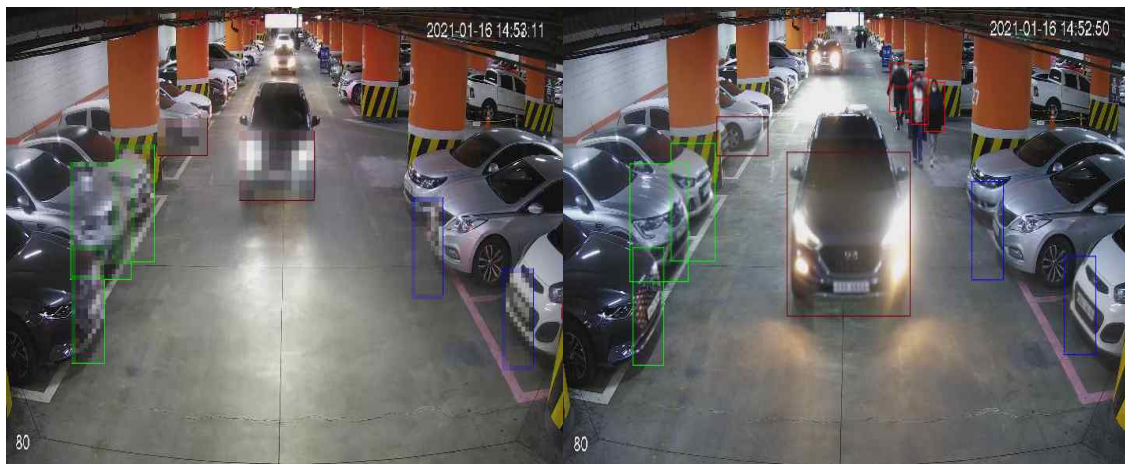
- output_videos 파일을 확인해보면 위와 같이 모자이크 효과가 추가된 영상이 저장됨

2.

주요 코드 설명

```
23  # 모자이크 ratio 설정  
24  # 최소값 0.2  
25  print("Before We Start")  
26  ratio = float(input("mosa ratio : "))
```

- Resize의 비율에 따라 모자이크 효과 세기를 정해주는 코드이며 Ratio가 작을수록 모자이크 효과가 강하게 나타난다. 통상적으로 Ratio의 값은 0.2~0.8의 값을 사용



ratio:0.2

ratio:0.8

```

47     # 프레임 조절
48     # 영상을 실시간으로 볼 때, 너무 느리면 사용
49     def process(Time):
50         count = 1
51         while True:
52             ret, frame = capture.read()
53             if count % (Time * frame_per_second) == 0:
54                 break
55             count += 1

```

- 프레임을 읽어 들이는 수를 조절하여 영상의 재생 속도를 조절

```

155     # Detecting된 Object의 Label이 Person일 때
156     if 'person' in label:
157         color = (0, 0, 255)
158         # 멀리 있는 사람과 가까이 있는 사람에 대해서 Rectangle
159         # 크기를 다르게 하기 위해 영역을 나눔
160         if 50 < y < 80:
161             # 사람의 Rectangle에서 사람의 머리 부분만 저장
162             # far_persons = cv2.rectangle(frame, (x, y), (x + w, (y + 25)), color, 1)
163             far_persons_list = np.append(far_persons_list, np.array([[x, y, x+w, y+25]]), axis=0)
164             # print('far_persons_list : %s' %far_persons_list)
165
166         elif y > 60:
167             # person의 Rectangle에서 사람의 머리 부분만 저장
168             # near_persons = cv2.rectangle(frame, (x, y), (x + w, (y + 60)), color, 1)
169             near_persons_list = np.append(near_persons_list, np.array([[x, y, x+w, y+60]]), axis=0)
170             # print('near_persons_list : %s' %near_persons_list)

```

- CCTV마다 angle이 다르기 때문에 그에 맞춰 사람이 멀리 있을 때($50 < y < 80$)와 가까이 있을 때($y > 60$)에 rect box의 크기를 조절해주어야 한다. 사람이 멀리 있을 때는 $y+25$ 로 박스의 크기를 작게 하고, 가까이 있을 때는 $y+60$ 로 크게 해줌으로서 얼굴 영역 박스 크기를 조절 해주어야 한다.


```

172 #Detecting된 Object의 Label이 Car일 때
173 elif 'car' in label:
174     if y > 75:
175         if 150 < x < 475:
176             # 움직이는 자만 Detecting후 리스트에 저장
177             color = (0, 0, 120)
178             # move = cv2.rectangle(frame, (x, y+30), (x+w, y+h), color, 1)
179             move_list = np.append(move_list, np.array([[x, y+30, x+w, y+h]]), axis=0)
180             # print('move_list : %s' %move_list)
181
182         elif x > 470:
183             # 오른쪽에 있는 자만 Detecting후 리스트에 저장
184             color = (255, 0, 0)
185             # rights = cv2.rectangle(frame, (x, y+35), (x+40, y+120), color, 1)
186             rights_list = np.append(rights_list, np.array([[x, y+35, x+40, y+120]]), axis=0)
187             # print('rights_list : %s' %rights_list)
188
189         elif x < 250:
190             # 왼쪽에 있는 자만 Detecting후 리스트에 저장
191             color = (0, 255, 0)
192             # lefts = cv2.rectangle(frame, (x+90, y+20), (x+w, y+140), color, 1)
193             lefts_list = np.append(lefts_list, np.array([[x+90, y+20, x+w, y+140]]), axis=0)
194             # print('lefts_list : %s' %lefts_list)

```

- CCTV마다 angle이 다르기 때문에 왼쪽 영역은 rect box를 오른쪽으로 해주어야 하고, 오른쪽 영역은 rect box를 왼쪽에 설정해주어야 한다. 위에 예시에서는 왼쪽 영역을 $x > 250$, rect box를 $[x+90, y+20, x+w, y+140]$, 오른쪽 영역을 $x > 470$, rect box를 $[x, y+35, x+40, y+120]$ 으로 설정해주었다. 이는 CCTV영상마다 grid를 참고하여 변경해주어야 번호판 영역만 비식별 처리할 수 있다.

3.

문제점

① 객체를 인식하지 못하는 경우.



< 사진 1 >



< 사진 2 >



< 사진 3 >



< 사진 4 >

Case1. 카메라 각도에 따라 detection하지 못할 때. (사진1,2)

-> 주차장 CCTV각도에서의 데이터 학습 필요.

Case2. 장애물이 object를 가렸을 때. (사진 3,4)

-> 가려지는 영역이 많으므로 detection 불가. (차량 또는 사람 등)

-> ID를 부여해 ID가 없어졌을 시 유지하도록 하는 코드 추가 필요.

② 에러가 발생하는 경우

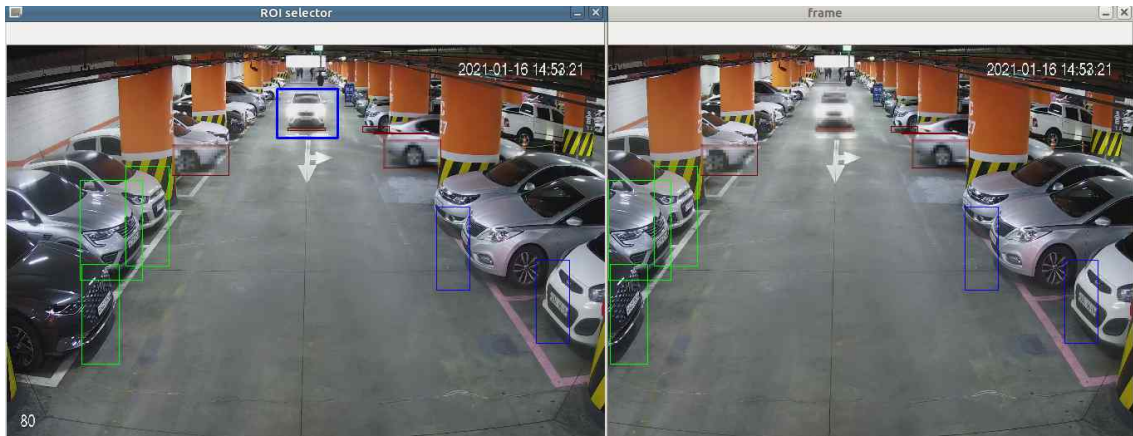


< 왼쪽부터 시계방향으로 사진 5, 사진 6, 사진 7 >

- rectangle이 너무 작게 잡힐 때 모자이크 부분에서 resize 오류 발생
- > 영상이 정지하고 ROI(Region Of Interest)를 설정하여 다시 영상 재생
- > ★★★주로 Move의 rect box범위 때문에 resize 수정이 필수★★★

4.

해결방안



- Frame창에는 모자이크가 풀리기 직전에 영상이 멈추고, 모자이크가 풀린 영상이 ROI selector에 뜬. 이 때 범위를 마우스로 선택하면 Frame영역에 모자이크가 처리되면서 다시 영상이 재생된다.

3.CCTV마다 Calibration 필요

```
# Detecting된 Object의 Label이 Person일 때
if 'person' in label:
    color = (0, 0, 255)
    # 멀리 있는 사람과 가까이 있는 사람에 대해서 Rectangle
    # 크기를 다르게 하기 위해 영역을 나눔
    if 50 < y < 80:
        # 사람의 Rectangle에서 사람의 머리 부분만 저장
        # far_persons = cv2.rectangle(frame, (x, y), (x + w, (y + 25)), color, 1)
        far_persons_list = np.append(far_persons_list, np.array([[x, y, x+w, y+25]]), axis=0)
        # print('far_persons_list : %s' %far_persons_list)

    elif y > 60:
        # person의 Rectangle에서 사람의 머리 부분만 저장
        # near_persons = cv2.rectangle(frame, (x, y), (x + w, (y + 60)), color, 1)
        near_persons_list = np.append(near_persons_list, np.array([[x, y, x+w, y+60]]), axis=0)
        # print('near_persons_list : %s' %near_persons_list)
```



```

#Detecting된 Object의 Label이 Car일 때
elif 'car' in label:
    if y > 75:
        if 150 < x < 475:
            # 움직이는 차만 Detecting후 리스트에 저장
            color = (0, 0, 120)
            # move = cv2.rectangle(frame, (x, y+30), (x+w, y+h), color, 1)
            move_list = np.append(move_list, np.array([[x, y+30, x+w, y+h]]), axis=0)
            # print('move_list : %s' %move_list)

        elif x > 470:
            # 오른쪽에 있는 차만 Detecting후 리스트에 저장
            color = (255, 0, 0)
            # rights = cv2.rectangle(frame, (x, y+35), (x+40, y+120), color, 1)
            rights_list = np.append(rights_list, np.array([[x, y+35, x+40, y+120]]), axis=0)
            # print('rights_list : %s' %rights_list)

        elif x < 250:
            # 왼쪽에 있는 차만 Detecting후 리스트에 저장
            color = (0, 255, 0)
            # lefts = cv2.rectangle(frame, (x+90, y+20), (x+w, y+140), color, 1)
            lefts_list = np.append(lefts_list, np.array([[x+90, y+20, x+w, y+140]]), axis=0)
            # print('lefts_list : %s' %lefts_list)

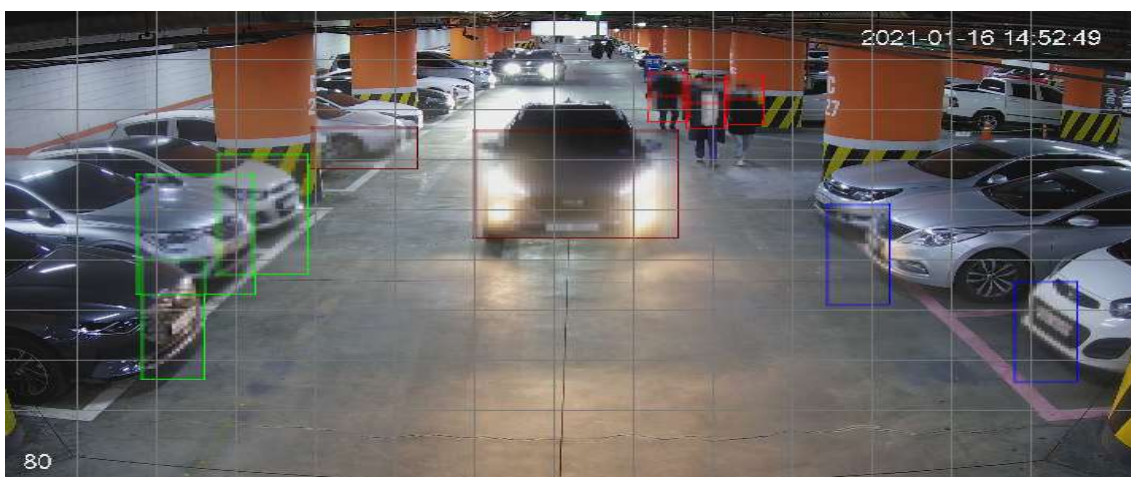
```

```

# 좌표값이 잘 저장되나 확인하기 위해서 frame에 grid 표시
# Calibration할 때 편리하기 위해서 표시
for k in range(0, 14):
    cv2.line(frame, (k * 50, 0), (k * 50, 480), (120, 120, 120), 1, 1)
for k in range(0, 9):
    cv2.line(frame, (0, k * 50), (720, k * 50), (120, 120, 120), 1, 1)

```

- 2번과 같은 오류가 발생하지 않기 위해서는 Calibration이 필요하다.
 - > CCTV마다 비추는 angle이 다르기 때문에 Detecting하는 영역과 움직이는 물체의 rectangle 영역을 Frame속 grid를 참고하여 Calibration
 - > 왼쪽 위 (0, 0) x, y축으로 눈금 한 칸에 50씩 증가
- 예시



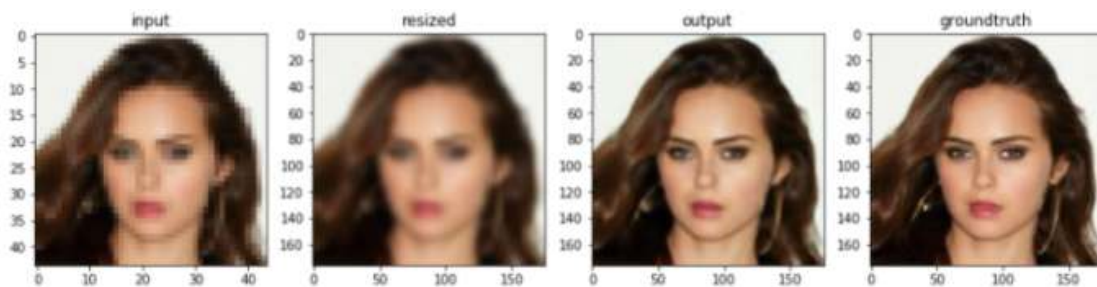
왼쪽 detect 영역은 $x < 250$, $y > 75$, 오른쪽 detect 영역은 $x > 470$, $y > 75$ 으로 설정해준다. 움직이는 차량의 detect 영역은 $y > 75$ 일 때 예시.

5.

이후 추가 할 내용

초해상도(Super-Resolution)

->딥러닝/ 머신 러닝을 이용하여 저해상도 영상을 고해상도 영상으로 변환시켜주는 기술. 초해상도 기술을 통하여 비식별 처리 된 저해상도 데이터를 고해상도로 변환시켜주어 데이터의 정보를 읽어 올 수 있다.



초해상도 구현 방법



참고자료

데이터 수집: <https://www.kaggle.com/dataturks/vehicle-number-plate-detection>

관련 코드: https://github.com/kairess/super_resolution
<https://github.com/beurtschipper/Depix>