친환경 차량 번호판 인식

Yolov5와 EasyOCR을 활용한 ALPR

상반기 베스텔라랩 인턴 장병희

목차

서론

1. 연구 목적

Yolo를 이용하여 차량에서 번호판을 검출한다. 검출한 번호판은 HSV Masking을 통해서 친환경 번호판인지 재확인을 한다. HSV를 이용하면 색상, 명도, 채도 값에 의해 색상 추출이 가능하게 된다. 그 후 Masking을 하여 색상을 추출하게 되면은 추출한 결과를 이진화시켜 픽셀 수를 계산한다. 그래서 픽셀 수가 특정 이상이면은 해당 번호판은 친환경 차량의 번호판이라고 판단하게 된다.

그림 1. 번호판에 따른 HSV Masking 결과

2. ALPR(Automatic number-plate recognition)

ALPR은 광학 문자 인식을 이용하여 차량의 번호판을 인식하는 것이다. 이 기술은 흔히 주차 정산 시스템에서 사용되어지고 있다. 이 기술을 위해서 특정 카메라들도 존재하고 있다. 본 개발에서는 특정 카메라를 사용하여 친환경 차량 번호인식을 하는 것이 아니라 관리용 목적으로 사용중인 카메라를 활용하여 차량 번호를 검출하는 것이다.

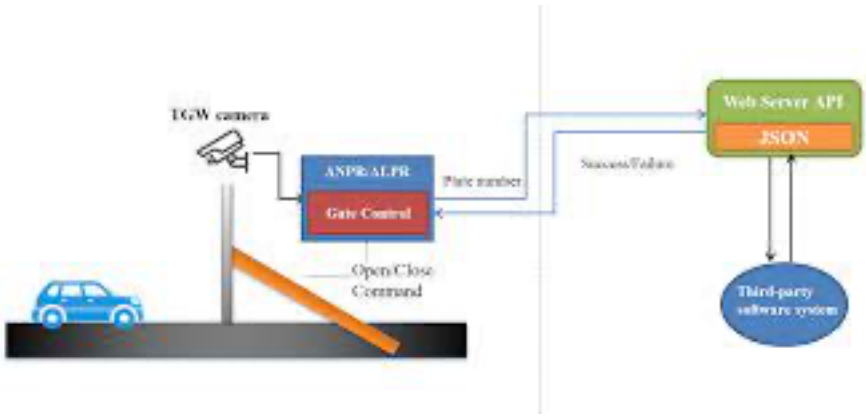

그림 2. ALPR 구조

본론

1. 개발환경

OS	CPU	RAM	GPU
Linux Mint 20.2	Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz	264GB	RTX 3090 24GB

개발 서버에서 실시하여 비교적 하드웨어 제원은 매우 높은 편이다. 그러나, 실질적으로 구동을 위해서 위와 같은 성능이 필요로 하지 않다. GPU가 장착되지 않더라도 실행하는 것에는 문제가 없으나 YOLO와 EasyOCR을 사용하기에 빠른 처리를 원한다면은 GPU 사용이 가능한 컴퓨터 환경이라면은 문제가 없다.

2. 개발 분석

1) YOLO 학습

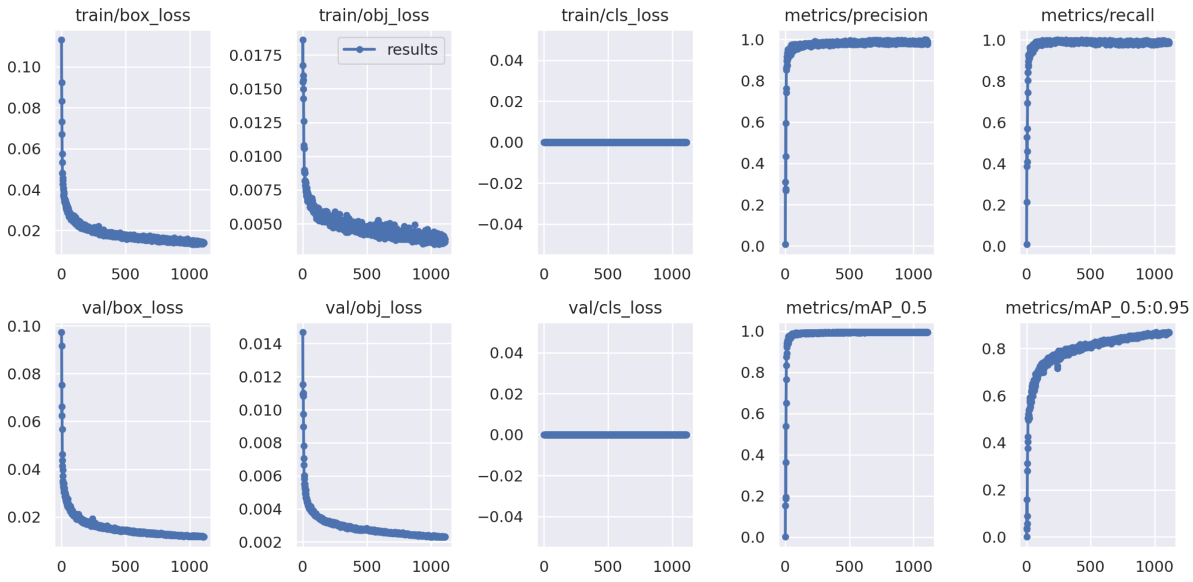

그림 3. 학습 결과

그림 4. 차량 내에 번호판 검출

학습 시킨 데이터를 통해서 추론 결과를 하면은 매우 높은 정확도로 추론이 되는 것을 확인할 수 있다. 초기 친환경 번호판이 없음에도 불구하고 번호판을 인식하는데 문제 없을 정도에 결과를 보여주었다. 본 학습을 위해서 400개 이상의 이미지 데이터를 사용하였는데 충분한 학습 결과가 나왔다고 판단되어 본 학습 결과를 사용하여 번호판을 검출하였다.

2) main.py

```
def detect(img,img_path):

    detects = model(img)

    for num,det in enumerate(detects.pandas().xyxy[0].values.tolist()):

        #Detect 결과
        [x1,y1,x2,y2,conf,cls,name] = det

        x1 = int(x1)
        y1 = int(y1)
        x2 = int(x2)
        y2 = int(y2)
        cls=int(cls)

        cv2.imwrite(f'{img_path}_crop_img.png', img[y1:y2, x1:x2])
        OCR_result = EasyOCR(f'{img_path}_crop_img.png')
        # OCR_result = kakao_ocr(img[y1:y2, x1:x2])
        if isElectronic(img[y1:y2,x1:x2],100)[0]:
            # 친환경 전기차 인경우
            print(f'친환경 자동차 OCR : {OCR_result}')
```

```
            # Using cv2.putText() method
            # img = cv2.putText(img, 'Electronic', (50, 50), cv2.FONT_HERSHEY_SIMPLEX,
            #                   0.3, (255, 0, 0), 2, cv2.LINE_AA)
        else:
            # 친환경 자동차 아님
            print(f'일반 자동차 OCR : {OCR_result}')

        img = cv2.putText(img,f'{name} {conf}', (x1,y1-10), cv2.FONT_HERSHEY_SIMPLEX,
0.5, COLORS[cls], 2, cv2.LINE_AA)
        img = cv2.rectangle(img,(x1,y1),(x2,y2),COLORS[cls],2)

    return img
```

코드 1. main.py(일부)

main.py 에서는 학습된 가중치를 통해서 YOLO를 통해서 차량에 부착되어 있는 번호판을 검출한다. 이때 pytorch hub를 사용하였기에 네트워크 환경이라면은 어떠한 곳에서든 최신의 Yolo를 사용할 수 있도록 하였다. Yolo를 통해서 얻어진 번호판을 통해서 EasyOCR을 통해서 문자를 인식한다. isElectronic 함수는 HSV 마스킹과 픽셀 수를 통해서 해당 번호판이 친환경 차량인지 파악하는 용도로 사용한다.

3) util/Easy_OCR.py - EasyOCR

```
def EasyOCR(img_path):
    reader = easyocr.Reader(['ko','en'],gpu=True)
    result = reader.readtext(img_path)
    text=""
    for sentence in result:
        text += sentence[-2]
    return text
```

코드 2. main_seq.py – getFrame

EasyOCR 라이브러리를 사용하여 문자를 인식한다. 이 때 GPU를 사용함으로 빠른 문자인식이 가능하도록 설정하였다.

4) detectElectronic.py

```
def isElectronic(img,thr):

    preprocessin_img = getElectronicColor(img)
    imgGray = cv2.cvtColor(preprocessin_img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
    imgThreshold = cv2.adaptiveThreshold(imgBlur, 255, cv2.ADAPTIVE_THRESH_GAUSSI
AN_C, cv2.THRESH_BINARY_INV, 19, 17)

    imgMedian = cv2.medianBlur(imgThreshold, 5)
    kernel = np.ones((3, 3), np.int8)
    imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)
```

```
    count = cv2.countNonZero(imgDilate)

    if count>thr:
        return True,count
    else:
        return False,count
```

```
def getElectronicColor(img):
    hsvLower = np.array([98, 50,68])  # 추출할 색의 하한(HSV)
    hsvUpper = np.array([108, 231, 201])  # 추출할 색의 상한(HSV)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)  # 이미지를 HSV으로

    hsv_mask = cv2.inRange(hsv, hsvLower, hsvUpper)  # HSV에서 마스크를 작성

    result = cv2.bitwise_and(img, img, mask=hsv_mask)

    return result
```

코드 3. main_seq.py – detect

본 개발에서 친환경 차량인지 구분하는 모듈이다. 먼저 Yolo를 통해서 얻어진 차량 번호판을 HSV를 통해서 masking을 한다. 그러면은 친환경 차량의 특정 색상만을 검출하게 된다. 그리고 검출된 결과를 이진화 시켜 0이 아니 픽셀들의 갯수를 파악한다. 그래서 특정 값 이상으로 픽셀 수가 많아지게 된다면은 해당 번호판을 친환경 차량이라고 판단하게 되는 것이다.

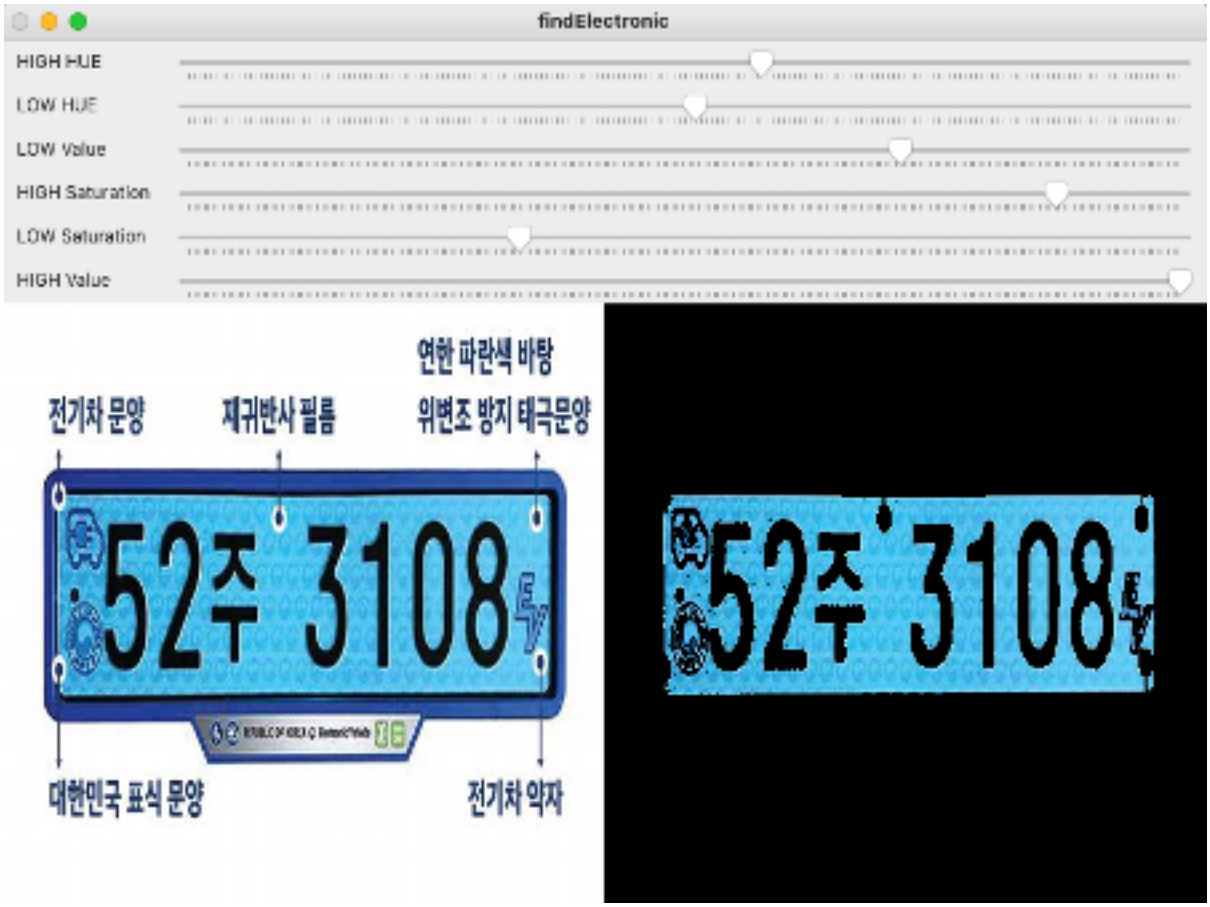

그림 5. HSV 마스킹

그림 6. 이진화후 얻어진 픽셀

3) 조건별 성능 분석

사용된 이미지 샘플

3.1 EasyOCR 사용

```
(easyocr) ves@ves-GPU:~/바탕화면/ANPR$ python main.py
Downloading: "https://github.com/ultralytics/yolov5/archive/master.zip" to /home
/ves/.cache/torch/hub/master.zip
YOLOv5 🚀 2022-5-13 torch 1.8.2 CUDA:0 (GeForce RTX 3090, 24268MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
친환경 자동차 OCR : 138차3415
일반 자동차 OCR : 30서 35569
일반 자동차 OCR : 확001-975
친환경 자동차 OCR : 33부 73145
일반 자동차 OCR : 0서:393나7327
```

EasyOCR 경우 추가적인 학습 없이 사용하였지반, 번호판 인식이 비교적 잘 이루어졌을 확인할 수 있었다. 또한 , GPU를 사용하였기에 API를 사용한 경우보다 더욱 빠른 주차판 번호 인식이 가능함을 확인할 수 있었다.

3.2 KaKao API 사용

```
(easyocr) ves@ves-GPU:~/바탕화면/ANPR$ python main.py
Downloading: "https://github.com/ultralytics/yolov5/archive/master.zip" to /home
/ves/.cache/torch/hub/master.zip
YOLOv5 🚀 2022-5-13 torch 1.8.2 CUDA:0 (GeForce RTX 3090, 24268MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
친환경 자동차 OCR : 3415 138차
일반 자동차 OCR : PICK 30서 3556
일반 자동차 OCR : 0 두팔간격 마스크착용 50 N 외교 001-975
친환경 자동차 OCR : - 33부 73144
일반 자동차 OCR : pay EXCELFx15 RYOACA 서울39- 나 7327
```

카카오에서 제공하는 API 경우 API 처리 과정에서 문자들의 위치를 찾아낸 후 문자들의 OCR을 진행하기 때문에 전체적인 이미지를 넣어야 OCR 결과를 얻을 수 있게 된다. EasyOCR에 비해서 모든 번호판 내용이 출력되기는 하나 번호판 내용외에 내용들이 많이 출력되게 된다. 그래서 현실적으로 사용하기에는 어려움이 보여진다.

3.결론

1) 고찰

그림 7. 구현 결과

본 개발을 위해서 Yolo를 사용해서 차량에 부착되어 있는 번호판들을 검출하였다. 그 후 검출한 번호판을 OCR을 통해서 차량 번호를 인식하였다. 거기 덧붙여서 HSV 마스킹을 통해서 친환경 번호판의 배경 색상만을 검출하였다. 그 후 검출한 색상을 이진화하여 픽셀 수를 계산함으로서 번호판이 친환경 번호판인지 일반 번호판인지 구분하였다. 본 개발에서

번호판을 검출하고 해당 번호판이 친환경 번호판인지 아닌지 구분하는 것은 효과적인 결과를 보여주었다. 그렇지만, OCR 경우 추가적인 학습 없이 OCR을 진행함으로 인해 인식율이 완벽하지는 않았다. 그러나 이 또한 OCR 학습을 진행함으로서 해결 가능성이 보여진다.