License plate Recognition

Final Report

2020. 12. 08

한선도, 김주호

1. Project Goal & Plans

System Definition and Business Model

System Definition

- YOLO를 활용하여 차량번호판 Object Detection
- Detection한 차량번호판 image를 가지고 OCR을 수행하여 차량번호판 Text Recognition
- Real-time으로 구현

Project Goal(Business Model; Real-Time System 구현)

1. 교내로 들어오는 차량들 중 bus의 차량번호판만 detection
2. Bus 차량번호판 text recognition
3. Bus의 출입시간 확인

Time Table(Midterm 이후)

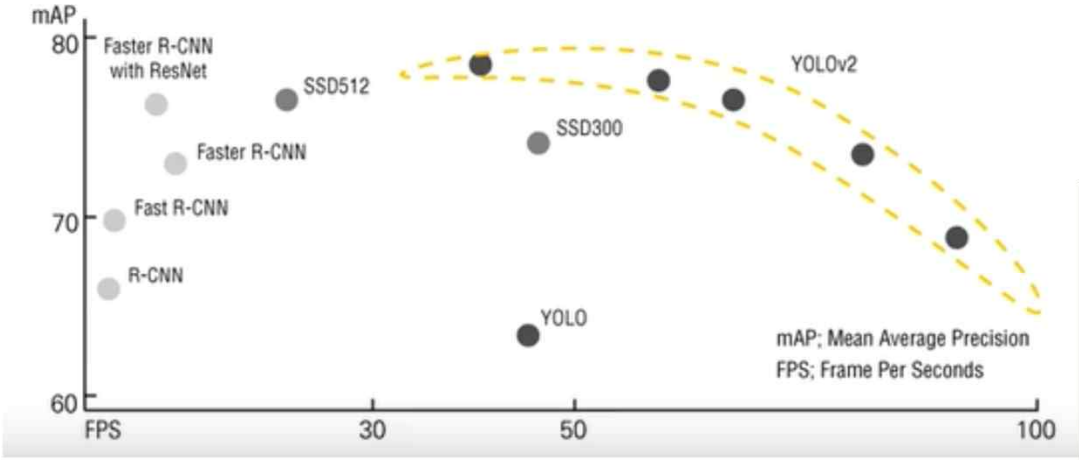
Week	계획	비고
11	영상으로부터 입력되는 Frame 수 변경 후 LPR수행	GREEN
12	Local 환경에서 Webcam을 통한 Object Detection 수행	GREEN
13	Local 환경에서 Webcam을 통한 Plate Recognition 수행 버그 수정 및 모델 튜닝	GREEN
14	Business model을 적용 및 영상환경의 차이에 따른 LPR 결과 분석 및 개선점 정리	GREEN
15	Final Report	GREEN

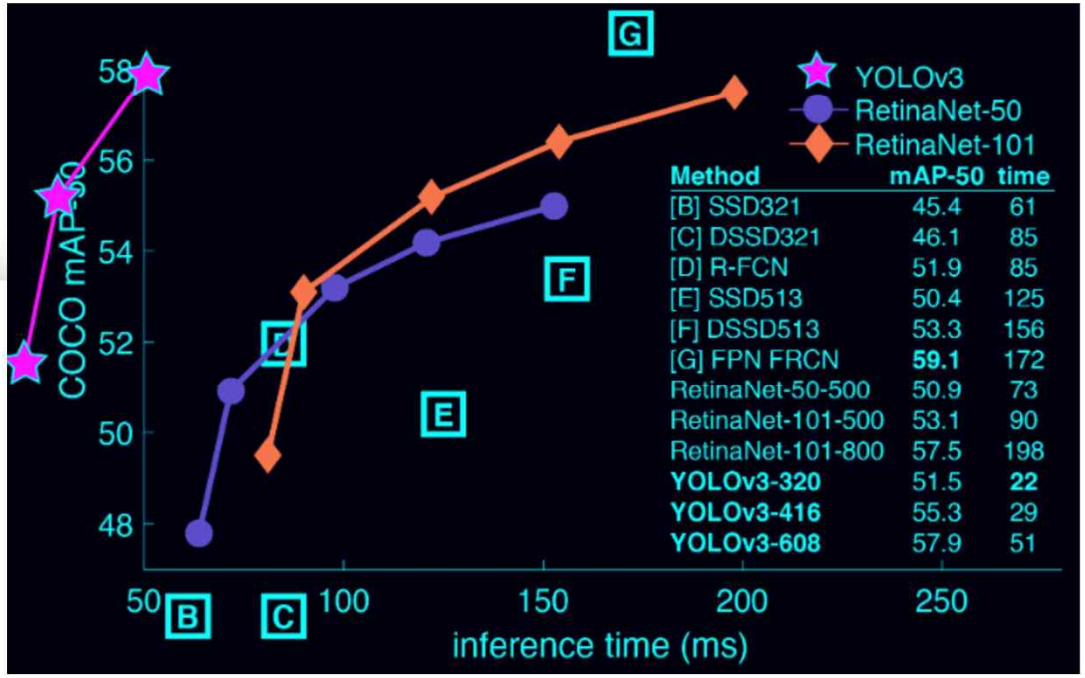
2. Project System Method

Machine Learning Methods Chosen

머신러닝 방법(YOLO)의 선택과 이유

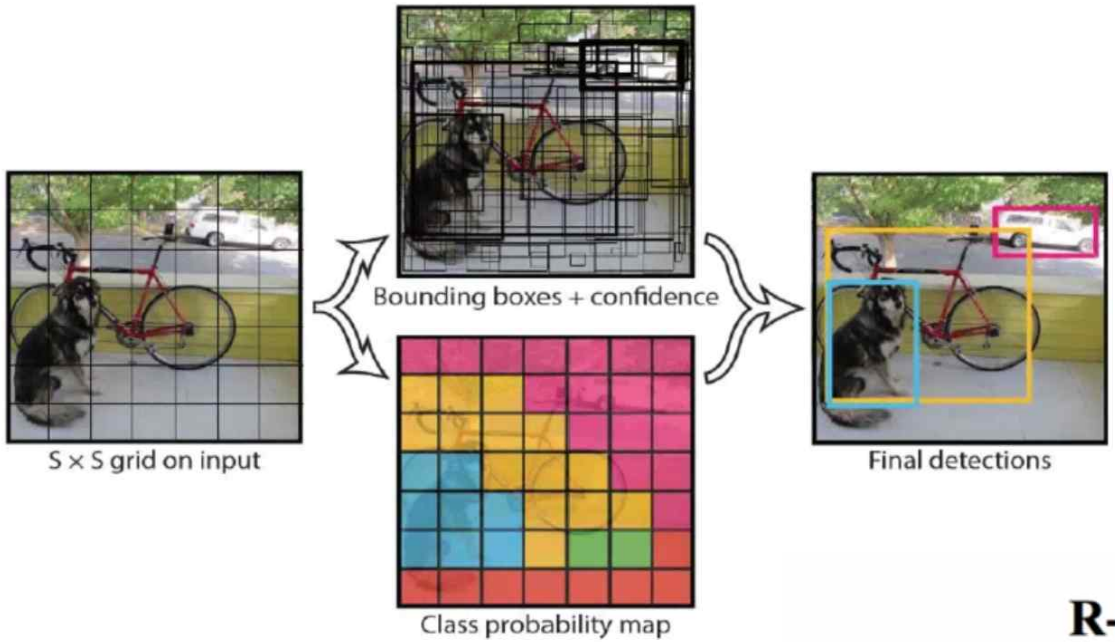
현재 사용되어지고 있는 주차장 차량관제시스템에서의 번호판 인식기술과의 차별성

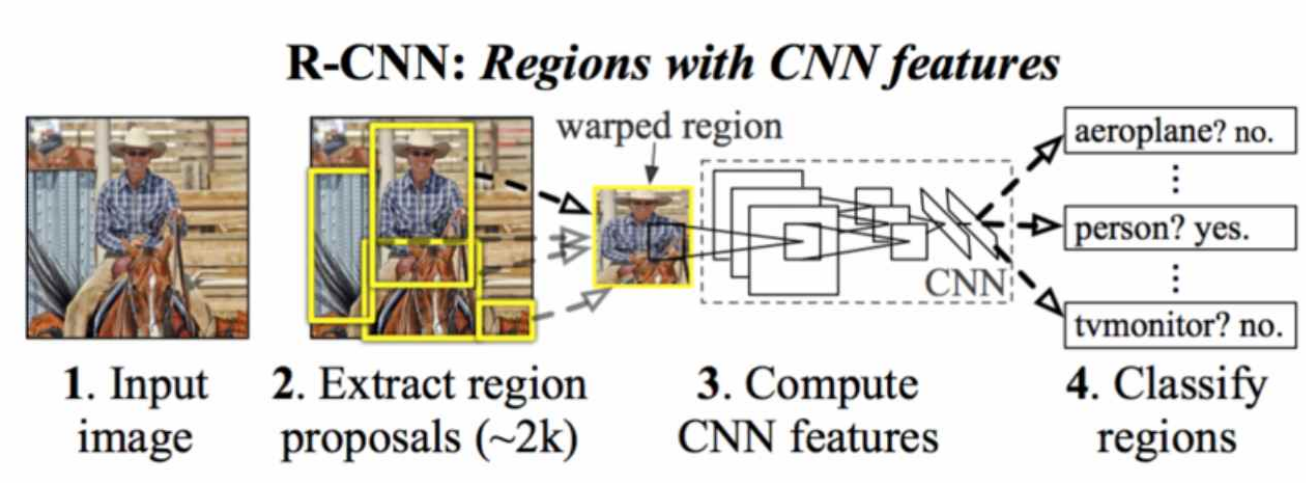




Major Lessons Learned

1. YOLO 인식 과정과 원리





Major Lessons Learned

1. YOLO 인식 과정과 원리

1. License Plate 이미지에 대한 weight 파일('lapi.weights')을 따로 다운로드 받아 Load 시켜 YOLO로 Image detection 할 수 있도록 함.

```
[7]  1 # 11 : Weight 파일을 로드하고 class name 입력해줌
     2
     3 net = cv2.dnn.readNet('gdrive/My Drive/CV/Object Detection_YOLO/lapi.weights',
     4                       'gdrive/My Drive/CV/Object Detection_YOLO/darknet-yolov3.cfg')
```

인식하고자 하는 대상에 대한 대량의 데이터와 훈련을 통해 필요로 되는 weight 파일(훈련모델)을 만들 수 있다.

1. YOLO 인식 과정과 원리

2. Classification에서 사용되는 class파일도 LP만 저장되어 있는 'classes.names' 파일을 사용한다.

```
5 classes = [] # detection 할 Object(Class) list 배열을 정의
6 with open('gdrive/My Drive/CV/Object Detection_YOLO/classes.names', 'r') as f:
```

Text 파일을 통해서 Classification할 대상의 이름을 정의할 수 있다.

1. YOLO 인식 과정과 원리

3. ‘Lapi.weight’가 적용된 학습된 YOLO모델을 사용하여 이미지 내의 차량번호판을 설정해둔 class name인 ‘LP’로 인식.

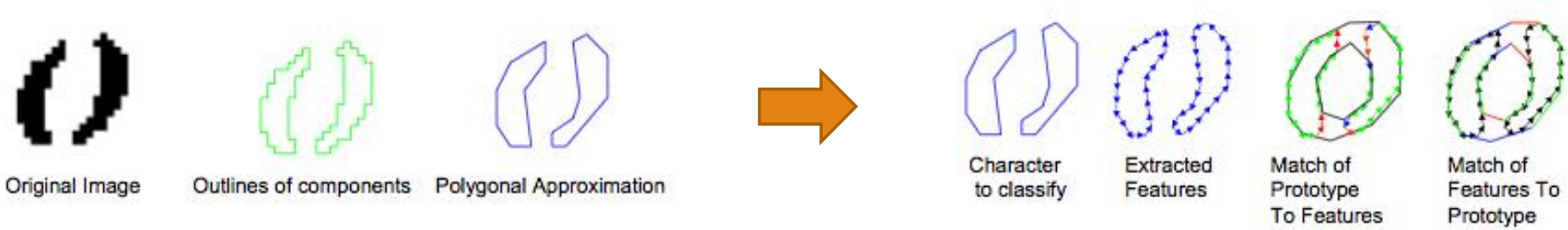
2. OCR (Optical Character Recognition; Tesseract)

Py-tesseract: 이미지 상의 텍스트 인식(OCR) - 번호판 텍스트 인식

How It Works?

Tesseract 는 오프라인 문자인식 기법으로 입력된 input 이미지의 특징점을 추출하고 그 특징점을 사용하여 문자를 인식한다.

문자의 특징점을 추출하는 방법은 원래의 image 에 outline 을 생성한 후에 방향성을 정하고 방향성에 따라 다각형에 근접하게 추출하게 된다.



2. OCR (Tesseract 사용 방법)

```
[10]  1 # 10 OCR 함수 정의
      2 def textRead(image):
      3   config = ('-l kor --oem 1 -- psm 7')
      4   text = pytesseract.image_to_string(image, config=config)
      5   print("OCR TEXT : {}₩n".format(text))
      6
      7   text = "".join([c if c.isalnum() else "" for c in text]).strip()
      8   print('Alpha numeric TEXT : {}₩n'.format(text))
      9   return text
```

<Tesseract 옵션 설정>

```
-- oem N

 0 = Original Tesseract only.
 1 = Neural nets LSTM only.
 2 = Tesseract + LSTM.
 3 = Default, based on what is available.


-- psm N

 0 = Orientation and script detection (OSD) only.
 1 = Automatic page segmentation with OSD.
 2 = Automatic page segmentation, but no OSD, or OCR. (not implemented)
 3 = Fully automatic page segmentation, but no OSD. (Default)
 4 = Assume a single column of text of variable sizes.
 5 = Assume a single uniform block of vertically aligned text.
 6 = Assume a single uniform block of text.
 7 = Treat the image as a single text line.
 8 = Treat the image as a single word.
 9 = Treat the image as a single word in a circle.
```

OCR engine mode 설정

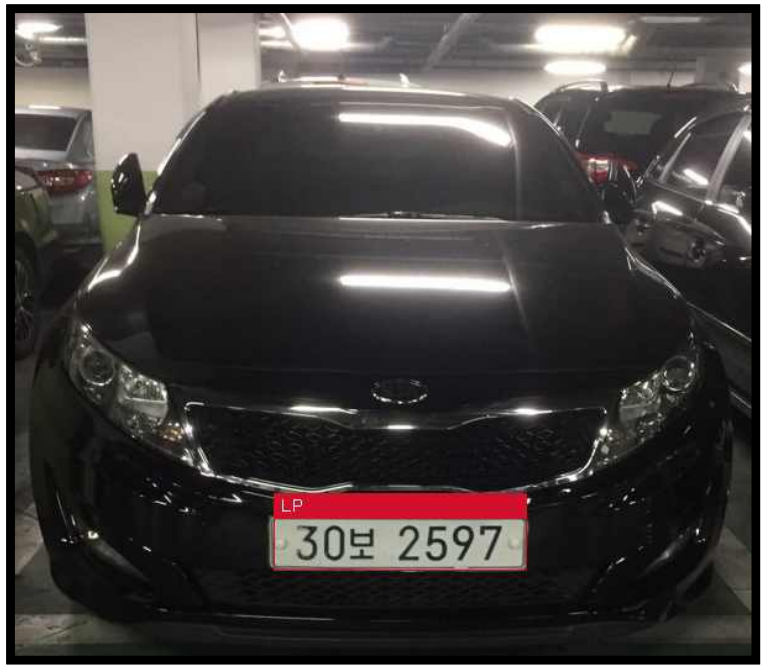
이미지에 대한 Layout 분석 방법을 결정

-> 차량 번호판이므로 'single text line' 선택

Major Lessons Learned

2. OCR (Tesseract의 정확도와 올바른 OCR 타겟 선정을 위한 방법)

1. YOLO를 통해 Plate Detection을 통해 정의한 **ROI(Crop image)** 사용으로 **올바른 OCR 타겟을 선정**한다.



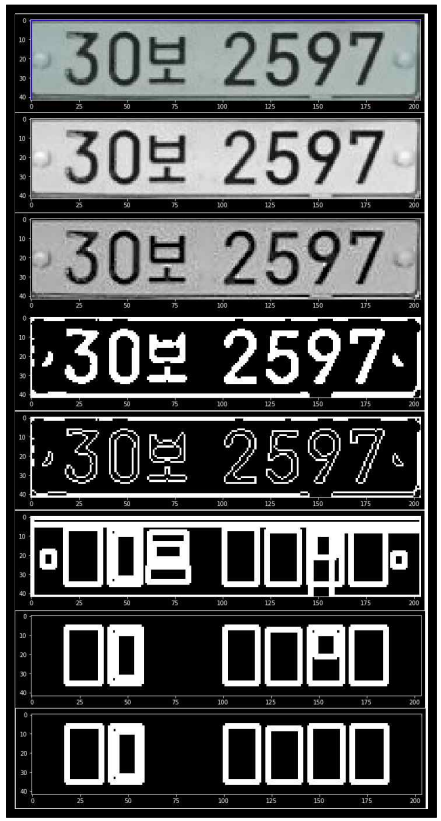

```
[ ]  1 # 15 crop한 이미지 출력
     2 cv2_imshow(crop)
```

Major Lessons Learned

2. OCR (Tesseract의 정확도와 올바른 OCR 타겟 선정을 위한 방법)

2. 이미지 전처리과정을 통해 OCR의 인식률을 높인다.

(문자가 있는 위치 탐색과 인식 정확도 향상 기능)



< Image 전처리 과정 >

1. Grayscale
2. TopHat, BlackHat, plus Grayscale
3. Thresholding, GaussianBlur
4. Find contours(특징점 찾기)
5. 사각형 그리기
6. 최소 사각형 크기에 제한을 두어 문자로 예측되는 사각형 식별

Major Lessons Learned

한글 첨자 인식에 대한 OCR(Tesseract) 의 낮은 인식률

'보' 를 '브' 라고 인식.

'고' 를 '끄' 라고 인식.

-> 한글 첨자 인식에 취약함

-> 한글 첨자 인식에 대한 추가적인 학습 필요

3. Business Model 구현

Project Goal(Business Model; Real-Time System 구현)

1. Real-Time System 구현

Real-Time에서의 LPR(License Plate Recognition)은 동영상에서의 LPR과 동일하게 이루어지며 Bus에 대한 model test가 여건상 연속적으로 test하기 어려우므로 동영상으로 model test를 수행함.

2. Bus의 차량번호판만 Detection, Bus 출입 시간 출력

3. 핵심기술은 Object Detection 과 OCR의 성능

Model이 적용되는 환경(차량 속도, 카메라 해상도, FPS)를 설정하여 model test를 진행하였으며, 차량번호판 인식 성능을 높이는데 초점을 맞추었다.

Project Goal(Business Model; Real-Time System 구현)

4. 필요조건

1) 버스를 가지고 지속적인 실험이 어려우므로 승용차를 통해 실험

2) 버스가 정류장으로 들어올 때의 속도를 기준으로 30km/h 이하의 속도에서 적용 가능한 모델 구현

5. 설치조건

버스가 들어올 때 속도가 느려지는 구간(ex. 과속방지턱 주변)에 설치

Model 적용시 조건문을 통해 OCR 결과로 출력되는 Text의 OCR의 신뢰도를 높인다.

```
length = len(result_chars.encode('utf-8'))
if len(result_chars) == 8 or len(result_chars) == 7:
  if length == 9 or length == 10:
    cv2_imshow(img_result)
    print(result_chars,img_result.shape) # 한 프레임 내 OCR로 인식된 번호들의 집합
    plate_chars.append(result_chars)
```

1. TEXT에 할당되는 byte 제한 조건
한글: 3 byte
숫자: 1 byte

보통 single line 차량 번호판일 경우
'00한0000' or '000글0000' 형식이므로
9, 10 byte로 조건을 만들어준다.

(3*1)+(1*6)= 9 byte
(3*1)+(1*7)= 10 byte

2. TEXT 글자 수에 대한 조건

보통 single line 차량 번호판일 경우
'00한0000' or '000글0000' 형식이므로
7~8개로 이루어진 결과만을 출력되도록 한다.

→ 올바른 결과 값의 형식을 가지지 않는 출력들을 1차적으로 제거할 수 있다.

촬영 환경에 대한 model의 성능 분석

해상도	차량 속도(km/h)	최초 OCR 인식 해상도
FHD 1920x1080 (30FPS)	10	**49x168**
	20	47x163
	30	47x166
FHD 1920x1080 (60FPS)	10	48x170
	20	52x186
	30	49x172
QHD 2560x1440 (30FPS)	10	**55x204**
	20	57x220
	30	60x239
UHD 3840x2160 (30FPS)	10	**73x290**
	20	73x307
	30	69x277
UHD 3840x2160 (60FPS)	10	68x281
	20	66x268
	30	72x308

높은 해상도 일수록 증가함

FHD -> QHD -> UHD
순으로 최초 OCR 인식 해상도증가.

YOLO Detection은 픽셀 수에 따라 수행되는 것이 아닌 Image에 gird를 만들어 detection하는 방식이므로 고해상도 일수록 만들어지는 gird의 픽셀 수가 증가하는 것이다.

최초 OCR 인식 해상도를 비교하여 보았을 때 FHD 카메라로도 충분히 ocr을 수행하는데 큰 무리가 없다.

촬영 환경에 대한 model의 성능 분석

해상도	차량 속도(km/h)	OCR 신뢰도(%)
FHD 1920x1080 (30FPS)	10	88.67
	20	83.33
	30	58.27
FHD 1920x1080 (60FPS)	10	86.33
	20	80.00
	30	78.79
QHD 2560x1440 (30FPS)	10	87.27
	20	93.33
	30	93.33
UHD 3840x2160 (30FPS)	10	90.63
	20	88.00
	30	93.75
UHD 3840x2160 (60FPS)	10	76.92
	20	82.35
	30	66.67

대부분의 결과에서 차량속도 증가에 따라 신뢰도 감소.

But, 신뢰도를 계산하는 방식 또는 촬영 변수 (차량과의 거리, 촬영 각도) 로 인해 그렇지 않은 경우도 존재.

프레임마다 출력된 것 들 중 가장 빈도수가 높은 것으로 최종 출력을 하는 모델이라 가정을 한다면, 현재 30km이하의 환경에서 모든 모델의 신뢰도가 50% 이상이므로 모든 모델에서 100%의 정확도를 보여준다.

모두 50% 이상의 신뢰도

→ **100% 정확도**

4. Summary

모델 평가

QHD 2560x1440 (30FPS)로 촬영한 **시속 30Km/h** 차량 LPR 결과

Project를 진행하면서 수행하기 어려웠던 점

1. Real-time 구현(느린 인식 속도)

➡ 초당 입력 받는 Frame 수를 용도에 맞게 조절 가능하게하여 해결

2. OpenCV2 환경에서의 한글 출력

➡ 유니코드를 지원하는 PIL 라이브러리를 통해 draw.text로 한글 출력

추가적으로 진행이 필요한 수행과정

1. 구형 번호판 인식

1) 서울 경기 등 지역명이 포함된 구 번호판

2) Double line 형식의 구 번호판

2. 신형번호판 인식

한글 앞 숫자가 세자리('000가0000' 형식)인 번호판

3. 다양한 번호판 인식 실험

다양한 나라의 번호판

전기차 인식(다른 번호판 색)

차량과 번호판을 동시에 인식 (차종+번호판 출력)