SSL SEMINAR

Deep Learning Based Vehicle Recognition Sys.

Kumoh National Institute of Technology
System Software Lab.
20200573 서준혁

SSL SEMINAR

PART 01

번호판 인식 최적화

최적화 알고리즘 구현

PART 02

프로그램 구현

구현 내용 및 소스 코드 분석

PART 03

기타 사항

향후 계획 및 고민

SSL SEMINAR

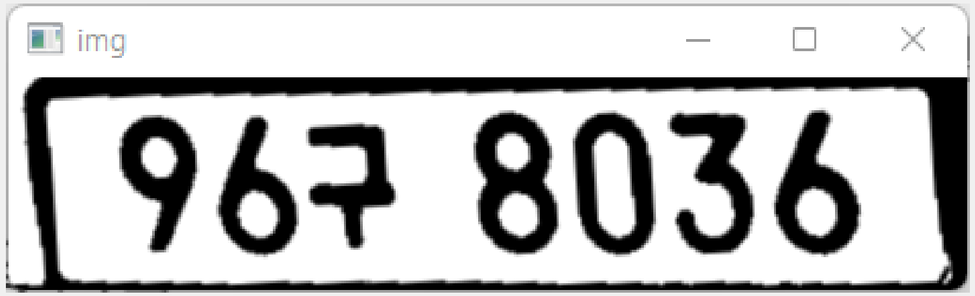
PART
01

최적화

번호판 인식 최적화 알고리즘

문제점


```
dataset\20150310100150606러3956.jpg 처리중...
6602 3956의
```

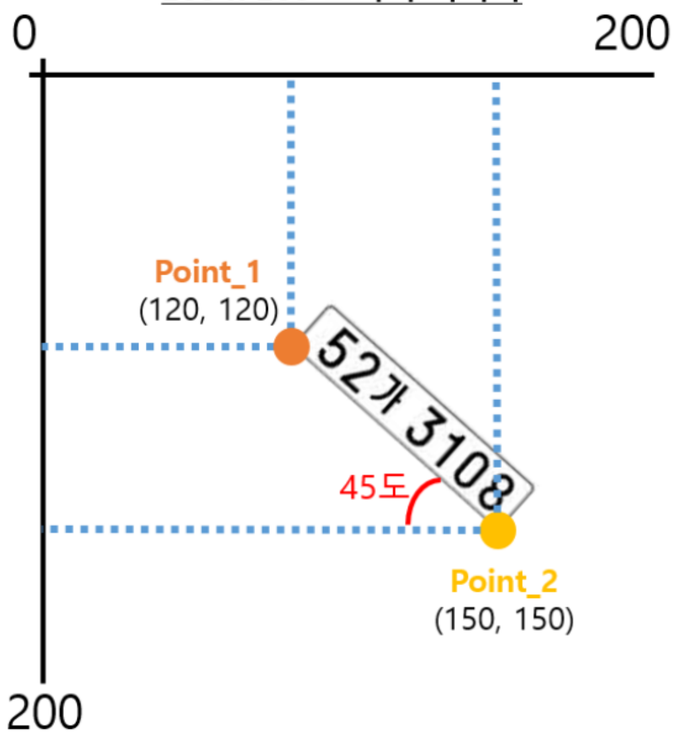


저번 발표까지 공부를 하면서 발견한 문제점은, 번호판을 찾는 과정에서 찾은 글자를 기준으로 N배 만큼 곱한 면적을 번호판으로 인식시켜버리기에 번호판 영역의 오차가 커지는 점, 또한 이로 인해 숫자 외에 다른 부분을 인식함으로써 발생하는 문자 오인식이었다.

또한, 번호판의 기울기가 완벽하게 수평이 되지 않고 여전히 기울어진 상태로 CROP 했기에 인식률이 낮아지는 일도 발생했었다.

추출된 번호판 영역

200 x 200 크기의 이미지



번호판 보정 수행

각도 계산 공식

$$\theta = \arctan\frac{|y_2 - y_1|}{|x_2 - x_1|} \times \frac{180}{\pi}$$

각도 계산 예시

Point 1, 2 가 (120, 120), (150, 150) 일 때,

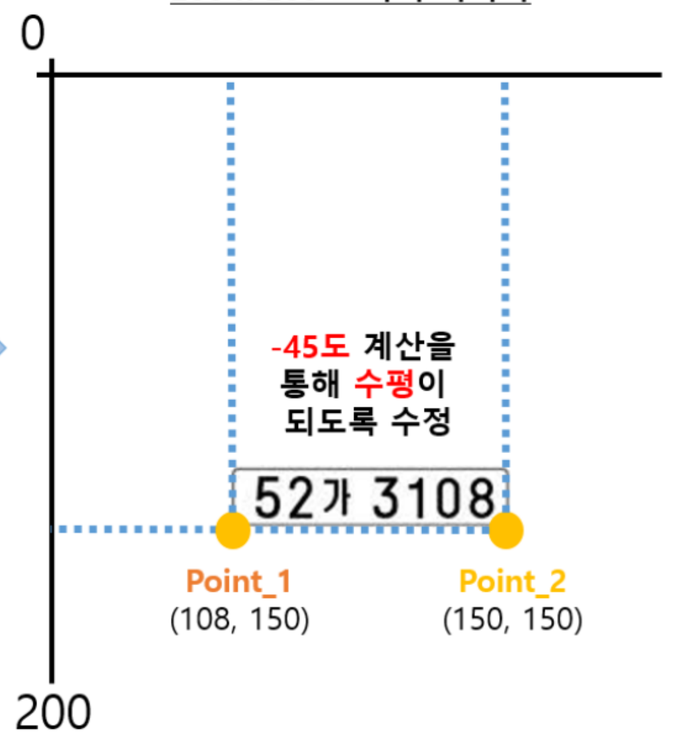
(1-1) dy = p2_y – p1_y = 150 - 120 = 30
(1-2) dx = p2_x – p1_x = 150 - 120 = 30

(2-1) arctan(dy / dx) = 0.7854
(2-2) (180.0 / PI) = 57.2958

(3) **Angle** = 0.7854 * 57.2958 = **45.0**

보정된 번호판 영역

200 x 200 크기의 이미지



수학적으로 탄젠트의 역함수인 ARCTANGENT를 사용하여 번호판을 수평으로 만들기 위한 방정식을 세울 수 있기 때문에,
해당 공식을 코드로 구현했다.

SSL SEMINAR

PART
02

프로그램 구현

구현 내용 및 소스 코드 분석

```
# Visualize Possible Contours
possibleResult = np.zeros((height, width, channel), dtype=np.uint8)

for items in matchResult:
    for item in items:
        cv2.rectangle(possibleResult, (item['x'], item['y']), (item['x'] + item['w'], item['y'] + item['h'])

for i, items in enumerate(matchResult):
    sortion = sorted(items, key = lambda x : x['cx'])
    return sortion

return possibleResult
```

우선 추출한 후보 컨투어의 'CX' (각 컨투어의 X축 기준 중간 지점)순으로 정렬하여 리턴한다.

```
# Visualize Possible Contours
possibleResult = np.zeros((height, width, channel), dtype=np.uint8)

for items in matchResult:
    for item in items:
        cv2.rectangle(possibleResult, (item['x'], item['y']), (item['x'] + item['w'], item['y'] + item['h'])

for i, items in enumerate(matchResult):
    sortion = sorted(items, key = lambda x : x['cx'])
    return sortion

return possibleResult
```

우선 추출한 후보 컨투어의 'CX' (각 컨투어의 X축 기준 중간 지점)순으로 정렬하여 리턴한다.

```
points = detectWhitePlate(gray)
LeftTop = (int(points[0]['x']), int(points[0]['y']))
LeftBottom = (int(points[0]['x']), int(points[0]['y'] + int(points[0]['h'])))
RightTop = (int(points[-1]['x']) + int(points[-1]['w']), int(points[-1]['y']))
RightBottom = (int(points[-1]['x']) + int(points[-1]['w']), int(points[-1]['y']) + int(points[-1]['h']))
```

전달받은 컨투어들의 첫 번째 값은 가장 왼쪽 컨투어, 마지막 값은 가장 오른쪽 컨투어라는 점을 이용하여 번호판 내부 숫자 영역의 정확한 네 개의 꼭짓점을 구한다.

```
dst = image.copy()
cv2.line(image, LeftTop, RightTop, (0, 255, 0), 3)
cv2.line(image, LeftBottom, RightBottom, (0, 255, 0), 3)
cv2.line(image, LeftTop, LeftBottom, (0, 255, 0), 3)
cv2.line(image, RightTop, RightBottom, (0, 255, 0), 3)

rows, cols, _ = dst.shape
pts1 = np.float32([LeftTop, LeftBottom, RightTop, RightBottom])
pts2 = np.float32([[0, 0], [0, rows], [cols, 0], [cols, rows]])
matrix = cv2.getPerspectiveTransform(pts1, pts2)
dst = cv2.warpPerspective(dst, matrix, (cols, rows))
```

CV2.LINE 함수를 이용하여 4개의 꼭짓점을 잇는 선을 그어 경계선을 표시하고, 앞서 설명한 기울기 보정을 수행할 수 있도록 2차원 행렬을 만들어 매개변수로 전달한다.

PTS1은 기존 영상 속 번호판의 영역
PTS2는 기존 영상 전체 영역을 의미하며

함수 내부에서 서로 대응하는 인자끼리 역탄젠트 연산을 통해 기울기 보정을 수행한다.


15바 1562


image
KIA
60러 3956


img
60러 3956

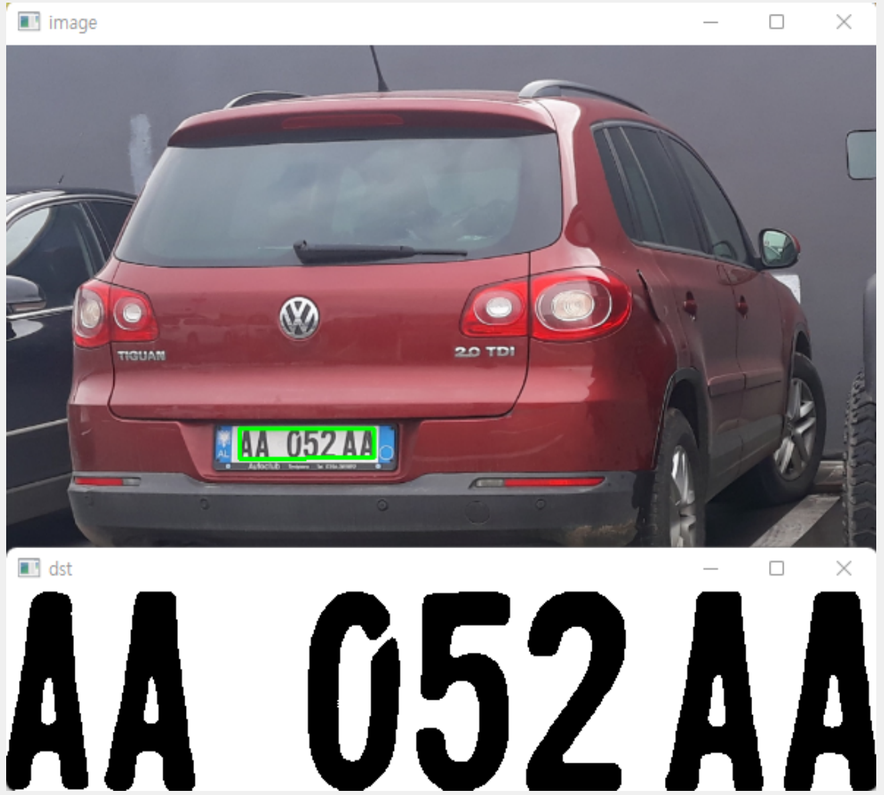

dst
60러 3956


```
dataset\20150310100015060러3956.jpg 처리중...
6602 3956의
```

오차 범위로 인해 발생하는 오류가 사라진 것을 확인할 수 있었음

이전에 구현했던 영상 처리 알고리즘을 보완해 구형 번호판과 특수 번호판도 어느정도 인식하는 것을 확인할 수 있었음.

(이번 구현 내용을 적용하지 않았기 때문에 대략적인 위치만 인식)



어둡거나 노이즈, 빛 반사가 조금 있는 영상도 추출 결과는 번짐이 조금 있지만 인식이 가능했으며
외국 번호판 또한 인식이 가능했음

번호판 추출과 노이즈 제거는 예상보다 아주 훌륭하게 됐지만

TESSERACT OCR의 한글 인식률이 생각보다 아주 좋지 않아서

TESSERACT 5.X 버전에서 제공하는 LSTM 기법을 이용해 자동차 번호판 폰트 학습이 필요할 것 같다.

타 OCR (KERAS-OCR, EASY OCR)도 사용해보고 싶었지만

빠르게 세팅하기에 WINDOWS 환경에서는 제한이 있어 아쉬운 면도 있었다.

EASY OCR이 한국 개발자가 학습 시켜놓은 OCR이라 한글 인식률이 더 좋다고하는 정보가 있어

기회가 된다면 리눅스 환경에서 세팅을 시도해보아야겠다.

SSL SEMINAR

Deep Learning Based
Vehicle Recognition Sys.

Contact

발표자 | 서준혁

TEL. 010-9802-5053 **E-mail** ssam2s@naver.com