TensorFlow Object Detection API를 활용한 건물번호판 인식과 Tesseract OCR을 이용한 자동 주소 인식

이주호, 박기수

인하대학교 컴퓨터정보공학과

(12161640@inha.edu)

Building License Plate recognition using TensorFlow Object Detection API and automatic address recognition using Tesseract OCR

Ju-Ho Lee, Ki-Soo Park

Dept. of Computer Science, Inha University

Abstract

The Republic of Korea requires that building license plates to be attached to all buildings in accordance with the Road Name Address Act. Currently a person have to take pictures of building license plates and manage the building address separately. We proceed this project to improve the method and develope a system that can be automatically registered in the database through automatic address recognition when taking pictures. We used TensorFlow Object Detection API to make building license plate detector. We trained the detector by Faster R-CNN inception model and it worked well. We also used Tesseract OCR library to recognize Korean address in building license plate. Experiments were conducted to increase the character recognition rate of Hangeul and the result of proper image processing came out.

In this document, we describe the image processing process for automatic address recognition and explain the experiments and results of the most appropriate image processing. It will also explain the methods, processes, and results of building license plate detector training. All image data and training data used in the building license plate detector training course are detailed in the following link.

https://github.com/JuhoLeedev/Winter-School-Building-License-Plate-Tensorflow

Ⅰ. 서 론

현재 대한민국에서는 2017년 시행된 도로명 주소법에 따라 각 건물마다 건물번호판을 부착하도록 되어있다. 또한 도로명주소안내시설규칙에 의거해 통일된 모양의 건물번호판을 부착해야 하며, 일부 자율형 건물번호판 또한 허용하고 있다. 각 지자체에서는 부착된 모든 건물번호판의 사진을 찍어서 관리하고 있는데, 현재까지의 관리 방법은 구청 직원이 직접 현장에 나가 해당 주소지를 따로 입력하고 건물번호판의 사진을 직접 찍어서 데이터베이스에 업로드 하는 방식으로 관리되고 있다.

최근 deep learning 기술과 OCR 기술의 발전에 따라 비슷한 License Plate라고 볼 수 있는 차량 번호판의 경우, 과속단속카메라나 주차장 출입구의 카메라 등이 자동으로 차량 번호판을 인식하고 차량번호를 읽어내는 방식이 이미 도입되어 있다.

이에 착안하여, 현재까지 사람이 수동으로 직접 주소지를 입력하고 사진을 찍어 따로 관리하는 방식을 개선하고 차량 번호판의 경우와 같이 사진을 찍으면 자동으로 주소를 인식해 데이터베이스에 업로드 되어지는 애플리케이션을 만들어보면 좋을 것 같다는 아이디어를 얻어 건물번호판 detector와 해당 번호판의 주소를 자동으로 인식하는 OCR 프로그램을 개발하게 되었다.

본 프로젝트에서는 직접 건물번호판 사진 data를 모아 TensorFlow Object Detection API를 활용해 건물번호판 detector를 직접 훈련시키고 Tesseract OCR을 활용해 인식한 건물번호판의 주소지를 읽어내는 과정을 설명하고 그 결과를 분석하고자 한다.

Ⅱ. 관련 연구

2.1 기존 차량 번호판 검출 연구

기존 차량의 License Plate 검출 방식에 대한 연구를 통해 건물번호판에 대한 검출 방식을 결정하고 개선된 방안을 연구한다. 기존 차량 번호판의 경우, 촬영된 사진, 영상에서 어느 곳에 번호판이 위치하는지 알 수 없기 때문에 번호판의 형태, 크기 색 등 통일된 번호판의 형태에 대한 사전 지식과 특징들을 이용하여 검출한다.

2.1.1 번호판 외각 경계선 및 에지를 이용한 검출

번호판은 사각형 모양으로 가로세로 비가 일정한 비율로 구성되어 있으며 모양 또한 통일되어 있기 때문에 번호판 외각 경계선(edge)를 이용하여 검출한 수직, 수평 경계선들의 정보를 가지고 번호판의 가로 세로 비율을 계산하여 검출한다. Edge 연산자를 이용한 번호판 검출은 빠르고 간단하지만, 사진의 질이나 날씨, 광원 등에 쉽게 영향을 받으며, 정확도가 높지 않다는 문제가 있다.

Figure 1. 에지 연산을 통해 검출한 차량 번호판

2.1.2 번호판 색상 및 특징을 이용한 검출

번호판 색상 및 특징을 이용한 검출 방법은 번호판 영역내의 문자를 제외한 바탕 색상 정보를 이용하여 검출한다. 사진의 모든 픽셀들을 HIS 색상 모델을 이용하여 번호판 색상 조합과 일치하는 해당 영역을 검출하는 방식으로 번호판이 일부 훼손되어 있거나 왜곡되어 있어도 검출하기 쉽다는 특징을 가지고 있지만, 번호판의 색상이 차량의 색상과 유사할 경우 오검출이 발생하는 문제가 있다.

2.2 TensorFlow Object Detection API

최근 구글에서 공개한 객체 인식 프레임 워크 오픈소스인 TensorFlow Object Detection API는, 구글의 machine learning, deep learning 엔진인 TensorFlow를 이용하여 이미지에서 객체를 인식할 수 있도록 개발된 API로, 라이브러리 형태로 제공되며 이미지 학습용으로 개발된 수많은 deep learning 알고리즘을 쉽게 이용할 수 있다[1].

사용자가 직접 데이터를 이용해 학습을 하여, 사용자의 시나리오에 맞는 모델을 사용해 이미지

또는 영상의 객체를 경계 상자로 감지하는데 사용할 수 있다. 응용 가능한 신경망 모델로는 Single Shot Multibox Detector(SSD), Region-Based Fully Convolutional Networks(R-FCN), Region-based Convolutional Neural Network(Faster R-CNN)등이 있다. 각 모델에 따라 속도와 성능, 탐지하는 객체의 개수에 따른 결과 차이가 있으며, 본 프로젝트에서는 Faster R-CNN 모델을 적용했다[2].

2.3 Faster R-CNN

영역 기반 심화학습 Faster R-CNN은 CNN 기반 객체 검출 기법으로 RPN(Region Proposal Network)와 Fast R-CNN의 하단 부분인 화전 연결층을 제외한 컨볼루션 네트워크와 특징 맵을 서로 공유하도록 구성된 기법이다[3]. 기존에 사용되던 Region Proposal 방법인 Selective Search는 CPU에서 계산되어 시간이 오래 걸렸다. Faster R-CNN은 GPU의 이점을 최대한 활용하고 CNN 내부에서 진행하기 위해 Region Proposal Network(RPN)을 도입했다. RPN은 각 위치의 object bounds와 objectness score를 동시에 예측하는 fully convolutional network이다[4]. Figure 2은 Faster R-CNN의 구조를 보여준다.

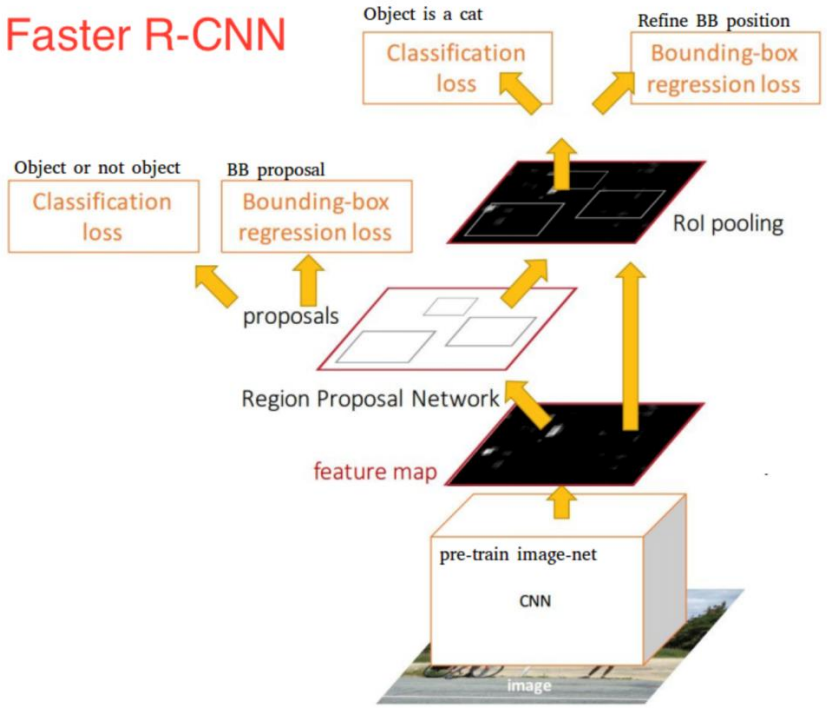

Figure 2. Faster R-CNN의 구조

RPN 은 특징 맵에서 3x3 격자(gird cell) 중심에 각 다른 3 개의 scale 과 3 개의 aspect ratio 를 이용하여 9 개의 anchor(box)를 구성하여 3x3 Sliding window Convolution 을 수행한다. 학습 구조 차원인 256 또는 512 차원의 벡터를 만든 후 객체 후보 영역 클래스에 대한 층과 객체 후보 영역의 좌표를 생성하는 층에 연결된다[3]. Figure 3 은 RPN 의 구조를 보여준다.

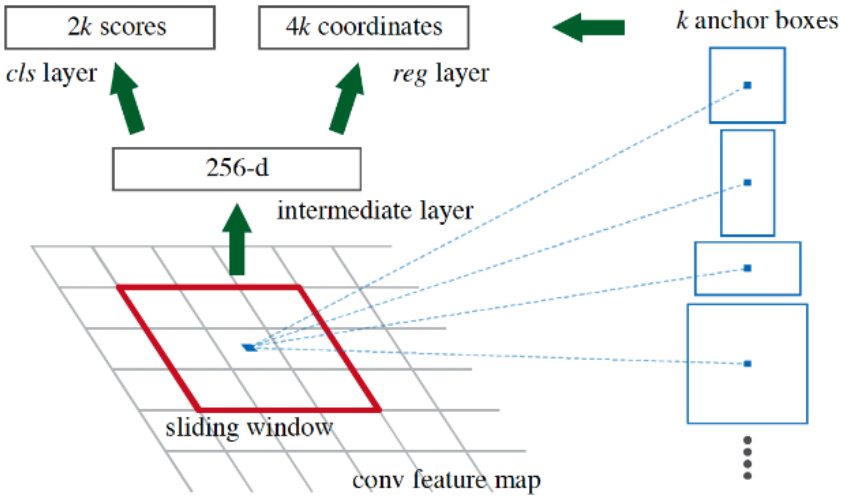

Figure 3. RPN의 구조

RPN을 수행할 때 모든 anchor마다 foreground, background를 정의한다. Anchor가 ground truth box와 IoU가 가장 크고 0.7 이상이면 foreground, 적으면 background로 분류한다. RPN을 지나면 서로 다른 크기의 proposed region이 나오는데 서로 다른 크기의 region을 동일한 크기로 맞추기 위해 Rol Pooling을 사용한다. 한 이미지에서 random으로 positive anchor와 non-positive anchor를 1:1 비율로 사용해 mini-batch만큼 anchor를 샘플링한다. 이때 보통 negative anchors가 더 많기 때문에 비율을 조정하지 않으면 학습이 한쪽으로 편향된다. 만약 postitive anchor가 128개보다 적으면 zero-padding을 시켜주거나 아예 positive가 없으면 IoU 값이 높은 값을 사용한다. object와 not object에 대한 손실 함수(loss function)은 다음과 같이 정의한다[4].

$$L(\{p_i\},t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*)$$

2.4 Tesserect-OCR과 이미지 전처리

Tesseract-OCR은 Google의 오픈소스 프로젝트로 문자인식 분야에서 광범위하게 사용된다. 최근 Tesseract-OCR 엔진의 성능이 LSTM(Long Short-Term Memory)을 기반으로 한 업데이트를 통해 비약적으로 향상되었으나, Tesserect-OCR은 저품질 이미지에서 성능이 급격히 하락한다. 이는 입력 이미지의 전처리 과정이 없거나, 유연하지 못한 것이 주된 원인이다. Tesseract-OCR의 인식률은 입력 이미지의 불필요한 영역(잡음, 그림자 등)에서 영향을 받는다. 입력 이미지는 스캐너, 카메라, 휴대폰 등으로 촬영되기 때문에 많은 문제점이 발생하는데, 이런 이미지를 그대로 엔진의 입력으로 사용할 경우 정확한 인식 결과를 도출할 수 없다, 따라서 정확한 인식을 위해 첫번째로 이미지로부터 문자 영역의 추출이 필요하다. 이는 TensorFlow Object Detection API를 활용한 객체 탐지로 처리를 진행하게 된다. 두번째로 이미지의 잡음을 제거하기 위해, multi-scale framework, binarization from noisy gray-scale character, projection denoising, block adaptive binarization 등의 다양한 기법을 사용한다[5].

Ⅲ. 건물번호판 detector 학습과 결과

3.1 TensorFlow 건물번호판 detector 구현

Faster R-CNN 모델을 사용하여 건물번호판 detector를 효율적으로 훈련시키기 위해서는 GPU가 필요하다. 본 프로젝트에서는 Ryzen 3700x 프로세서와 RTX 2080 그래픽 카드를 사용해서 훈련을 진행하였다. 건물번호판 detector 학습에 사용된 이미지 데이터들과 학습된 frozen inference graph, train 데이터는 다음 깃허브 링크에 모두 업로드 되어있다. readme에는 TensorFlow 설치부터 detector 학습까지 모든 과정을 상세하게 적어놨다.

https://github.com/JuhoLeedev/Winter-School-Building-License-Plate-Tensorflow

3.1.1 이미지 수집 및 라벨링(Labelling)

TensorFlow Object Detection 라이브러리로 이미지를 학습시키기 위해서는 대략 100장 이상의 이미지가 필요하다. 이미지는 직접 시내를 돌아다녀서 건물번호판 사진을 스마트폰 카메라로 촬영해 수집했으며, 총 267장의 저녁 시간대와 오후 시간대의 사진을 수집하였다. 수집된 사진들은 labelImg 프로그램을 사용해 라벨링을 진행했고, 수집된 건물번호판의 양식이 2개이기 때문에

2가지 object로 구별해 진행하였다. Figure 4는 수집된 건물번호판의 양식이고 Figure 5는 라벨링이 된 이미지의 그림이다.

Figure 4. 다른 양식의 건물번호판

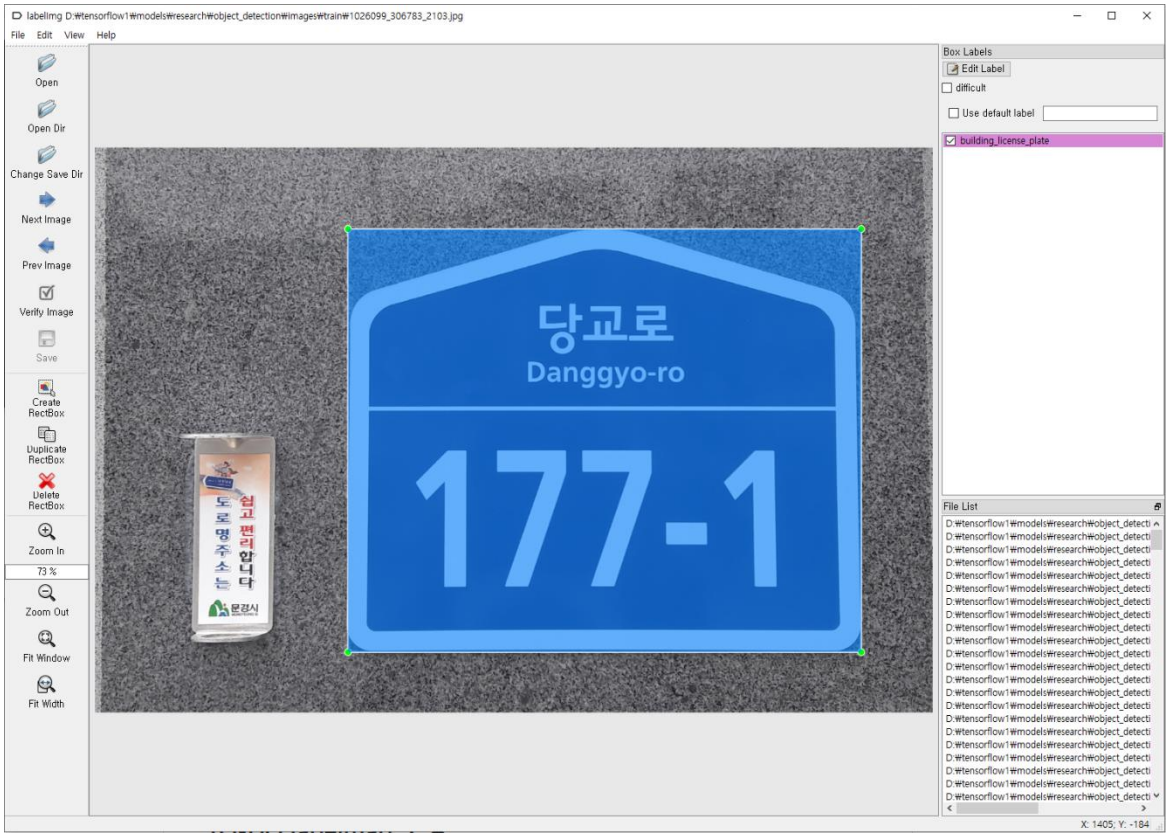

Figure 5. 이미지 라벨링

3.1.2 Training 실행

수집된 267장의 이미지에 대한 라벨링 작업 이후 Faster R-CNN Inception v2 모델을 사용해 학습을 진행하였다. 학습은 GPU와의 비교를 위한 CPU만을 사용한 학습 1회와 GPU를 사용한 학습 3회 도합 4회를 실시하였다. GPU의 학습은 객체 인식률을 올리기 위해 이미지 data를 수정하여 반복 실시하였다. 최종 결과본이 출력되는 Python 코드의 경우 이후 Tesserat OCR를 이용해 주소를 인식하는 작업을 위해 건물번호판이 인식될 경우 해당 객체를 crop하여 출력하도록 했다. 다음 표는 진행된 학습에 대한 시간, 횟수, 1개의 이미지당 걸린 평균 시간과 인식률을 보여준다. 인식률 테스트의 경우 테스트 이미지로 훈련에 이용된 사진이 아닌 44개의 이미지를 정해 진행했으며 44개의 이미지 중, 정확하게 객체를 인식한 경우만 성공한 것으로 계산했다.

학습방법	학습 횟수	학습 시간	이미지당 평균 학습 시간	인식률
CPU-Only	5492	6h 33min 25sec	4.2980sec	86.3%
GPU 1	45466	7h 25min 12sec	0.5875sec	90.9%
GPU 2	57835	8h 14min 27sec	0.5129sec	90.9%
GPU 3	64078	9h 13min 43sec	0.5184sec	88.6%

3.1.3 건물번호판 detector 훈련 결과

우선 가장 두드러지는 부분은 CPU-Only 학습과 GPU를 활용한 학습의 이미지당 평균 학습 시간이다. 최대 837%의 시간차이를 보였으며 GPU를 활용하는 학습이 CPU-Only 학습에 비해 압도적인 효율이 나타났다. 이는 기존 R-CNN 학습에서 GPU의 이점을 살리기 위해 RPN을 도입한 부분에서 속도적인 측면의 확연한 발전이 있었음을 시사한다. 그에 반해 학습횟수가 많게는 11배 정도까지 차이가 남에도 불구하고 CPU-Only 학습과 GPU를 활용한 학습의 인식률이 차이가 크게 나지 않았다. 이는 건물번호판의 양식이 어느정도 정형화되어 있고 특징이 확연한 객체라는 점에서 Obeject detector가 감지하기 비교적 쉬운 물체이기 때문일 것이다.

건물번호판 detector의 인식률을 올리기 위해 이미지 set을 바꿔가며 여러 번 학습을 해 테스트를 진행했지만 인식률 개선이 뚜렷하게 나타나지 않았다. 이는 테스트에 사용된 일부 이미지가 객체가 명확하게 나타나지 않는 원거리 사진이었기 때문이다. 4번의 학습으로 만들어진 각각의 detector에서 주로 탐지하지 못한 5개의 테스트 이미지가 건물번호판을 상당히 먼 거리에서 찍은 사진이었다. 이는 테스트용 이미지에 적합하지 않은 사진이었으며, 1개의 저화질 사진 역시 인

식이 잘 되지 않았다. 그 외의 사진들은 공통적으로 인식이 잘 되었다.

Figure 6. 건물번호판 detector의 실행 모습

Ⅳ. Tesseract OCR 주소 인식 시스템

4.1 이미지 전처리

관련 연구에서 이미 살펴보았듯이 Tesseract OCR의 고질적인 문제로 이미지의 품질이 떨어지거나 화질이 저하될 경우 글자 인식률이 현저하게 떨어지는 문제가 있다. 따라서 Tesseract OCR의 인식률을 올리기 위해서는 이미지 전처리 과정이 필수적이다.

4.1.1 Gray-Scale 변환

글자의 인식률을 올리기 위해 가장 먼저 처리되는 과정이 Gray-Scale 변환이다. 이는 기존의

컬러 이미지를 흑백 이미지로 전환해 명암의 차이가 더욱 드러나도록 해 글자 인식률이 올라가도록 하는 과정이다.

R, G, B는 각각 0~255의 값을 가지는데, Gray-Scale 변환 공식에 따라 변환할 경우 연산양과 연산 속도를 대폭 줄일 수 있다. 대표적인 Gray-Scale 변환 알고리즘은 Lightness, Average, Luminosity가 있다. Luminosity는 사람의 인식에 따라 색깔 별로 가중치를 다르게 적용시킨다. 따라서 가장 민감하게 인식하는 녹색의 가중치가 가장 높다. 다음은 각 알고리즘 공식이다.

$$Lightness = \frac{Max(R,G,B) + Min(R,G,B)}{2}$$

$$Average = \frac{R+G+B}{3}$$

$$Luminosity = 0.21R + 0.72G + 0.07B$$

Figure 7.1 원본 이미지

Figure 7.2 Gray-Scale로 변환한 이미지

4.1.2 블러(blur)처리(Smoothing)

이미지의 노이즈들은 글자를 인식할 때 방해가 된다. 노이즈를 제거하고 영상의 디테일을 낮추기 위해 사용하는 처리이다. 주로 사용되는 기법은 Gaussian Blur, Bilateral Filter가 있다.

Gaussian Blur는 이미지의 각 픽셀에 적용할 변환을 계산하기 위해 Gaussian function을 사용하는 이미지 블러링 필터 유형이다. 중심 화소에서의 거리에 따른 가중치를 적용한 MASK를 사용해서 영상을 부드럽게 만든다. 다만 영상의 경계선이 흐릿해진다는 단점이 있다.

Bilateral Filter는 Gaussian Blur처럼 가중치를 적용한 MASK를 사용하지만 결정적으로 다른 점은 두 픽셀과의 거리 뿐만 아니라 두 픽셀의 명암 값의 차이도 커널에 넣어서 가중치로 곱한다. 까라서 픽셀 값의 차이가 너무 크면 가중치가 0에 가까운 값이 되어 합쳐지지 않으므로 영역과 영역 사이의 경계선이 잘 보존된다.

Figure 8.1 Gaussian Blur

Figure 8.2 Bilateral Filter

4.1.3 이진화(Binarization)

이진화 작업은 임계값(Threshold)을 기준으로 임계값보다 낮은 값을 가지는 픽셀은 0, 높은 값을 가지는 픽셀은 255로 만든다. 이렇게 함으로써 영상에서 글자부분을 뚜렷하게 특징지어 나타낼 수 있으며, 영상의 크기 또한 줄어든다.

Figure 9.1 Gray-Scale 및 Binarization

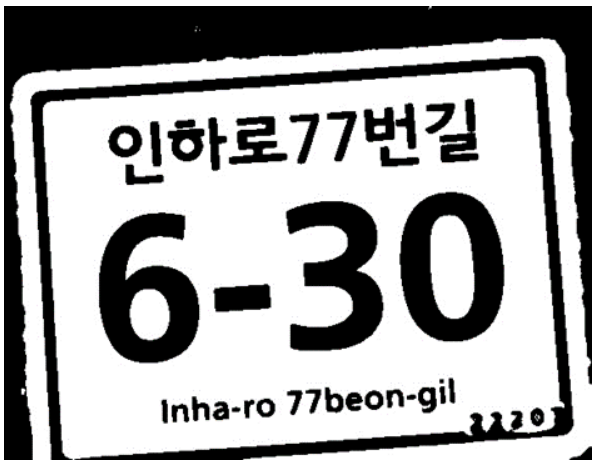

Figure 9.2 Bilateral Filter 및 Binarization

4.1.4 Morphology 연산

Morphology 연산은 Thresholding에서 얻어진 binary 이미지에서 노이즈나 너무 작은 feature들을 제거하는 연산이다. 크게 4가지의 방식이 존재한다.

Erosion 방식은 전경(Foreground)이 되는 이미지의 경계부분을 침식시켜서 배경(Background) 이미지로 전환한다. 원하지 않는 작은 노이즈를 제거할 수 있다. 커널 영역의 픽셀 값이 모두 1이면 1, 그렇지 않으면 0을 결과 이미지에 기록한다. 침식 연산을 적용하면 밝은 영역이 줄어들고 어두운 영역이 늘어난다.

Dilation 방식은 erosion의 반대 연산이다. 커널 영역의 픽셀 값이 모두 0이면 0, 그렇지 않으면 1을 결과 이미지에 기록한다. 침식 연산을 적용하면 어두운 영역이 줄어들고 밝은 영역이 늘어난다.

Opening 방식은 erosion을 적용한 뒤 dilation을 적용하는 것으로 영역이 점점 둥글게 되므로 점 잡음, 작은 물체, 돌기 등을 제거하는데 적합하다.

Closing 방식은 opening 방식의 반대로 dilation을 먼저 적용하고 erosion을 적용하는 것으로 전체적인 윤곽을 파악하는데 적합하다.

Figure 10.1 Gaussian Blur, Binarization 및 Morphology

(왼쪽 위부터 시계방향으로 Dilation, Erosion, Closing, Opening)

Figure 10.2 Bilateral Filter, Binarization 및 Morphology

(왼쪽 위부터 시계방향으로 Dilation, Erosion, Closing, Opening)

4.2 로테이션(Rotation) 작업

건물번호판 사진을 찍을 때, 반드시 똑바로 찍히기만 하지는 않을 것이다. 번호판이 기울어져 있을 때, 이미지를 회전시키지 않고 글자를 인식할 경우, 글자를 제대로 읽지 못한다. 따라서 글자를 인식시키기 위해 이미지를 rotate 하는 작업이 반드시 필요하다. 우선 글자 영역을 찾기 위해 번호판의 윤곽선(Contour)을 찾고, 윤곽선을 감싸는 사각형을 찾는다.

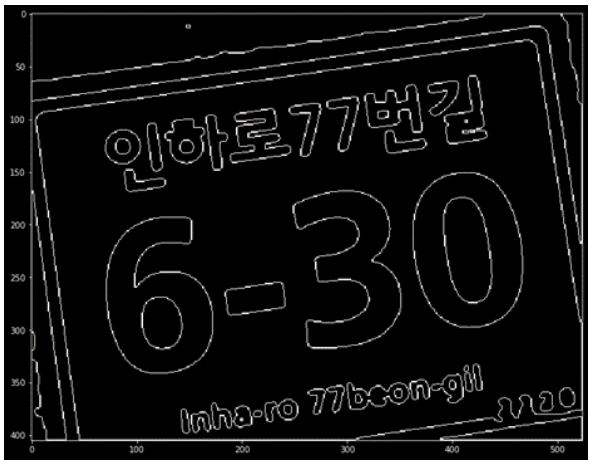

Figure 11.1 번호판의 윤곽선(Contour)

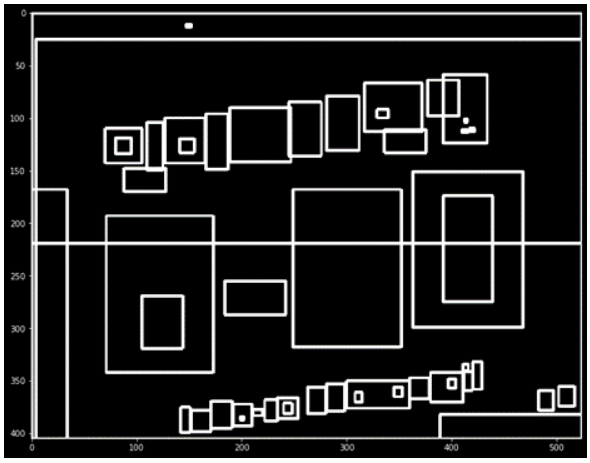

Figure 11.2 윤곽선을 감싸는 사각형

이때, 노이즈가 많은 이미지의 경우 글자 영역이 아닌 노이즈에도 윤곽선과 사각형이 그려진다. 건물번호판의 경우 글씨체의 비율이 일정하기 때문에 글자 영역을 둘러싼 사각형과 글자가 아닌 영역을 둘러싼 사각형을 구분할 수 있다. 사각형의 넓이, 비율 등 조건을 설정하여 글자 영역에 해당되지 않는 사각형들을 제외시킨다. 다음으로, 사각형의 중심 좌표가 이루는 각도, 거리 등을 계산하여 최종 글자영역 후보를 찾는다. 그 후에는 사각형의 중심 좌표가 이루는 각도를 계산하여 이미지를 기울어진 각도만큼 회전시킨다.

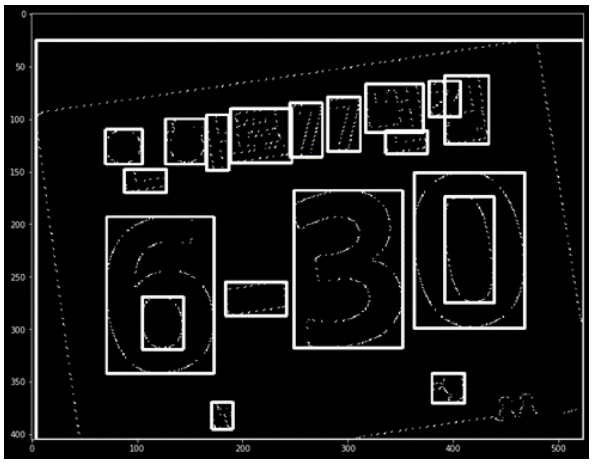

Figure 12.1 한글 및 숫자 영역을 감싸는 사각형

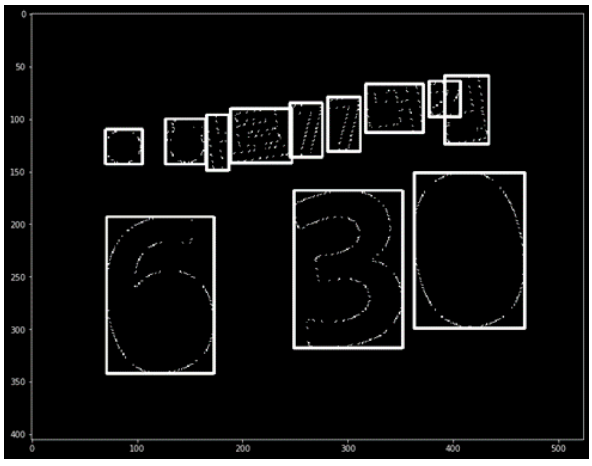

Figure 12.2 최종 글자 영역 후보

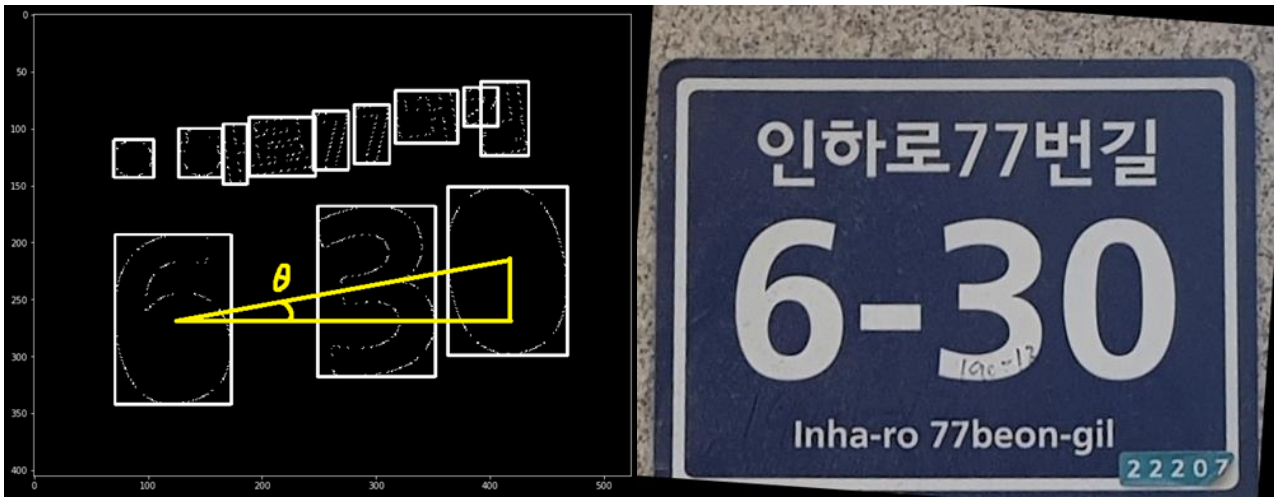

Figure 13 최종적으로 회전된 이미지

4.3 글자 인식

우리는 글자 인식을 위해 Google에서 공개한 Tesseract OCR을 사용할 것이다. Tesseract의 경우 이미지 전처리 과정과 로테이션 과정을 거치지 않으면 인식률이 매우 낮지만, 해당 과정을 모두 처리한 이후에는 글자를 인식하는 것을 확인할 수 있다.

```
text = pytesseract.image_to_string(erosion_bf, lang = 'kor+eng', config = '--psm 1 0c preserve_interword_space=1')
print(text)
plt.figure(figsize=(12, 10))
plt.imshow(img_ori)
```

```
<matplotlib.image.AxesImage at 0x23a02db9ac8>
```

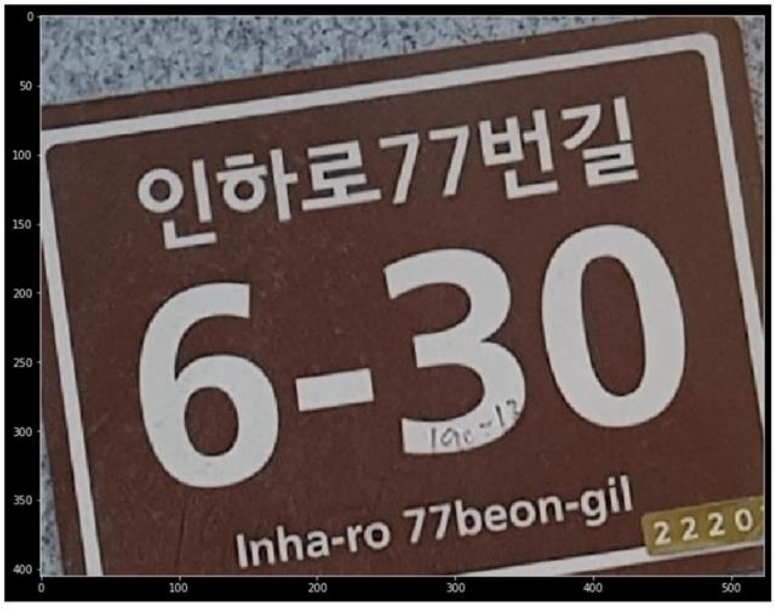

Figure 14 회전하지 않은 이미지의 글자 인식 결과

```
text = pytesseract.image_to_string(rotated_binary, lang = 'kor+eng', config = '--psm 1 0c preserve_interword_space=1')
print(text)
plt.figure(figsize=(12, 10))
plt.imshow(rotated_ori)
```

```
인하로77번길

6-30

Inha-ro 77beon-gil
232208
```

Figure 15 회전시킨 이미지의 글자 인식 결과

4.4 글자 인식률 개선을 위한 실험

지금까지 다양한 방법의 이미지 전처리 과정과 이미지 로테이션을 적용해 Tesseract 글자 인식을 진행해보았다. 본 프로젝트에서는 가장 좋은 문자 인식률을 보이는 이미지 처리과정이 무엇인지 확인하기 위해 각 이미지 전처리 과정에 따른 글자 인식 실험을 진행하였다. 일반적으로 이미지 전처리 과정에서 글자 인식률을 높이기 위해 Gray-Scale, Blur, 이진화 과정은 공통적으로 시행한다. 문제는 그 이후 이미지의 노이즈를 제거하는 Morphology 연산의 4가지 기법 중 어느 것이 가장 효율적인지 확인하는 것이었다. 따라서 실험은 Gray-Scale, Blur, 이진화 작업을 공통으로 진행하고 각 4가지 Morphology 방식의 차이에 따른 인식률을 확인해보는 방향으로 진행되었다.

실험은 6가지 이미지에 대해 각 10번씩 60회 진행되었으며, 이미지 전처리를 하지 않은 경우(None), Erosion 처리, Dilation 처리, Opening 처리, Closing 처리 5가지 경우에 대해서 글자 인식 테스트를 진행했다. 인식 기준은 한글 주소가 정확히 인식되는지에 대한 여부로 확인했다.

4.5 글자 인식률 개선을 위한 실험 결과

이미지 처리 방법	None	Erosion	Dilation	Opening	Closing
글자 인식률	1.67%	75%	56.67%	81.67%	63.33%

실험 결과 이미지 전처리를 하지 않은 이미지는 글자 인식률이 1.67%로 거의 불가능한 수준이었다. 이진화까지의 공통적인 이미지 전처리 후, 4가지 Morphology 방식의 차이에 따른 글자 인식률에서는 Erosion 방식과 Opening 방식이 다른 두 방식에 비해 상대적으로 인식률이 높았다. Erosion 방식과 Opening 방식의 공통점은 이진화 처리 이후 Erosion 처리를 먼저 진행한다는 점이다. 이에 반해 단일로 Dilation 처리를 하거나 Dilation 후 Erosion 처리를 하는 Closing 방식에서는 인식률이 약 60% 정도로 저조한 결과를 보였다. 결과에 따르면 Opening 방식이 Tesseract OCR을 사용해 글자 인식을 할 때 가장 좋은 방식으로 보여진다.

다만 그럼에도 불구하고, 가장 높은 글자의 인식률이 81.67%에 불과한 점은 매우 아쉽다. 이는 Tesseract의 한글 인식률이 저조하기 때문인데 이 때문에 현재 TensorFlow Object Detection API를 활용한 한글 인식 OCR 연구가 활발하게 이루어지고 있다.

Ⅴ. 결론

본 프로젝트에서는 현재 대한민국에서 건물번호판을 관리하는 방식의 불편함을 개선하고 deep learning 및 OCR 기술을 활용해 보다 편리한 자동 주소 인식 시스템을 구축하기 위해 건물번호판 detector 훈련과 자동 주소 인식 OCR 실험을 진행했다. 건물번호판 detector의 경우 최종 구현 결과에서 약 90% 정도의 정확도를 보인 것으로 나타났지만, 이는 테스트에 사용된 이미지가 적절치 못했던 까닭이다. 실제로 건물번호판을 인지하기 어려운 원거리에서 촬영된 이미지와 심한 저해상도 이미지를 제외하고는 99%의 인식률을 보였으며, 이는 건물번호판의 형태가 통일되어 있고, 객체의 특성상 특징을 찾기 쉽기 때문인 것으로 보인다. Tesseract OCR의 글자 인식률을 높이기 위한 실험에서는 글자 인식 전의 이미지 전처리를 하지 않을 경우 1.67%의 인식률을 보이는 만큼 이미지 전처리 과정은 필수적인 것으로 나타났으며, Gray-Scale, Blur, 이진화 작업 후 Opening 방식의 Morphology 처리를 하는 것이 81.67%의 글자 인식률을 보이며 가장 좋은

것으로 나타났다. 그러나 모든 처리 후에도 여전히 한글의 인식률이 81.67%를 보이는 점은 매우 아쉬우며, 한글의 글자 인식률을 더욱 높이기 위해서는 Tesseract OCR의 고질적인 한글 인식 문제 자체를 해결해야 하는 것으로 보인다. 따라서 TensorFlow Object Detection API등을 활용해 직접 한글 데이터 셋을 이용한 OCR 개발을 통해 한글의 OCR 인식률을 높여야 할 것으로 보인다.

Reference

[1] 원진영, 김창균, 최승현, 염세경, 강용신. (2018). 독거노인 위급 상황 탐지를 위한 TensorFlow Object Detection API 기반 자세 인식 절차. 한국정보과학회 학술발표논문집, 726-728.

[2] 이소연, 박지훈, 김대영. (2019). 효율적인 가로등 전력 관리를 위한 TensorFlow Object Detection API 기반 객체 인식 절차. 한국정보과학회 학술발표논문집, 1746-1748.

[3] 김병준. (2017). Faster R-CNN 기법을 이용한 옥외 영상 차량 번호판 검출 시스템.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems 28.

[5] 김원표, 이정근, 고영웅, 손철준, 진창규. (2018). Tesserect-OCR의 인식률 향상을 위한 병렬 전처리 모델. 한국정보과학회 학술발표논문집, 641-643.