기말 프로젝트 최종보고

사진 속의 텍스트를 추출하는 프로그램 OCR의 중간보고입니다.

원래 기존의 계획은 cv2.findContours()함수와 cv2.matchShape()의 매칭점수를 이용하여 사진의 텍스트를 라이브러리와 비교하여 텍스트를 추출하려고 했으나 인식률이 저조해서 두 이미지의 차이를 이용하는 bitwise_xor 연산을 사용하는 방법으로 변경했습니다.

라이브러리 제작은 유니코드를 이용하여 for 반복문으로 해당 유니코드의 문자를 출력후에 저장했습니다.

한글의 유니코드는 0xAC00(가) ~ 0xD79F(힟) 까지 총 11172개입니다

그 외 문장에 필요한 문자와 알파벳은 0x0021(!) 부터 0x007E(~) 까지 총 94개 입니다.

한글과 한글이 아닌 것을 구분하기 위해 맨 앞의 자릿수를 다르게 하기위해 5자리를 이용해서 0번~1만대는 한글, 2만대는 한글이 아닌 다른 글자들로 나누었습니다.

함수들을 설명하겠습니다

print_result() 함수는 현재 결과값 result의 문자열을 출력하는 함수입니다.

get_text_name() 함수는 파일명을 주면 파일의 유니코드의 문자로 바꾸어 주는 함수입니다.

라이브러리의 이름 형식은 폴더/폴더/XXXXX.png인데 여기서 변환에 필요한 부분은 뒤에서 5번째부터 9번째까지입니다. 하지만 슬라이싱할 때 마지막 에 하나 빼고 들어가므로 file_name[-9:-4]가 됩니다. 이후에는 2만보다 작은 한글과 2만보다 큰 한글이 아닌 글자들을 chr()을 이용사용해서 문자로 변환합니다.

draw_blueline() 함수는 texts에 있는 x좌표 y좌표 최대 최소 값을 이용하여 파란 사각형을 그려주는 함수입니다.

texts는 img이미지의 글자들의 영역(x좌표 y좌표 최대 최소 값)의 리스트입니다.

combine_Texts() 함수는 img의 도형들을 분리하고, 또 자음 모음 받침 나눠져 있는 한글이나 알파벳 소문자 i, j를 한 영역으로 묶어주는 함수입니다.

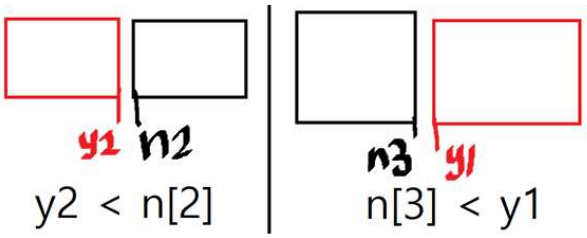
img의 도형들을 최소 단위로 컨투어를 이용하여 (컨투어의 x좌표 y좌표 최대 최소 값) 도형의 영역을 지정합니다.

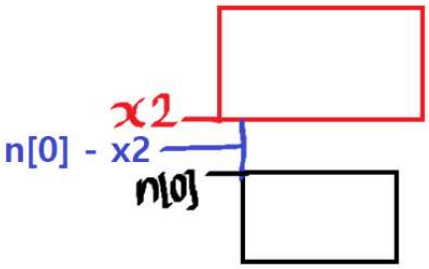
3조건에 의해 최소 단위의 도형들이 하나의 글자인지를 구별하여 영역을 합칩니다.

1. if (y2 < n[2]) | (y1 > n[3]) == False: / if n[0]-x2 <= space_size_x:

위 아래 조금 떨어진 글자를 합치는 조건식입니다.

두 조건을 중첩해서 사용했습니다. 붙이려는 모형들은 가로선상에 좌표가 겹치는 부분이 있어야 되기 때문에 겹치지 않을 때의 부정조건을 사용했고,





if n[0]-x2 <= space_size_x: 에서

길이의 제한을 두지 않으면 모든 가로선상 겹치는 모든 그림들이 합쳐져서 최대 공백의 크기를 나타내는 변수space_size_x을 사용했습니다.

컨투어의 특성상 findContours 함수를 사용하면 그림의 아래쪽의 오른쪽부터 생성됩니다.

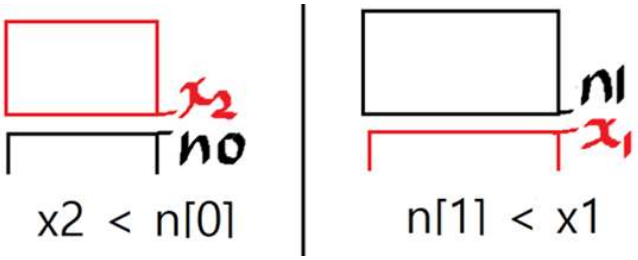
나중에 생성된 그림이 위쪽으로 두 그림 사이의 위치 차이는 기존 그림의 제일 큰 세로 위치 값 n[0]에서 현재 제일 작은 세로 값x2를 뺀 n[0] – x2가 됩니다.

2. if (x2 < n[0]) | (x1 > n[1]) == False:

if y1-n[3] <= space_size_y and n[2]-y2 <= space_size_y:

좌우 조금 떨어진 글자를 합치는 조건식입니다.

마찬가지로 중첩으로 사용했습니다.

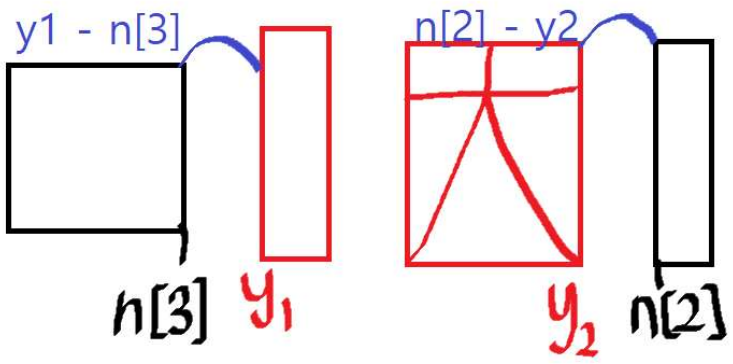


마찬가지로 같은 세로선상에 있어야 되므로 겹치지 않을 때의 부정조건을 사용했고,

if y1-n[3] <= space_size_y and n[2]-y2 <= space_size_y:

좌우 합치는 건 예외 케이스가 있어서 조건이 두 개입니다. 보통 findContours 함수를 사용하면

자음에 ㅏ ㅖ ㅣ ㅑ 같은 모음들 붙을 때 세로의 시작 좌표가 더 큰 자음들이 먼저 함수를 통과하는데 자음의 ㅊ 같은 경우는 상단부분이 길어서 세로의 시작 좌표가 작아서 나중에 통과하게 됩니다.

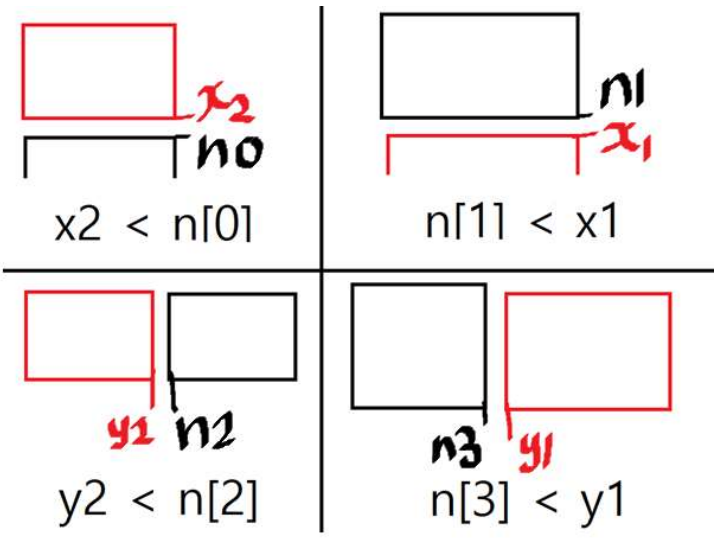


여기서 수식 사이에 and 인이유는

or로 하게 되면 수식에서 세로 공백의 최대 크기를 나타내는 변수 **space_size_y**가 양수이기 때문에 수식 음수인 경우 거리의 제한이 사라지기 때문에 모든 도형이 하나로 합쳐지고, 또 앞에 수식이 참인 경우, 뒤의 수식이 음수가 나오므로 뒤에 수식도 자동으로 참이다. 반대의 경우도 마찬가지여서 and를 사용했습니다.

3. if ((x2 < n[0]) | (x1 > n[1]) | (y2 < n[2]) | (y1 > n[3])) == False:

영역이 겹치는 글자를 합치는 조건식입니다.



영역이 겹치지 않을 때의 부정조건을 사용했습니다. 즉 영역이 조금이라도 겹칠 때 입니다.

sort_Texts() 함수는 combine_Texts()를 거친 texts리스트를 왼쪽에서 아래로 글을 읽는 순서대로 정렬해주는 함수입니다.

컨투어의 특성상 아래쪽에서 오른쪽으로 순서대로 리스트가 추가됩니다.

그래서 texts리스트에 순서를 뒤집는 reverse() 함수를 이용합니다.

그리고 임시 리스트 texts의 복사본 c_texts와 t_texts를 생성하고 texts를 모두 비웁니다

c_texts는 정렬이 완료된 요소들을 지우면서 정렬하기 위해, t_texts는 c_texts가 원소를 삭제하면 순서가 꼬이기 때문에 올바른 순서를 알기 위해 선언했습니다.

c_texts가 모두 지워질 때까지 무한 반복합니다.

반복문 내부에서 같은 줄에 있는 문자들을 temp_list에 저장하고, c_texts에서는 삭제를 합니다.

후에 temp_list를 x축 최소값(시작좌표) 순으로 정렬하여 기존 texts에 추가합니다

이후 temp_list를 비우고 다음줄을 진행합니다.

c_texts를 모두 비웠으면 is_sort(정렬여부)(정렬된 texts를 다시 정렬하지 않기 위한 변수)를 참으로 변경합니다.

check_Texts() 함수는 정렬된 texts를 라이브러리 파일들과 비교하는 함수입니다.

texts의 글자들을 모두 컨투어를 이용해 영역을 나타낸 것처럼 라이브러리 파일들 또한 컨투어를 이용하여 영역을 지정합니다.

texts의 한 글자씩 영역을 roi_file로 관심 영역으로 지정하고, 또 라이브러리 파일의 한글자 영역 roi_text로 관심 영역으로 지정한 후에 같은 크기로 맞추고 bitwise_xor연산을 이용하여 이미지를 생성합니다. xor연산으로 이미지가 겹치는(같은) 부분은 검은색으로 표현되고 다른 부분은 흰색으로 표현됩니다. 결과 이미지가 검정색에 가까울수록 두 이미지가 서로 유사함을 나타냅니다.

texts의 한 글자 한 글자를 라이브러리에 모든 파일과 xor연산을 반복합니다. 흰색 면적의 비율이 제일 적을 때의 라이브러리 파일 이름을 저장합니다.

get_text_name()를 이용하여 파일이름이 나타내는 유니코드의 문자를 알아내서 result 결과 문자열에 저장합니다.

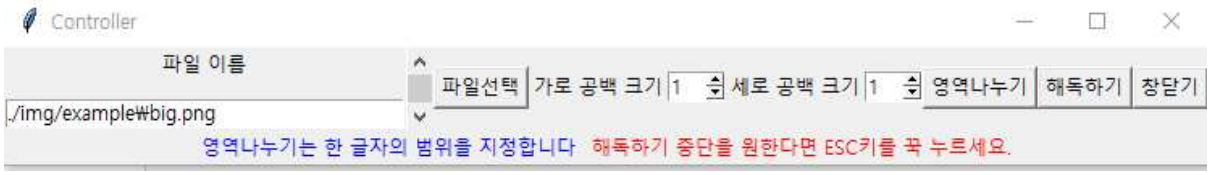
(저장하기 전에 띄어쓰기와 줄바꿈을 먼저 인식해서 result 결과 문자열에 저장합니다.

이때 파일의 인덱스가 0이 아닌 경우, 바로 전 인덱스의 리스트의 x축 최대값 좌표와 현재 x축 최소 값의 좌표의 차이를 이용하여 거리를 구한다음 일정 이상(**space_size_y의 3배**) 떨어져 있으면 띄어쓰기로 인식합니다. 또 (b_y2 > y1) 이전 문자의 최대 x좌표 > 현재 문자의 최소 x좌표인 경우 같은 선상에 있는 글자는 뒤에 나오는 글자가 x좌표의 값이 더 크지만 줄이 바뀌는 경우 후에 나오는 글자의 x좌표가 더 줄어들게 된다. 따라서 줄바꿈이 일어난 것이다.)

그리고 ESC를 누르면 해독이 중지되도록 하였습니다.

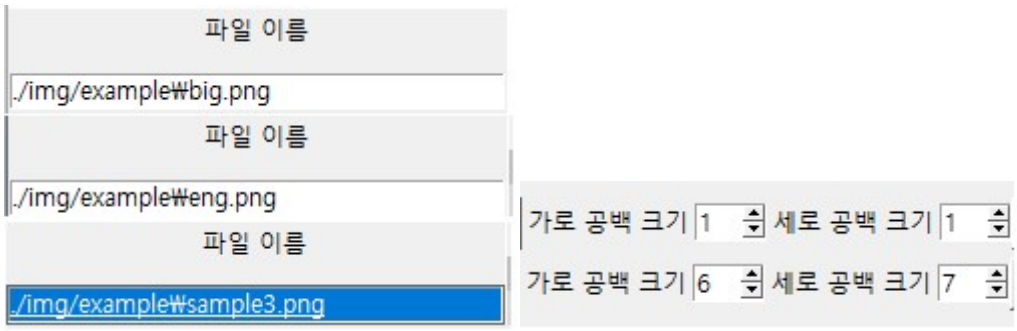
(중지하면 알림창으로 인해 소리가 울리고, 중지전까지의 결과를 출력합니다.)

(글자 하나에 대하여 라이브러리 탐색하는 반복문에 코드를 작성할 경우 waitKey()로 인해 속도가 너무 느려져서 반복문이 종료되고 다음 글자의 반복문이 시작되기전 구간에 추가했습니다)



새로 제작한 컨트롤러입니다. 스크롤을 이용하여 파일을 선택할 수 있습니다.

공백의 크기도 사용자가 지정할 수 있습니다. (이후 **space_size_x,y** 의 값이 됩니다.)

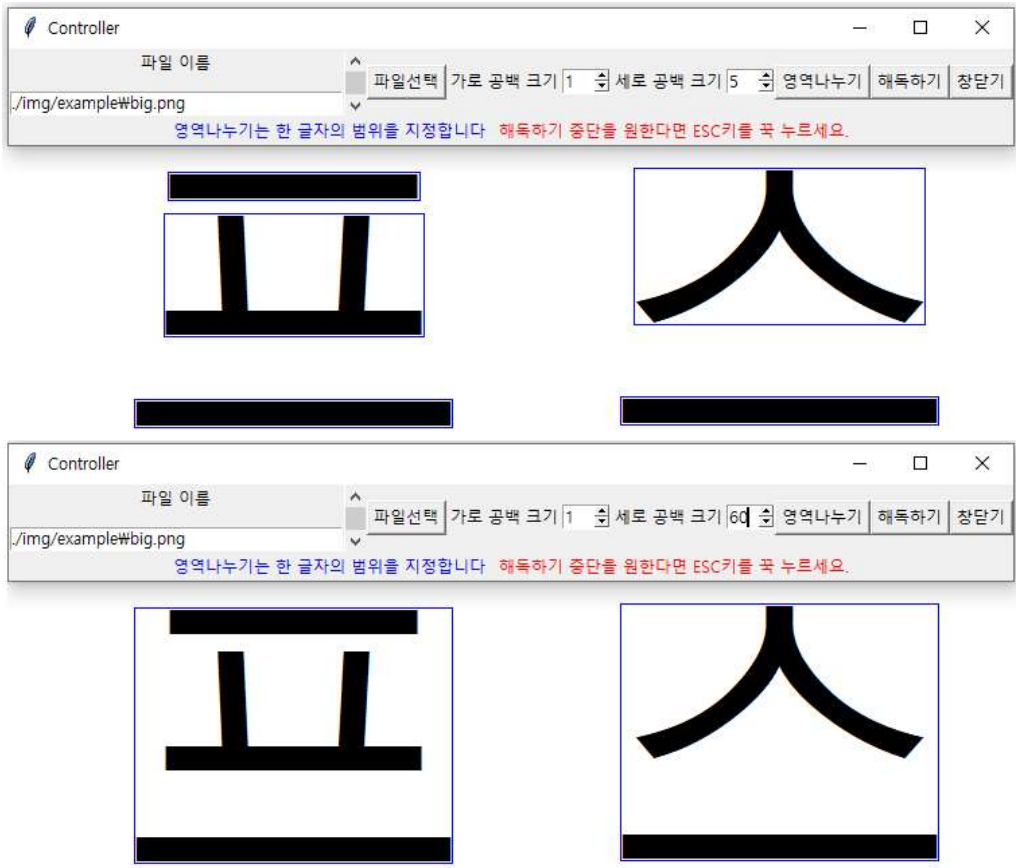


아래 함수는 컨트롤러의 작동을 위한 함수입니다.

choose_file()는 '파일선택' 버튼을 눌렀을 때 cv2이미지창에 대한 초기 세팅을 하는 함수입니다.

destroy()는 '창닫기' 버튼을 눌렀을 때 이미지 창을 닫기 위한 이벤트를 처리를 위한 함수입니다.

spinbox_change()는 '영역 나누기' 버튼을 눌렀을 때 사용자가 입력한 가로 공백 크기와 세로 공백 크기를 space_size_x, y(띄어쓰기와 도형 합칠 때 사용하는 변수)로 바꾸고 **combine_Texts()**를 이용하여 한글자로 취급하는 글자의 영역을 지정하는 함수입니다.



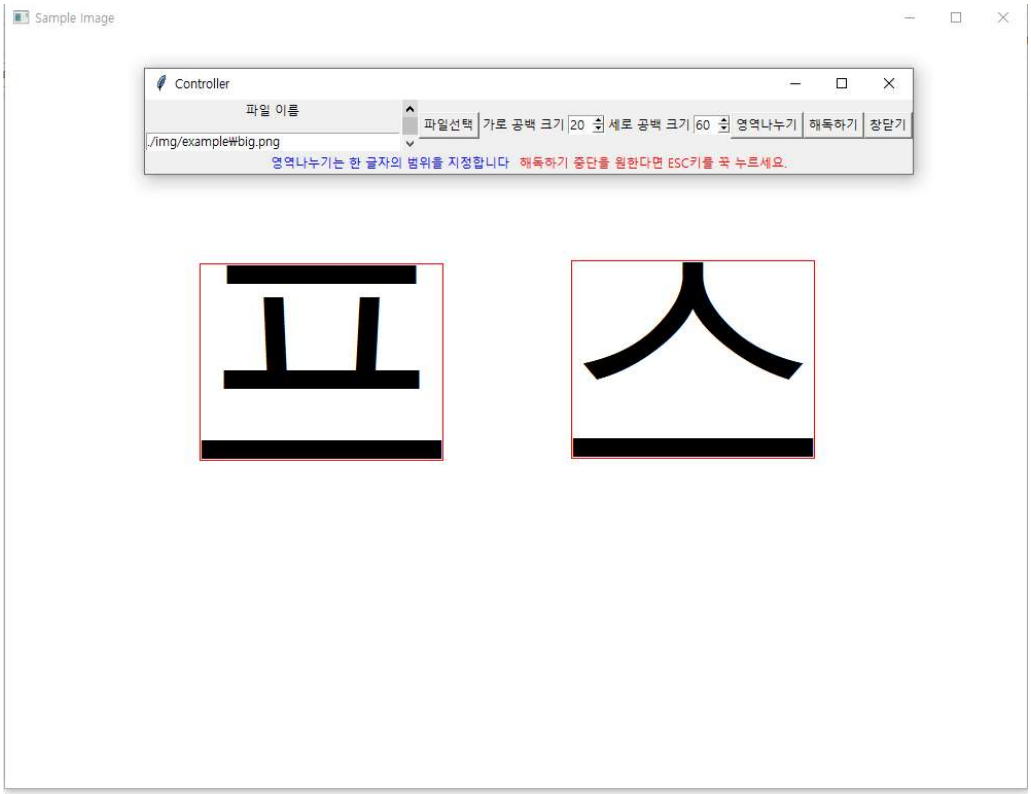
(세로 공백 크기를 조정함으로 한 글자의 영역을 사용자가 지정할 수 있습니다.)

decord()함수는 '해독하기'버튼의 이벤트 처리를 위한 함수입니다.

spinbox_change()으로 영역을 나누었다면 decord()는 texts를 **sort_Texts()** 이용하여 정렬하고 **check_Texts()**를 수행합니다.

결과입니다. (라이브러리는 맑은 고딕 글씨체를 사용했습니다.

./img/example₩big.png

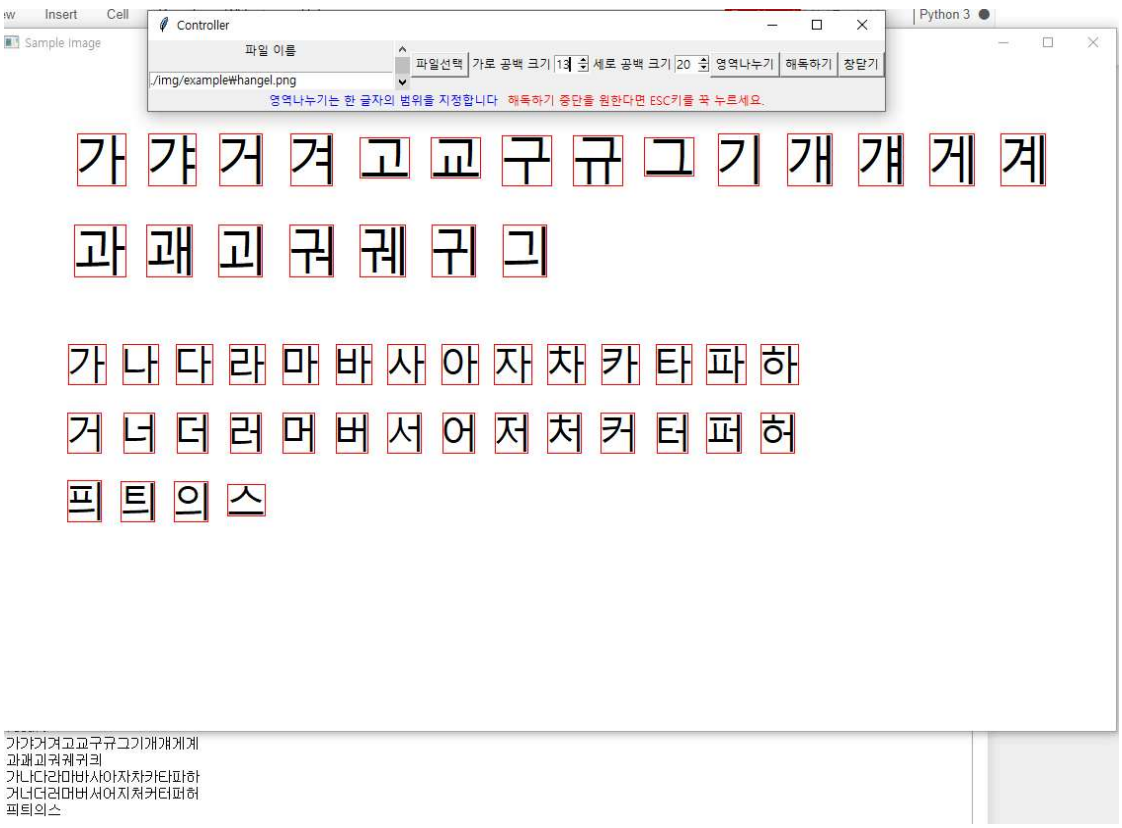



```
./img/example₩big.png 파일 선택완료
result >>
ㅍ ㅅ
```

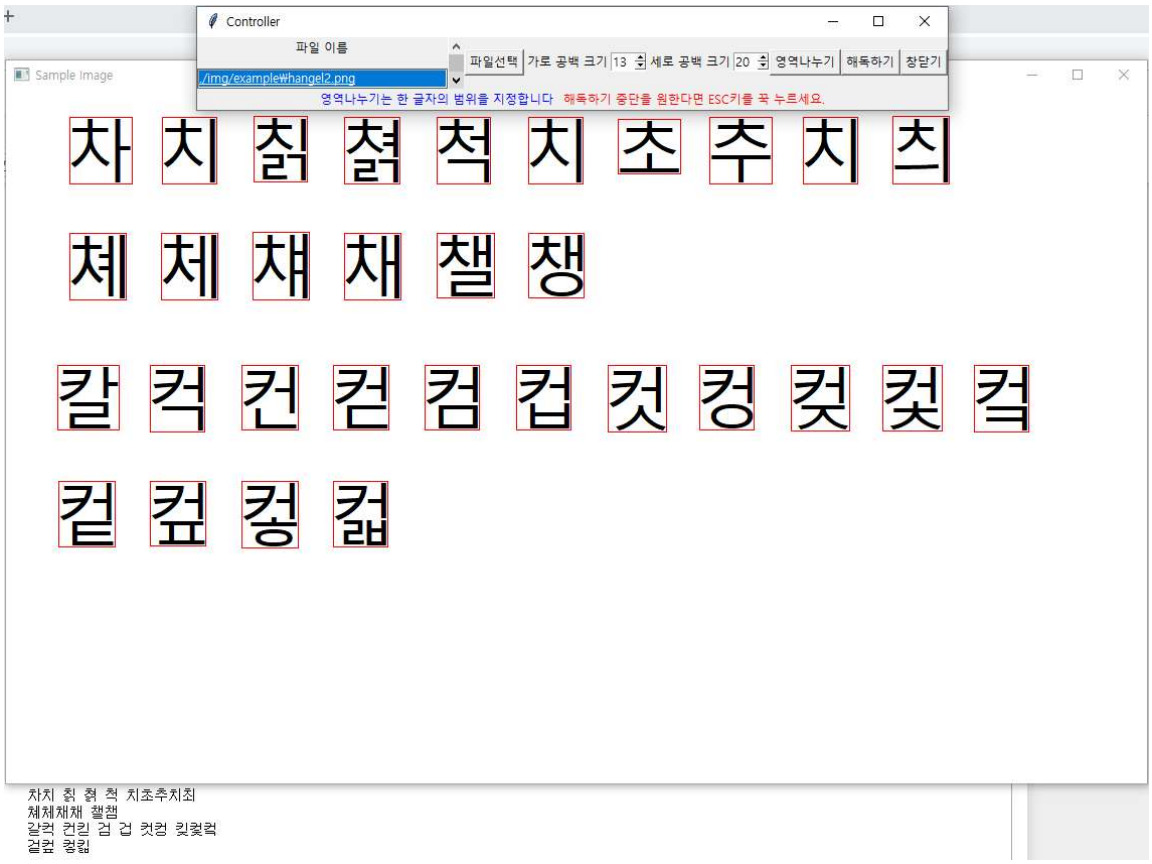
./img/example₩eng.png(바탕체)(글씨체가 다른 경우 인식률이 현저히 떨어집니다)


```
./img/example₩eng.png 파일 선택완료
result >>
AL LL FFLlh{]LLvN
UFULSTLv₩XY/
Bh C JO fFhi] LlmLO r LlrS
[LY₩Xy/
```

./img/example₩hangel.png

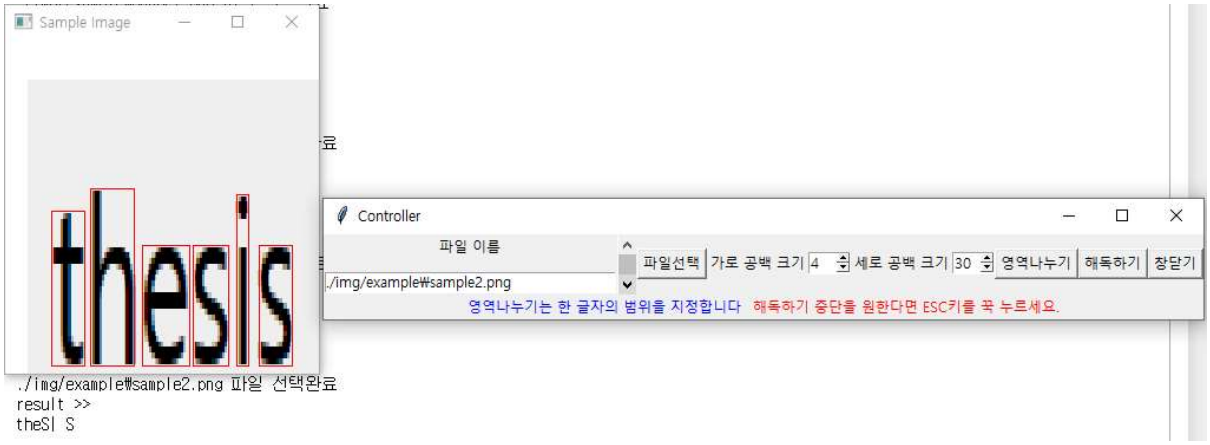


./img/example₩hangel2.png

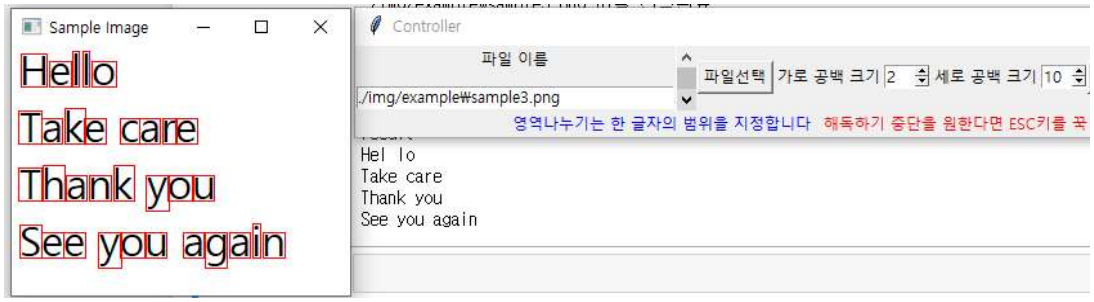


./img/example₩sample.png

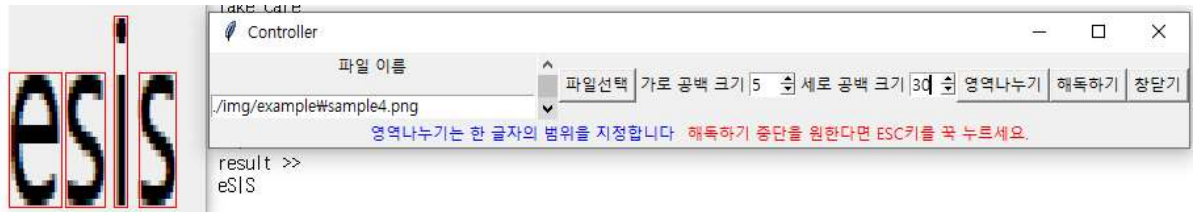
./img/example₩sample2.png



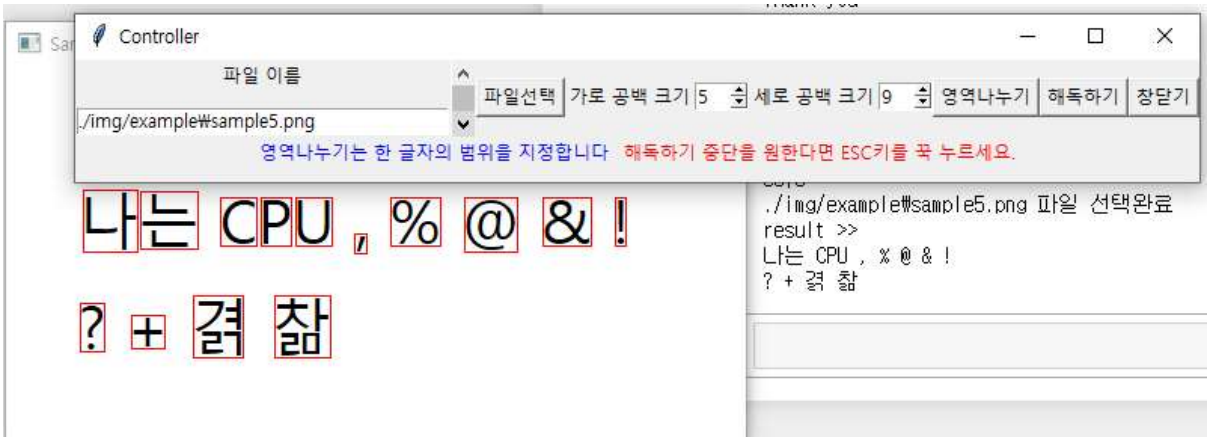
./img/example₩sample3.png



./img/example₩sample4.png



./img/example₩sample5.png



생각대로 잘 되지 않는 부분은 역시나 인식률입니다.

인식률에 관해서는 글씨체가 다르면 물론이고, 종종 정확하지 않습니다, 특히 소문자i에 대한 인식과 대문자S와 소문자s를 구별하는데 제한이 됩니다. 인식률을 높이려면 라이브러리의 이미지 파일을 크게 제작하면 될 것 같습니다.