<손글씨에서 작성된 문자 위치 범위 좌표 구하기>

1. 이미지 로드 및 전처리
   - 이미지 로드 : 손글씨 이미지 로드
   - 흑백 변환 : 이미지를 흑백으로 변환하여 글자 강조
2. 이진화 : 이미지를 흑백으로 변환하여 글자 부분 강조
3. 노이즈 제거 : 노이즈 제거 -> 윤곽선 검출 정확성 높임
4. 윤곽선 검출 : 각 문자의 경계를 찾음
5. 경계 상자 계산 : 각 윤곽선에 대해 경계 상자를 계산하여 문자 위치 파악

<코드>

```
import cv2

# 이미지 로드
image = cv2.imread('handwriting.jpg')
# 흑백 변환
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# 이진화 (흰색 배경에서 검정 글자를 강조)
_, binary_image = cv2.threshold(gray_image, 128, 255, cv2.THRESH_BINARY_INV)
# 커널 생성
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
# 노이즈 제거
cleaned_image = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)
# 윤곽선 검출
contours, _ = cv2.findContours(cleaned_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
# 각 문자에 대한 경계 상자 좌표 추출
positions = []
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    positions.append((x, y, w, h))  # (x, y, 너비, 높이)
    print(f'문자 위치: x={x}, y={y}, 너비={w}, 높이={h}')
```

<코드설명>

'handwriting.jpg' : 손글씨가 작성된 이미지 파일의 이름

cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) : 컬러 이미지를 흑백 이미지로

cv2.threshold() : 이미지의 픽셀 값이 특정 임계값(여기서는 128)보다 크면 흰색(255)으로, 작으면 검정색(0)으로 변환

THRESH_BINARY_INV : 이진화 후 색상을 반전시킴

- 첫 번째 인자: 흑백으로 변환된 이미지.
- 두 번째 인자: 임계값 (128).
- 세 번째 인자: 최대 값 (255).
- 네 번째 인자: 적용할 방법 (여기서는 이진 반전)

cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)): 3x3 크기의 직사각형 구조 요소를 만듦. 이 구조 요소는 모폴로지 연산에서 사용

cv2.morphologyEx() : 모폴로지 연산을 수행

MORPH_CLOSE : 작은 구멍이나 노이즈를 제거하는 데 유용

cv2.findContours() : 이진화된 이미지에서 윤곽선을 찾음

- 첫 번째 인자: 윤곽선을 찾을 이진 이미지
- 두 번째 인자: 윤곽선의 계층 구조를 결정하는 방법 (RETR_EXTERNAL: 외부 윤곽선만을 찾음)
- 세 번째 인자: 윤곽선을 압축하는 방법 (CHAIN_APPROX_SIMPLE: 단순한 윤곽선만 저장)

cv2.boundingRect(contour): 주어진 윤곽선에 대해 최소한의 경계 상자를 계산

반환값은 (x, y) 좌표(상자의 왼쪽 위 모서리)와 너비(w), 높이(h)

positions.append((x, y, w, h)): 각 문자의 위치를 저장하기 위해 리스트에 추가

<피드백>

- 노이즈 제거, 기울임을 종합적으로 볼 수 있는 OCR
- User study 에서 ai의 노력을 좀 덜어줄 필요
- 고정된 상황 (터치스크린에서 하겠다, 고정시켜서 하겠다)
- 물론 그래도 OCR 제대로 해야함

참고문헌: https://xiellehera.tistory.com/13

```python
import cv2
import numpy as np
import pytesseract

# 이미지 로드
image = cv2.imread('handwriting.jpg')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 이진화
_, binary_image = cv2.threshold(gray_image, 128, 255, cv2.THRESH_BINARY_INV)

# 노이즈 제거
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
cleaned_image = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)

# 윤곽선 검출
contours, _ = cv2.findContours(cleaned_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```python
# 각 윤곽선의 기울임 각도 계산
angles = []
for contour in contours:
    rect = cv2.minAreaRect(contour)
    angle = rect[2]
    if angle < -45:
        angle += 90
    angles.append(angle)

# 평균 각도 계산
average_angle = sum(angles) / len(angles)
print(f'평균 기울임 각도: {average_angle}')

# 기울임 보정
(h, w) = cleaned_image.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, average_angle, 1.0)
corrected_image = cv2.warpAffine(cleaned_image, M, (w, h))

# Tesseract 경로 설정
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# OCR 수행
recognized_text = pytesseract.image_to_string(corrected_image, lang='kor')
print("인식된 텍스트:", recognized_text)
```

Pytesseract: Tesseract OCR 엔진을 Python에서 사용할 수 있도록 해주는 라이브러리

<기울임 각도 계산>
cv2.minAreaRecr() 사용 : 최소 외접 사각형 찾고, 각도 계산
각도가 -45 이하인 경우, 90도를 더하여 올바른 각도 얻기
average_angle : 모든 각도의 평균을 계산하여 저장

<기울임 보정>
cleaned_image.shape[:2] : 이미지의 높이와 너비 가져옴
cv2.getRotationMatrix2D() : 회전 행렬을 생성 (이미지의 중심을 기준으로 평균 각도만큼 회전)
cv2.warpAffine() : 회전 행렬을 사용하여 이미지를 회전
corrected_image : 기울임이 보정된 이미지입니다.

Tesseract OCR : 설치된 경로를 지정
pytesseract.image_to_string() : 보정된 이미지에서 텍스트를 인식
lang='kor' : 한글 인식을 위한 설정

```
import cv2
import numpy as np
import pytesseract

# 이미지 로드
image = cv2.imread('handwriting.jpg')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 이진화
_, binary_image = cv2.threshold(gray_image, 128, 255, cv2.THRESH_BINARY_INV)

# 노이즈 제거
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
cleaned_image = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)

# 윤곽선 검출
contours, _ = cv2.findContours(cleaned_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# 각 윤곽선의 기울임 각도 계산
angles = []
for contour in contours:
    rect = cv2.minAreaRect(contour)
    angle = rect[2]
    if angle < -45:
        angle += 90
    angles.append(angle)

# 평균 각도 계산
average_angle = sum(angles) / len(angles)
print(f'평균 기울임 각도: {average_angle}')

# 기울임 보정
(h, w) = cleaned_image.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, average_angle, 1.0)
corrected_image = cv2.warpAffine(cleaned_image, M, (w, h))

# Tesseract 경로 설정
pytesseract.pytesseract.tesseract_cmd = r'C:₩Program Files₩Tesseract-OCR₩tesseract.exe'

# OCR 수행
recognized_text = pytesseract.image_to_string(corrected_image, lang='kor')
print("인식된 텍스트:", recognized_text)
```

- 이렇게 할 수 없음 : 왜냐? 한글자 or 한 자음, 모음은 작아서 구별 불가 -
이진화 -> 중심점 계산 -> 기울임 각도 계산 -> 이미지 회전

```
# 경계선에서 중심점 계산
for cnt in contours:
    M = cv2.moments(cnt)
    if M['m00'] != 0:
        cX = int(M['m10'] / M['m00'])
        cY = int(M['m01'] / M['m00'])
# 직선 피팅
for cnt in contours:
    [vx, vy, x, y] = cv2.fitLine(cnt, cv2.DIST_L2, 0, 0.01, 0.01)
    angle = np.arctan2(vy, vx) * (180.0 / np.pi)
# 회전 행렬 생성
M = cv2.getRotationMatrix2D((cX, cY), angle, 1.0)
rotated = cv2.warpAffine(image, M, image.shape[1::-1], flags=cv2.INTER_CUBIC)
```

cv2. Moments (cnt) : 경계선의 모멘트 계산 / 모멘트: 중심점 계산하는데 사용
If M['m00'] !=0 : 분모가 0이 아닌경우에만 계산
M['m00'] : 모멘트의 0차 모멘트, 경계선의 면적
cX & cY : 경계선의 중심점 계산

cv2. Fitline (cnt, cv2. DIST_L2, 0, 0.01, 0.01) : 경계선에 가장 잘 맞는 직선 찾음
vx, vy : 직선 방향 벡터 / x, y : 직선의 한 점
np.arctan2(vy, vx) : 방향벡터로 기울임 각도 계산

cv2.getRotationMatrix2D((cX, cY), angle, 1.0 : 회전 변환을 위한 행렬 생성
(cX, cY) : 회전 중심 / angle : 회전 각도 / 1.0 : 스케일
cv2.warpAffine(image, M, image.shape[1::-1], flags=cv2.INTER_CUBIC) : 이미지 회전
M은 : 회전 행렬 / image.shape[1::-1] : 회전 후 이미지의 크기 지정
flags=cv2.INTER_CUBIC : 보간 방법, 이미지 품질을 높임