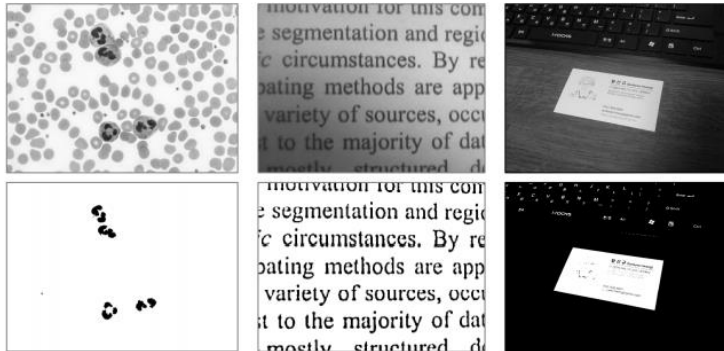


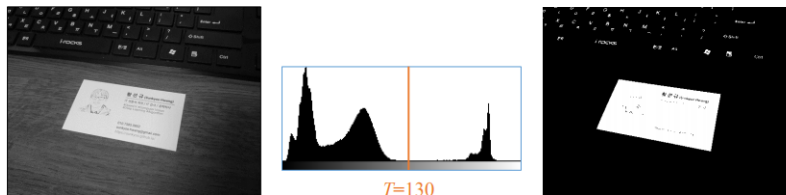
이진화



영상의 이진화(Binarization)이란?

: 영상의 픽셀 값을 0 또는 1(255)로 만드는 연산

- 배경 vs 객체
- 관심 영역 vs 비관심 영역



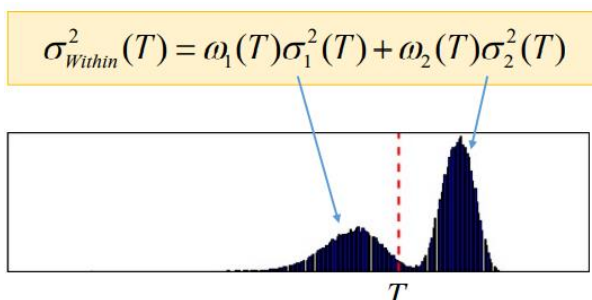
T(Threshold)를 이용하여 이진화!(cv2.Threshold)

threshold를 임의로 설정해줘야 하는데 이 과정에서 가장 적합한 threshold를 정하기가 쉽지 않다.

임계값(threshold) 결정 방법

자동 임계값 결정 방법 : **Otsu 방법**

- 입력 영상이 배경(background)와 객체(object) 두 개로 구성되어 있다고 가정(bimodal histogram)
- 두 픽셀 분포의 분산의 합이 최소가 되는 임계값을 선택
- 효과적인 수식 전개와 재귀 식을 이용하여 빠르게 임계값을 결정



cv2.ThresholdTypes : **cv2.THRESH_OTSU**(Otsu 알고리즘으로 임계값 결정)으로 이진화
예시)

```
th, src_bin = cv2.threshold(src_gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
```

앞의 0, 255는 무시가 된다. 자동으로 threshold를 설정해준다.

th는 OTSU 방법으로 자동 결정된 임계값이 들어간다.

객체 단위 분석

: 객체 위치 및 크기 정보, ROI 추출



레이블링(Connected Component Labeling)

- cv2.connectedComponent(), cv2.connectedComponentWithStats()

- 서로 연결되어 있는 객체 픽셀에 고유한 번호를 지정

- 각 객체의 바운딩 박스, 무게 중심 좌표로 함께 반환

외곽선 검출(Contour Tracing)

- cv2.findContours()


- 각 객체의 외곽선 좌표를 모두 검출

빠른 것은 레이블링이 외곽선 검출보다 더 빠르다.

따라서 위치나 크기를 찾고자 하면 레이블링을 사용하고, 구멍이나 다른 세부 사항을 파악하고 싶으면 외곽선 검출(Contour Tracing)을 하는 것이 좋다.

외곽선 검출의 mode, method는 다양하다.

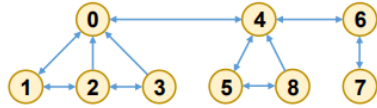
Mode

▪ RETR_EXTERNAL 

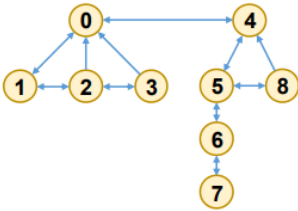
▪ RETR_LIST



▪ RETR_CCOMP

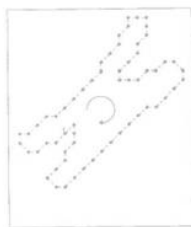


▪ RETR_TREE

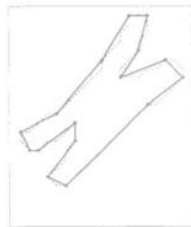


method:

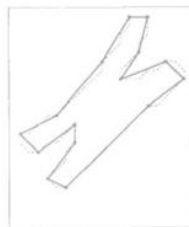
- ☐ CHAIN_APPROX_NONE : 근사화 없음
- ☐ CHAIN_APPROX_SIMPLE : 수직선, 수평선, 대각선에 대해 끝점만 사용하여 압축
- ☐ CHAIN_APPROX_TC89_L1 : Teh & Chin L1 근사화
- ☐ CHAIN_APPROX_TC89_KCOS : Teh & Chin k cos 근사화



contours



L1



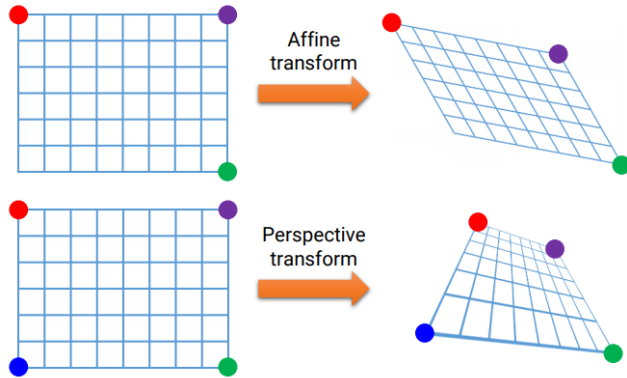
k cos

- 바운딩 박스(외곽선을 외접하여 둘러싸는 가장 작은 사각형) 구하기
- 바운딩 서클(외곽선을 외접하여 둘러싸는 가장 작은 원) 구하기
- 외곽선 근사화

: 끝점을 연결하여 margin을 설정해서 포함하고 있으면 없애고, 포함하지 않으면 없애지 않는다. 이런 방식으로 근사화를 한다.

영상의 기하학적 변환

- Affine, Perspective



투시 변환 행렬 구하기

```
cv2.getPerspectiveTransform(src, dst, solveMethod=None) -> retval
```

- src: 4개의 원본 좌표점. `numpy.ndarray`. `shape=(4, 2)`
e.g) `np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]], np.float32)`
- dst: 4개의 결과 좌표점. `numpy.ndarray`. `shape=(4, 2)`
- 반환값: 3x3 크기의 투시 변환 행렬

영상의 투시 변환

```
cv2.warpPerspective(src, M, dsize, dst=None, flags=None,  
borderMode=None, borderValue=None) -> dst
```

- src: 입력 영상
- M: 3x3 변환 행렬, 실수형
- dsize: 결과 영상의 크기. (0, 0)을 지정하면 src와 같은 크기.
- dst: 출력 영상
- flags: 보간법. 기본값은 `cv2.INTER_LINEAR`
- borderMode: 가장자리 픽셀 확장 방식.
- borderValue: `cv2.BORDER_CONSTANT`일 때 사용할 상수 값.
기본값은 0.

Tesseract

- 광학 문자 인식(OCR) 라이브러리
- <https://github.com/tesseract-ocr/tesseract>
- 1985~1994년에 휴렛 팩커드에서 개발 -> 2005년 오픈소스 -> 2006년부터 구글에서 관리
- 2018년 4.0이 발표되면서 LSTM(Long Short-Term Memory) 기반 OCR 엔진 및 모델이 추가
- 총 116개의 언어가 제공

- Apache License v2.0

[출처] https://github.com/sunkyoo/T_Academy_Python_OpenCV