파이썬

OpenCV 프로그래밍 3

명함 검출과 인식

- 일반적인 명함 사진의 조건
  - ✓ 명함은 흰색 배경에 검정색 글씨이다.
  - ✓ 명함은 충분히 크게 촬영되었다.
  - ✓ 명함은 각진 사각형 모양이다.

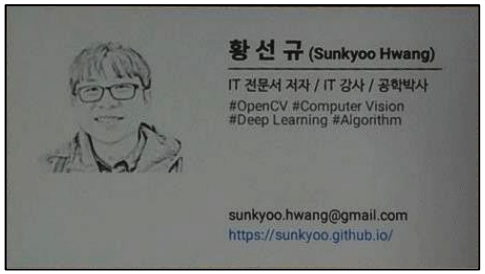
명함 검출과 인식

☐ 명함 검출 및 인식 진행 과정

이진화

외곽선 검출
&
다각형 근사화

투영 변환



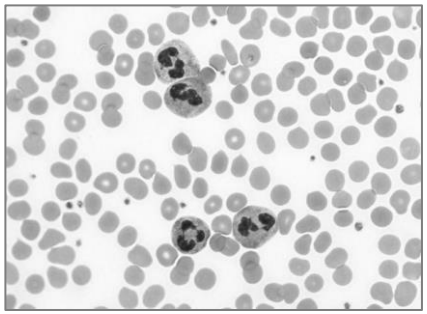
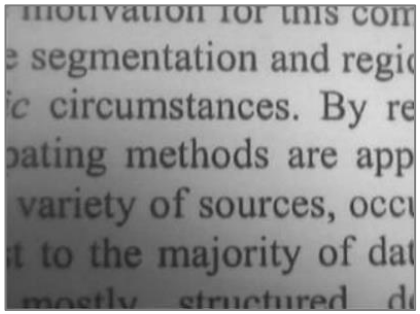
OCR

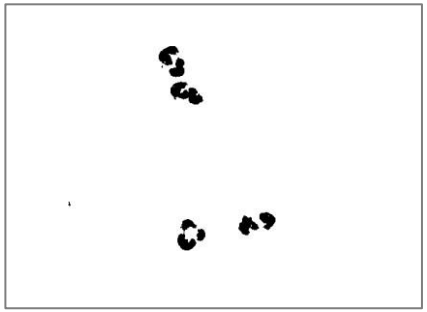
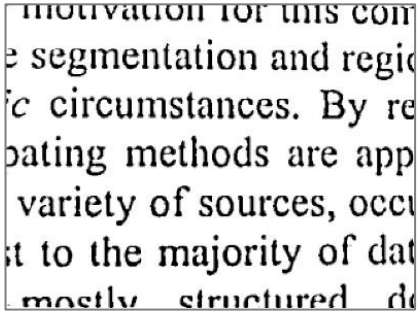
황 선 규
(Sunkyoo Hwang)
IT 전문서 저자 / IT 강사 / 공 학 박사

#OpenCV #Computer Vision #Deep Learning #Algorithm
…

이진화

- 영상의 이진화(Binarization)란?
  - ✓ 영상의 픽셀 값을 0 또는 1(255)로 만드는 연산
    - 배경(background) vs. 객체(object)
    - 관심 영역 vs. 비관심 영역

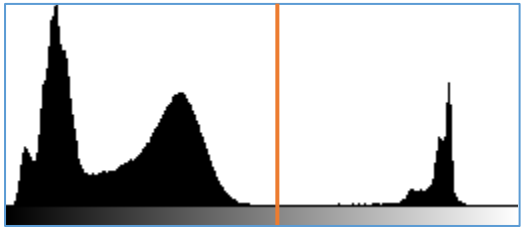




이진화

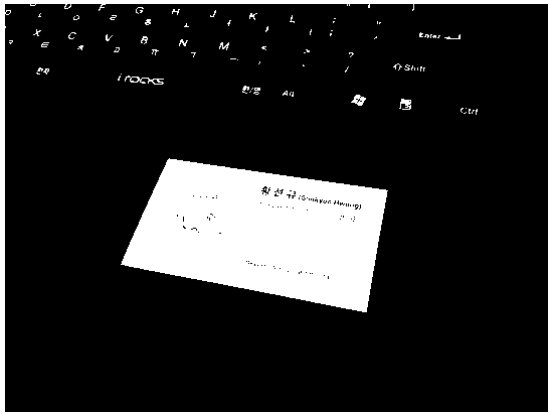
- 그레이스케일 영상의 이진화

$$g(x,y)=\begin{cases}0 & \text{if } f(x,y)\le T\\ 255 & \text{if } f(x,y)>T\end{cases}$$

- T : 임계값, 문턱치,threshold



T=130



이진화: OpenCV API

☐ 임계값 함수

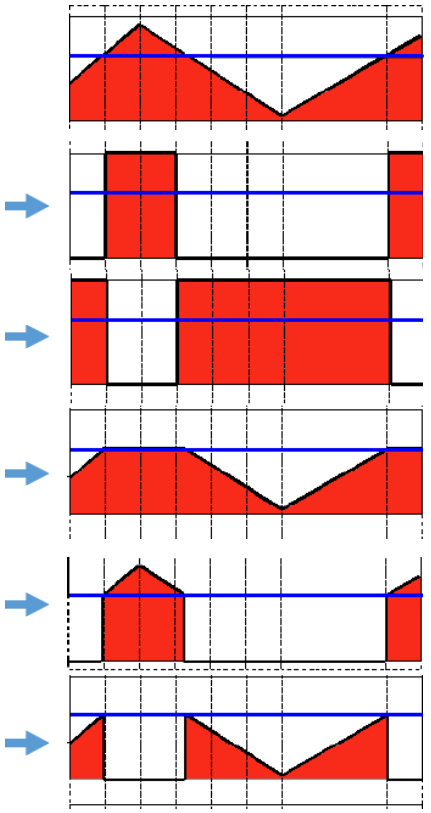
```
cv2.threshold(src, thresh, maxval, type, dst=None)
-> retval, dst
```

- ✓ src: 입력 영상. (다채널, 8비트 또는 32비트 실수형)
- ✓ thresh: 임계값
- ✓ maxval: THRESH_BINARY 또는 THRESH_BINARY_INV 방법을 사용할 때의 최댓값 지정
- ✓ type: 임계값에 의한 변환 함수 지정 또는 자동 임계값 설정 방법 지정 (cv.ThresholdTypes)
- ✓ retval: 사용된 임계값
- ✓ dst: (출력) 입계값 영상 (src와 동일 크기, 동일 타입)

이진화: OpenCV API

- cv2.ThresholdTypes

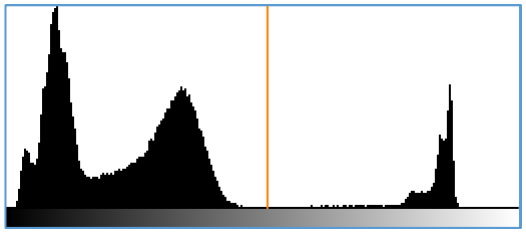
cv2.THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
cv2.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
cv2.THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
cv2.THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
cv2.THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
cv2.THRESH_MASK	
cv2.THRESH_OTSU	Otsu 알고리즘으로 임계값 결정
cv2.THRESH_TRIANGLE	삼각 알고리즘으로 임계값 결정



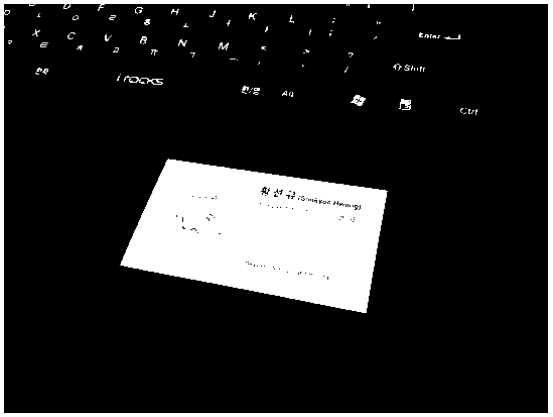
이진화: 임계값 결정 방법

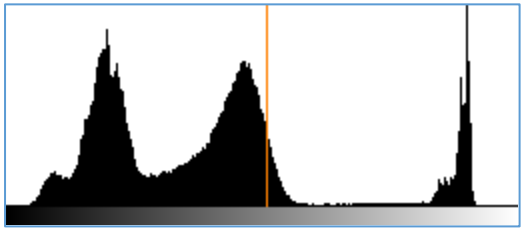
실습: threshold.py

□ 입력 영상의 밝기가 다른 경우

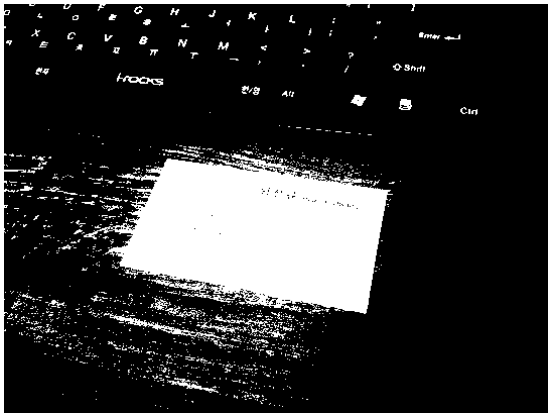


T=130





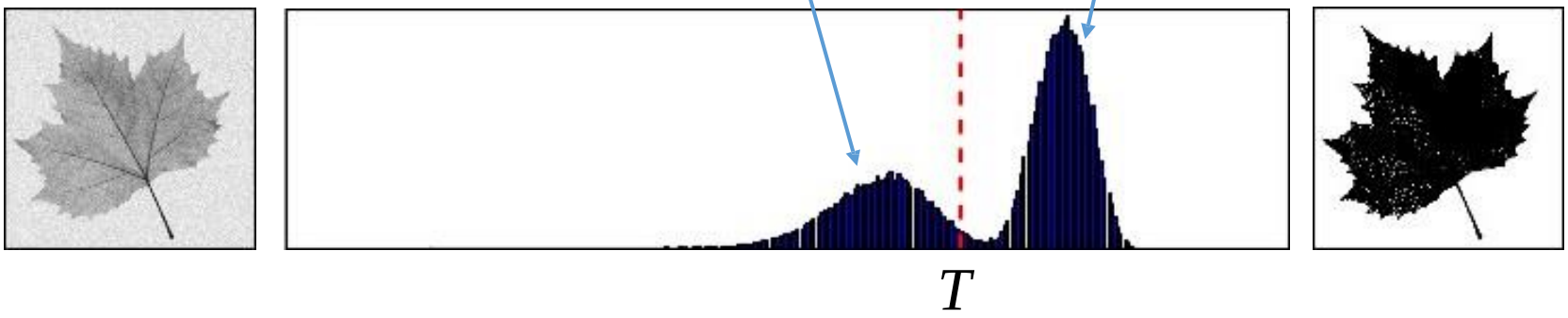
T=130



이진화: 임계값 결정 방법

- 자동 임계값 결정 방법: Otsu 방법
  - ✓ 입력 영상이 배경(background)과 객체(object) 두 개로 구성되어 있다고 가정 ❼ bimodal histogram
  - ✓ 두 픽셀 분포의 분산의 합이 최소가 되는 임계값을 선택 (Minimize within-class variance)
  - ✓ 효과적인 수식 전개와 재귀식을 이용하여 빠르게 임계값을 결정

$$\sigma^2_{Within}(T) = \omega_1(T)\sigma^2_1(T) + \omega_2(T)\sigma^2_2(T)$$



자동 임계값 결정 방법 – Otsu 방법

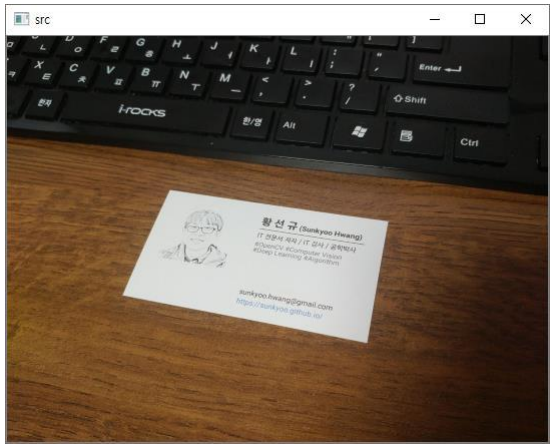
실습: otsu.py

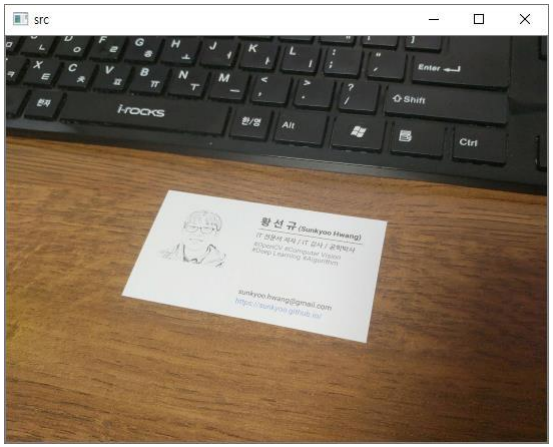
☐ Otsu 방법을 이용한 자동 이진화

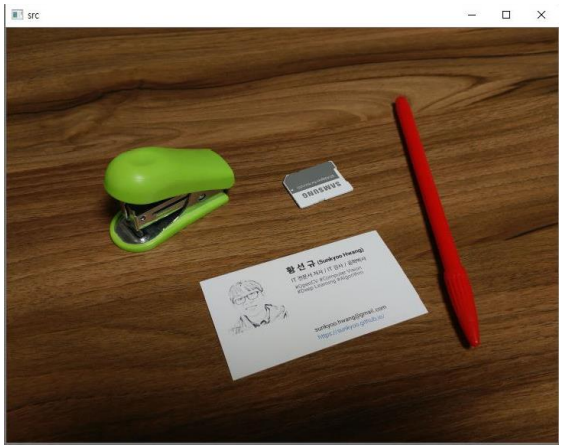
```
import cv2

src = cv2.imread('namecard1.jpg', cv2.IMREAD_GRAYSCALE)

th, src_bin = cv2.threshold(src, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
print('threshold:', th)

cv2.imshow('src', src)
cv2.imshow('dst', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

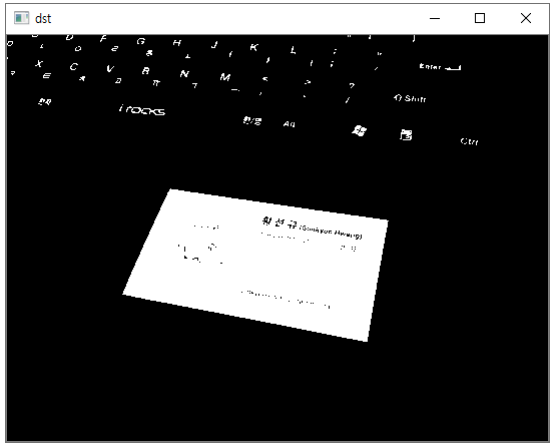
자동 임계값 결정 방법 – Otsu 방법

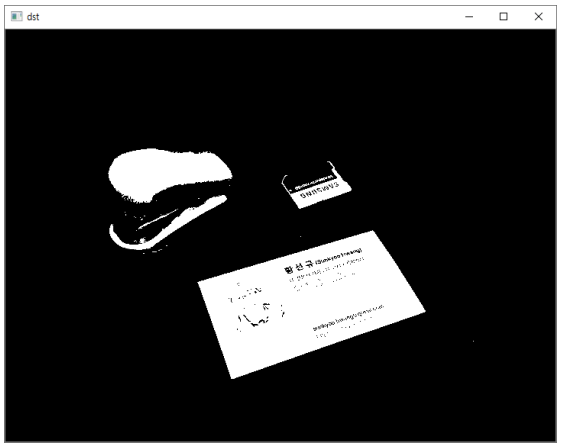
- Otsu 방법을 이용한 자동 이진화









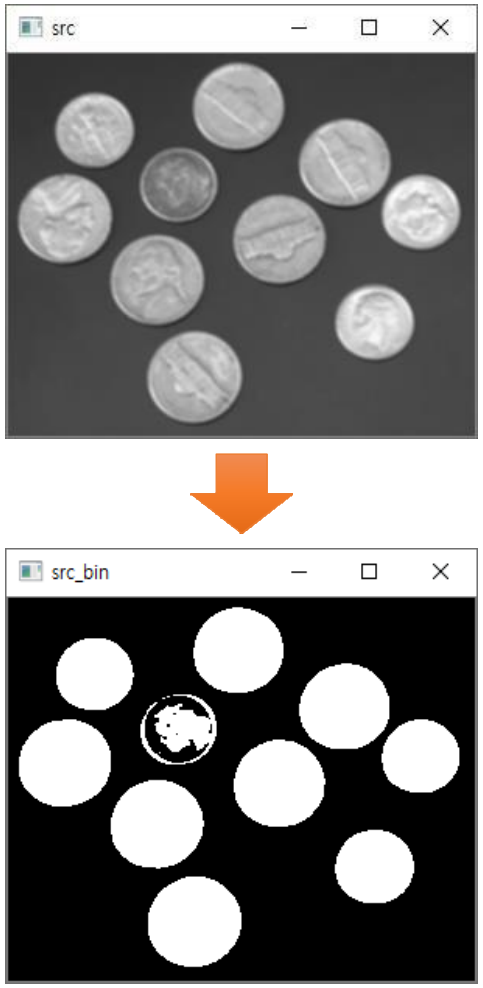


객체 단위 분석

- 객체 단위 분석
  - ✓ 흰색으로 표현된 객체를 분할하여 특징을 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

객체 단위 분석

실습: labeling.py



레이블링

- 영역 기반 모양 분석
- 레이블맵, 바운딩 박스, 픽셀 개수, 무게 중심 좌표를 반환

외곽선 검출

- 외곽선 기반 모양 분석
- 외곽선 점들의 좌표와 계층 구조 를 반환
- 다양한 외곽선 처리 함수에서 활용 가능 (근사화, 컨벡스헐 등)

외곽선 검출

- 외곽선 검출이란?
  - ✓ 객체의 외곽선 좌표를 모두 추출하는 작업
  - ✓ 바깥쪽 & 안쪽(홀) 외곽선
  - ✓ 외곽선의 계층 구조도 표현 가능
- 객체 하나의 외곽선 표현 방법
  - ✓ numpy.ndarray
  - ✓ shape=(K,1, 2), dtype=int32 (K는 외곽선 좌표 개수)
- 여러 객체의 외곽선 표현 방법
  - ✓ "객체 하나의 외곽선"을 원소로 갖는 리스트
  - ✓ 리스트 길이 = 외곽선 개수

외곽선 검출: OpenCV API

☐ 외곽선 검출

```
cv2.findContours(image, mode, method, contours=None,  hiera
        rchy=None, offset=None) -> contours, hierarchy
```

- ✓ image: 입력 영상. non-zero 픽셀을 객체로 간주함.
- ✓ mode: 외곽선 검출 모드
- ✓ method: 외곽선 근사화 방법
- ✓ contours: 검출된 외곽선 좌표. numpy.ndarray로 구성된 리스트.
  len(contours)=N,
  contours[i].shape=(K, 1, 2)
- ✓ hierarchy: 외곽선 계층 정보. numpy.ndarray.
  shape=(1, N, 4). hierarchy[0, i, 0~3]가 순서대로
  next, prev, child, parent 외곽선 인덱스를 가리킴.
  해당 외곽선이 없으면 -1을 가짐.

외곽선 검출: OpenCV API

- 외곽선 검출 (Con't)
  - ✓ mode:
    - **RETR_EXTERNAL**: 가장 바깥쪽 외곽선만 검출 (hierarchy[i][2]=hierarchy[i][3]=-1)
    - **RETR_LIST**: 계층 관계없이 모든 외곽선 검출 (hierarchy[i][2]=hierarchy[i][3]=-1)

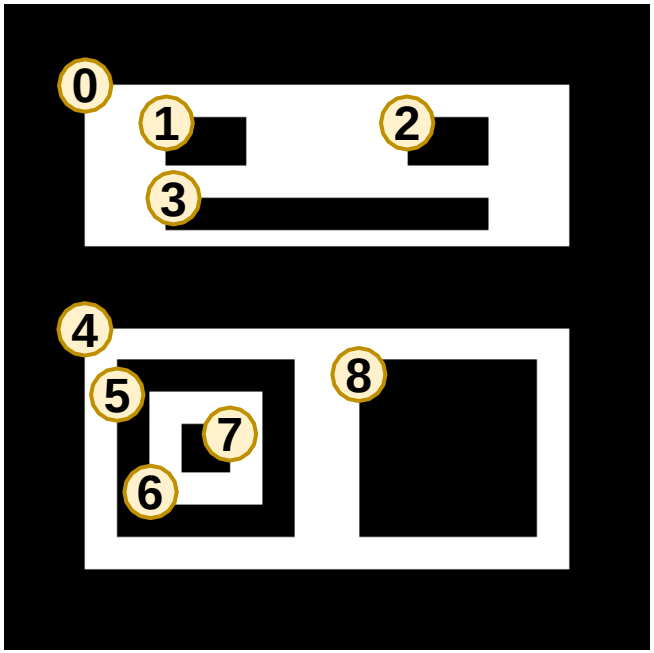
    } 계층 정보 X

    - **RETR_CCOMP**: 2레벨 계층 구조로 외곽선 검출 상위 레벨은 (흰색) 객체 외곽선, 하위 레벨은 (검정색) 구멍(hole) 외곽선.
    - **RETR_TREE**: 계층적 트리 구조로 모든 외곽선 검출

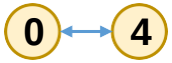
    } 계층 정보 O

외곽선 검출: OpenCV API

- 외곽선 검출 (Con't)
  - ✓ mode:

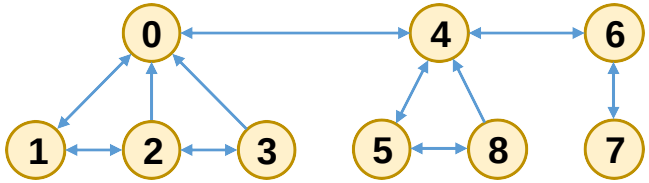


- RETR_EXTERNAL

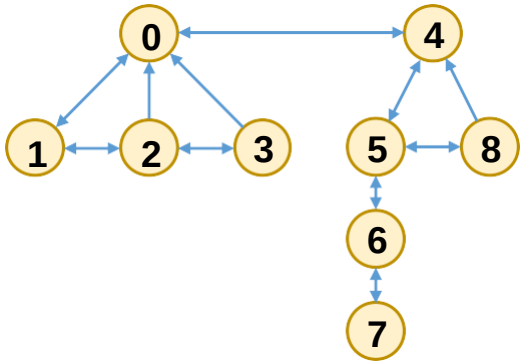


- RETR_LIST

- RETR_CCOMP

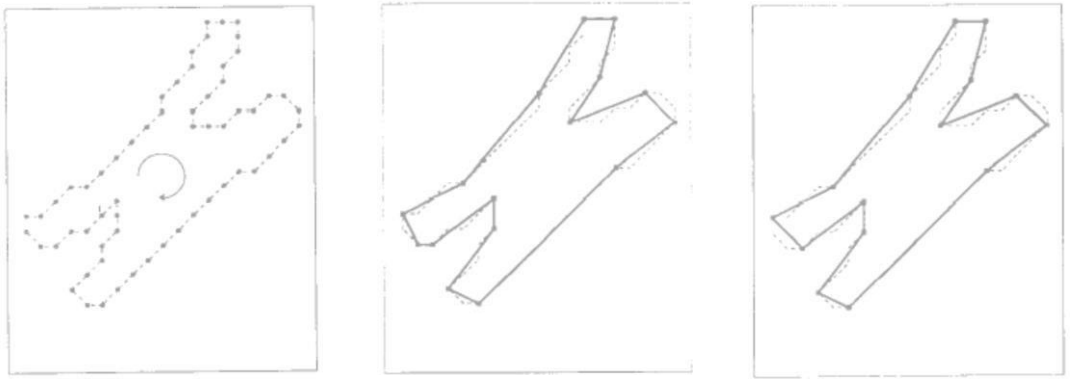


- RETR_TREE



외곽선 검출: OpenCV API

- 외곽선 검출 (Con't)
  - ✓ 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

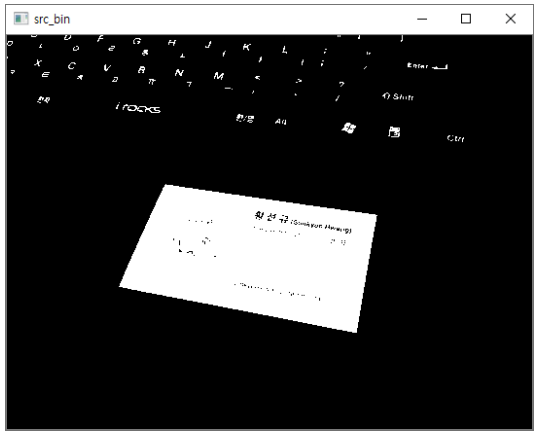
외곽선 검출

실습: contours.py

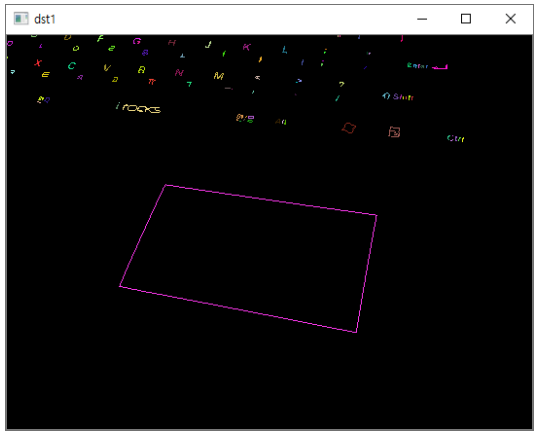
```
src = cv2.imread('namecard1.jpg', cv2.IMREAD_GRAYSCALE)

h, w = src.shape[:2]
dst1 = np.zeros((h, w, 3), np.uint8)
dst2 = np.zeros((h, w, 3), np.uint8)

# 이진화
_, src_bin = cv2.threshold(src, 0, 255, cv2.THRESH_OTSU)

# 외곽선 검출
contours1, _ = cv2.findContours(src_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
contours2, _ = cv2.findContours(src_bin, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)

for i in range(len(contours1)):
    c = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    cv2.drawContours(dst1, contours1, i, c, 1)

for i in range(len(contours2)):
    c = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    cv2.drawContours(dst2, contours2, i, c, 1)

...
```

외곽선 검출

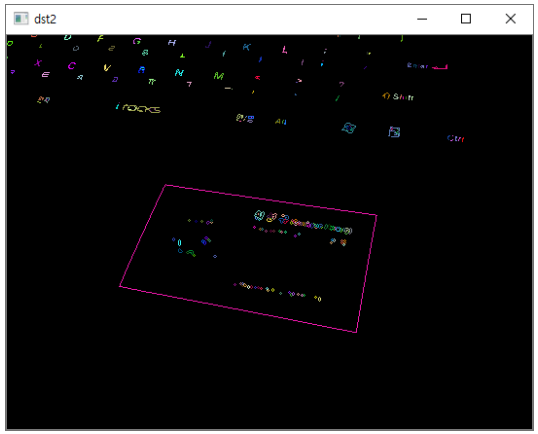
입력 영상



이진화



RETR_EXTERNAL



RETR_LIST

외곽선 관련 OpenCV API

면적 구하기

```
cv2.contourArea(contour, oriented=None) -> retval
```

- ✓ contour: 외곽선 좌표. numpy.ndarray. shape=(K, 1, 2)
- ✓ oriented: True이면 외곽선 진행 방향에 따라 부호 있는 면적을 반환
- ✓ retval: 외곽선으로 구성된 면적

외곽선 길이 구하기

```
cv2.arcLength(curve, closed) -> retval
```

- ✓ curve: 외곽선 좌표. numpy.ndarray. shape=(K, 1, 2)
- ✓ closed: True이면 폐곡선으로 간주
- ✓ retval: 외곽선 길이

외곽선 관련 OpenCV API

- 바운딩 박스(외곽선을 외접하여 둘러싸는 가장 작은 사각형) 구하기

```
cv2.boundingRect(array) -> retval
```

- ✓ array: 외곽선 좌표. numpy.ndarray. shape=(K, 1, 2)
- ✓ retval: 사각형. (x, y, w, h)

- 바운딩 서클(외곽선을 외접하여 둘러싸는 가장 작은 원) 구하기

```
cv2.minEnclosingCircle(points) -> center, radius
```

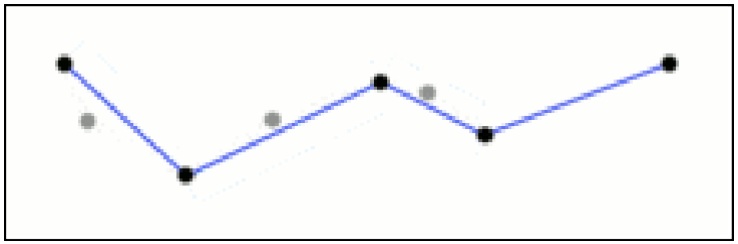
- ✓ points: 외곽선 좌표. numpy.ndarray. shape=(K, 1, 2)
- ✓ center: 바운딩 서클 중심 좌표 (x, y)
- ✓ radius: 바운딩 서클 반지름

외곽선 관련 OpenCV API

☐ 외곽선 근사화

```
cv2.approxPolyDP(curve, epsilon, closed, approxCurve=None)
-> approxCurve
```

- ✓ curve: 입력 곡선 좌표. numpy.ndarray. shape=(K, 1,2).
- ✓ epsilon: 근사화 정밀도 조절. 입력 곡선과 근사화 곡선 간의 최대 거리. e.g) (외곽선 전체 길이) * 0.02
- ✓ closed: True를 전달하면 폐곡선으로 간주
- ✓ approxCurve: 근사화된 곡선 좌표. numpy.ndarray. shape=(K', 1,2)



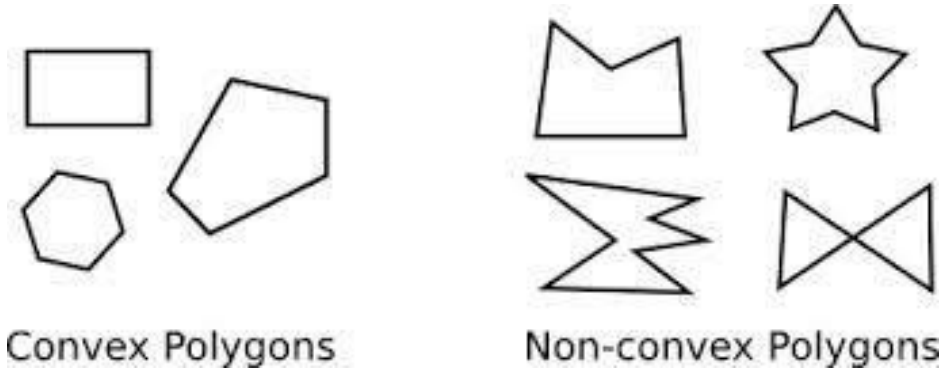
https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

외곽선 관련 OpenCV API

☐ 컨벡스(Convex) 검사

```
cv2.isContourConvex(contour) -> retval
```

- ✓ contour: 입력 곡선 좌표. numpy.ndarray. shape=(K,1, 2).
- ✓ retval: 컨벡스이면 True, 아니면 False.

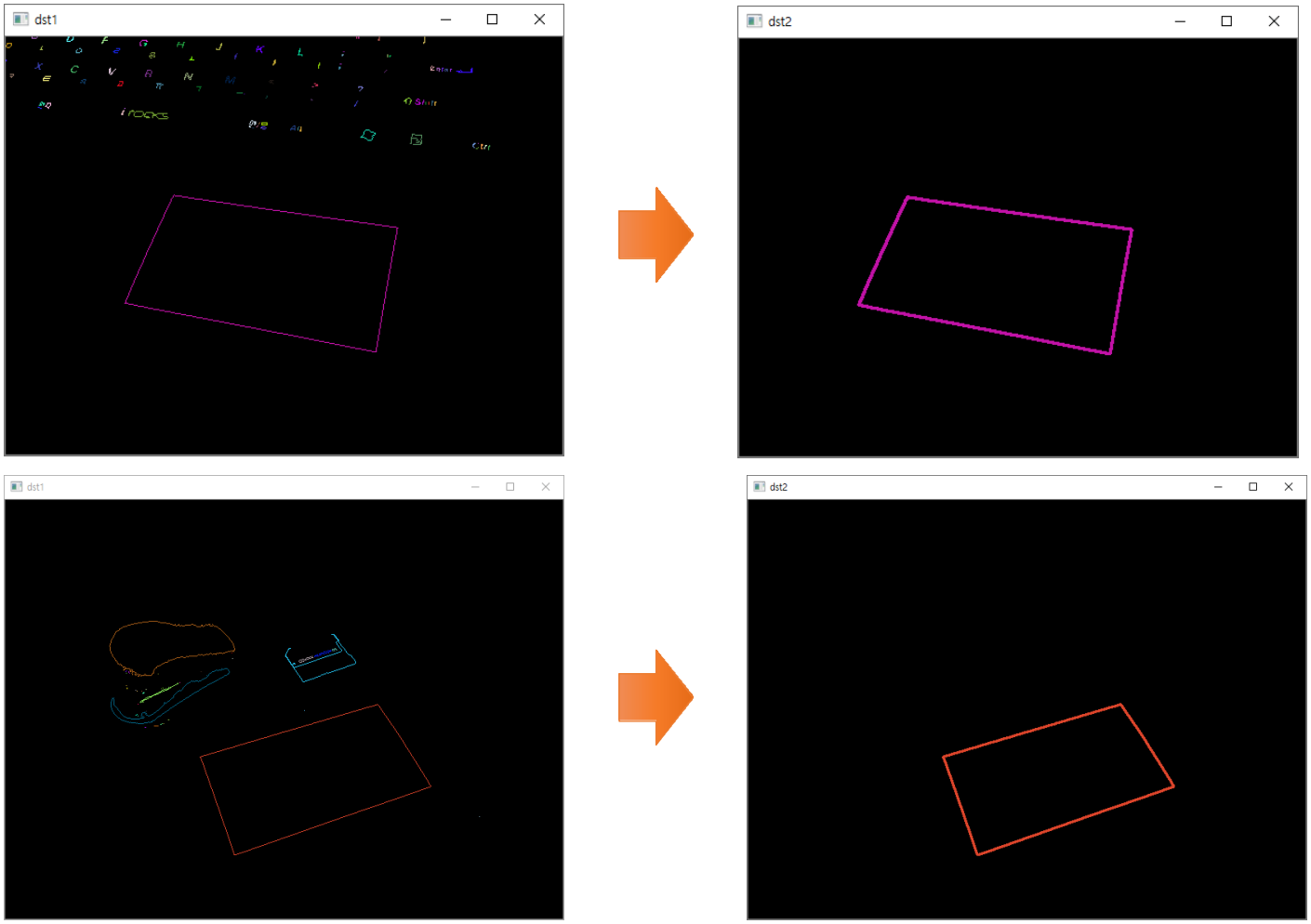


외곽선을 이용한 명함 검출

실습: contours2.py

```
src = cv2.imread('namecard1.jpg')
src_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

h, w = src.shape[:2]
dst1 = np.zeros((h, w, 3), np.uint8)
dst2 = np.zeros((h, w, 3), np.uint8)

_, src_bin = cv2.threshold(src_gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
contours, _ = cv2.findContours(src_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE
)

for i in range(len(contours)):
    pts = contours[i]
    if (cv2.contourArea(pts) < 1000):
        continue
    approx = cv2.approxPolyDP(pts, cv2.arcLength(pts, True)*0.02, True)
    if not cv2.isContourConvex(approx):
        continue

    if len(approx) == 4:
        cv2.drawContours(dst2, contours, i, c, 2)
```

외곽선을 이용한 명함 검출



영상의 기하학적 변환

- 영상의 기하학적 변환(geometric transformation)이란?
  - ✓ 영상을 구성하는 픽셀의 배치 구조를 변경함으로써 전체 영상의 모양을 바꾸는 작업
  - ✓ Image registration, removal of geometric distortion, etc.

영상의 기하학적 변환

Affine Transform

Translation

Shear

Scaling

Rotation

Combinations

Original

Perspective Transform

2x3 matrix

3x3 matrix

어파인 변환과 투시 변환

☐ 어파인 변환(Affine Transform)

✓ 6DOF

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

☐ 투시 변환(Perspective Transform)

✓ 8DOF

$$\begin{pmatrix} wx' \\ wy' \\ w \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

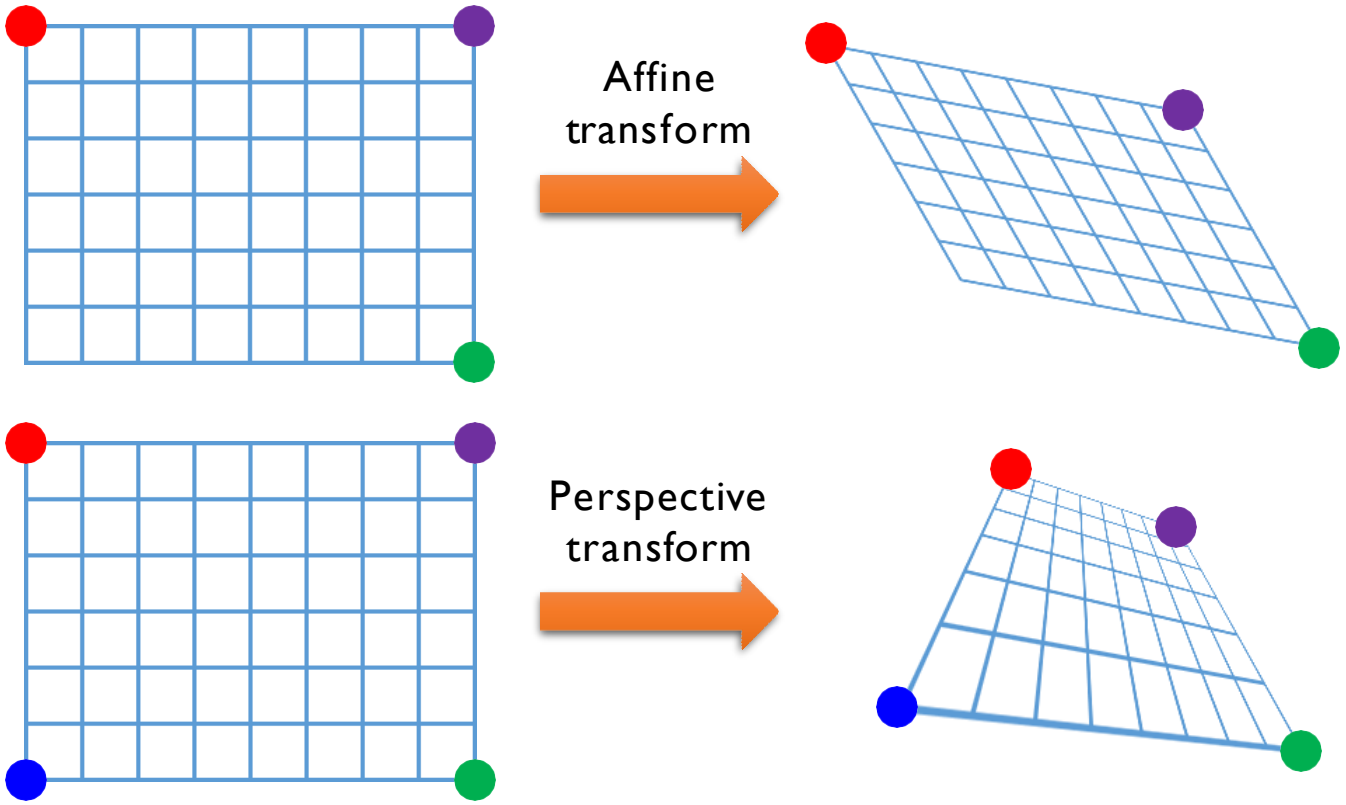
$$x' = \frac{p_{11}x + p_{12}y + p_{13}}{p_{31}x + p_{32}y + 1}$$

$$y' = \frac{p_{21}x + p_{22}y + p_{23}}{p_{31}x + p_{32}y + 1}$$

- DOF: Degree of Freedom

어파인 변환과 투시 변환

- 어파인 변환 vs. 투시 변환



OpenCV API

☐ 투시 변환 행렬 구하기

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

- ✓ src: 4개의 원본 좌표점. numpy.ndarray. shape=(4,2)
  e.g) np.array([[x_1, y_1], [x_2, y_2], [x_3, y_3], [x_4, y_4]], np.float32)
- ✓ dst: 4개의 결과 좌표점. numpy.ndarray. shape=(4,2)
- ✓ 반환값: 3x3 크기의 투시 변환 행렬

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \mathbf{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

where $dst(i) = (x'_i, y'_i), src(i) = (x_i, y_i), i = 0, 1, 2, 3$

OpenCV API

☐ 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. cv2.BORDER_CONST
- ✓ borderValue: ANT일 때 사용할 상수 값.
  기본값은 0.

투시 변환 예제

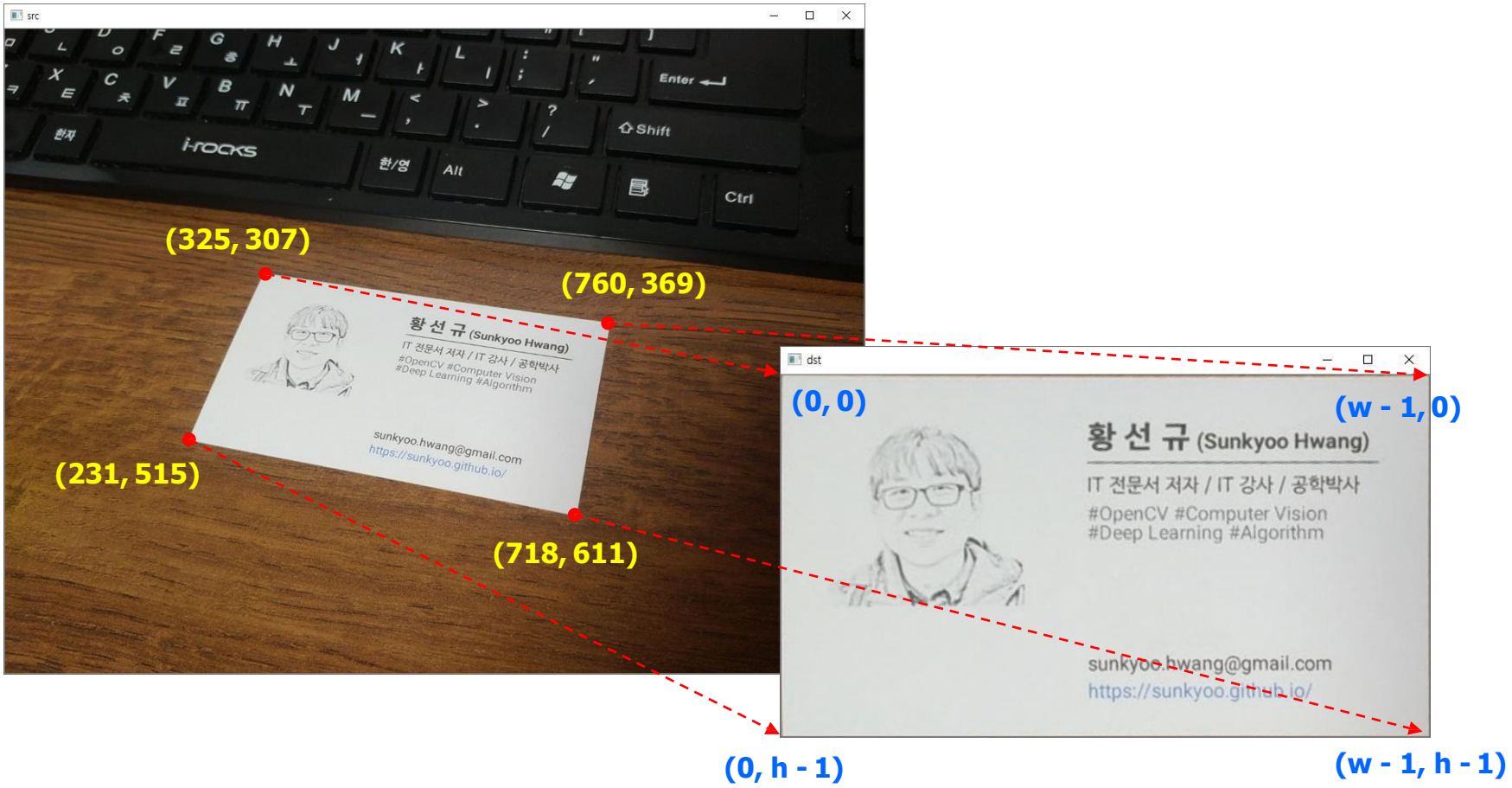
실습: perspective.py

☐ 찌그러진 명함 펴기

```
import sys
import numpy as np
import cv2

src = cv2.imread('namecard1.jpg')

w, h = 720, 400
srcQuad = np.array([[325, 307], [760, 369], [718, 611], [231, 515]], np.float32)
dstQuad = np.array([[0, 0], [w-1, 0], [w-1  h-1], [0, h-1]], np.float32)
,
pers = cv2.getPerspectiveTransform(srcQuad, dstQuad)
dst = cv2.warpPerspective(src, pers, (w, h))

cv2.imshow('src', src)
cv2.imshow('dst', dst)
cv2.waitKey()  cv2.dest
royAllWindows()
```

투시 변환 예제

- 찌그러진 명함 펴기 실행 화면



Tesseract 사용하기

- Tesseract
  - ✓ 광학 문자 인식(OCR) 라이브러리
  - ✓ https://github.com/tesseract-ocr/tesseract
  - ✓ 1985년~1994년 사이에 휴렛 팩커드에서 개발 ➐ 2005년 오픈 소스 ➐ 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

- Windows Pre-built 설치 파일 다운로드
  - ✓ https://github.com/UB-Mannheim/tesseract/wiki
  - ✓ 독일 만하임 대학교(Mannheim University) 도서관에서 오래된 신문에 대해 OCR을 수행하기 위해 Tesseract를 사용

Tesseract 사용하기

- pytesseract 설치하기
  - ✓ https://github.com/madmaze/pytesseract
  - ✓ 파이썬에서 Tesseract를 사용하기 위해 필요한 패키지

```
> pip install pytesseract
```

- OpenCV/numpy와 사용하기
  - ✓ Tesseract는 numpy ndarray객체를 지원
  - ✓ 다만 Tesseract는 RGB 컬러 포맷을 사용하므로 cv2.cvtColor() 함수를 이용하여 BGR 순서를 RGB 순서로 변경해야 함.

```
img = cv2.imread('digits.png')

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
print(pytesseract.image_to_string(img_rgb))
```

명함 검출 및 인식 프로그램

실습: namecard.py

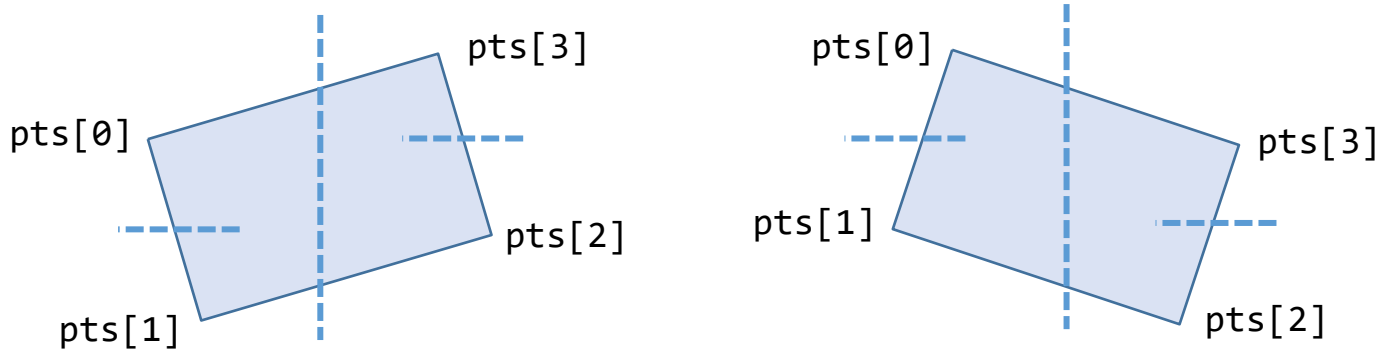
```
import sys
import numpy as np
import cv2  pytess
import eract

filename = 'namecard1.jpg'
if len(sys.argv) > 1:
    filename = sys.argv[1]

src = cv2.imread(filename)

dw, dh = 720, 400
srcQuad = np.array([[0, 0], [0, 0], [0, 0], [0, 0]], np.float32)
dstQuad = np.array([[0, 0], [0, dh], [dw, dh], [dw, 0]], np.float32)
dst = np.zeros((dh, dw), np.uint8)

src_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
th, src_bin = cv2.threshold(src_gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU
)

contours, _ = cv2.findContours(src_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

명함 검출 및 인식 프로그램

```
for pts in contours:
    if cv2.contourArea(pts) < 1000:
        continue

    approx = cv2.approxPolyDP(pts, cv2.arcLength(pts, True)*0.02, True)

    if not cv2.isContourConvex(approx) or len(approx) != 4:
        continue

    srcQuad = reorderPts(approx.reshape(4, 2).astype(np.float32))

    pers = cv2.getPerspectiveTransform(srcQuad, dstQuad)
    dst = cv2.warpPerspective(src, pers, (dw, dh), flags=cv2.INTER_CUBIC)

    dst_rgb = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)  print(pytesser
    act.image_to_string(dst_rgb, lang='Hangul+eng'))

cv2.imshow('src', src)
cv2.imshow('dst', dst)
cv2.waitKey()  cv2.dest
royAllWindows()
```

명함 검출 및 인식 프로그램

```
def reorderPts(pts):
    idx = np.lexsort((pts[:, 1], pts[:, 0])) # 칼럼0 -> 칼럼1 순으로 정렬한 인덱스를 반환
    pts = pts[idx]  # x좌표로 정렬

    if pts[0, 1] > pts[1, 1]:
        pts[[0, 1]] = pts[[1, 0]]

    if pts[2, 1] < pts[3, 1]:
        pts[[2, 3]] = pts[[3, 2]]

    return pts
```



명함 검출 및 인식 프로그램

```
관리자: C:₩Windows₩System32₩cmd.exe - python  namecard.py namecard3.jpg
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

c:₩coding₩python₩opencv₩t-academy₩3교시>python namecard.py
황 선 규 (Sunkyoo Hwang)

IT 전문서 저자 / IT 강사 / 공 학 박사

#OpenCV #Computer Vision
#Deep Learning #Algorithm

sunkyoo.hwang@gmail.com
https://sunkyoo.github.io/

c:₩coding₩python₩opencv₩t-academy₩3교시>
c:₩coding₩python₩opencv₩t-academy₩3교시>python namecard.py namecard3.jpg
황 선 규 (Sunkyoo Hwang)

17 전문서 저자 / 17 강사 / 공 학 박사

#OpenCV #Computer Vision
#Deep Learning #Algorithm

sunkyoo.hwang@gmail.com
https://sunkyoo.github.io/
```