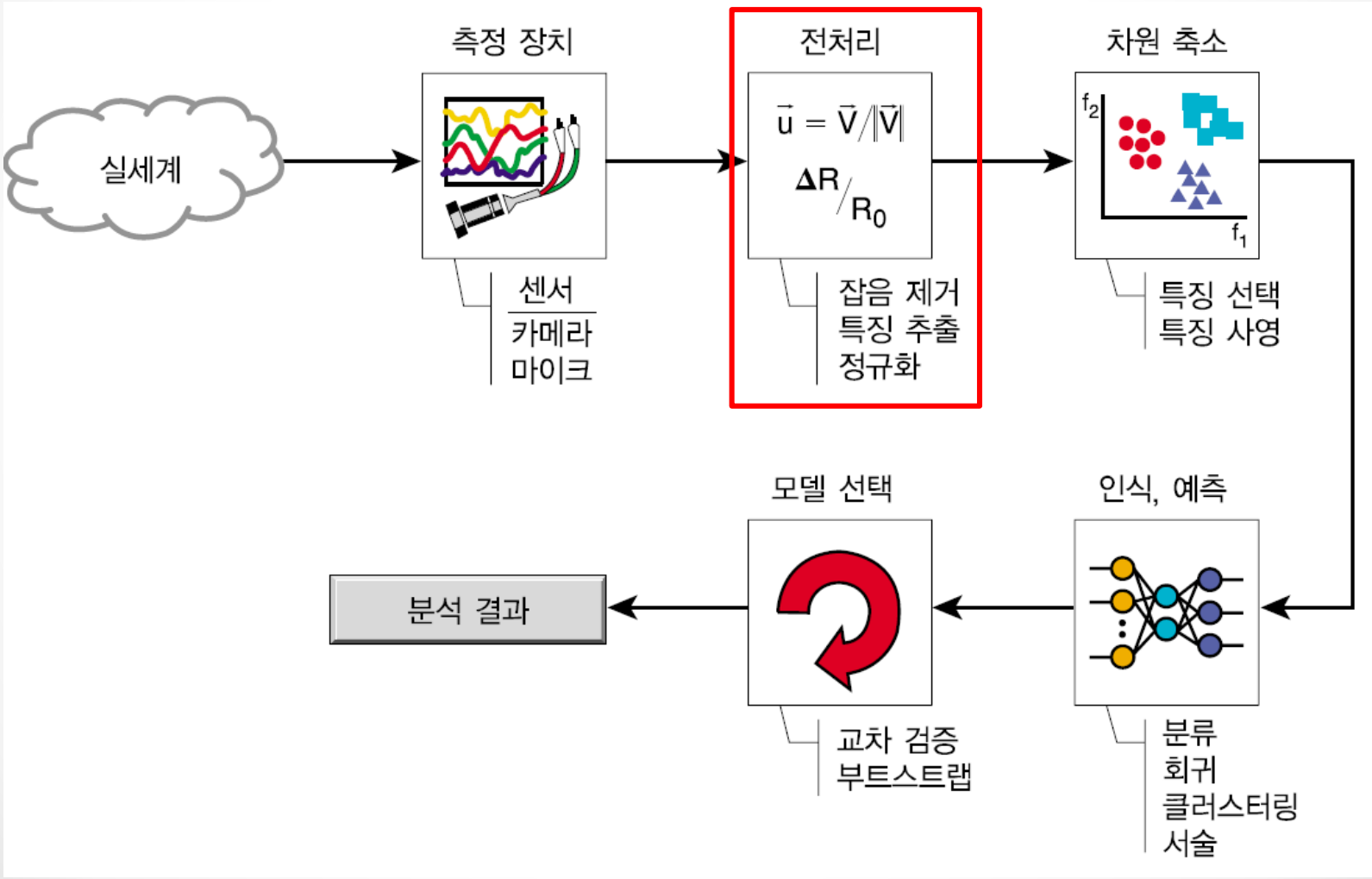
Python

Application2

Application

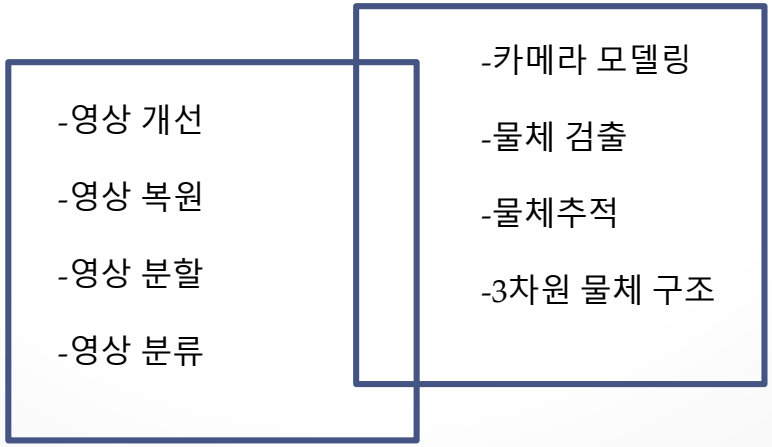


Application

분석 결과

Application

- 1. 영상처리와 컴퓨터 비전
  - 디지털 영상처리는 컴퓨터를 사용하여 입력 영상을 보다 질 좋은 출력 영상을 얻는 과정(Ex.영상 대비 개선, 관심영역 강조,영상 압축)
  - 컴퓨터 비전은 카메라에 의해 획득되는 입력 영상으로부터 영상에 대한 의미 있는 정보를 추출해 내는 분야로 주로 실시간 응용에 적용
  - (Ex.문자인식,제품 결함검사, 지문인식,움직임검출, 물체 추적)
  - 영상처리와 컴퓨터 비전은 모두 영상을 처리하기 때문에 많은 내용이 중복
  - 대략적인 구분은 영상을 컴퓨터를 사용하여 처리하는 모든 분야를 영상처리라 하고, 인간의 눈 대신 카메라에 의한 영상을 입력, 인간의 뇌 대신에 컴퓨터를 사용하여 영상으로부터 의미 있는 정보를 추출하는 분야를 컴퓨터 비전이라고 할 수 있음



Application

- OpenCV 설치

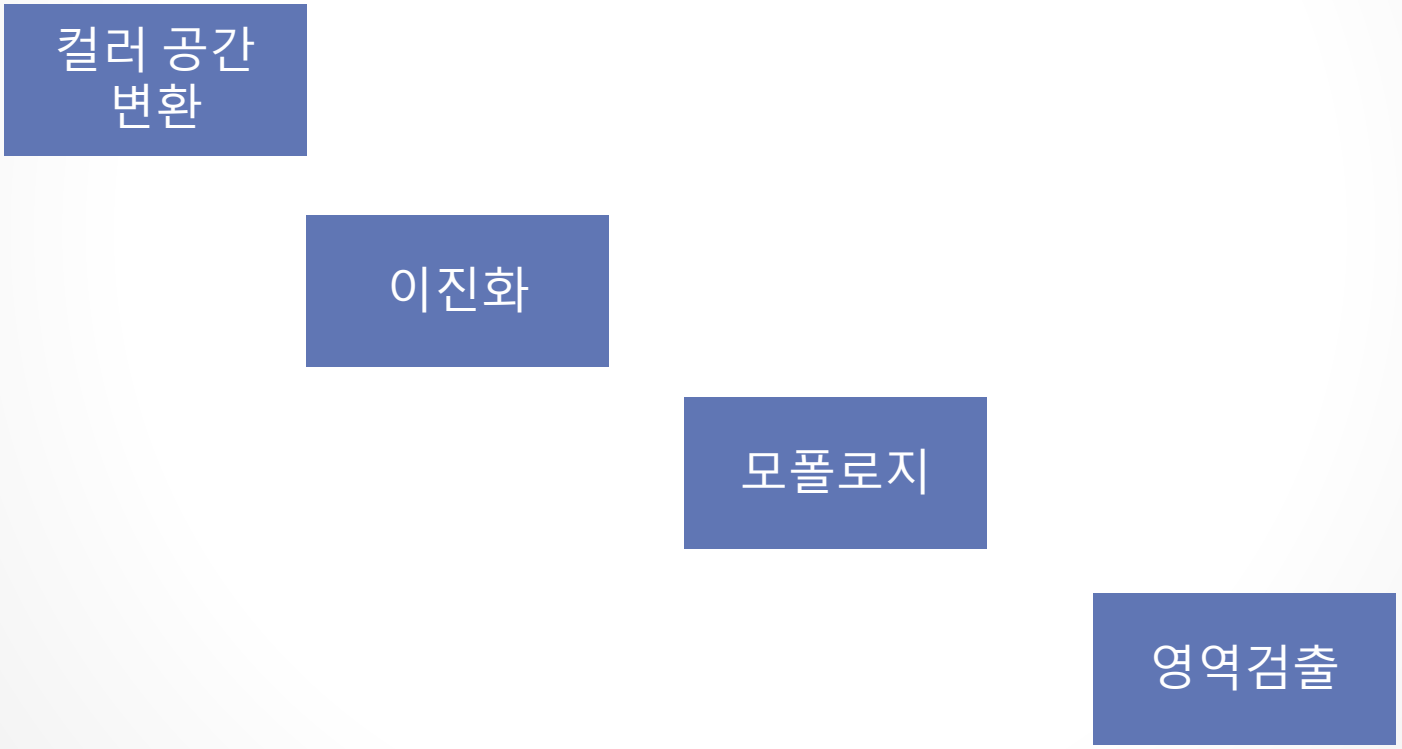
python -m pip install opencv-python

or

PyCharm 에서 opencv-python 패키지 설치

Application

- 숫자 영역 추출



Application

- cvtColor()

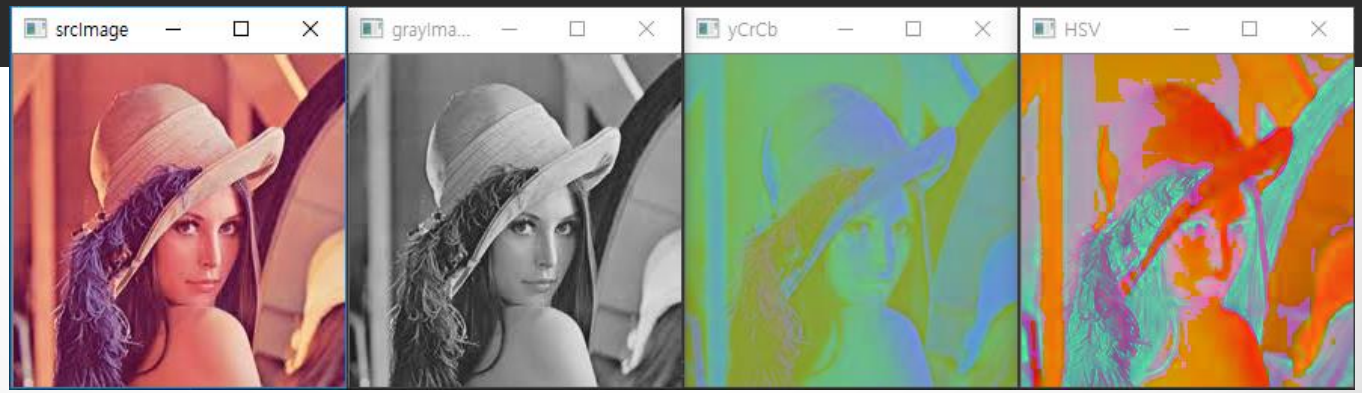
cvtColor(src, code) → dst

  - src는 8비트, 16비트 또는 32비트 실수 입력영상
  - dst는 src와 같은 크기 같은 깊이를 가지며, 채널의 수는 다를 수 있는 출력 영상
  - **RGB -> GRAY**
    - COLOR_BGR2GRAY는 BGR로 표현된 영상을 dst에 GRAY 영상으로 변환
  - **RGB -> YCrCb**
    - COLOR_BGR2YCrCb는 src BGR 컬러모델을 YCrCb로 변환하여 dst에 저장
  - **RGB -> HSV**
    - COLOR_BGR2HSV는 src BGR 컬러모델을 HSV로 변환하여 dst에 저장

Application

- cvtColor()

```
import cv2

srcImage = cv2.imread("lenna.jpg", cv2.IMREAD_COLOR)

grayImage = cv2.cvtColor(srcImage, cv2.COLOR_BGR2GRAY)
yCrCb = cv2.cvtColor(srcImage, cv2.COLOR_BGR2YCrCb)
HSV = cv2.cvtColor(srcImage, cv2.COLOR_BGR2HSV)

cv2.imshow('srcImage', srcImage)
cv2.imshow('grayImage', grayImage)
cv2.imshow('yCrCb', yCrCb)
cv2.imshow('HSV', HSV)

cv2.waitKey(0)
```



Application

- cv2.threshold()

cv2.threshold(src, thresh, maxval, type) → retval, dst

  - Parameters:
    - src – input image로 single-channel 이미지.(grayscale 이미지)
    - thresh – 임계값
    - maxval – 임계값을 넘었을 때 적용할 value
    - type – thresholding type

Application

- cv2.threshold()

$$f(x) = \begin{cases} \max\_value, & r > threshold \\ 0, & otherwise \end{cases}$$

- CV_THRESH_BINARY

$$f(x) = \begin{cases} \max\_value, & r > threshold \\ 0, & otherwise \end{cases}$$

- CV_THRESH_BINARY_INV

$$f(x) = \begin{cases} 0, & r > threshold \\ \max\_value, & otherwise \end{cases}$$

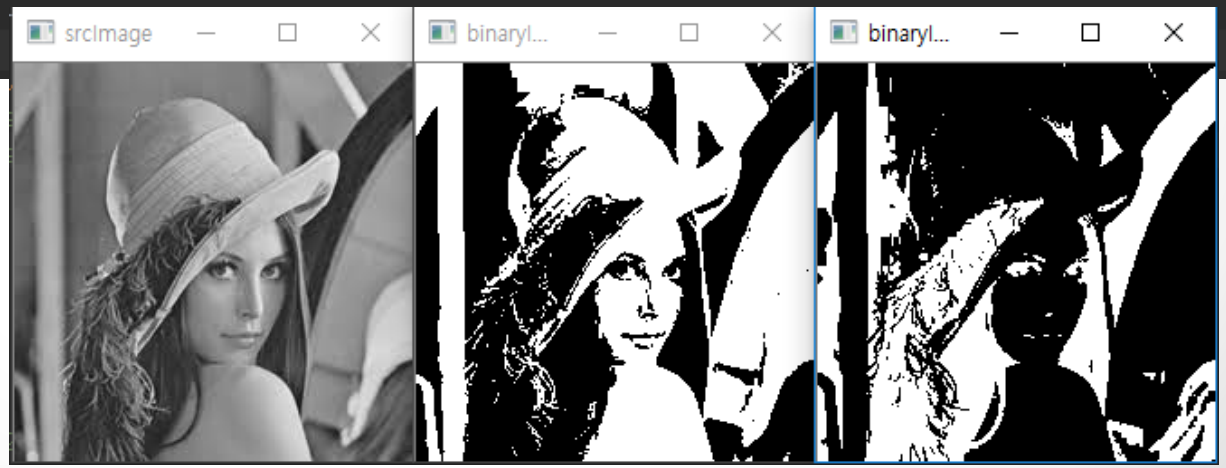
- CV_THRESH_TRUNC

$$f(x) = \begin{cases} threshold, & r > threshold \\ r, & otherwise \end{cases}$$

Application

- cv2.threshold()

```
import cv2

srcImage = cv2.imread("lenna.jpg", 0)

ret1, binaryImage1 = cv2.threshold(srcImage, 125, 255, cv2.THRESH_BINARY)
ret2, binaryImage2 = cv2.threshold(srcImage, 100, 255, cv2.THRESH_BINARY_INV)

cv2.imshow('srcImage', srcImage)
cv2.imshow('binaryImage1', binaryImage1)
cv2.imshow('binaryImage2', binaryImage2)

cv2.waitKey(0)
```



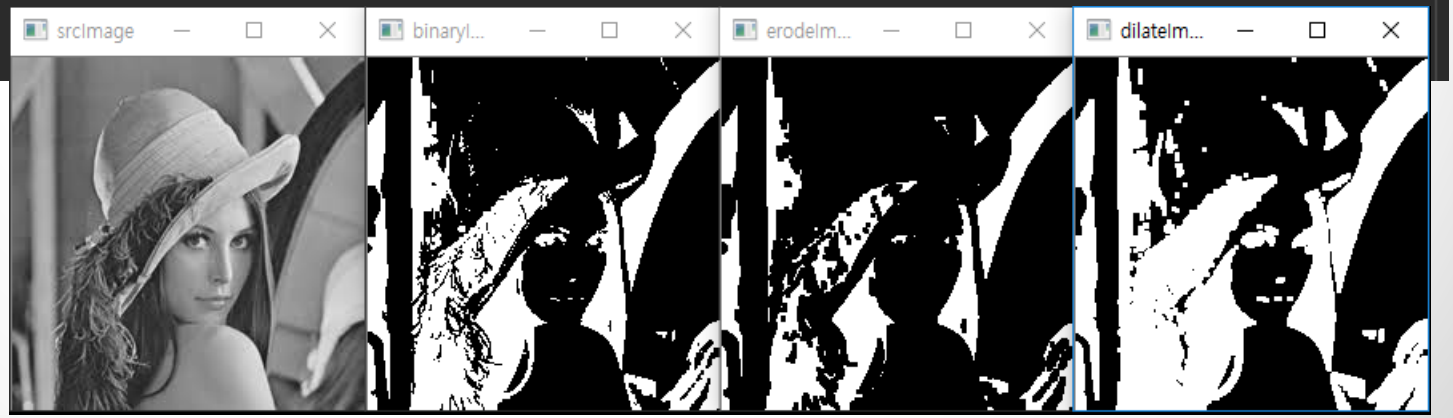
Application

- 모폴로지 연산
  - 모폴로지 연산은 구조요소를 이용하여 반복적으로 영역을 확장시켜 떨어진 부분 또는 구멍을 채우거나, 잡음을 축소시켜 제거하는 등의 연산으로 침식(erode), 팽창(dilate), 열기(opening), 닫기(closing)등이 있음.
  - erode
    - cv2.erode(src, kernel, dst, anchor, iterations, borderType, borderValue)
  - dilate
    - cv2.dilation(src, kernel, dst, anchor, iterations, borderType, borderValue)

Application

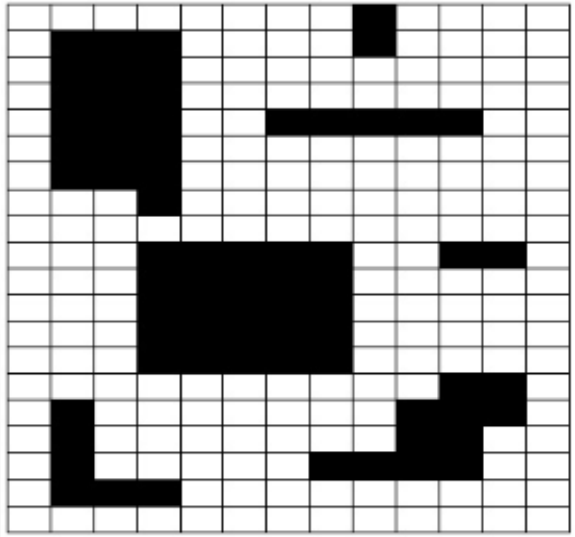
- 모폴로지 연산

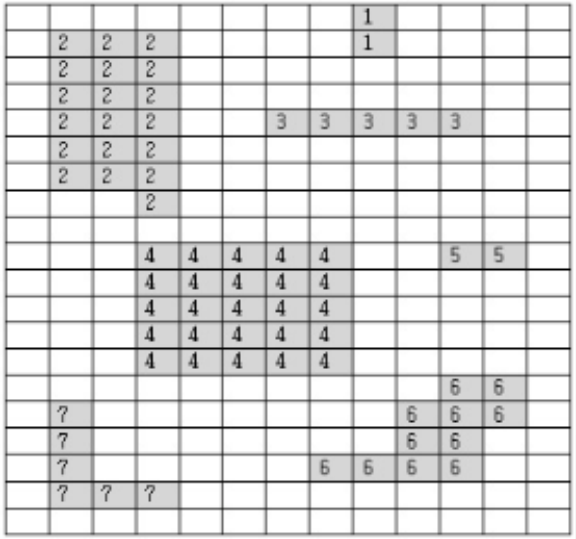
```
import numpy as np
import cv2

srcImage = cv2.imread("lenna.jpg", 0)

ret, binaryImage = cv2.threshold(srcImage, 100, 255, cv2.THRESH_BINARY_INV)

kernel = np.ones((3, 3), np.uint8)
erodeImage = cv2.erode(binaryImage, kernel, iterations=1)
dilateImage2 = cv2.dilate(binaryImage, kernel, iterations=1)

cv2.imshow('srcImage', srcImage)
cv2.imshow('binaryImage', binaryImage)
cv2.imshow('erodeImage', erodeImage)
cv2.imshow('dilateImage2', dilateImage2)

cv2.waitKey(0)
```



Application

- 레이블링
  - 이진화된 영상은 아직 영역이 분리된 상태가 아니라 단지 각각의 화소들이 배경과 구분되어 있는 상태.
  - 물체 부분에 해당하는 각각의 화소들을 하나의 영역으로 묶어 각 영역에 레이블 값을 할당하는 과정을 레이블링이라 함.





Application

- 숫자 영역 추출

```
import numpy as np
import cv2
# 이미지 불러오기
srcImage = cv2.imread("plate1.jpg")

# 컬러 변환
grayImage = cv2.cvtColor(srcImage, cv2.COLOR_BGR2GRAY)

# 이진 이미지 생성
ret, binaryImage = cv2.threshold(grayImage, 100, 255,
cv2.THRESH_BINARY_INV)

# 이진 이미지 필터링
kernel = np.ones((3, 3), np.uint8)
binaryImage = cv2.erode(binaryImage, kernel, iterations=3)
binaryImage = cv2.dilate(binaryImage, kernel, iterations=3)
```

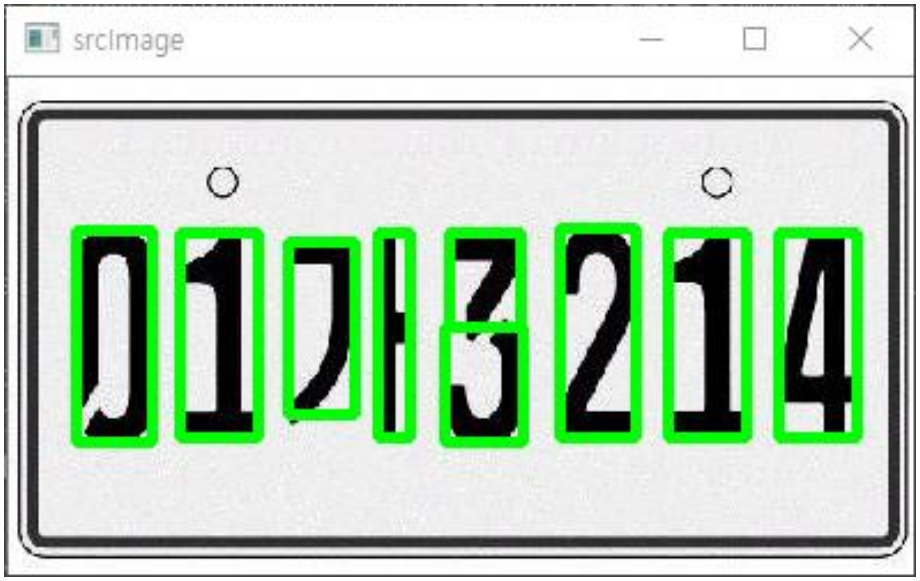
Application

- 숫자 영역 추출

```
# 객체 레이블링
nlabels, labels, stats, centroids = cv2.connectedComponentsWithStats(binaryImage)

# 레이블링 결과 출력
for I in range(1, nlabels):
    rect = (stats[i][0], stats[i][1], stats[i][2], stats[i][3])

    cv2.rectangle(srcImage, rect, (0, 255, 0), 3)
# 이미지 출력
cv2.imshow('srcImage', srcImage)
cv2.imshow('binaryImage', binaryImage)
cv2.waitKey(0)
```

Application

- 숫자 영역 추출



Application

- 숫자 영역 추출