울산대학교 공과대학 IT융합학부
IT융합전공

Chapter 12. 영상처리 응용 사례 II

2024-1학기

IT융합학부 IT융합전공

김대환

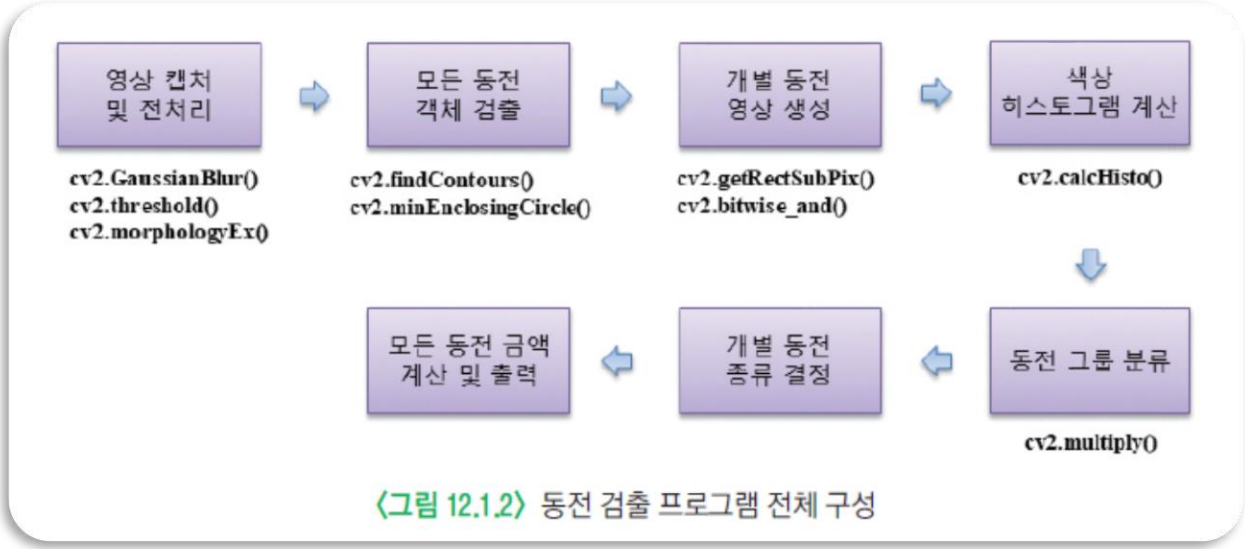
목차

12-1. 동전 인식 프로그램 (1) – 영상 캡처 및 전처리

- 대량의 동전 계산 – 주화 계수기

- 일상생활에서 동전 계산
  - 디지털 영상 처리를 이용하면 소량의 동전들 인식 → 금액 계산 가능

- 동전 인식 및 계산 시스템 전체 구성



〈그림 12.1.2〉 동전 검출 프로그램 전체 구성

12-1. 동전 인식 프로그램 (2) – 영상 캡처 및 전처리

- 동전 캡처 시스템 구성
  - 동전을 놓는 받침대
  - USB 카메라
  - 카메라 고정 스탠드

  - 캡처된 동전 영상 파일 제공

12-1. 동전 인식 프로그램 (3) – 영상 캡처 및 전처리

- 전처리
  - 명암도 영상 및 가우시안 블러링 수행
  - 이진화와 모폴로지 열림 (Open) 연산

```
def preprocessing(coin_no):                                    # 전처리 함수
    fname = "images/coin/{0:02d}.png".format(coin_no)
    image = cv2.imread(fname, cv2.IMREAD_COLOR)                # 영상 읽기
    if image is None: reutrn None, None                        # 예외처리는 메인에서

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)             # 명암도 영상 변환
    gray = cv2.GaussianBlur(gray, (7, 7), 2, 2)                # 블러링
    flag = cv2.THRESH_BINARY + cv2.THRESH_OTSU                 # 오츠(otsu) 이진화 지정
    _, th_img = cv2.threshold(gray, 130, 255, flag)            # 이진화

    mask = np.ones((3, 3), np.uint8)
    th_img = cv2.morphologyEx(th_img, cv2.MORPH_OPEN, mask)    # 열림 연산
    reutrn image, th_img
```

동전 파일 폴더

히스토그램에서 두 영역으로 구분하도록 임계값 정해줌

원본 영상과 이진 영상 모두 반환

12-1. 동전 인식 프로그램 (4) – 동전 객체 검출

- 동전 검출 과정
  - 낱개의 동전객체 인식 → 중심 좌표와 반지름 계산
  - Find_coins() 함수에서 구현

```
01  def find_coins(image):
02        results= cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
03        contours = results[0] if int(cv2.__version__[0]) >= 4 else results[1]
04
05        ## 반복문 방식
06        # circles = []
07        # for contour in contours:
08        #     center, radius = cv2.minEnclosingCircle(contour)          # 외각 감싸는 원 검출
09        #     circle = (tuple(map(int, center)), int(radius))           # 소스 최적화 위해
10        #     if radius>25: circles.append(circle)
11
12        ## 리스트 생성 방식
13        circles = [cv2.minEnclosingCircle(c) for c in contours]     # 외각 감싸는 원 검출
14        circles = [(tuple(map(int, center)), int(radius))
15                   for center, radius in circles if radius>25]
16        reutrn circles
```

검출 외곽선들

정수형 자료로 구성

중심좌표와 반지름

12-1. 동전 인식 프로그램 (5) – 동전 객체 검출

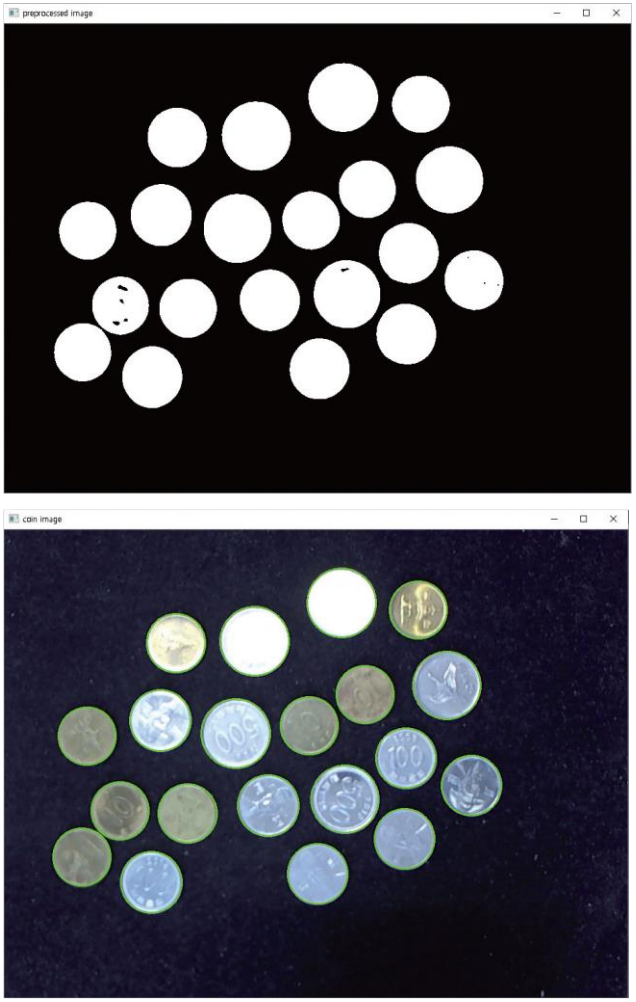
예제 12.1.1 동전 객체 검출 – header/coin_preprocess.py

```
01  import numpy as np, cv2
02
03  def preprocessing(coin_no): ···
17
18  def find_coins(image): ···
```

예제 12.1.1 동전 객체 검출 – 01.find_coins.py

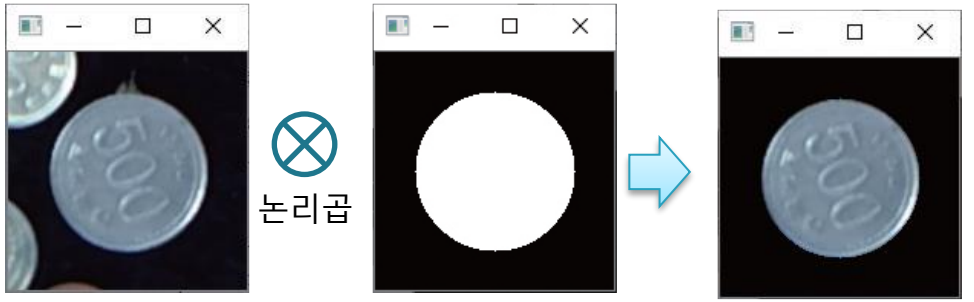
```
01  from coin_preprocess import *                    #
02
03  image, th_img = preprocessing(70)                #
04  if image is None: raise Exception("영상파일 읽기 에러")
05
06  circles = find_coins(th_img)                     #
07  for center, radius in circles:
08      cv2.circle(image, center, radius, (0, 255, 0), 2)   #
09
10  cv2.imshow("preprocessed image", th_img)
11  cv2.imshow("coin image", image)
12  cv2.waitKey(0)
```

| 실행결과 |



12-1. 동전 인식 프로그램 (6) – 개별 동전 영상 생성

- 개별 동전영상 가져오기
  - cv2::getRectSubPix() 함수
    - 입력 영상에서 지정된 좌표를 중심으로 크기만큼 영상 가져옴
    - 중심점 기준으로 특정 크기의 영역을 가져오기 편리함
  - 개별 동전 영상
    - 주위에 다른 동전 포함 및 주위 잡음 존재 가능
    - 원형 마스크를 통한 논리곱 (cv2::bitwise_and()) 연산으로 주위 배경 제거 가능



동전 영상 원형 마스크

```
def make_coin_img(src, circles):
    coins = []
    for center, radius in circles:
        r = radius * 3                                          # 검
        cen = (r // 2, r // 2)                                  # 마
        mask = np.zeros((r, r, 3), np.uint8)                    # 마
        cv2.circle(mask, cen, radius, (255, 255, 255), cv2.FILLED)
        # cv2.imshow("mask_" + str(center), mask)               # 마

        coin = cv2.getRectSubPix(src, (r, r), center)           # 동
        coin = cv2.bitwise_and(coin, mask)                      # 마
        coins.append(coin)                                      # 동
    reutrn coins
```

12-1. 동전 인식 프로그램 (7) – 색상 히스토그램 계산

```python
def calc_histo_hue(coin):
    hsv = cv2.cvtColor(coin, cv2.COLOR_BGR2HSV)                    #
    hsize, ranges = [32], [0, 180]                                 #
    hist = cv2.calcHist([hsv], [0], None, hsize, ranges)           #
    reutrn hist.flatten()                                          #
```

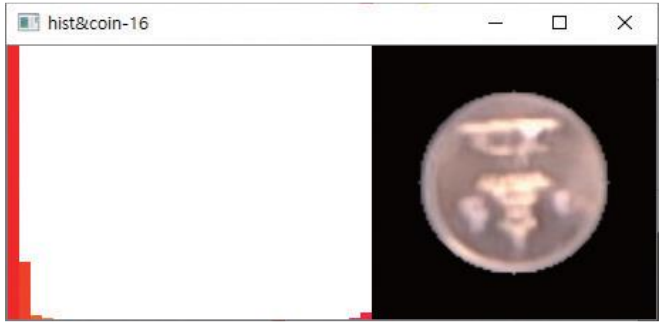
```python
def make_palette(rows):                                            # hue 채널 팔
    ## 리스트 생성 방식
    hue = [round(i * 180 / rows) for i in range(rows)]             # hue 값 리스
    hsv = [[[h, 255, 255]] for h in hue]                           # (hue, 255,2
    hsv = np.array(hsv, np.uint8)                                  # 정수(uint8)형
    return cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)                    # HSV 컬러-

def draw_hist_hue(hist, shape=(200, 256, 3)):                      # 색상 히스토
    hsv_palette = make_palette(hist.shape[0])                      # 색상 팔레트
    hist_img = np.full(shape, 255, np.uint8)
    cv2.normalize(hist, hist, 0, shape[0], cv2.NORM_MINMAX)        # 영상

    gap = hist_img.shape[1] / hist.shape[0]                        # 한 ?
    for i, h in enumerate(hist):
        x, w = int(round(i * gap)), int(round(gap))
        color = tuple(map(int, hsv_palette[i][0]))                 # 정수
        cv2.rectangle(hist_img, (x, 0, w, int(h)), color, cv2.FILLED)

    return cv2.flip(hist_img, 0)
```

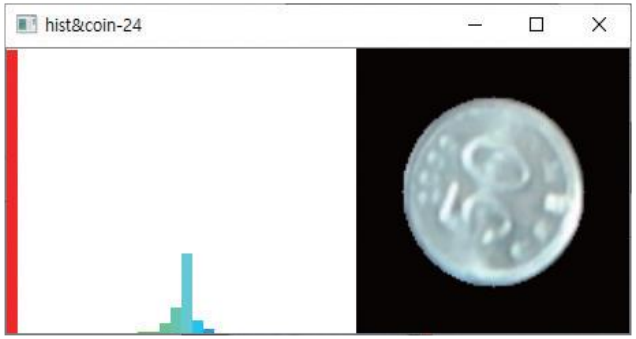
예제 12.1.2 동전 객체 히스토그램 그리기 – 02.draw_coin_histo.p

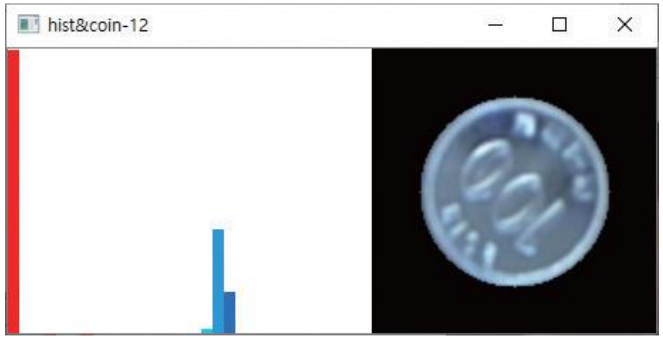
```python
from coin_preprocess import *
from coin_utils import *                                           # 모
from Common.histogram import draw_histo_hue                        # 색

coin_no = 15                                                       # 동
image, th_img = preprocessing(coin_no)                             # 전
circles = find_coins(th_img)                                       # 동
coin_imgs = make_coin_img(image, circles)
coin_hists = [calc_histo_hue(coin) for coin in coin_imgs]

for i, img in enumerate(coin_imgs):
    h, w = 200, 256
    hist_img = draw_hist_hue(coin_hists[i], (h, w, 3))

    merge = np.zeros((h, w+h, 3), np.uint8)
    merge[:, :w] = hist_img                                        # 결
    merge[:, w:] = cv2.resize(img, (h, h))                         # 결
    cv2.imshow("hist&coin - " + str(i), merge)

cv2.waitKey(0)
```

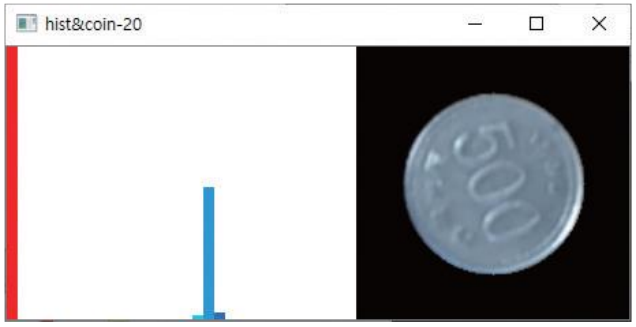
12-1. 동전 인식 프로그램 (8) – 색상 히스토그램 계산

| 실행결과 |



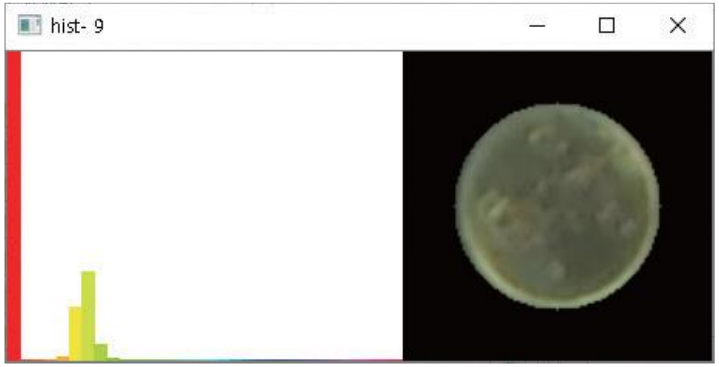


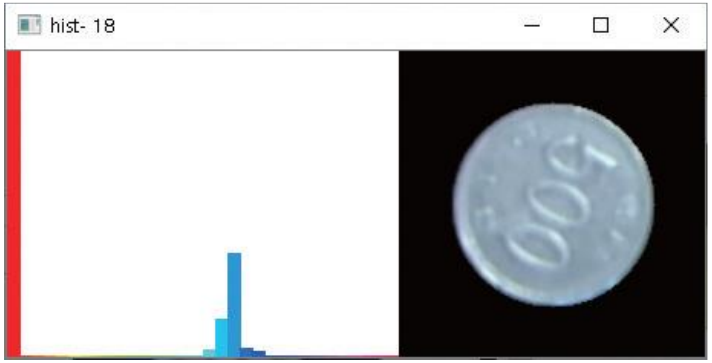




12-1. 동전 인식 프로그램 (9) – 동전 그룹 분류

- 동전 인식
  - 일반적으로 동전 객체의 크기 이용
  - 동전 객체 검출과정에서 해당 동전이 완벽하게 검출되지 않을 수 있음
    - 크기 비슷한 금액 오인식 가능성 높음 (10원/50원, 10원/100원)
  - 10원과 다른 동전 색상 차리 확실함
    - 색상을 동전 분류에 활용
    - 색상 히스토그램 이용





12-1. 동전 인식 프로그램 (10) – 동전 그룹 분류

- 히스토그램 계산 함수

$$S = \frac{\sum H(i) * \omega(i)}{\sum H(i)}, \quad i\{0,\ 2,\ \cdots,\ N\} \quad \begin{cases} Group1, & where\ S > 1.2 \\ Group0, & otherwise \end{cases}$$

```
def grouping(hists):
    ws = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3,
          4, 5, 6, 8, 6, 5, 4, 3, 2, 1, 0, 0, 0, 0, 0, 0]        # 가중치 32개 원소 지정

    sim = np.multiply(hists, ws)                                  # 히스토그램과 가중치 곱
    similaritys = np.sum(sim, axis=1) / np.sum(hists, axis=1)     # 가중치 곱의 합/히스토그램 합
    groups = [1 if s > 1.2 else 0 for s in similaritys]

    ## 결과 보기
    # x = np.arange(len(ws)) # plt.plot(x, ws, 'r'), plt.show(), plt.tight_layout()  # 가중치 그래프 보기
    # for i, s in enumerate(similaritys):                                            # 그룹핑 결과 출력
    #     print("%d %5.0f  %d" % (i, s, groups[i]))
    reutrn groups
```

개별 동전들의 그룹 판정

x방향 합 계산

12-1. 동전 인식 프로그램 (11) – 개별 동전 종류 결정

- 그룹에 따라 반지름 달리 적용

동전 종류		동전 반지름
그룹 0	500 원	50 ≤ raduis < 56
	100 원	47 ≤ raduis < 50
	10 원	26 ≤ raduis < 47
그룹 1	500 원	50 ≤ raduis < 56
	100 원	44 ≤ raduis < 50
	50 원	36 ≤ raduis < 44
	10 원	26 ≤ raduis < 36

동전별 개수

```
def classify_coins(circles, groups):
    ncoins = [0] * 4
    g = np.full((2, 70), -1, np.int)                    # 2행으로 두 개 그룹 설정
    g[0, 26:47], g[0, 47:50], g[0, 50:] = 0, 2, 3       # 10원 그룹– 10원 가능성 확대
    g[1, 36:44], g[1, 44:50], g[1, 50:] = 1, 2, 3       # 50원 그룹– 50원 100원 가능성 확대

    for group, (_, radius) in zip(groups, circles):     # 동전 객체 순회
        coin = g[group, radius]                         # 동전 종류 확정
        ncoins[coin] += 1                               # 동전별 개수 산정
    reutrn np.array(ncoins)                             # 넘파이 행렬로 반환
```

12-1. 동전 인식 프로그램 (12) – 최종 동전 계산 프로그램

예제 12.1.3 동전 계산 프로그램 완성 – 03.calc_coins.py

오타

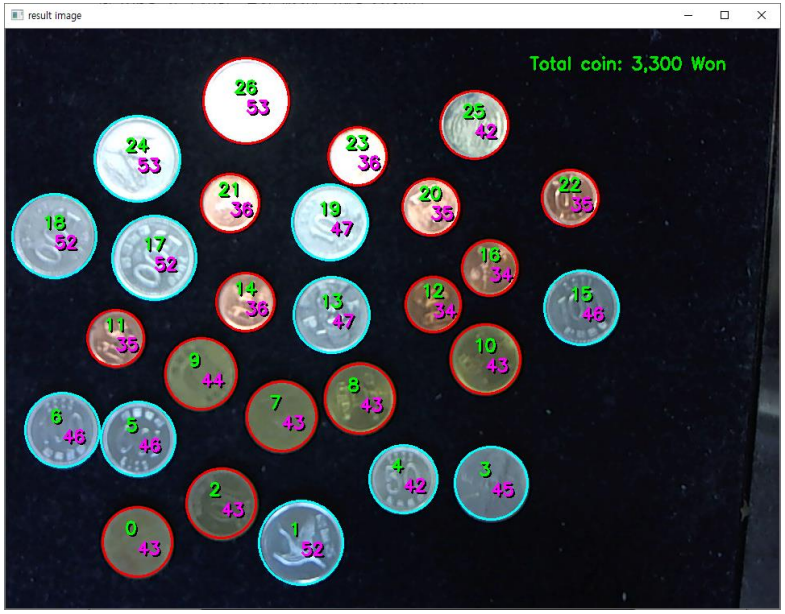
```
01  from coin_preprocess import *
02  from coin_utils import *                          # 관련 함수 임포트
03  from Common.utlis import put_string               # 문자열 표시 함수 임포트
04
05  coin_no = int(input("동전 영상 번호: "))
06  image, th_img = preprocessing(coin_no)            # 전처리 수행
07  circles = find_coins(th_img)                      # 객체(중심점, 반지름) 검출
08  coin_imgs = make_coin_img(image, circles)         # 개별 동전 영상 생성
09
10  coin_hists= [calc_histo_hue(coin) for coin in coin_imgs]      # 동전 영상 히스토그램
11  groups = grouping(coin_hists)                     # 동전 영상 그룹 분리
12
13  ncoins = classify_coins(circles, groups)          # 동전 인식
14  coin_value = np.array([10, 50, 100, 500])         # 동전 금액
15  for i in range(4):
16      print("%3d원: %3d개" % (coin_value[i], ncoins[i]))  # 동전별 개수 출력
17
```

검출 동전별 개수 출력

12-1. 동전 인식 프로그램 (13) – 최종 동전 계산 프로그램

```
18  total = sum(coin_value * ncoins)                    # 동전 금
19  str = "Total coin: {:,} Won".format(total)          # 계산 금
20  print(str)                                          # 실행창
21  put_string(image, str, (10, 50), '', (0, 230, 0))  # 영상에
22
23  ## 동전 객체에 정보(반지름, 금액) 표시
24  color = [(0, 0, 250), (255, 255, 0), (0, 250, 0), (250, 0, 255)]
25  for i, (c, r) in enumerate(circles):
26      cv2.circle(image, c, r, color[groups[i]], 2)
27      put_string(image, i, (c[0] - 15, c[1] - 10), "", color[2])
28      put_string(image, r, (c[0], c[1] + 15), "", color[3])
29
30  cv2.imshow("result image", image)
31  cv2.waitKey(0)
```

| 실행결과 |




```
Run:  03.calc_coins
C:\Python\python.exe D:/source/chap12/03.calc_coins.py
동전 영상 번호: 24
 10원:  15개
 50원:   1개
100원:   6개
500원:   5개
Total coin: 3,300 Won
```

12-2. SVM을 이용한 차량 번호 검출 프로그램 (1)

- 차량 번호 인식 (LPR: License Plate Recognition) 시스템
  - 아파트 주차장, 공영 주차장, 빌딩 등에서 진입차량의 번호판 자동 인식 및 식별
  - 차량과 주차관기를 제어하는 시스템

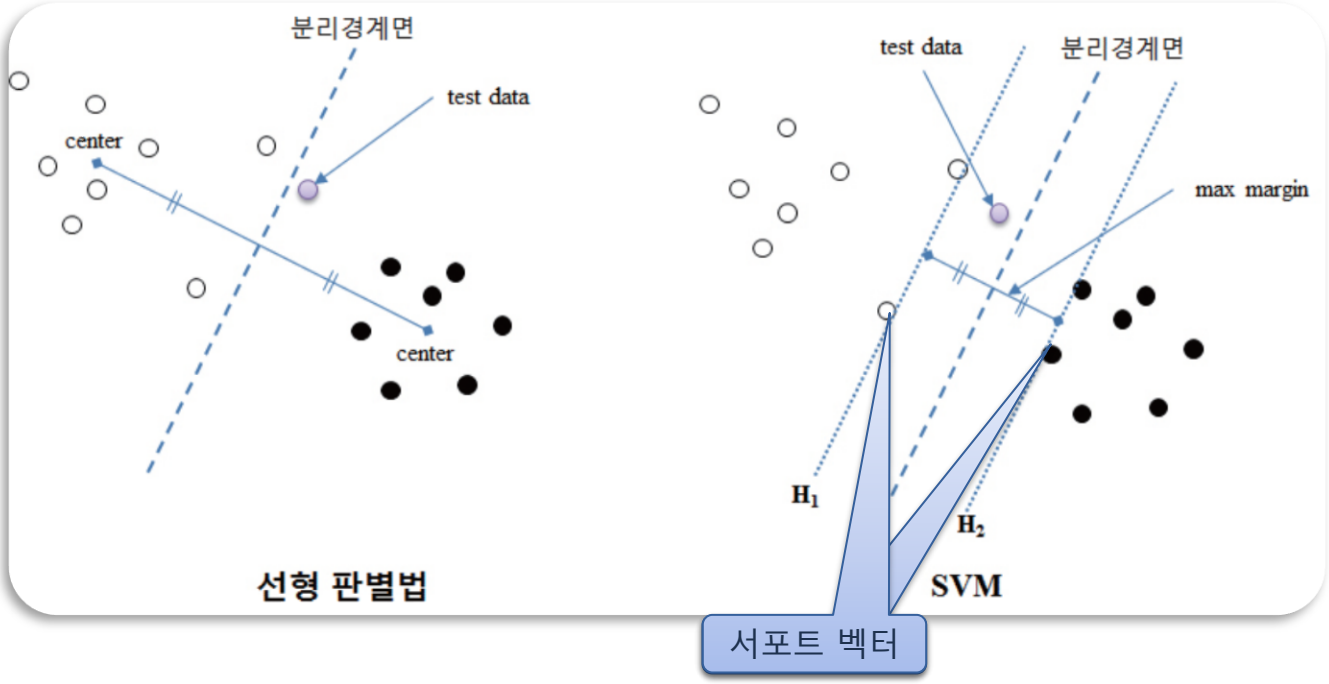
- 차량 번호 인식 프로그램 2단계
  - 입력 영상에서 번호판 영역 검출 단계
  - 검출된 번호판 영역에서 숫자나 문자 인식하는 단계

- 본 응용 예제 – 자동차 번포한 영역 검출
  - 기계 학습 (Machine Learning) 알고리즘 중의 하나인 SVM (Support Vector Machine) 이용

12-2. SVM을 이용한 차량 번호 검출 프로그램 (2) – SVM 개념

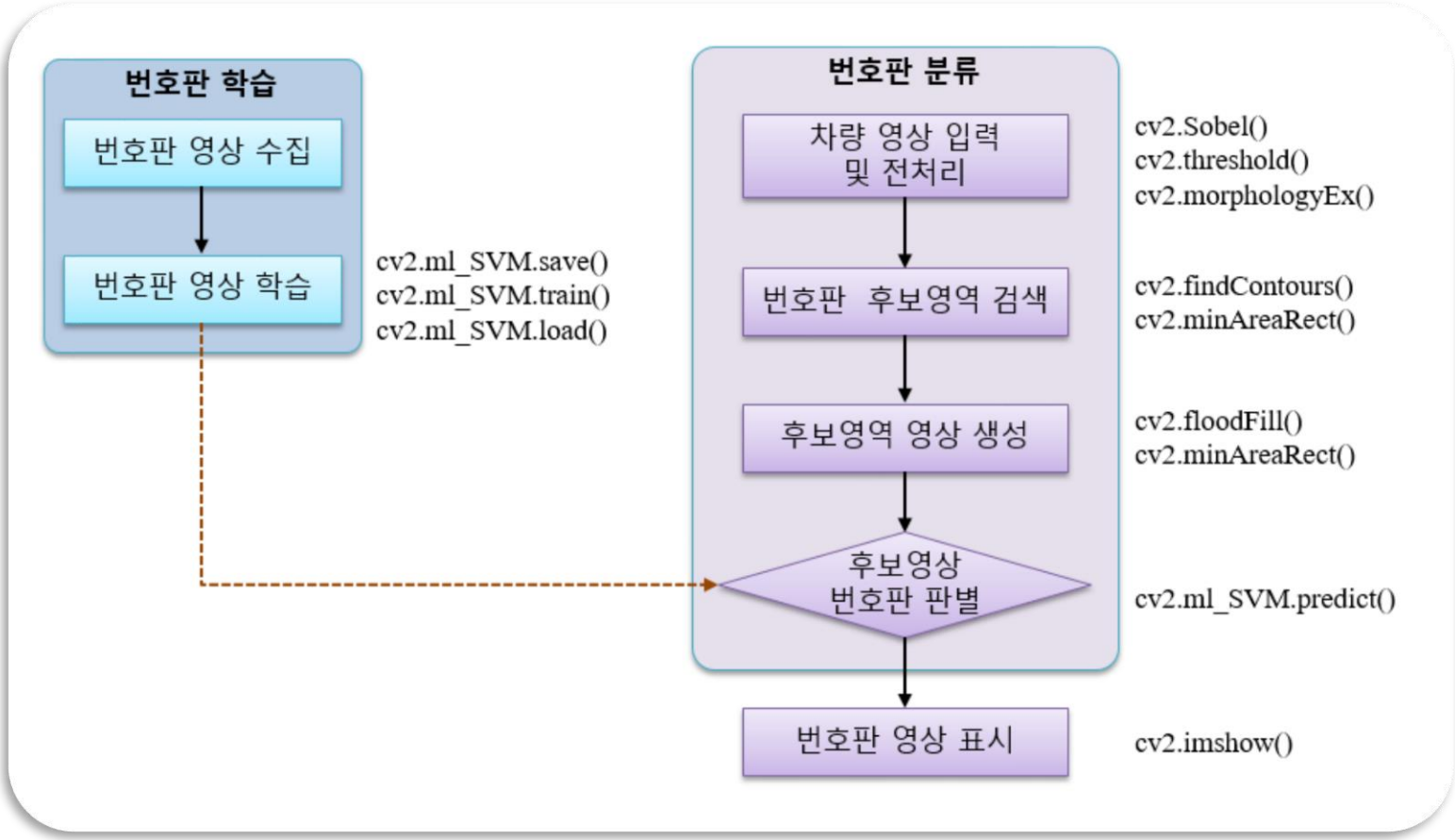
- 선형 판별법
  - 각 그룹 내에서 데이터간 거리 측정 ➔ 중심점 계산
  - 두 중심점에서 중간에 최적 분리 경계면 → 이 경계를 기준으로 새로운 데이터 분류 수행

- 서포트 벡터 머신 (SVM, Support Vector Machine)
  - 데이터들을 분리하는 분리 경계면 중에서 분류 데이터들과의 거리가 가장 먼 분리 경계면 찾음
  - 분리 경계와 실제 데이터들 사이의 마진 (Margin)이 가장 크도록 분리 경계를 설정



12-2. SVM을 이용한 차량 번호 검출 프로그램 (3) – 처리 과정

- 자동차 번호판 검출 전체 과정
  - 번호판 학습 모듈과 번호판 분류 모듈로 구성



12-2. SVM을 이용한 차량 번호 검출 프로그램 (4) – 번호판 영상 학습

- 번호판 영상의 수집
  - SVM을 수행하려면 먼 저 학습 데이터를 생성해서 학습 수행해야 함
    - 학습데이터 – 번호판 영상과 번호판 아닌 영상
  - 일반 차량 번호 공개 불가
    - 학습데이터 xml 파일로 제공: SVMtrain.xml
    - 직접 학습하고 싶다면
      - 번호판 영상 폴더 '../images/plate'
      - 번호판 영상 000.png ~ 069.png 로 저장
      - 번호판 아닌 영상 070.png ~ 139.png로 저장
      - 학습 영상의 크기 144 x 28

12-2. SVM을 이용한 차량 번호 검출 프로그램 (5) – 직접 학습 데이터 만들기

예제 12.2.1 번호판 영상 수집 및 학습 – 04.collect_carimage.py

```
01  import numpy as np, cv2
02
03
04  def SVM_create(type, max_iter, epsilon):
05      svm = cv2.ml.SVM_create()                          # SVM 객체 선언
06      ## SVM 파라미터 지정
07      svm.setType(cv2.ml.SVM_C_SVC)                      # C-Support Vector Classifica
08      svm.setKernel(cv2.ml.SVM_LINEAR)                   # 선형 SVM
09      svm.setGamma(1)                                    # 커널 함수의 감마값
10      svm.setC(1)                                        # 최적화를 위한 C 파라미터
11      svm.setTermCriteria((type, max_iter, epsilon))     # 학습 반복 조건 지정
12      reutrn svm
13
14  nsample = 140                                          # 학습 영상 총 개수
15  trainData = [cv2.imread("images/plate/%03d.png" %i, 0) for i in range(nsample)]
16  trainData = np.reshape(trainData, (nsample, -1)).astype('float32')
17  labels = np.zeros((nsample, 1), np.int32)              # 레이블 행렬
18  labels[:70] = 1                                        # 번호판 레이블 번호
19
20  print("SVM 객체 생성")
21  svm = SVM_create(cv2.TERM_CRITERIA_MAX_ITER, 1000, 1e-6)      # SVM 객체 생성
22  svm.train(trainData, cv2.ml.ROW_SAMPLE, labels)               # 학습 수행
23  svm.save("SVMtrain.xml")                                      # 학습된 데이터 저
24  print("SVM 객체 저장 완료")
```

오타

학습 영상 폴더

분류 결과로 지정될 값

파일로 제공됨

| 실행결과 |

- SVMtrain.xml 파일 내용 -

SVMtrain.xml - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말

```
<?xml version="1.0"?>
<opencv_storage>
<opencv_ml_svm>
 <format>3</format>
 <svmType>C_SVC</svmType>
 <kernel>
  <type>LINEAR</type></kernel>
 <C>1.</C>
 <term_criteria><iterations>1000</iterations></term_criteria>
 <var_count>4032</var_count>
 <class_count>2</class_count>
 <class_labels type_id="opencv-matrix">
  <rows>2</rows>
  <cols>1</cols>
  <dt>i</dt>
  <data>
   0 1</data></class_labels>
 <sv_total>1</sv_total>
 <support_vectors>
  <_>
   2.42166261e-05 4.39279438e-06 -2.54070528e-06 4.73219188e-06
   5.13863552e-06 9.51447419e-07 -1.26075986e-06 8.52291009e-07
   -2.42310171e-06 8.48960099e-07 2.61173932e-06 1.32524667e-06
   -1.74666255e-07 2.78180096e-06 6.92730009e-06 6.51901155e-06
```

Windows (CRLF) Ln 1, Col 22

12-2. SVM을 이용한 차량 번호 검출 프로그램 (6) – 번호판 후보영역 검색

- 전처리 함수 구현
  - 명암도 영상 변환, 블러링 및 소벨 에지 검출
  - 이진화 및 모폴로지 닫힘 (Close) 연산 수행

```
def preprocessing(car_no):
    image = cv2.imread("images/car/%02d.jpg" %car_no, cv2.IMREAD_COLOR)
    if image is None: return None, None

    kernel = np.ones((5, 17), np.uint8)                    # 닫힘 연산 마스크(커널)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)         # 명암도 영상 변환
    gray = cv2.blur(gray, (5, 5))                          # 블러링
    gray = cv2.Sobel(gray, cv2.CV_8U, 1, 0, 3)             # 수직 에지 검출

    th_img = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)[1]      # 이진화 수행
    morph = cv2.morphologyEx(th_img, cv2.MORPH_CLOSE, kernel, iterations=3)

    #cv2.imshow("th_img", th_img); cv2.imshow("morph", morph)      # 결과표시
    reutrn image, morph
```

가로로 긴 닫힘 연산 마스크

수직 소벨마스크 적용

닫힘 연산 3번 반복수행

12-2. SVM을 이용한 차량 번호 검출 프로그램 (7) – 번호판 후보영역 검색

| 실행결과 | - 전처리 수행 결과

소벨마스크 및
이진화 적용

닫힘 연산 3번 수행

12-2. SVM을 이용한 차량 번호 검출 프로그램 (8) – 번호판 후보영역 검색

- 번호판 후보영역 판정 함수

```
def verify_aspect_size(size):
    w, h = size
    if h == 0 or w == 0: reutrn False

    aspect = h / w if h > w else w / h          # 세로가 길면 역수 취함
    chk1 = 3000 < (h*w) < 12000                 # 번호판 넓이 조건
    chk2 = 2.0 < aspect < 6.5                   # 번호판 종횡비 조건
    reutrn (chk1 and chk2)

def find_candidates(image):
    results = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours = results[0] if int(cv2.__version__[0]) >= 4 else results[1]

    rects = [cv2.minAreaRect(c) for c in contours]    # 회전 사각형 반환
    candidates = [(tuple(map(int, center)), tuple(map(int, size)), angle) # 정수형 변환
                  for center, size, angle in rects if verify_aspect_size(size)]
    reutrn candidates
```

번호판 유사 모양 판단 기준
– 넓이 & 종횡비

12-2. SVM을 이용한 차량 번호 검출 프로그램 (9) – 번호판 후보영역 검색

예제 12.2.2 header/plate_preprocess.py

```
01  import numpy as np, cv2
02
03  def preprocessing(car_no): ···
17
18  def verify_aspect_size(size): ···
28
29  def find_candidates(image): ···
```

예제 12.2.2 번호판 후보 영역 검색 – 05.find_plates.cpp

```
01  from plate_preprocess import *                          # 전처
02
03  car_no = int(input("자동차 영상 번호(0~15): "))
04  image, morph = preprocessing(car_no)                    # 전처
05  if image is None: Exception("영상파일 읽기 에러")
06
07  candidates = find_candidates(morph)                     # 번호
08  for candidate in candidates:                            # 후보
09      pts = np.int32(cv2.boxPoints(candidate))
10      cv2.polylines(image, [pts], True, (0, 225,255), 2)  # 다중
11      print(candidate)
12
13  if not candidates:                                      # 리스
14      print("번호판 후보 영역 미검출")
15  cv2.imshow("image", image)
16  cv2.waitKey(0)
```

| 실행결과 | - 전처리 수행 결과

```
Run:  05.find_plates
C:\Python\python.exe D:/source/chap12/05.find_pla
자동차 영상 번호 (0~15): 0
((224, 271), (32, 171), -88.03858184814453)
((373, 184), (96, 33), -21.614778518676758)
((428, 45), (41, 136), -65.77225494384766)
```

12-2. SVM을 이용한 차량 번호 검출 프로그램 (10) – 번호판 후보영역 영상 생성

- 번호판 후보영역 영상 생성 과정
  - 후보 영역 개선 → 후보영역 재검증 → 후보영상 회전 보정 → 후보영상 생성

  - 후보영상 개선
    - 검출한 번호판 후보영역들은 이전영상으로 검출
    - 컬러 영상 이용해 번호판과 좀 더 유사한 영역으로 개선
    - Refine_candidate() 함수로 구현
      - cv2.floodFill() 함수 – 영역 채움 함수
      - cv2.minAreaRect() 함수 – 회전 사각형의 최소 영역 반환

12-2. SVM을 이용한 차량 번호 검출 프로그램 (11) – 번호판 후보영역 영상 생성

- 후보영상 개선 함수 – 컬러 활용

```
01  def color_candidate_img(image, candi_center):
02      h, w = image.shape[:2]
03      fill = np.zeros((h + 2, w + 2), np.uint8)                    # 채움 행렬
04      dif1, dif2 = (25, 25, 25), (25, 25, 25)                      # 채움 색상 범위
05      flags = 0xff00 + 4 + cv2.FLOODFILL_FIXED_RANGE               # 채움 방향 및 방법
06      flags += cv2.FLOODFILL_MASK_ONLY                             # 결과 영상만 채움
07
08      ## 후보 영역을 유사 컬러로 채우기
09      pts = np.random.randint( -15, 15, (20, 2) )                  # 임의 좌표 20개 생성
10      pts = pts + center                                           # 중심 좌표로 평행이동
11      for x, y in pts:                                             # 임의 좌표 순회
12          if 0 <= x < w and 0 <= y < h:                            # 후보 영역 내부이면
13              _, _, fill, _ = cv2.floodFill(image, fill, (x, y), 255, dif1, dif2, flags)  #채움누적
14
15          reutrn cv2.threshold(fill, 120, 255, cv2.THRESH_BINARY)[1]
```

채움 점검 방향 -4방향

입력영상 변경 않고, 결과영상만 채움 수행

-15~15사이 랜덤 좌표 20개 생성

후보영역 중심 근처 20개 좌표

입력영상 변경 않고, 결과영상만 채움 수행

후보영역 중심 근처와 유사한 색을 채움 수행 – 20번 반복 수행해서 오류 최소화

12-2. SVM을 이용한 차량 번호 검출 프로그램 (12) – 번호판 후보영역 영상 생성

- 후보영상 보정 함수 rotate_plate() 구현
  - 검출 사각형이 회전각도 만큼 기울어져 있으면
    - 각도만큼 역회전 → 후보 영상 (번호판)을 수평 직사각형으로 배치 가능
    - 종횡비가 1 이상 → 세로가 긴 사각형 → 가로와 세로 값을 맞바꾸고 → 회전할 각도 90도 더하여 90도 회전
  - 회전 수행
    - 회전 변환 행렬 계산 - cv::getRotationMatrix2D()
    - 회전 변환 행렬로 원본 영상 회전 - cv::warpAffine()
    - 회전된 후보 영상 가져오기 (crop) - cv::getRectSubPix()
    - 144 x 28로 정규화 - cv::resize()

12-2. SVM을 이용한 차량 번호 검출 프로그램 (13) – 번호판 후보영역 영상 생성

- 후보영상 보정 함수

```
def rotate_plate(image, rect):
    center, (w, h), angle = rect                                    # 중심 좌표, 크기, 회전 각도
    if w < h :                                                      # 세로가 긴 영역이면
        w, h = h, w                                                 # 가로와 세로 맞바꿈
        angle += 90                                                 # 회전 각도 조정

    size = image.shape[1::-1]                                       # 행태와 크기는 역순
    rot_mat = cv2.getRotationMatrix2D(center, angle, 1)             # 회전 행렬 계산
    rot_img = cv2.warpAffine(image, rot_mat, size, cv2.INTER_CUBIC)     # 회전 변환

    crop_img = cv2.getRectSubPix(rot_img, (w, h), center)           # 후보 영역 가져오기
    crop_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)           # 명암도 영상
    reutrn cv2.resize(crop_img, (144, 28))                          # 크기변경 후 반환
```

세로로 긴 영역이면

회전 변환 행렬

회전사각형 중심점에서 상하좌우로 후보영역만큼 영상 가져오기

12-2. SVM을 이용한 차량 번호 검출 프로그램 (14) – 번호판 후보영역 영상 생성

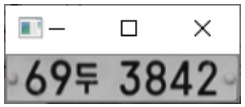
예제 12.2.3 후보 영상 검증 헤더 파일 – header/plate_candida

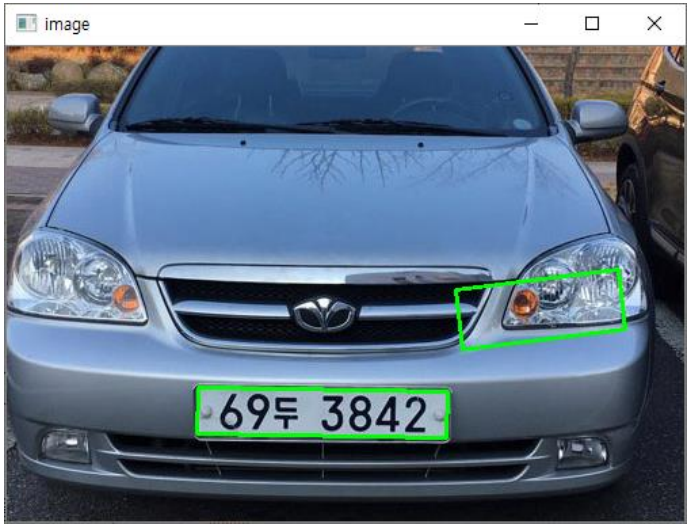
```
01  import numpy as np, cv2
02
03  def color_candidate_img(image, center): …
19
20  def rotate_plate(image, rect): …                    #
```

예제 12.2.3 번호판 영상 생성 – 06.correct_plate.py

```
01  from plate_preprocess import *              # 전처리 및 후보 영역 검출 함수
02  from plate_candidate import *               # 후보 영역 개선 및 후보 영상 생성 함수
03
04  car_no = 0
05  image, morph = preprocessing(car_no)                  # 전처리– 소벨 & 열림연산
06  candidates = find_candidates(morph)                   # 번호판 후보 영역 검색
07
08  fills = [color_candidate_img(image, size) for size, _, _ in candidates]    # 후보 영역 재생성
09  new_candis = [find_candidates(fill) for fill in fills]          # 재생성 영역 검사
10  new_candis = [cand[0] for cand in new_candis if cand]           # 재후보 있으면 저장
11  candidate_imgs =[rotate_plate(image, cand) for cand in new_candis]  # 후보 영역 영상
12
13  for i, img in enumerate(candidate_imgs):                        # 후보 영상 표시
14      cv2.polylines(image, [np.int32(cv2.boxPoints(new_candis[i]))], True, (0, 225,255), 2)
15      cv2.imshow("candidate_img - " + str(i), img)
16
17  cv2.imshow("image" , image)
18  cv2.waitKey()
```

| 실행결과 |





12-2. SVM을 이용한 차량 번호 검출 프로그램 (15) – 후보 영상의 번호판 판별

- 번호판 판별 위해 SVM 분류 수행
  - SVM_create() 함수 호출 – SVM 클래스 객체 생성
  - svm 객체의 train() 함수 호출 – 학습 수행

  - 후보영상 분류 수행 함수 classify_plates() 구현
    - 입력 인수 – 학습을 수행한 svm 객체, 번호판 후보 영상들
    - 분류 수행 위해
      - 회전 보정된 번호판 후보영상을 1행 데이터로 변환
      - 자료형 CV_32F로 변경
    - svm 객체의 predict() 함수 호출 → 분류 수행
      - 학습 및 분류를 위한 행렬 데이터는 1행(1열) 데이터, float형

12-2. SVM을 이용한 차량 번호 검출 프로그램 (16) – 후보 영상의 번호판 판별

예제 12.2.4 번호판 인식 프로그램 완성 – 07.classify_plate.py

```
01  from plate_preprocess import *              # 전처리 및 후보 영역 검출 함수
02  from plate_candidate import *               # 후보 영역 개선 및 후보 영상 생성 함수
03
04  car_no = int(input("자동차 영상 번호(0~15): "))
05  image, morph = preprocessing(car_no)
06  candidates = find_candidates(morph)         # 번호판 후보 영역 검색
07
08  fills = [color_candidate_img(image, size) for size, _, _ in candidates]    # 후보 영역 재생성
09  new_candis = [find_candidates(fill) for fill in fills]          # 재생성 영역 검사
10  new_candis = [cand[0] for cand in new_candis if cand]           # 재후보 있으면 저장
11  candidate_img =[rotate_plate(image, cand) for cand in new_candis]    # 후보 영역 영상
12
13  svm = cv2.ml.SVM_load("SVMtrain.xml")                           # 학습된 데이터 적재
14  rows = np.reshape(candidate_imgs, (len(candidate_imgs), -1))    # 1행 데이터들로 변환
15  _, results = svm.predict(rows.astype('float32'))                # 분류 수행
16  correct = np.where(results == 1)[0]                             # 정답 인덱스 찾기
17
18  print("분류 결과:\n", results)
19  print("번호판 영상 인덱스:", correct )
20
```

학습하여 저장된 xml 파일 로드

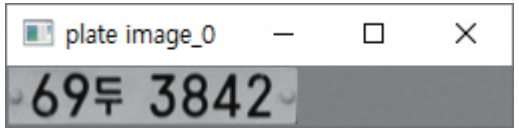
분류 결과 행렬로 반환

results 행렬에서 번호판으로 분류된 인덱스

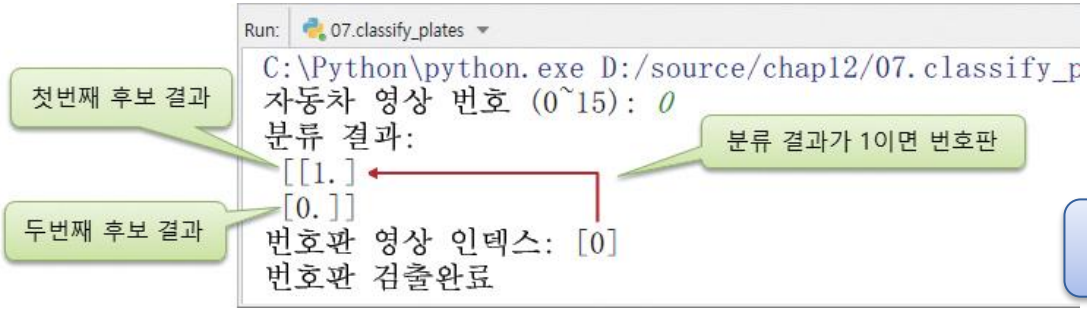
12-2. SVM을 이용한 차량 번호 검출 프로그램 (17) – 후보 영상의 번호판 판별

```
21  for i, idx in enumerate(correct):
22      cv2.imshow("plate image_" + str(i), candidate_imgs[idx])
23      cv2.resizeWindow("plate image_" + str(i), (250,28))          # 윈도우
24
25  for i, candi in enumerate(new_candis):
26      color = (0, 255, 0) if i in correct else (0, 0, 255)      # 후보영역
27      cv2.polylines(image, [np.int32(cv2.boxPoints(candi))], True, color, 2)
28
29  print("번호판 검출완료") if len(correct)>0 else print("번호판 미검출")
30
31  cv2.imshow("image", image)
32  cv2.waitKey(0)
```

후보영상 번호로 후보영상들에서 가져와 윈도우에 표시

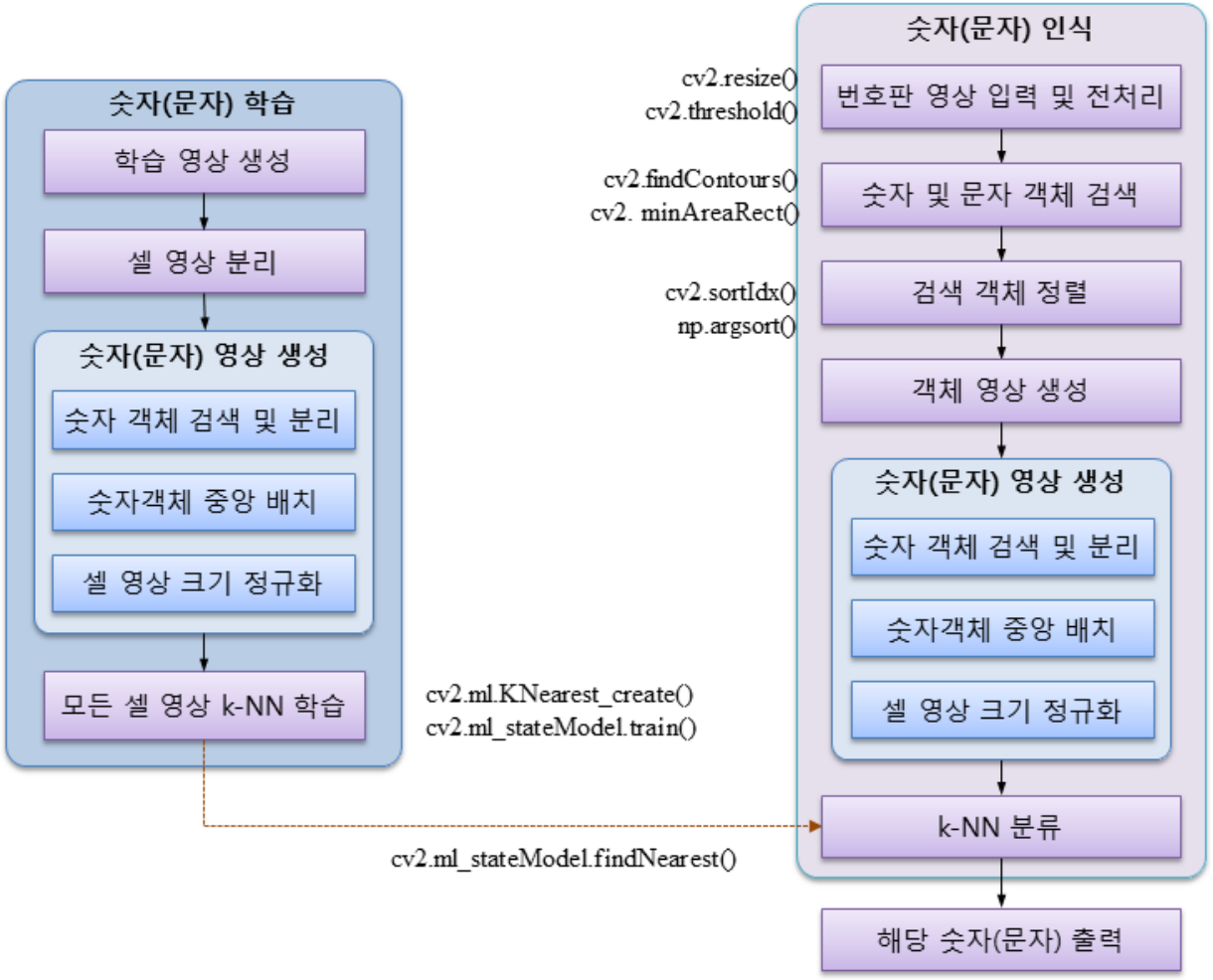


| 실행결과 |



12-3. k-NN을 이용한 차량 번호 인식 (1) – 전체 처리 과정

- 차량 번호판의 숫자와 문자 인식
  - 숫자(문자) 학습 모듈과 숫자(문자) 인식 모듈로 구성



12-3. k-NN을 이용한 차량 번호 인식 (2) – 영상 학습

- 숫자 및 문자 인식에 k-NN 알고리즘 사용
  - 예제_10.3.4에서 KNN_number.py 소스
    - 학습 데이터 로드 → knn 객체 생성 → 학습수행을 위한 kNN_train() 함수로 변경

선출 이웃 수

학습 데이터 전체 영상 파일

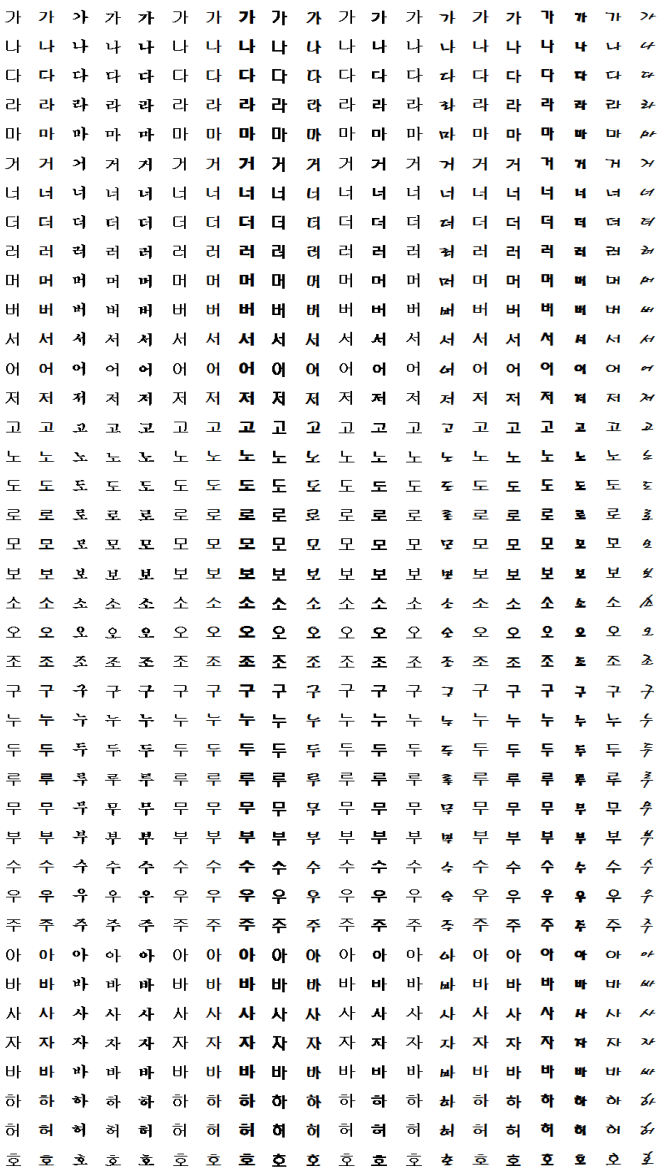
학습 영상 이진화

1행이 하나의 샘플

```
def kNN_train(train_fname, K, nclass, nsample):
    size = (40, 40)                                          # 숫자 영상 크기
    train_img = cv2.imread(train_fname, cv2.IMREAD_GRAYSCALE) # 학습 영상 읽기
    h, w = train_img.shape[:2]
    dx = w % size[0] // 2                                    # 좌우 여백 크기
    dy = h % size[1] // 2                                    # 상하 여백 크기
    train_img = train_img[dy:h-dy-1, dx:w-dx-1]               # 학습 영상 여백 제거
    cv2.threshold(train_img, 32, 255, cv2.THRESH_BINARY, train_img)
    cells = [np.hsplit(row, nsample) for row in np.vsplit(train_img, nclass)] # 셀 영상 분리
    nums = [find_number(c) for c in np.reshape(cells, (-1, 40, 40))]          # 숫자 객체 검출
    trainData = np.array([place_middle(n, size) for n in nums])               # 객체 중앙 배치
    labels = np.array([i for i in range(nclass) for j in range(nsample)], np.float32)

    knn = cv2.ml.KNearest_create()
    knn.train(trainData, cv2.ml.ROW_SAMPLE, labels)                           # k-NN 학습 수행
    reutrn knn
```

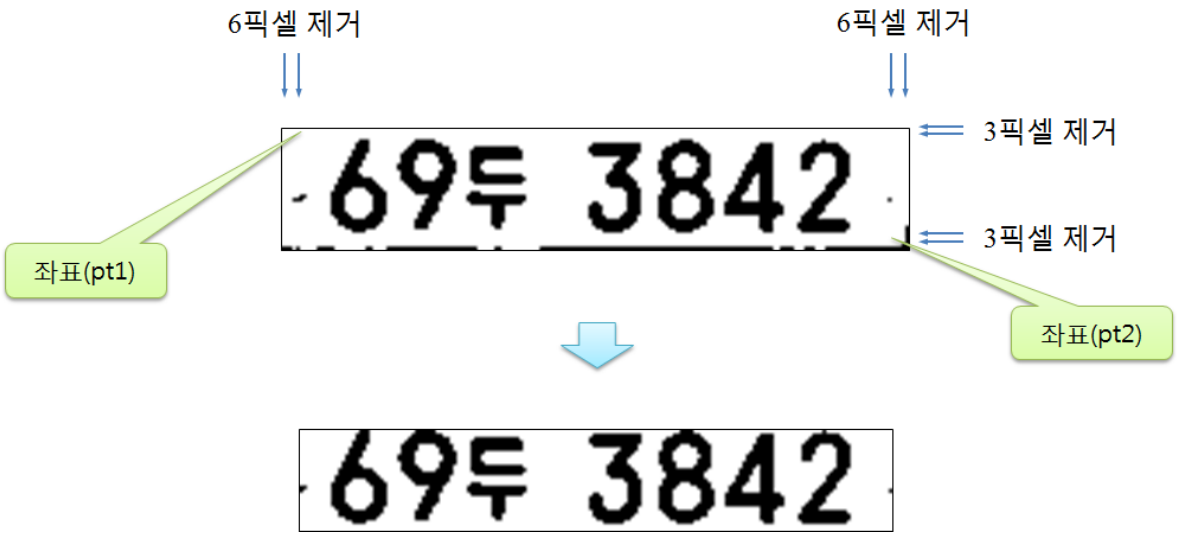
12-3. k-NN을 이용한 차량 번호 인식 (3) – 영상 학습

- 숫자 학습 영상
  - 10.3.4절 예시파일 그대로 사용
    - 'train_numbers.png'
- 문자 학습 영상 구성
  - 아래 한글 파일로 작성
  - 영상 파일
  - 숫자 학습
    - kNN_train(..., K1, 10, 20)
  - 문자 학습
    - kNN_train(..., K2, 25, 20)



12-3. k-NN을 이용한 차량 번호 인식 (4) – 전처리

- 전처리 과정
  - 번호판 영상의 크기 180x35로 변경한 후에 이진화 수행
  - 모서리 부분 (좌우로 6픽셀, 상하 3픽셀) 제거
    - 상하좌우 모서리 부분에서 잡음



12-3. k-NN을 이용한 차량 번호 인식 (5) – 전처리

- 전처리 수행 함수

```
def preprocessing_plate(plate_img):
    plate_img = cv2.resize(plate_img, (180, 35))                # 번호판 영상 크기 정규화
    flag = cv2.THRESH_BINARY | cv2.THRESH_OTSU                  # 이진화 방법
    cv2.threshold(plate_img, 32, 255, flag, plate_img)          # 이진화

    h, w = plate_img.shape[:2]
    dx, dy = (6, 3)                                             # 좌우 6화소, 상하 3화소
    ret_img= plate_img[dy:h-dy, dx:w-dx]                        # 여백 제거
    reutrn ret_img
```

모서리 제거 사각형

12-3. k-NN을 이용한 차량 번호 인식 (6) – 객체 검색

- 숫자와 문자 객체 찾기 함수 find_objects() 로 구현
  - Cv::findContours() 함수로 객체 외곽선 검출
  - 검출 객체 넓이로 잡은 제거
    - 150보다 작은 것은 잡음으로 처리
  - 검출 객체 종횡비로 잡은 제거
    - 가로로 긴 객체는 숫자, 문자 아님

  - 번호판은 숫자와 문자 공존
    - 번호판 영상의 크기가 일정하기 때문에 문자 영역 위치 확인 가능
      - 45~80 픽셀 범위 객체는 문자로 인지
    - 문자객체 여러 부분으로 분리되어 검출 가능
      - 문자 객체 영역들을 논리합으로 누적하여 하나의 영역으로 만듦
      - 문자 객체 잡음은 넓이 60미만

12-3. k-NN을 이용한 차량 번호 인식 (7) – 객체 검색

```
01  def find_objects(img):
02      results=cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
03      contours = results[0] if int(cv2.__version__[0]) >= 4 else results[1]
04
05      rois = [cv2.boundingRect(contour) for contour in contours]
06      rois = [(x, y, w, h, w*h) for x, y, w, h in rois if w / h < 2.5]
07      text_rois = [(x, y, x+w, y+h) for x, y, w, h, a in rois if 45<x<80 and a > 60]
08      num_rois = [(x, y, w, h) for x, y, w, h, a in rois if not(45<x<80) and a > 150]
09
10      if text_rois:                                   # 분리된 문자 영역 누적
11          pts= np.sort(atext_rois, axis=0)            # 세로 방향 정렬
12          x0, y0 = pts[0, 0:2]                        # 시작좌표 중 최소인 x, y 좌표
13          x1, y1 = pts[-1, 2:]                        # 종료좌표 중 최대인 x, y 좌표
14          w, h = x1-x0, y1-y0                         # 너비, 높이 계산
15          num_rois.append((x0, y0, w, h))             # 문자 영역 구성 및 저장
16      reutrn num_rois
```

객체 외곽선 검출

문자 객체 잡음 기준

숫자 객체 잡음 기준

12-3. k-NN을 이용한 차량 번호 인식 (8) – 숫자, 문자 인식

- 숫자와 문자 사각형들 벡터(object_rects)에 저장
  - Cv2.findContours() 함수로 객체 외곽선 검출시
    - 검출 사각형들 위치 정렬되어 있지 않음 → 각 사각형이 몇 번째 숫자/문자 인지 확인 불가
  - 검출 객체를 x좌표 순으로 정렬 필요
    - np.argsort() 함수 사용하여 정렬 인덱스 사용

- 검출 객체 영상 분류하여 숫자 및 문자 인식
  - Classify_numbers() 함수로 구현
    - 10장 3절 숫자 인식 예제에서 사용한 함수 사용
      - find_numbers() 함수 – 숫자 객체 검출
      - find_middle() 함수 – 숫자 중앙 배치 및 1행 데이터 생성

12-3. k-NN을 이용한 차량 번호 인식 (9) – 숫자, 문자 인식

```
01  def classify_numbers(cells, nknn, tknn, K1, K2, object_rois):
02      if len(cells) != 7:
03          print("검출된 숫자(문자)가 7개가 아닙니다.")
04          reutrn
05
06      texts =  "가나다라마거너더러머버서어저고노도로모보" \
07               "소오조구누두루무부수우주아바사자바하허호"          # 학습 문자 집합
08      numbers = [find_number(cell) for cell in cells]          # 숫자 객체 찾기
09      datas = [place_middle(num, (40,40)) for num in numbers]  # 중심 배치
10      datas = np.reshape(datas, (len(datas), -1                # 환
11
12      idx = np.argsort(object_rois, axis=0).T[0]               # x좌표 정렬 인덱스
13      text = datas[idx[2]].reshape(1,-1)
14      _, resp1, _, _ = nknn.findNearest(datas, K1)             # 숫자 k-NN 분류
15      _, [[resp2]], _, _ = tknn.findNearest(text, K2)          # 문자 k-NN 분류
16
17      resp1 =  resp1.flatten().astype('int')                   # 전개 및 정수변환
18      results = resp1[idx].astype('str')
19      results[2] = texts[int(resp2)]
20
21      print("정렬 인덱스:", idx)
22      print("숫자 객체 결과:", resp1)
23      print("문자 객체 결과:", int(resp2))
24      print("분류 결과: ", " ".join(results))
```

문자 객체 분류 결과인 레이블값에 대응하는 문자를 배열로 저장

각 검출객체 x좌표에 따라 순서 정함

x좌표로 세번째가 문자 객체

6개 객체에 대한 분류 결과를 2차원 행렬

문제 객체 분류 결과는 1차원 행렬로

오타

숫자 객체를 x좌표 순서에 맞게 정렬 후 문자로 변환

숫자들을 하나로 합쳐 출력

12-3. k-NN을 이용한 차량 번호 인식 (10) – 숫자, 문자 인식

예제 12.3.1 번호판 숫자 인식 프로그램 완성 – 08.classify_numbers.cpp

```
01  from plate_preprocess import *                          # 전처리 및 후보 영역 검출 함수
02  from plate_candidate import *
03  from plate_classify import *                            # k-NN 학습 및 분류 함수 임포트
04
05  car_no = int(input("자동차 영상 번호(0~15): "))
06  image, morph = preprocessing(0)                         # 전처리
07  candidates = find_candidates(morph)                     # 후보 영역 검색
08
09  fills = [refine_candidate_img(image, size) for size, _, _ in candidates]
10  new_candi = [find_candidates(fill) for fill in fills]
11  new_candi = [cand[0] for cand in new_candi if cand]
12  candidate_imgs = [rotate_plate(image, cand) for cand in new_candi]
13
14  svm = cv2.ml.SVM_load("SVMTrain.xml")                   # 학습된 데이터 적재
15  rows = np.reshape(candidate_imgs, (len(candidate_imgs), -1))   # 1행 데이터들로 변환
16  _, results = svm.predict(rows.astype('float32'))        # 분류 수행
17  result = np.where(results.flatten() == 1)[0]            # 1인 값의 위치 찾기
18
19  plate_no = result[0] if len(result)>0 else -1           # 번호판 판정
20
21  K1, K2 = 10, 10
22  nknn = kNN_train("images/train_numbers.png", K1, 10, 20)    # 숫자 학습
23  tknn = kNN_train("images/train_texts.png", K2, 40, 20)      # 문자 학습
```

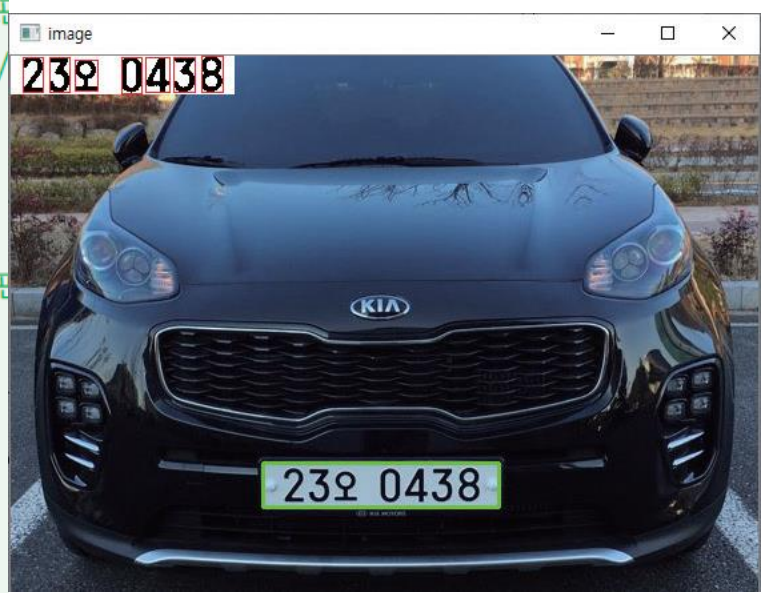
12-3. k-NN을 이용한 차량 번호 인식 (11) – 숫자, 문자 인식

```
25  if plate_no >= 0:
26      plate_img = preprocessing_plate(candidate_imgs[plate_no]) # 번호판
27      cells_roi = find_objects(cv2.bitwise_not(plate_img))
28      cells = [plate_img[y:y+h, x:x+w] for x, y, w, h in cells_roi]   # 셀(숫
29
30      classify_numbers(cells, nknn, tknn, K1, K2, cells_roi)      # 숫자/
31
32      pts = np.int32(cv2.boxPoints(candidates[plate_no]))
33      cv2.polylines(image, [pts], True,  (0, 255, 0), 2)          # 번호판
34
35      color_plate = cv2.cvtColor(plate_img, cv2.COLOR_GRAY2BGR)  # 번호판
36      for x, y, w, h in cells_roi:                                # 숫자/
37          cv2.rectangle(color_plate, (x,y), (x+w, y+h), (0, 0, 255), 1)
38
39      h, w = color_plate.shape[:2]
40      image[0:h, 0:w] = color_plate                               # 번호판
41  else:
42      print("번호판 미검출")
43
44  cv2.imshow("image", image)
45  cv2.waitKey(0)
```

번호판 영상을
원본 영상 상단에 복사

| 실행결과 |

```
Run:  08.classify_number
C:\Python\python.exe D:/source/chap1
자동차 영상 번호 (0~20): 1
정렬 인덱스: [5 0 6 4 3 2 1]
숫자 분류 결과: [3 8 3 4 0 2 2]
문자 분류 결과: 21
분류결과: 2 3 오 0 4 3 8
```



Q & A