

영상 분할과 객체 검출

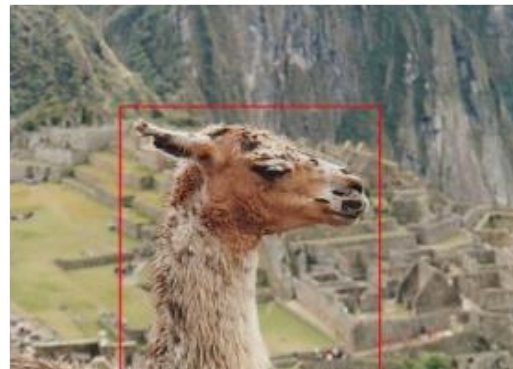
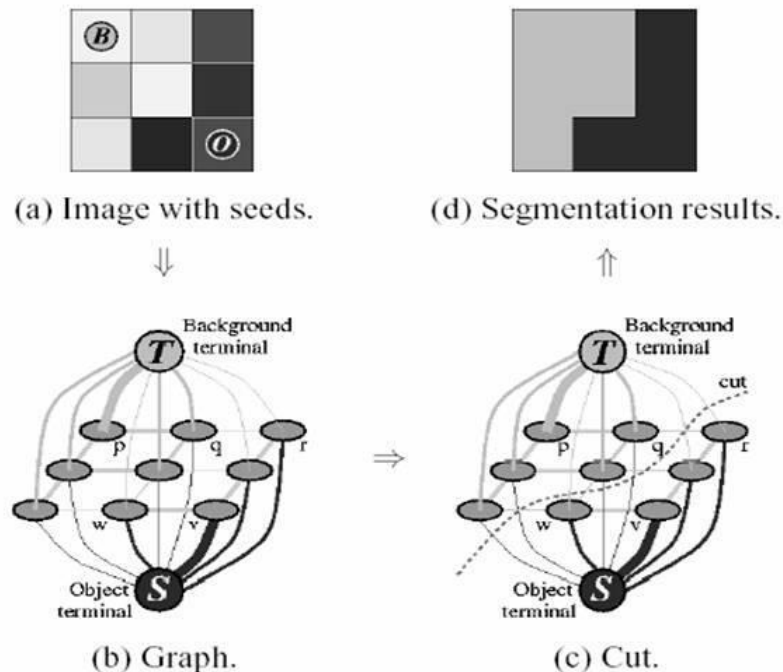
그랩컷

✔ 그랩컷(GrabCut)이란?

- 그래프 컷(graph cut)기반 영역 분할 알고리즘
- 영상의 픽셀을 그래프 정점으로 간주하고 픽셀들을 두 개의 그룹으로 나누는 최적의 컷 (Max Flow Minimum Cut) 을 찾는 방식

✔ 그랩컷 영상 분할 동작 방식

- 사각형 지정 자동 분할
- 사용자가 지정한 전경 배경 정보를 활용하여 영상 분할



그랩컷

✓ 그랩컷 함수

```
cv2.grabCut(img, mask, rect, bgdModel, fgdModel, iterCount, mode=None)  
--> mask, bgdModel , fgdModel
```

- img: 입력 영상. 8 비트 3 채널.
- mask: 입출력 마스크. cv2.GC_BGD(0), cv2.GC_FGD(1), cv2.GC_PR_BGD(2), cv2.GC_PR_FGD(3) 네 개의 값으로 구성됨 .
cv2.GC_INIT_WITH_RECT 모드로 초기화
- rect: ROI 영역. cv2.GC_INIT_WITH_RECT 모드에서만 사용됨
- bgdModel: 임시 배경 모델 행렬. 같은 영상 처리 시에는 변경 금지
- fgdModel: 임시 전경 모델 행렬 . 같은 영상 처리 시에는 변경 금지
- iterCount: 결과 생성을 위한 반복 횟수 .
- mode: cv2.GC_ 로 시작하는 모드 상수 . 보통 cv2.GC_INIT_WITH_RECT 모드로 초기화하고,
cv2.GC_INIT_WITH_MASK 모드로 업데이트함

그랩컷

✔ 그랩컷 영상 분할 예제

```
1 import sys
2 import numpy as np
3 import cv2
4
5 # 입력 영상 불러오기
6 src = cv2.imread('nemo.jpg')
7
8 if src is None:
9     print('Image load failed!')
10    sys.exit()
11
12 # 사각형 지정을 통한 초기 분할
13 rc = cv2.selectROI(src)
14 mask = np.zeros(src.shape[:2], np.uint8)
15
16 cv2.grabCut(src, mask, rc, None, None, 5, cv2.GC_INIT_WITH_RECT)
```

그랩컷

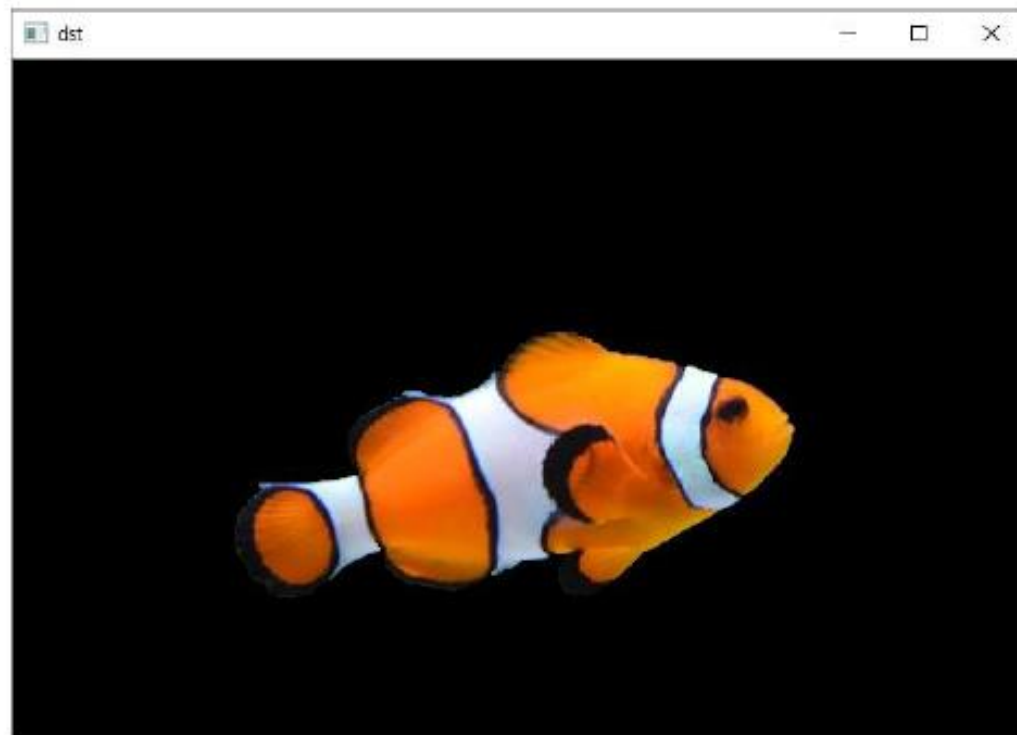
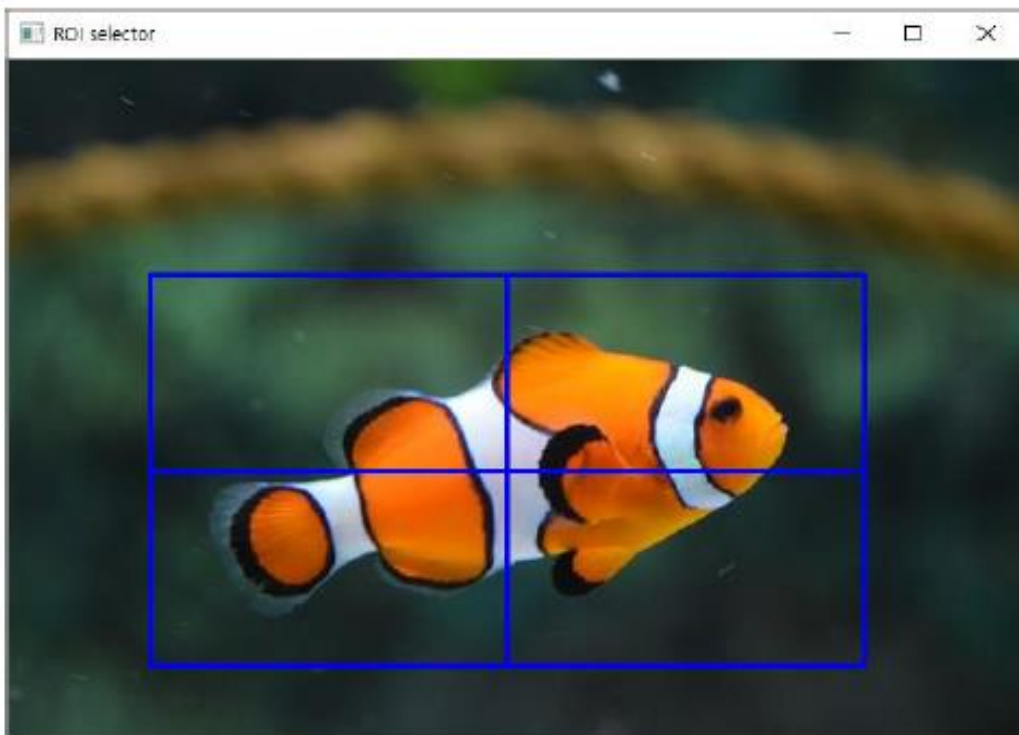
✓ 그랩컷 영상 분할 예제

```
17
18 # 0: cv2.GC_BGD, 2: cv2.GC_PR_BGD
19 mask2 = np.where((mask == 0) | (mask == 2), 0, 1).astype('uint8')
20 dst = src * mask2[:, :, np.newaxis]
21
22 # 초기 분할 결과 출력
23 cv2.imshow('dst', dst)
24 cv2.waitKey()
25 cv2.destroyAllWindows()
```

mask 행렬에서 값이 0 또는 2인 원소는 0으로, 그렇지 않은 원소는 1로 설정

그랩컷

✔ 그랩컷 영상 분할 예제 실행 결과



그랩컷

✔ 마우스를 활용한 그랩컷 영상 분할 예제

```
1  import sys
2  import numpy as np
3  import cv2
4
5  # 입력 영상 불러오기
6  src = cv2.imread('messi5.jpg')
7
8  if src is None:
9      print('Image load failed!')
10     sys.exit()
11
12 # 사각형 지정을 통한 초기 분할
13 mask = np.zeros(src.shape[:2], np.uint8) # 마스크
14 bgdModel = np.zeros((1, 65), np.float64) # 배경 모델
15 fgdModel = np.zeros((1, 65), np.float64) # 전경 모델
16
17 rc = cv2.selectROI(src)
```

그랩컷

✔ 마우스를 활용한 그랩컷 영상 분할 예제

```
18
19 cv2.grabCut(src, mask, rc, bgdModel, fgdModel, 1, cv2.GC_INIT_WITH_RECT)
20
21 # 0: cv2.GC_BGD, 2: cv2.GC_PR_BGD
22 mask2 = np.where((mask == 0) | (mask == 2), 0, 1).astype('uint8')
23 dst = src * mask2[:, :, np.newaxis]
24
25 # 초기 분할 결과 출력
26 cv2.imshow('dst', dst)
27
```


그랩컷

✔ 마우스를 활용

```
28      # 마우스 이벤트 처리 함수 등록
29      def on_mouse(event, x, y, flags, param):
30          if event == cv2.EVENT_LBUTTONDOWN:
31              cv2.circle(dst, (x, y), 3, (255, 0, 0), -1)
32              cv2.circle(mask, (x, y), 3, cv2.GC_FGD, -1)
33              cv2.imshow('dst', dst)
34          elif event == cv2.EVENT_RBUTTONDOWN:
35              cv2.circle(dst, (x, y), 3, (0, 0, 255), -1)
36              cv2.circle(mask, (x, y), 3, cv2.GC_BGD, -1)
37              cv2.imshow('dst', dst)
38          elif event == cv2.EVENT_MOUSEMOVE:
39              if flags & cv2.EVENT_FLAG_LBUTTON:
40                  cv2.circle(dst, (x, y), 3, (255, 0, 0), -1)
41                  cv2.circle(mask, (x, y), 3, cv2.GC_FGD, -1)
42                  cv2.imshow('dst', dst)
43              elif flags & cv2.EVENT_FLAG_RBUTTON:
44                  cv2.circle(dst, (x, y), 3, (0, 0, 255), -1)
45                  cv2.circle(mask, (x, y), 3, cv2.GC_BGD, -1)
46                  cv2.imshow('dst', dst)
47
```

그랩컷

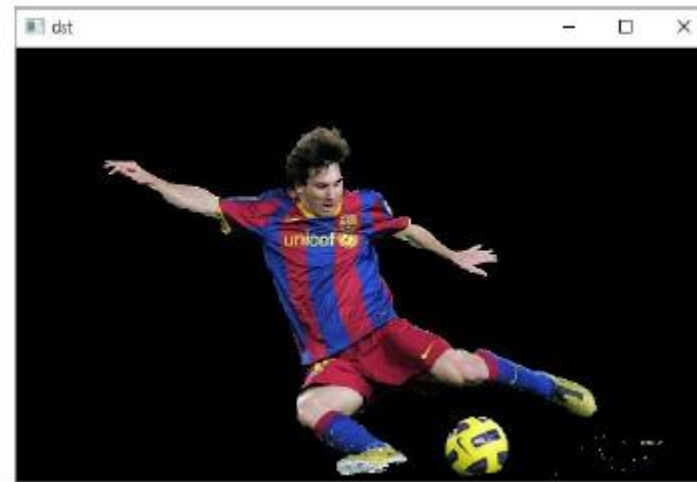
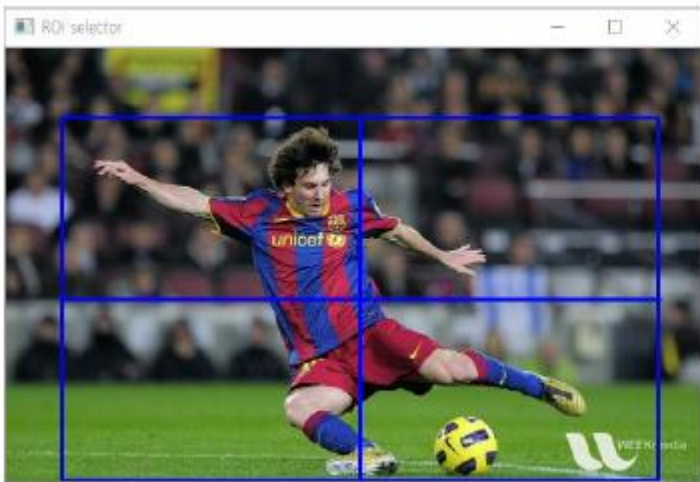
✔ 마우스를 활용한 그랩컷 영상 분할 예제

```
48     cv2.setMouseCallback('dst', on_mouse)
49
50     while True:
51         key = cv2.waitKey()
52         if key == 13: # ENTER
53             # 사용자가 지정한 전경/배경 정보를 활용하여 영상 분할
54             cv2.grabCut(src, mask, rc, bgdModel, fgdModel, 1, cv2.GC_INIT_WITH_MASK)
55             mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
56             dst = src * mask2[:, :, np.newaxis]
57             cv2.imshow('dst', dst)
58         elif key == 27:
59             break
60
61     cv2.destroyAllWindows()
```

그랩컷

✔ 마우스를 활용한 그랩컷 영상 분할 예제

- 초기 영역은 ROI selector 창에서 사각형 지정
- 초기 분할 결과 dst 창에서 전경은 마우스 왼쪽 버튼 드래그 파란색), 배경은 마우스 오른쪽 버튼 드래그 빨간색)
→ ENTER 키 입력 시 영상 재분할



모멘트 기반 객체 검출

✓ 모멘트(Moments)란?

- 영상의 형태를 표현하는 일련의 실수값
- 특정 함수 집합과의 상관 관계(correlation) 형태로 계산

$$\text{(e.g.) Geometric moments: } m_{pq} = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} x^p y^q f(x, y)$$

- Geometric moments, Central moments, Normalized central moments, Legendre moments, Complex moments, Zernike moments, ART(Angular Radial Transform), etc.

✓ Hu의 7개 불변 모멘트(Hu's seven invariant moments)

- 3차 이하의 정규화된 중심 모멘트를 조합하여 만든 7개의 모멘트 값
- 영상의 크기, 회전, 이동, 대칭 변환에 불변

모멘트 기반 객체 검출

✓ 모양 비교 함수

cv2.matchShapes(contour1, contour2, method, parameter) --> retval

- contour1: 첫 번째 외곽선 또는 그레이스케일 영상
- contour2: 두 번째 외곽선 또는 그레이스케일 영상
- method: 비교 방법 지정 . cv2.CONTOURS_MATCH_I1, cv2.CONTOURS_MATCH_I2, cv2.CONTOURS_MATCH_I3 중 하나 사용 .
- parameter: 사용되지 않음 . 0 지정
- retval: 두 외곽선 또는 그레이스케일 영상 사이의 거리 (
- 참고사항
 - Hu 의 불변모멘트를 이용하여 두 외곽선 또는 영상의 모양을 비교
→ 크기, 회전, 이동, 대칭 변환에 강인

모멘트 기반 객체 검출

✓ 모양 비교 함수

- method

Enumerator	
CONTOURS_MATCH_I1 Python: cv.CONTOURS_MATCH_I1	$I_1(A, B) = \sum_{i=1...7} \left \frac{1}{m_i^A} - \frac{1}{m_i^B} \right $
CONTOURS_MATCH_I2 Python: cv.CONTOURS_MATCH_I2	$I_2(A, B) = \sum_{i=1...7} m_i^A - m_i^B $
CONTOURS_MATCH_I3 Python: cv.CONTOURS_MATCH_I3	$I_3(A, B) = \max_{i=1...7} \frac{ m_i^A - m_i^B }{ m_i^A }$

where m_i^A and m_i^B are log-scale Hu's moments.

모멘트 기반 객체 검출

✓ 모멘트 기반 객체 검출 예제

```
1  import sys
2  import numpy as np
3  import cv2
4
5  # 영상 불러오기
6  obj = cv2.imread('spades.png', cv2.IMREAD_GRAYSCALE)
7  src = cv2.imread('symbols.png', cv2.IMREAD_GRAYSCALE)
8
9  if src is None or obj is None:
10     print('Image load failed!')
11     sys.exit()
12
13  # 객체 영상 외곽선 검출
14  _, obj_bin = cv2.threshold(obj, 128, 255, cv2.THRESH_BINARY_INV)
15  obj_contours, _ = cv2.findContours(obj_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
16  obj_pts = obj_contours[0]
```

모멘트 기반 객체 검출



```
17
18 # 입력 영상 분석
19 _, src_bin = cv2.threshold(src, 128, 255, cv2.THRESH_BINARY_INV)
20 contours, _ = cv2.findContours(src_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
21
22 # 결과 영상
23 dst = cv2.cvtColor(src, cv2.COLOR_GRAY2BGR)
24
25 # 입력 영상의 모든 객체 영역에 대해서
26 for pts in contours:
27     if cv2.contourArea(pts) < 1000:
28         continue
29
30     rc = cv2.boundingRect(pts)
31     cv2.rectangle(dst, rc, (255, 0, 0), 1)
32
33 # 모양 비교
34 dist = cv2.matchShapes(obj_pts, pts, cv2.CONTOURS_MATCH_I3, 0)
35
```

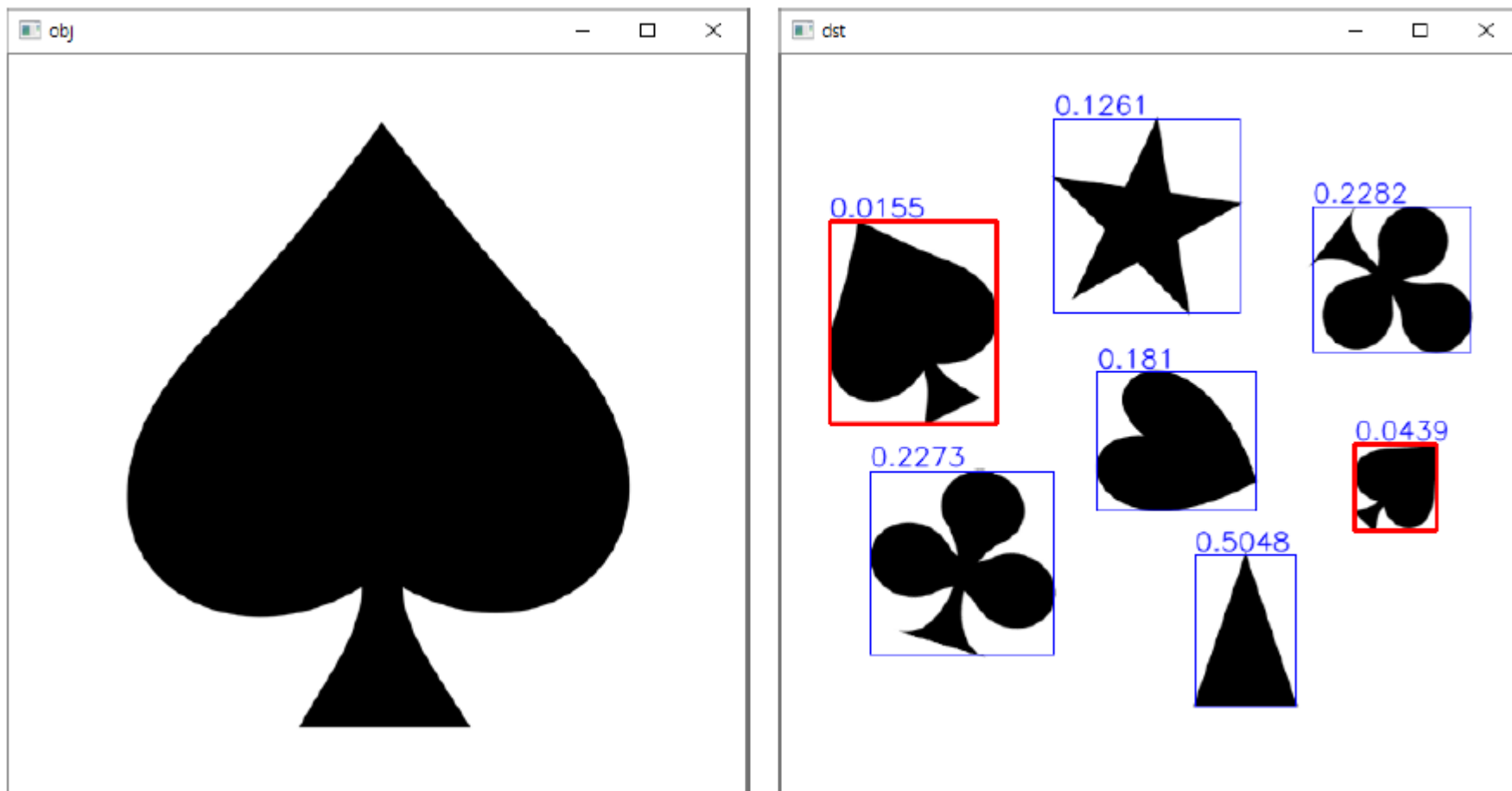

모멘트 기반 객체 검출

✓ 모멘트 기반 객체 검출 예제

```
36         cv2.putText(dst, str(round(dist, 4)), (rc[0], rc[1] - 3),
37         |             cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 1, cv2.LINE_AA)
38
39         if dist < 0.1:
40         |             cv2.rectangle(dst, rc, (0, 0, 255), 2)
41
42     cv2.imshow('obj', obj)
43     cv2.imshow('dst', dst)
44     cv2.waitKey(0)
```

모멘트 기반 객체 검출

✓ 모멘트 기반 객체 검출 예제 실행 결과



템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭(Template matching)이란?

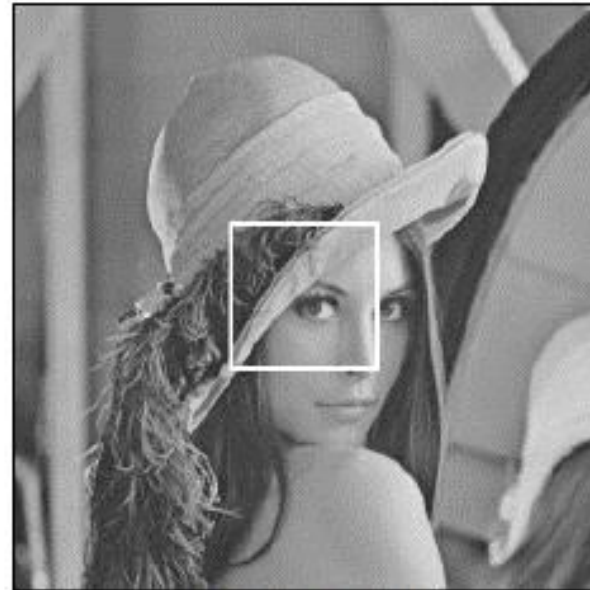
- 입력 영상에서 (작은 크기의) 템플릿 영상과 일치하는 부분을 찾는 기법
- 템플릿: 찾을 대상이 되는 작은 영상. 패치(patch).



입력 영상



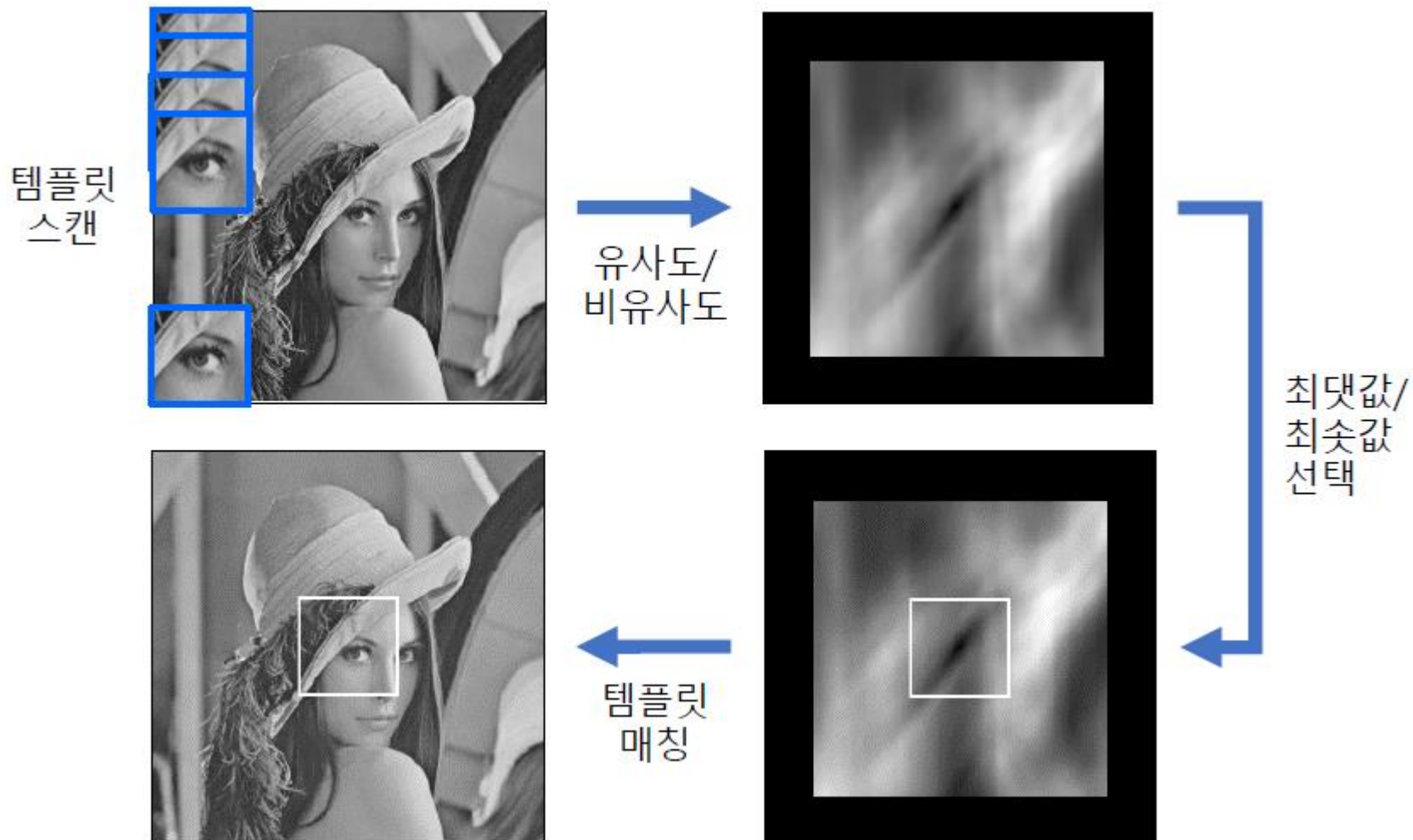
템플릿



검출 결과

템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭(Template matching)이란?



템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 함수

cv2.matchTemplate(image, temp1, method, result=None, mask=None) --> result

- image: 입력 영상 8 비트 또는 32 비트
- temp1 : 템플릿 영상 image 보다 같거나 작은 크기, 같은 타입
- method: 비교 방법. cv2.TM_으로 시작하는 플래그 지정

TM_SQDIFF / TM_SQDIFF_NORMED	Sum of squared difference
TM_CCORR / TM_CCORR_NORMED	(Cross) Correlation
TM_CCOEFF / TM_CCOEFF_NORMED	Correlation Coefficient

- result: 비교 결과 행렬. numpy.ndarray.dtype=numpy.float32
image 의 크기가 W x H 이고, temp1의 크기가 w x h 이면
result 크기는 (W - w + 1) x (H - h + 1)

템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 방법

method	설명
cv2.TM_SQDIFF	$R(x, y) = \sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2$ <div>완전히 같으면 0, 다르면 값이 커짐.</div>
cv2.TM_SQDIFF_NORMED	$R(x, y) = \frac{\sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$ <div>[0, 1] 정규화</div>
cv2.TM_CCORR	$R(x, y) = \sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))$ <div>같으면 큰 값, 다르면 작은 값.</div>
cv2.TM_CCORR_NORMED	$R(x, y) = \frac{\sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$ <div>[0, 1] 정규화</div>

템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 방법

method	설명
cv2.TM_CCOEFF	$R(x, y) = \sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))$ $T'(x', y') = T(x', y') - 1 / (w \cdot h) \cdot \sum_{x'', y''} T(x'', y'')$ $I'(x + x', y + y') = I(x + x', y + y') - 1 / (w \cdot h) \cdot \sum_{x'', y''} I(x + x'', y + y'')$
cv2.TM_CCOEFF_NORMED	$R(x, y) = \frac{\sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x', y'} T'(x', y')^2 \cdot \sum_{x', y'} I'(x + x', y + y')^2}}$

평균 보정 후
Correlation 연산

완전히 일치하면 1,
역일치하면 -1,
상호 연관성이 없으면 0.

템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 방법

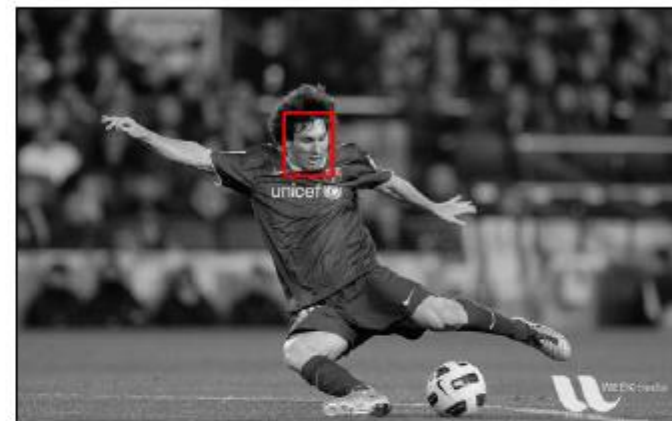
cv2.TM_SQDIFF



cv2.TM_CCORR



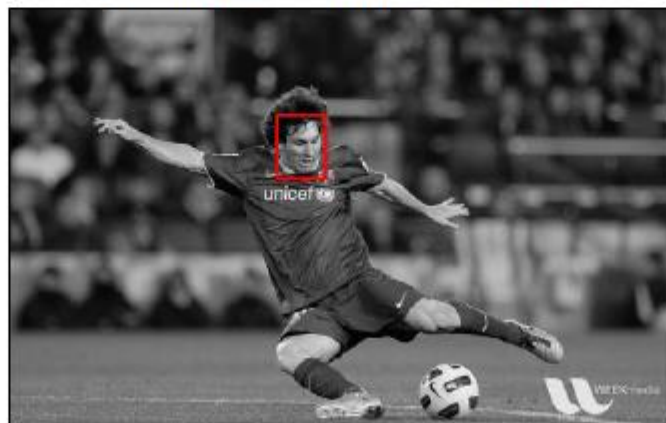
cv2.TM_CCOEFF



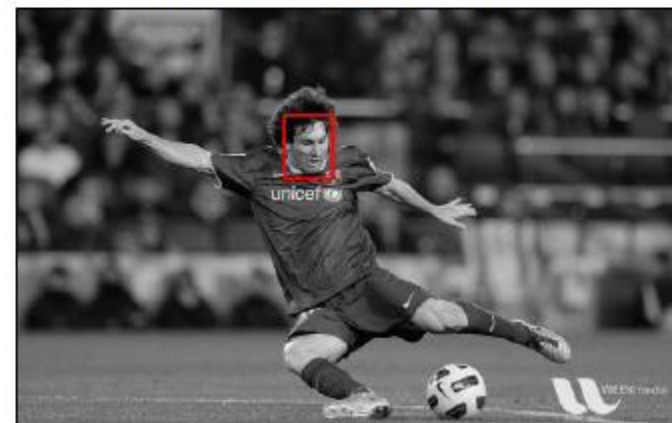
cv2.TM_SQDIFF_NORMED



cv2.TM_CCORR_NORMED



cv2.TM_CCOEFF_NORMED



템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 방법

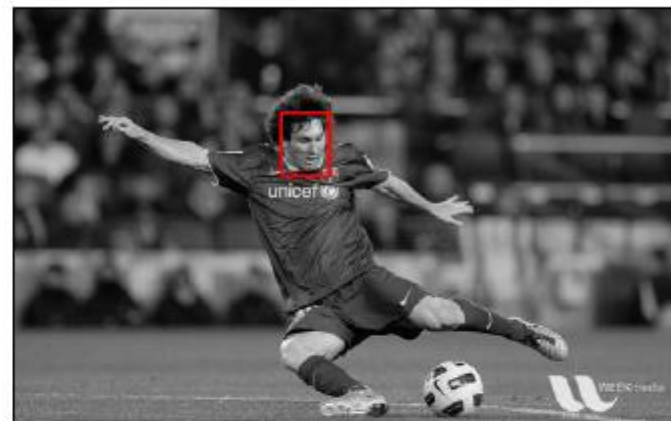
cv2.TM_SQDIFF



cv2.TM_CCORR



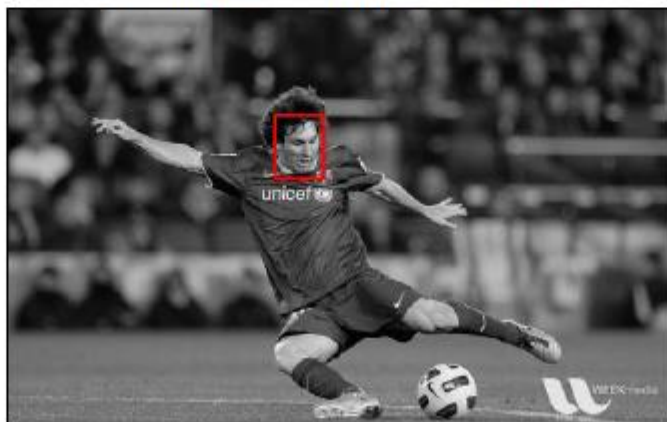
cv2.TM_CCOEFF



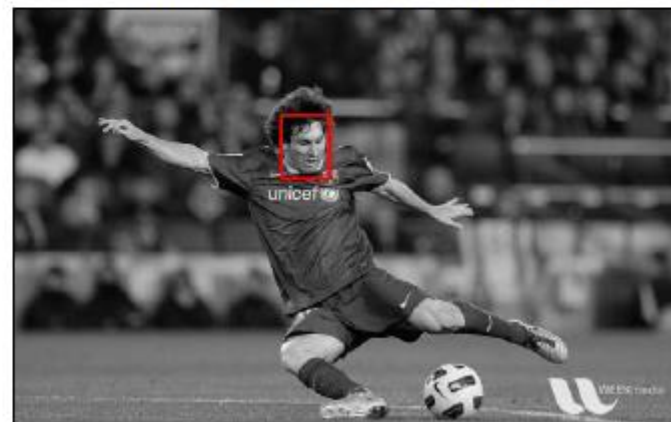
cv2.TM_SQDIFF_NORMED



cv2.TM_CCORR_NORMED



cv2.TM_CCOEFF_NORMED



템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 예제

```
1  import sys
2  import numpy as np
3  import cv2
4
5  # 입력 영상 & 템플릿 영상 불러오기
6  src = cv2.imread('circuit.bmp', cv2.IMREAD_GRAYSCALE)
7  templ = cv2.imread('crystal.bmp', cv2.IMREAD_GRAYSCALE)
8
9  if src is None or templ is None:
10     print('Image load failed!')
11     sys.exit()
12
13  # 입력 영상 밝기 50증가, 가우시안 잡음(sigma=10) 추가
14  noise = np.zeros(src.shape, np.int32)
15  cv2.randn(noise, 50, 10)
16  src = cv2.add(src, noise, dtype=cv2.CV_8UC3)
17
```

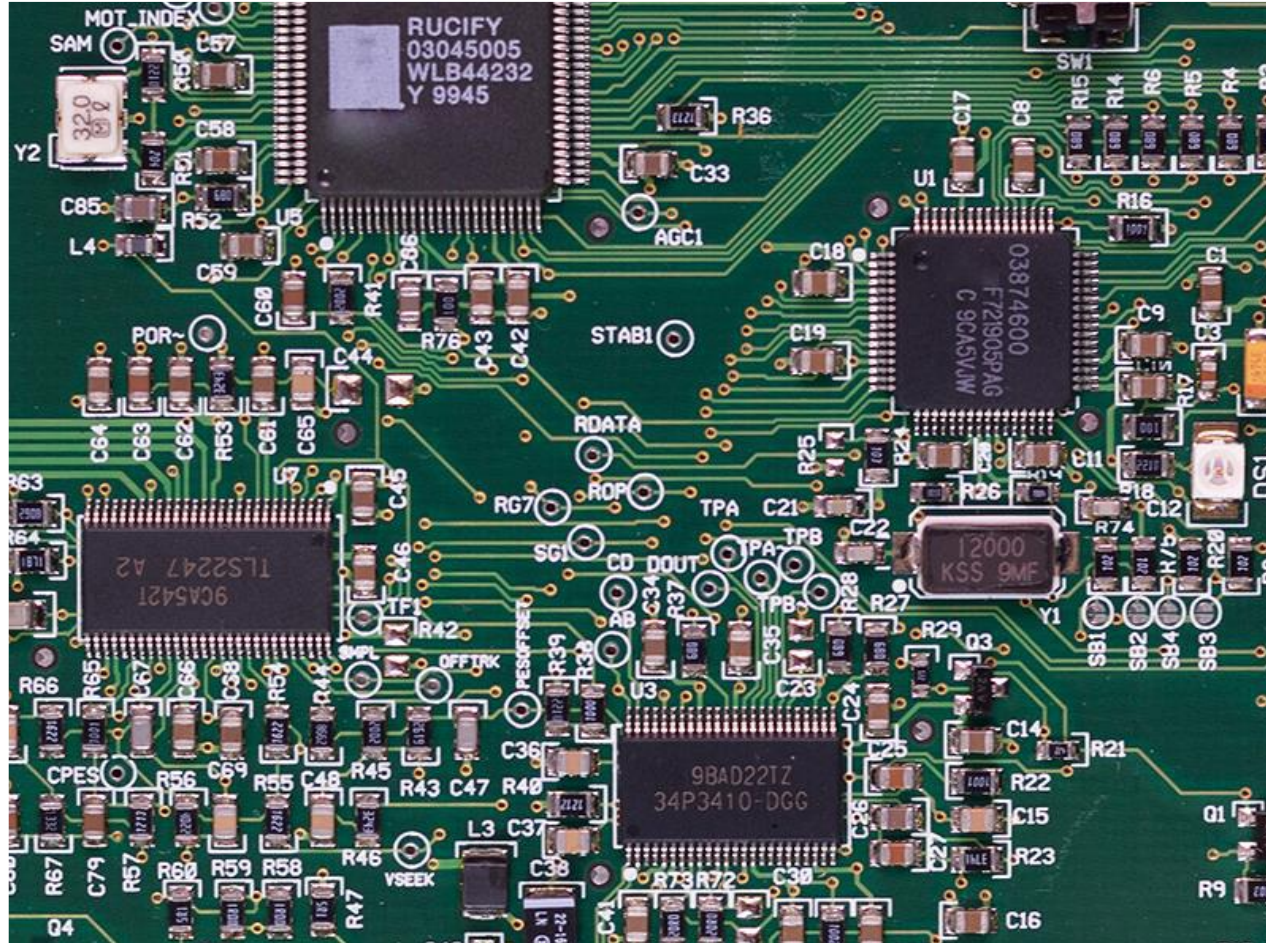
템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 예제

```
18  # 템플릿 매칭 & 결과 분석
19  res = cv2.matchTemplate(src, templ, cv2.TM_CCOEFF_NORMED)
20  res_norm = cv2.normalize(res, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
21
22  _, maxv, _, maxloc = cv2.minMaxLoc(res)
23  print('maxv:', maxv)
24  print('maxloc:', maxloc)
25
26  # 매칭 결과를 빨간색 사각형으로 표시
27  th, tw = templ.shape[:2]
28  dst = cv2.cvtColor(src, cv2.COLOR_GRAY2BGR)
29  cv2.rectangle(dst, maxloc, (maxloc[0] + tw, maxloc[1] + th), (0, 0, 255), 2)
30
31  # 결과 영상 화면 출력
32  cv2.imshow('res_norm', res_norm)
33  cv2.imshow('dst', dst)
34  cv2.waitKey()
35  cv2.destroyAllWindows()
```

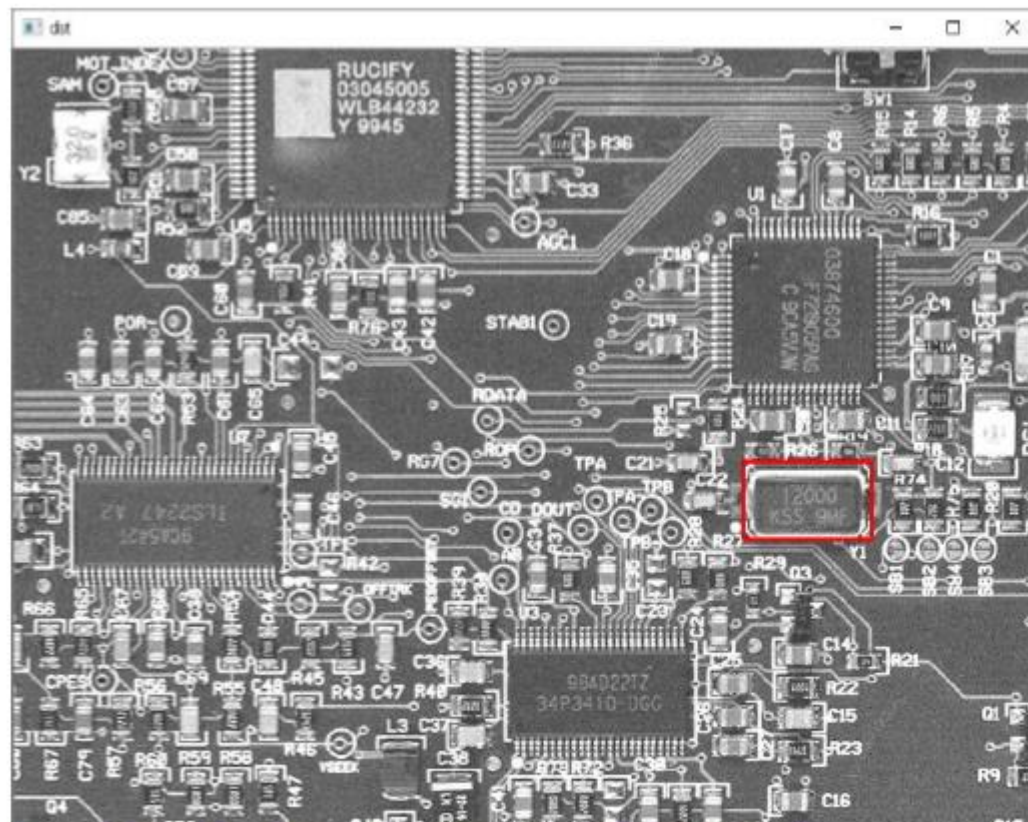
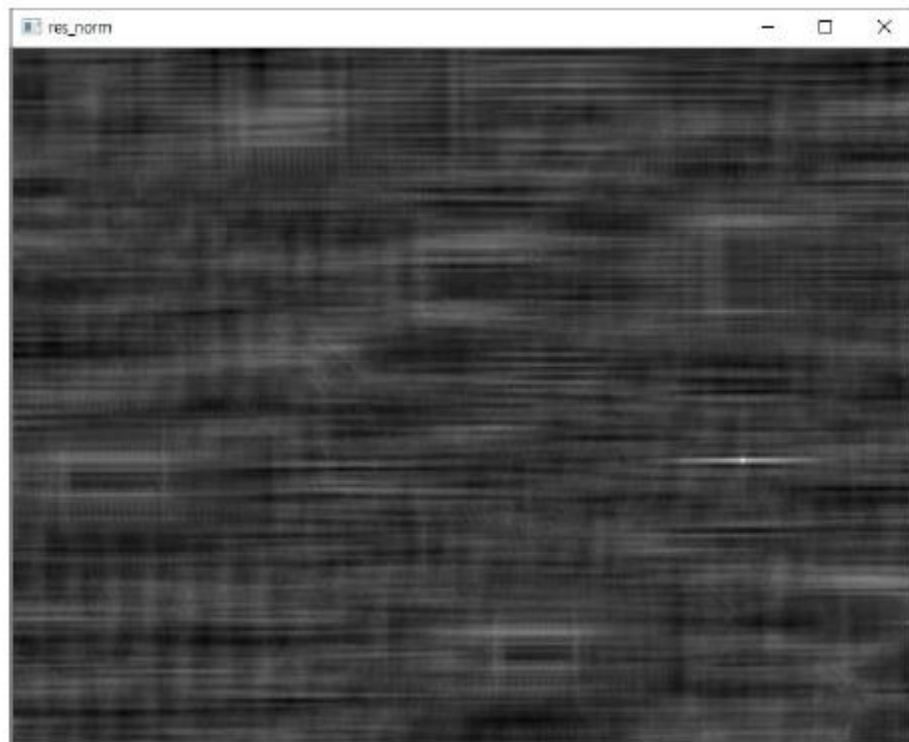

템플릿 매칭 (1): 이해하기

✓ 템플릿 매칭 예제 입력 영상



템플릿 매칭 (1): 이해하기

✅ 템플릿 매칭 예제 실행 결과

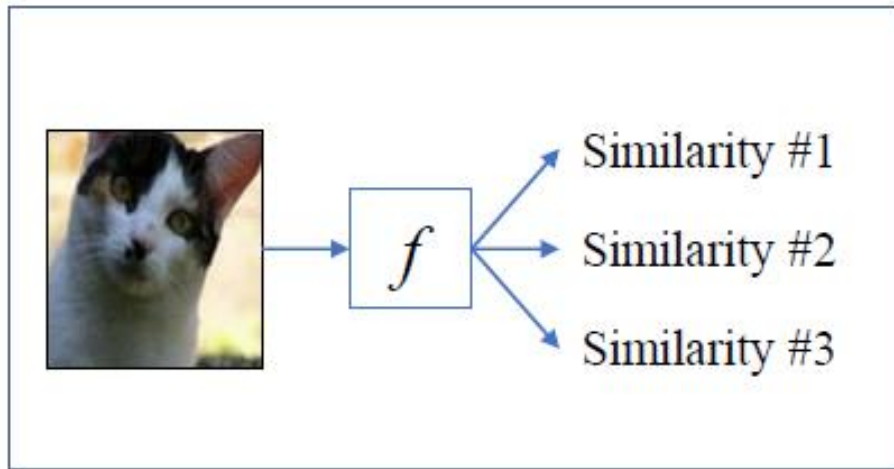





원본 영상에서 밝기 50 증가, 가우시안 잡음($\sigma=10$) 추가

템플릿 매칭 (2): 인쇄체 숫자 인식

✓ 인식(Recognition)이란?

- Classifying a detected object into different categories.
- 여러 개의 클래스 중에서 가장 유사한 클래스를 선택



			
cat	3.45	0.51	1.42
car	2.90	6.04	4.64
flog	0.09	2.31	3.65
	Good!	Good!	NG!

템플릿 매칭 (2): 인쇄체 숫자 인식

✓ 숫자 템플릿 영상 생성

- Consolas 폰트로 쓰여진 숫자 영상을 digit0.bmp ~ digit9.bmp 로 저장
- 각 숫자 영상의 크기는 100x150 크기로 정규화



digit0.bmp



digit1.bmp



digit2.bmp



digit3.bmp



digit4.bmp



digit5.bmp



digit6.bmp



digit7.bmp



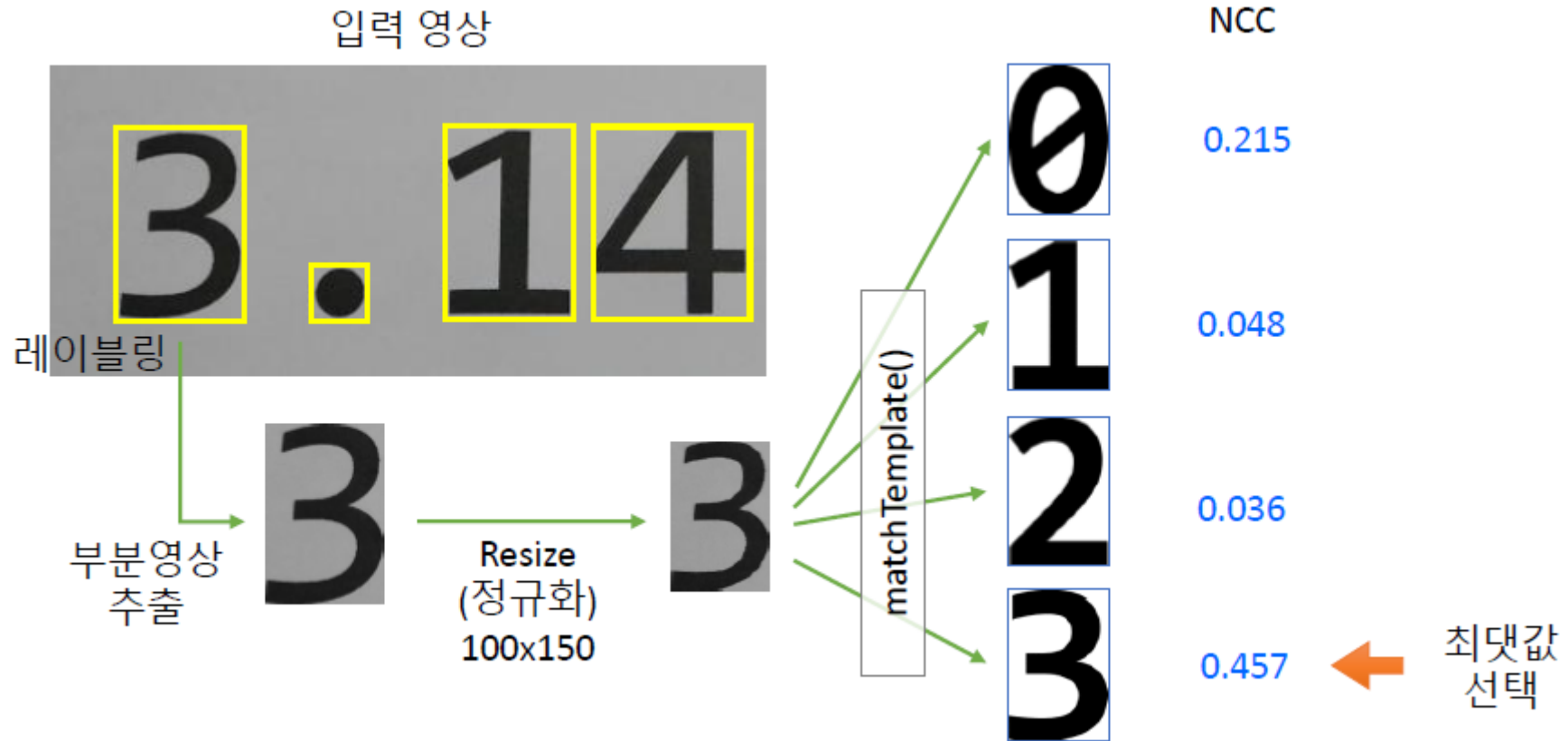
digit8.bmp



digit9.bmp

템플릿 매칭 (2): 인쇄체 숫자 인식

✓ 인쇄체 숫자 인식 방법



템플릿 매칭 (2): 인쇄체 숫자 인식

✓ 인쇄체 숫자 인식 예제

```
1  import sys
2  import numpy as np
3  import cv2
4
5  def load_digits():
6      img_digits = []
7
8      for i in range(10):
9          filename = './digits/digit{}.bmp'.format(i)
10         img_digits.append(cv2.imread(filename, cv2.IMREAD_GRAYSCALE))
11
12         if img_digits[i] is None:
13             return None
14
15     return img_digits
16
```

템플릿 매칭 (2): 인쇄체 숫자 인식

✓ 인쇄체 숫자 인식 예제

```
17  def find_digit(img, img_digits):
18      max_idx = -1
19      max_ccoeff = -1
20
21      # 최대 NCC 찾기
22      for i in range(10):
23          img = cv2.resize(img, (100, 150))
24          res = cv2.matchTemplate(img, img_digits[i], cv2.TM_CCOEFF_NORMED)
25
26          if res[0, 0] > max_ccoeff:
27              max_idx = i
28              max_ccoeff = res[0, 0]
29
30      return max_idx
31
```

테프리 매치 (2): 이생체 숫자 이식

```
32 def main():
33     # 입력 영상 불러오기
34     src = cv2.imread('digits_print.bmp')
35
36     if src is None:
37         print('Image load failed!')
38         return
39
40     # 100x150 숫자 영상 불러오기
41     img_digits = load_digits() # list of ndarray
42
43     if img_digits is None:
44         print('Digit image load failed!')
45         return
46
47     # 입력 영상 이진화 & 레이블링
48     src_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
49     _, src_bin = cv2.threshold(src_gray, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
50     cnt, _, stats, _ = cv2.connectedComponentsWithStats(src_bin)
51
```

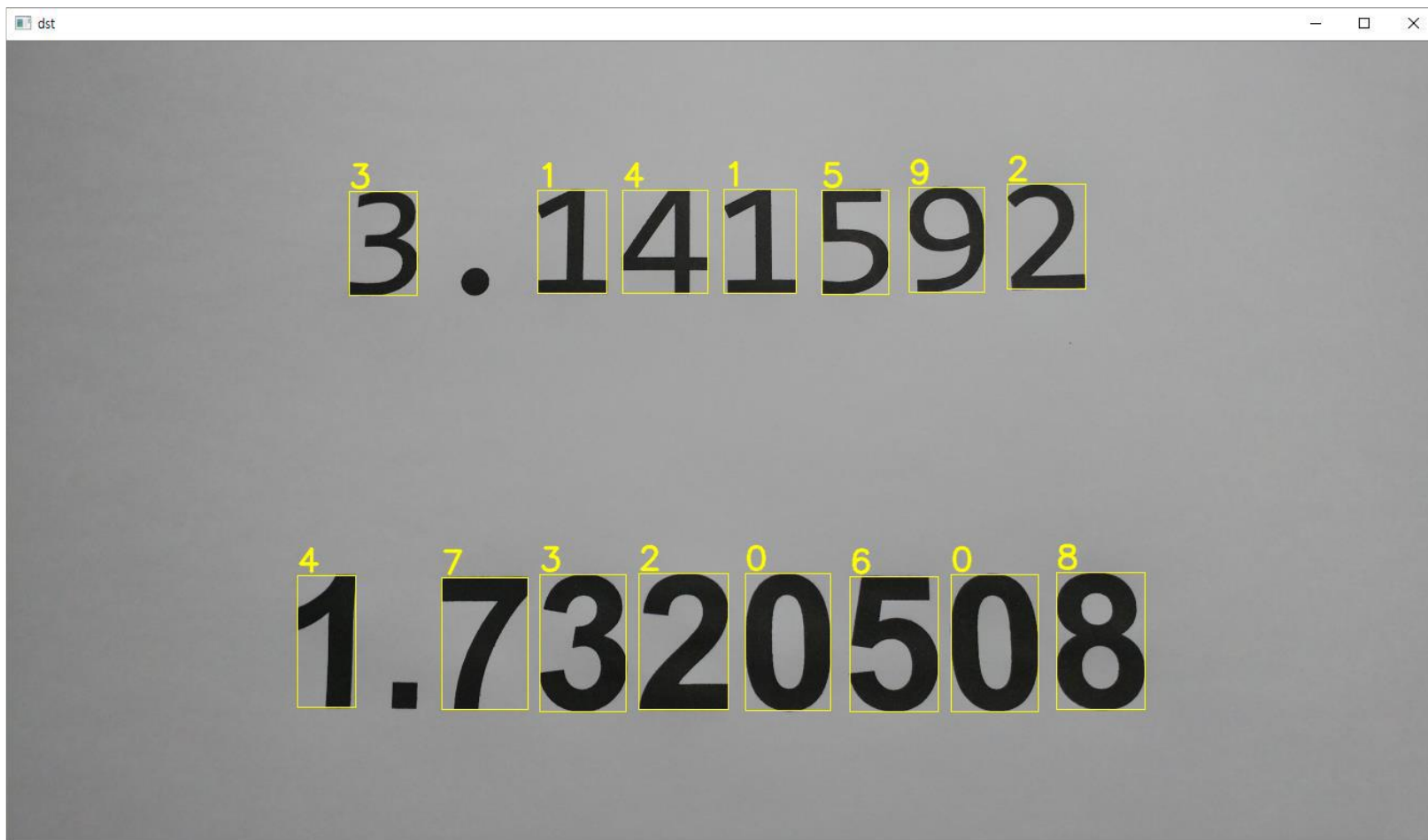
템플릿 매치 (2) · 이체체 숫자 이체

✓ 인쇄

```
52     # 숫자 인식 결과 영상 생성
53     dst = src.copy()
54     for i in range(1, cnt):
55         (x, y, w, h, s) = stats[i]
56
57         if s < 1000:
58             continue
59
60         # 가장 유사한 숫자 이미지를 선택
61         digit = find_digit(src_gray[y:y+h, x:x+w], img_digits)
62         cv2.rectangle(dst, (x, y, w, h), (0, 255, 255))
63         cv2.putText(dst, str(digit), (x, y - 4), cv2.FONT_HERSHEY_SIMPLEX,
64                     1, (0, 255, 255), 2, cv2.LINE_AA)
65
66         cv2.imshow('dst', dst)
67         cv2.waitKey()
68     cv2.destroyAllWindows()
69
70     if __name__ == '__main__':
71         main()
```

템플릿 매칭 (2): 인쇄체 숫자 인식

✓ 인쇄체 숫자 인식 예제 실행 결과



캐스케이드 분류기 : 얼굴 검출

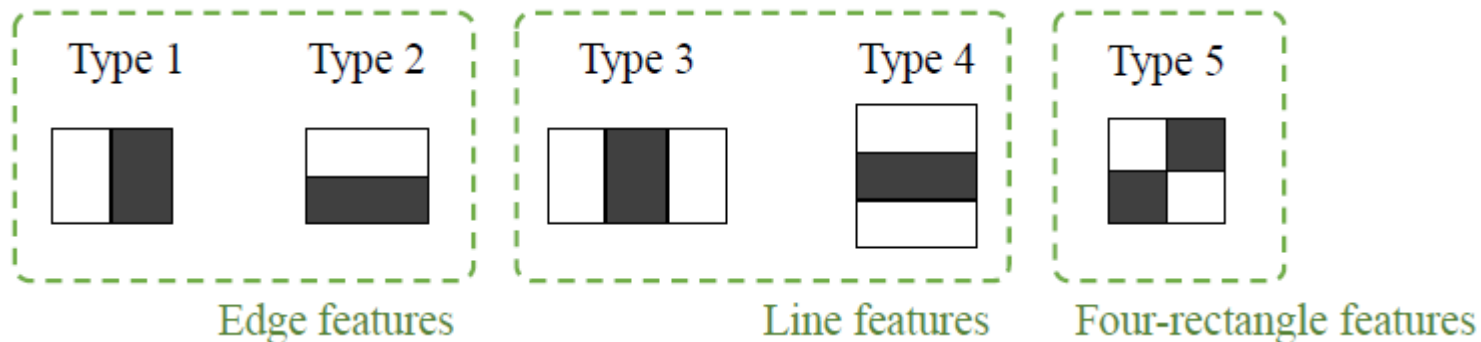
✓ Viola-Jones 얼굴 검출기

- Positive 영상 얼굴 영상 과 negative 영상 얼굴 아닌 영상 을 훈련하여 빠르고 정확하게 얼굴 영역을 검출
- 기존 방법과의 차별점
 - 유사 하르(Haar-like) 특징을 사용
 - AdaBoost에 기반한 강한 분류 성능
 - 캐스케이드(cascade) 방식을 통한 빠른 동작 속도
- 기존 얼굴 검출 방법보다 약 15 배 빠르게 동작

캐스케이드 분류기 : 얼굴 검출

✓ 유사 하르 특징(Haar-like features)

- 사각형 형태의 필터 집합을 사용
- 흰색 사각형 영역 픽셀 값의 합에서 검정색 사각형 영역 픽셀 값을 뺀 결과 값을 추출



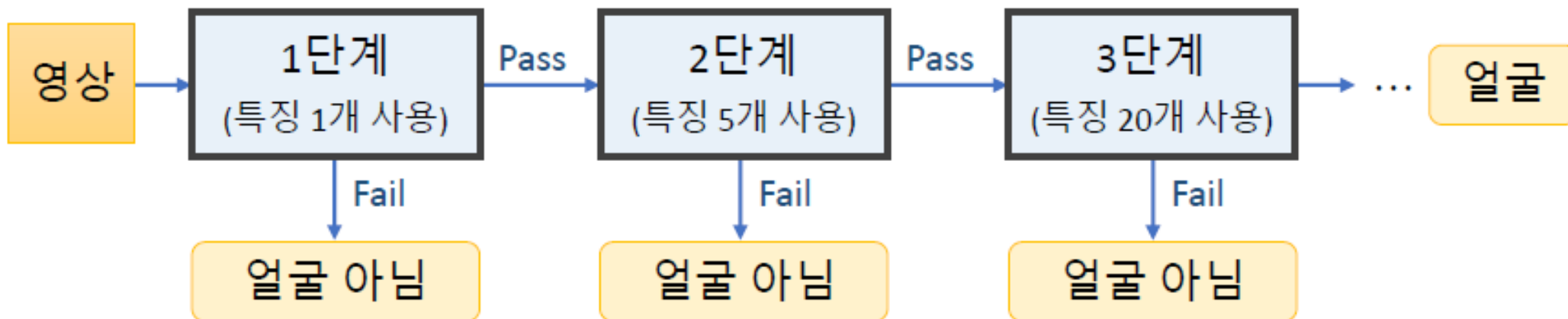
- 24x24 부분 영상에서 얼굴 판별에 유용한 유사 하르 특징을 선별



캐스케이드 분류기 : 얼굴 검출

✓ 캐스케이드 분류기 (Cascade classifier)

- 일반적인 영상에는 얼굴이 한 두개 있을 뿐, 나머지 영역은 대부분 non-face 영역
- Non-face 영역을 빠르게 skip하도록 다단계 검사 수행



캐스케이드 분류기 : 얼굴 검출

✓ 캐스케이드 분류기 얼굴 검출 과정 시각화



캐스케이드 분류기 : 얼굴 검출

✓ cv2.CascadeClassifier 객체 생성 및 학습 데이터 불러오기

```
cv2.CascadeClassifier(          ) --> <CascadeClassifier object>  
cv2.CascadeClassifier(filename) --> <CascadeClassifier object>
```

```
cv2.CascadeClassifier.load(filename) --> retval
```

- filename: XML 파일 이름
- retval 성공하면 True, 실패하면 False.

캐스케이드 분류기 : 얼굴 검출

✓ 미리 학습된 XML 파일 다운로드

- <https://github.com/opencv/opencv/tree/master/data/haarcascades>

XML 파일 이름	검출 대상
haarcascade_frontalface_default.xml haarcascade_frontalface_alt.xml haarcascade_frontalface_alt2.xml haarcascade_frontalface_alt_tree.xml	정면 얼굴 검출
haarcascade_profileface.xml	측면 얼굴 검출
haarcascade_smile.xml	웃음 검출
haarcascade_eye.xml haarcascade_eye_tree_eyeglasses.xml haarcascade_lefteye_2splits.xml haarcascade_righteye_2splits.xml	눈 검출
haarcascade_frontalcatface.xml haarcascade_frontalcatface_extended.xml	고양이 얼굴 검출
haarcascade_fullbody.xml	사람의 전신 검출
haarcascade_upperbody.xml	사람의 상반신 검출
haarcascade_lowerbody.xml	사람의 하반신 검출
haarcascade_russian_plate_number.xml haarcascade_licence_plate_rus_16stages.xml	러시아 자동차 번호판 검출

캐스케이드 분류기 : 얼굴 검출

✓ 미리 학습된 XML 파일 다운로드

```
cv2.CascadeClassifier.detectMultiScale(image, scaleFactor=None, minNeighbors=None, flags=None, minSize=None, maxSize=None) --> result
```

- image: 입력 영상 (cv2.CV_8U)
- scaleFactor: 영상 축소 비율. 기본값은 1.1.
- minNeighbors: 얼마나 많은 이웃 사각형이 검출되어야 최종 검출 영역으로 설정할지를 지정.
기본값은 3.
- flags: (현재) 사용되지 않음
- minSize: 최소 객체 크기. (w, h) 튜플
- maxSize: 최대 객체 크기. (w, h) 튜플
- result: 검출된 객체의 사각형 정보 (x, y, w, h)를 담은 numpy.ndarray.shape=(N, 4). dtype=numpy.int32

캐스케이드 분류기 : 얼굴 검출

✓ 정면 얼굴 검출 예제

```
1 import sys
2 import numpy as np
3 import cv2
4
5 src = cv2.imread('lenna.bmp')
6
7 if src is None:
8     print('Image load failed!')
9     sys.exit()
10
11 classifier = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')
12
13 if classifier.empty():
14     print('XML load failed!')
15     sys.exit()
16
17 faces = classifier.detectMultiScale(src)
```

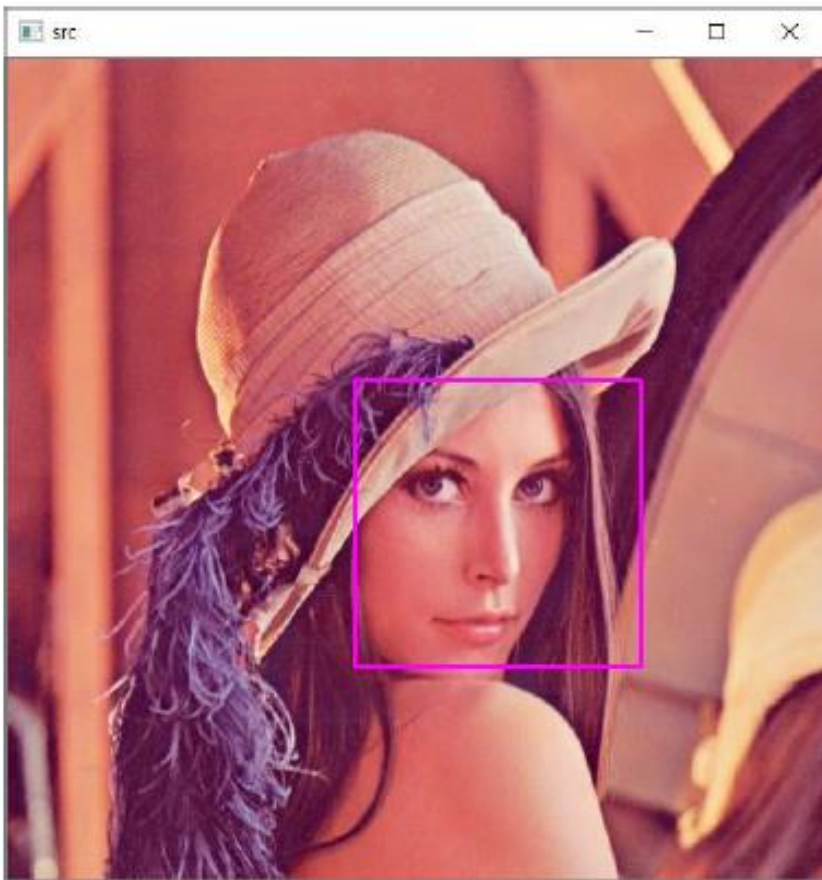
캐스케이드 분류기 : 얼굴 검출

✓ 정면 얼굴 검출 예제

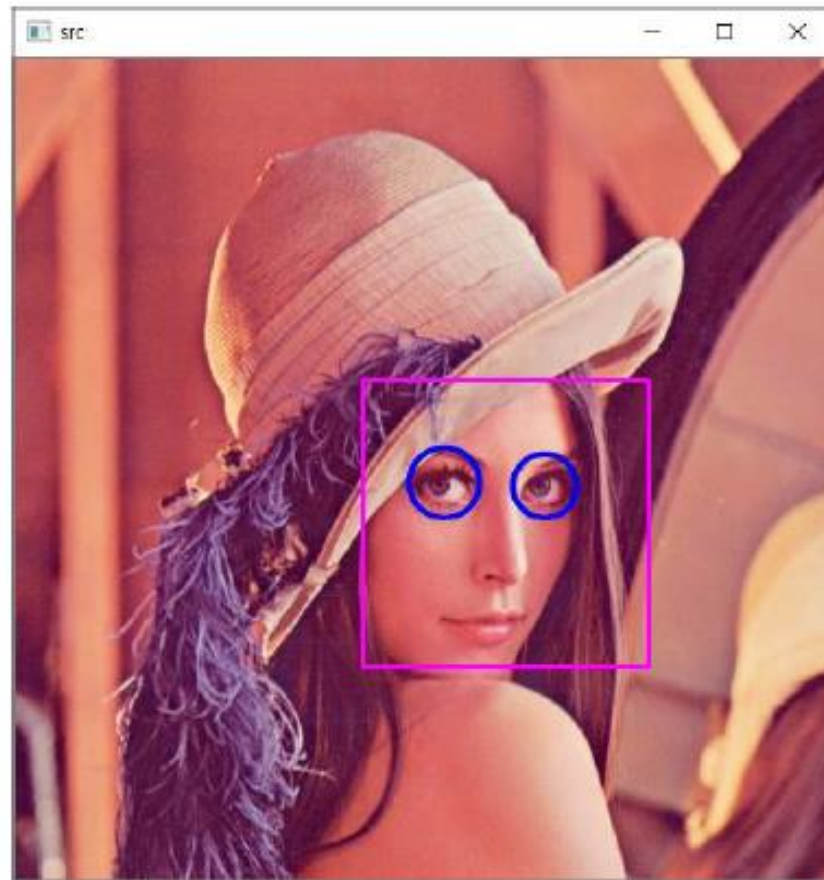
```
18
19     for (x, y, w, h) in faces:
20         cv2.rectangle(src, (x, y, w, h), (255, 0, 255), 2)
21
22     cv2.imshow('src', src)
23     cv2.waitKey()
24     cv2.destroyAllWindows()
```

캐스케이드 분류기 : 얼굴 검출

✔ 정면 얼굴 검출 예제 실행 결과



얼굴 검출



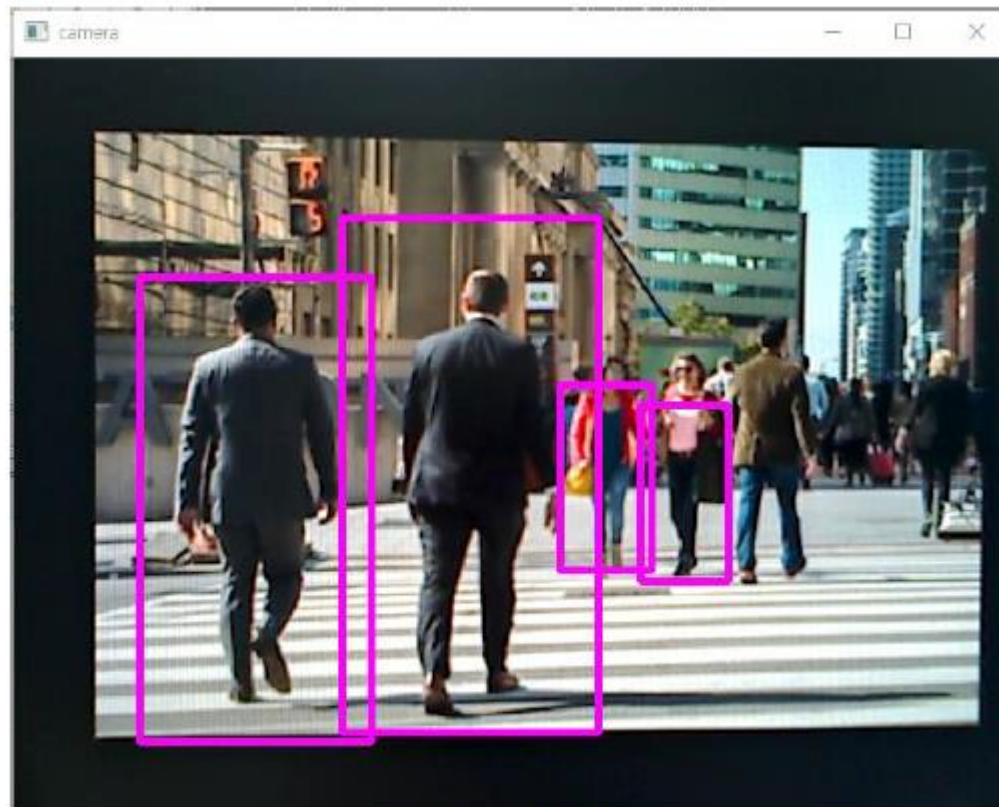
얼굴 & 눈 검출 (eyedetect.py)

캐스케이드 분류기 : 얼굴 검출

✔ 기타 객체 검출 결과



러시아 자동차 번호판 검출
(haarcascade_russian_plate_number.xml)



사람 전신 검출
(haarcascade_fullbody.xml)

HOG 보행자 검출

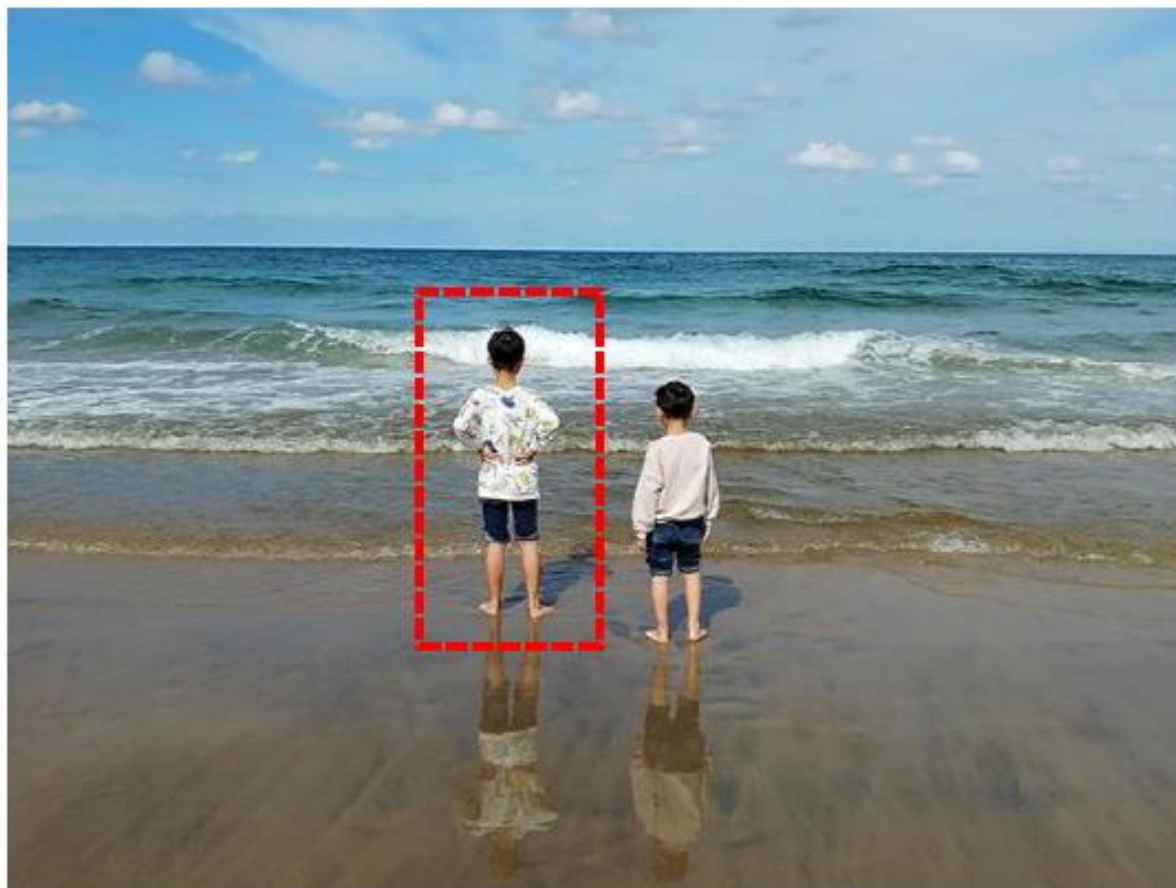
✓ HOG (Histogram of Oriented)란?

- 영상의 지역적 그래디언트 방향 정보를 특징 벡터로 사용
- 2005 년 CVPR 학회에서 보행자 검출 방법으로 소개되어 널리 사용되기 시작함
- 이후 다양한 객체 인식에서 활용됨



HOG 보행자 검출

✓ HOG 알고리즘



입력 영상



부분 영상
추출



크기 정규화
64x128

HOG 보행자 검출

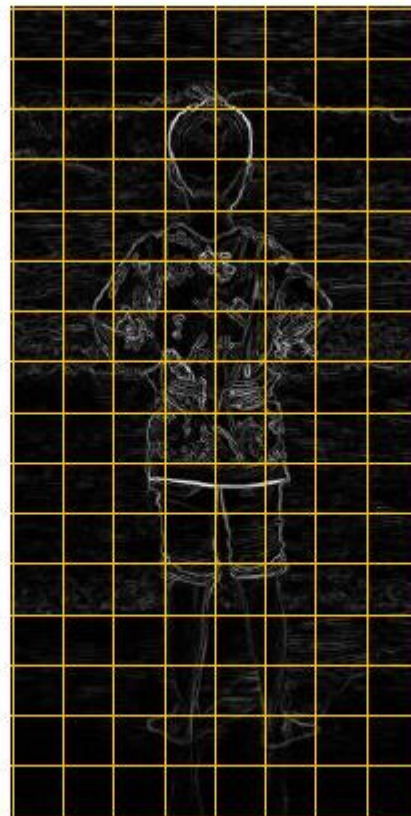
✓ HOG 알고리즘



64x128 영상



그래디언트
계산



8x8 크기의
셀(cell) 분할



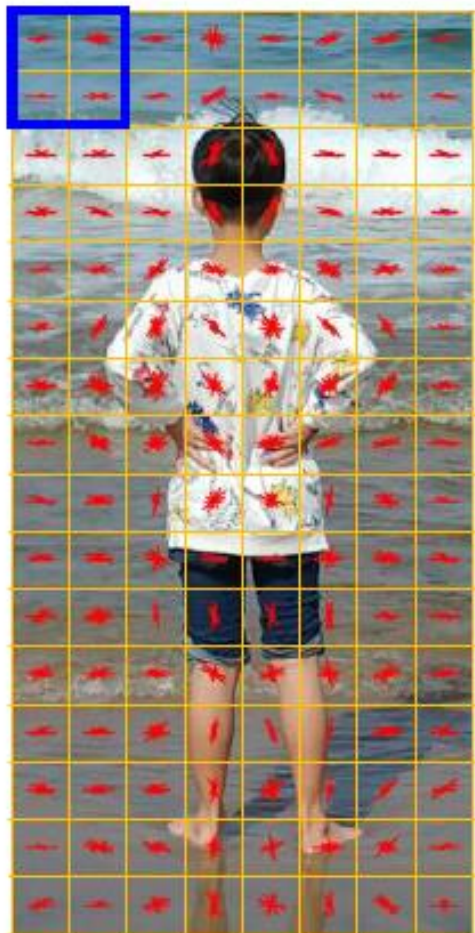
각 셀마다
방향과 크기
성분을 이용하여
방향 히스토그램
계산



방향 히스토그램의
빈 개수 = 9

HOG 보행자 검출

✓ HOG 알고리즘



128x64 영상

[블록 히스토그램 구하기]

- 8x8 셀 4개를 하나의 블록을 지정
→ 즉, 블록 하나의 크기는 16×16
→ 8픽셀 단위로 이동: $\text{stride} = 8$
→ 각 블록의 히스토그램 빈(bin) 개수는 $4 \times 9 = 36$ 개

[특징 벡터의 차원]

하나의 부분 영상 패치에서의 특징 벡터 크기

$$\rightarrow 7 \times 15 \times 36 = 3780$$

HOG 보행자 검출

✓ HOG 기술자 객체 생성 및 보행자 검출을 위해 학습된 분류기 계수 불러오기

```
cv2.HOGDescriptor( ) --> <CascadeClassifier object>
```

```
cv2.HOGDescriptor_getDefaultPeopleDetector( ) --> retval
```

- retval : 미리 훈련된 특징 벡터. `numpy.ndarray.shape=(3781, 1)`.
`dtype=numpy.float32`

✓ SVM 분류기 계수 등록하기

```
cv2.HOGDescriptor.setSVMDetector(svmdetector) --> None
```

- svmdetector : 선형 SVM 분류기를 위한 계수

HOG 보행자 검출

✓ HOG 멀티스케일 객체 검출 함수

```
cv2.HOGDescriptor.detectMultiScale(img, hitThreshold=None, winStride=None,  
padding=None, scale=None, finalThreshold=None, useMeanshiftGrouping=None)  
--> foundLocations , foundWeights
```

- img: 입력 영상. cv2.CV_8UC1 또는 cv2.CV_8UC3.
- hitThreshold: 특징 벡터와 SVM 분류 평면까지의 거리에 대한 임계값
- winStride: 셀 윈도우 이동 크기. (0, 0) 지정 시 셀 크기와 같게 설정
- padding: 패딩 크기
- scale: 검색 윈도우 크기 확대 비율 . 기본값은 1.05.
- finalThreshold: 검출 결정을 위한 임계값
- useMeanshiftGrouping: 겹쳐진 검색 윈도우를 합치는 방법 지정 플래그
- foundLocations: (출력) 검출된 사각형 영역 정보
- foundWeights: (출력) 검출된 사각형 영역에 대한 신뢰도

HOG 보행자 검출

✓ HOG 보행자 검출 예제

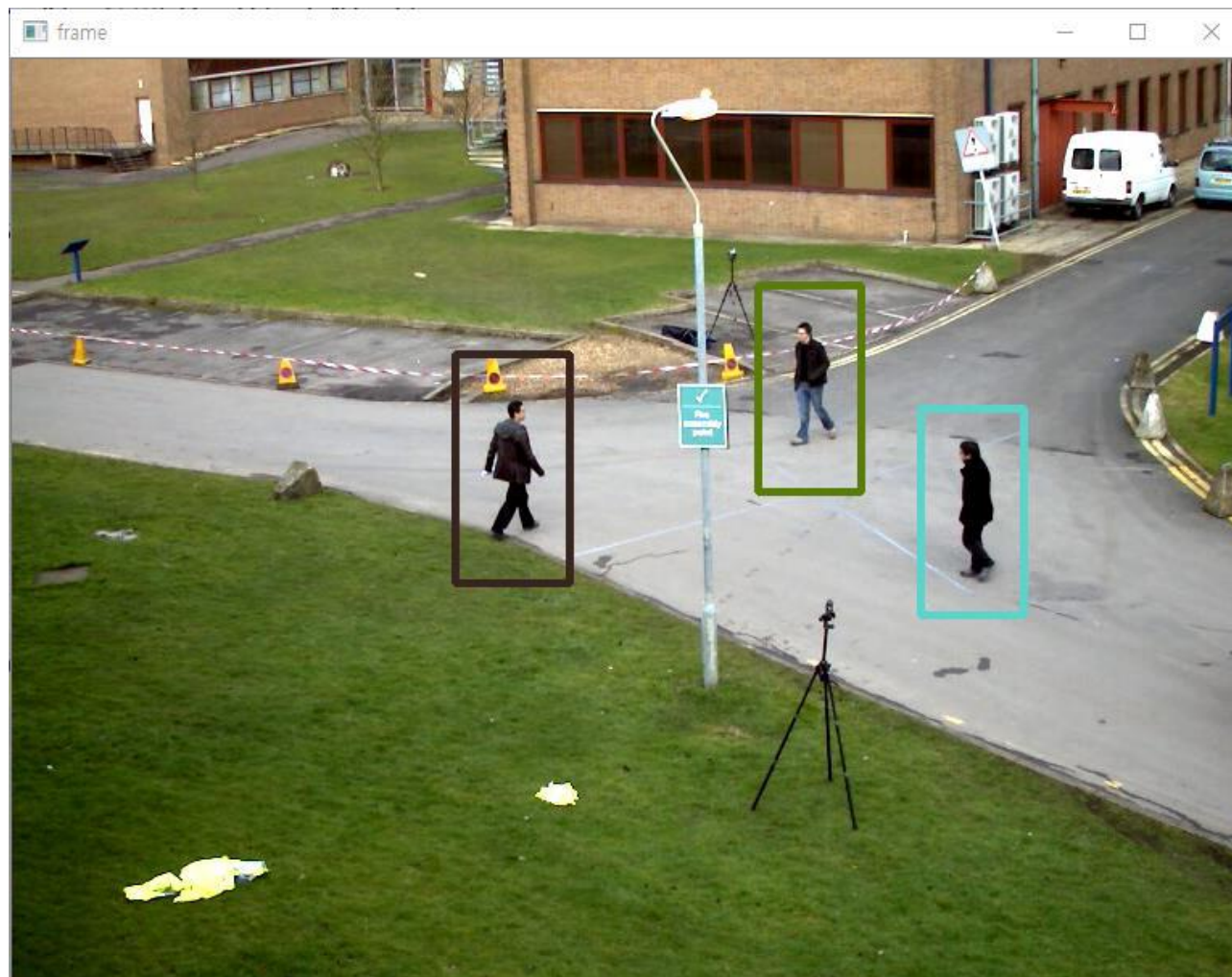
```
1 import sys
2 import random
3 import numpy as np
4 import cv2
5
6 # 동영상 불러오기
7 cap = cv2.VideoCapture('vtest.avi')
8
9 if not cap.isOpened():
10     print('Video open failed!')
11     sys.exit()
12
13 # 보행자 검출을 위한 HOG 기술자 설정
14 hog = cv2.HOGDescriptor()
15 hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
16
```

HOG 보행자 검출

```
17 while True:
18     ret, frame = cap.read()
19
20     if not ret:
21         break
22
23     # 매 프레임마다 보행자 검출
24     detected, _ = hog.detectMultiScale(frame)
25
26     # 검출 결과 화면 표시
27     for (x, y, w, h) in detected:
28         c = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
29         cv2.rectangle(frame, (x, y, w, h), c, 3)
30
31     cv2.imshow('frame', frame)
32     if cv2.waitKey(10) == 27:
33         break
34
35 cv2.destroyAllWindows()
```


HOG 보행자 검출

✓ HOG 보행자 검출 예제 실행 결과



[프로젝트] 간단 스노우앱

✓ 간단 스노우앱

- 카메라 입력 영상에서 얼굴을 검출하여 재미있는 그래픽을 합성하는 프로그램



✓ 구현할 기능

- 카메라 입력 영상에서 얼굴 눈 검출하기
- 눈 위치와 맞게 투명한 PNG 파일 합성하기
- 합성된 결과를 동영상으로 저장하기

[프로젝트] 간단 스노우앱

✓ 얼굴 & 눈 검출

- 캐스케이드 분류기 사용
 - XML 파일 다운로드 <https://github.com/opencv/opencv/tree/master/data/haarcascades>
 - 얼굴 검출 XML: haarcascade_frontalface_alt2.xml
 - 눈 검출 XML 파일 : haarcascade_eye.xml
- 눈 검출하기
 - 얼굴 검출 영역 내에서만 눈 검출
 - 눈이 두 개 검출되었을 경우에만 그래픽 합성
 - 왼쪽 눈과 오른쪽 눈 좌표를 계산하여 합성 시 사용

[프로젝트] 간단 스노우앱

✓ 눈 위치와 맞게 투명한 PNG 파일 합성하기

- 사용할 PNG 파일 : glasses.png
- PNG 영상 크기 조절 및 위치 계산
 - 입력 영상에서 두 눈사이의 거리에 맞게 PNG 안경 영상 크기를 Resize
 - 왼쪽 눈 위치가 일치하도록 PNG 영상의 합성 위치를 계산
- 합성 방식
 - Alpha 채널 값이 255 면 완전한 불투명 , 0 이면 완전한 투명 , 중간값이면 가중합 연산



BGR 채널



Alpha 채널



단순 복사



가중합 연산

[프로젝트] 간단 스노우앱

✓ snowapp.py

```
1  import sys
2  import numpy as np
3  import cv2
4
5  # 3채널 img 영상에 4채널 item 영상을 pos 위치에 합성
6  def overlay(img, glasses, pos):
7      # 실제 합성을 수행할 부분 영상 좌표 계산
8      sx = pos[0]
9      ex = pos[0] + glasses.shape[1]
10     sy = pos[1]
11     ey = pos[1] + glasses.shape[0]
12
13     # 합성할 영역이 입력 영상 크기를 벗어나면 무시
14     if sx < 0 or sy < 0 or ex > img.shape[1] or ey > img.shape[0]:
15         return
16
```

[프로젝트] 간단 스노우앱

✓ snowapp.py

```
17      # 부분 영상 참조. img1: 입력 영상의 부분 영상, img2: 안경 영상의 부분 영상
18      img1 = img[sy:ey, sx:ex]  # shape=(h, w, 3)
19      img2 = glasses[:, :, 0:3] # shape=(h, w, 3)
20      alpha = 1. - (glasses[:, :, 3] / 255.) # shape=(h, w)
21
22      # BGR 채널별로 두 부분 영상의 가중합
23      img1[..., 0] = (img1[..., 0] * alpha + img2[..., 0] * (1. - alpha)).astype(np.uint8)
24      img1[..., 1] = (img1[..., 1] * alpha + img2[..., 1] * (1. - alpha)).astype(np.uint8)
25      img1[..., 2] = (img1[..., 2] * alpha + img2[..., 2] * (1. - alpha)).astype(np.uint8)
26
27
28      # 카메라 열기
29      cap = cv2.VideoCapture(0)
30
31      if not cap.isOpened():
32          print('Camera open failed!')
33          sys.exit()
34
```

[프로젝트] 간단 스노우앱

✓ snowapp.py

```
35 w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
36 h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
37
38 fourcc = cv2.VideoWriter_fourcc(*'DIVX')
39 out = cv2.VideoWriter('output.avi', fourcc, 30, (w, h))
40
41 # Haar-like XML 파일 열기
42 face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')
43 eye_classifier = cv2.CascadeClassifier('haarcascade_eye.xml')
44
45 if face_classifier.empty() or eye_classifier.empty():
46     print('XML load failed!')
47     sys.exit()
48
49 # 안경 PNG 파일 열기 (Image from http://www.pngall.com/)
50 glasses = cv2.imread('glasses.png', cv2.IMREAD_UNCHANGED)
51
```


[프로젝트] 간단 스노우앱

```
52 if glasses is None:
53     print('PNG image open failed!')
54     sys.exit()
55
56 ew, eh = glasses.shape[:2] # 가로, 세로 크기
57 ex1, ey1 = 240, 300 # 왼쪽 눈 좌표
58 ex2, ey2 = 660, 300 # 오른쪽 눈 좌표
59
60 # 매 프레임에 대해 얼굴 검출 및 안경 합성
61 while True:
62     ret, frame = cap.read()
63
64     if not ret:
65         break
66
67     # 얼굴 검출
68     faces = face_classifier.detectMultiScale(frame, scaleFactor=1.2,
69                                             minSize=(100, 100), maxSize=(400, 400))
70
```


[프로젝트] 간단 스노우앱

✓ snowapp.py

```
71     for (x, y, w, h) in faces:
72         #cv2.rectangle(frame, (x, y, w, h), (255, 0, 255), 2)
73
74         # 눈 검출
75         faceROI = frame[y:y + h // 2, x:x + w]
76         eyes = eye_classifier.detectMultiScale(faceROI)
77
78         # 눈을 2개 검출한 것이 아니라면 무시
79         if len(eyes) != 2:
80             continue
81
82         # 두 개의 눈 중앙 위치를 (x1, y1), (x2, y2) 좌표로 저장
83         x1 = x + eyes[0][0] + (eyes[0][2] // 2)
84         y1 = y + eyes[0][1] + (eyes[0][3] // 2)
85         x2 = x + eyes[1][0] + (eyes[1][2] // 2)
86         y2 = y + eyes[1][1] + (eyes[1][3] // 2)
87
```

[프로젝트] 간단 스노우앱

✓ snowapp.py

```
88     if x1 > x2:
89         x1, y1, x2, y2 = x2, y2, x1, y1
90
91     #cv2.circle(faceROI, (x1, y1), 5, (255, 0, 0), 2, cv2.LINE_AA)
92     #cv2.circle(faceROI, (x2, y2), 5, (255, 0, 0), 2, cv2.LINE_AA)
93
94     # 두 눈 사이의 거리를 이용하여 스케일링 팩터를 계산 (두 눈이 수평하다고 가정)
95     fx = (x2 - x1) / (ex2 - ex1)
96     glasses2 = cv2.resize(glasses, (0, 0), fx=fx, fy=fx, interpolation=cv2.INTER_AREA)
97
98     # 크기 조절된 안경 영상을 합성할 위치 계산 (좌상단 좌표)
99     pos = (x1 - int(ex1 * fx), y1 - int(ey1 * fx))
100
101     # 영상 합성
102     overlay(frame, glasses2, pos)
```

[프로젝트] 간단 스노우앱

✓ snowapp.py

```
103
104     # 프레임 저장 및 화면 출력
105     out.write(frame)
106     cv2.imshow('frame', frame)
107
108     if cv2.waitKey(1) == 27:
109         break
110
111 cap.release()
112 out.release()
113 cv2.destroyAllWindows()
```