

# Python OpenCV

---

## 중급 교육 자료

이미지 필터링과 얼굴 인식 기초

# 1. 이미지 필터링과 블러링

## 1.1 평균 블러 (Average Blur)

가장 간단한 블러 방법으로, 커널 영역의 평균값으로 픽셀을 대체합니다.

```
import cv2
import numpy as np

img = cv2.imread('image.jpg')

# 5x5 커널을 사용한 평균 블러
blurred = cv2.blur(img, (5, 5))

# 다양한 커널 크기 비교
blur1 = cv2.blur(img, (3, 3))
blur2 = cv2.blur(img, (7, 7))
blur3 = cv2.blur(img, (15, 15))

cv2.imshow('Original', img)
cv2.imshow('Blur 3x3', blur1)
cv2.imshow('Blur 7x7', blur2)
cv2.imshow('Blur 15x15', blur3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 1.2 가우시안 블러 (Gaussian Blur)

가장 널리 사용되는 블러 기법으로, 가우시안 커널을 사용합니다.

```
import cv2

img = cv2.imread('image.jpg')

# 가우시안 블러 적용
# 커널 크기는 홀수여야 함
gaussian = cv2.GaussianBlur(img, (5, 5), 0)

# 시그마 값 지정
gaussian_sig = cv2.GaussianBlur(img, (5, 5), sigmaX=2)

cv2.imshow('Original', img)
cv2.imshow('Gaussian Blur', gaussian)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

### 가우시안 블러의 장점:

- 노이즈 제거에 효과적
- 이미지의 세부 정보를 부드럽게 유지
- 경계선 보존이 비교적 좋음

## 1.3 미디언 블러 (Median Blur)

소금-후추 노이즈 제거에 매우 효과적입니다.

```
import cv2
import numpy as np

img = cv2.imread('image.jpg')

# 노이즈 추가
noise = np.random.randint(0, 2, img.shape[:2])
noisy = img.copy()
noisy[noise == 1] = 255

# 미디언 블러로 노이즈 제거
median = cv2.medianBlur(noisy, 5)
```

```
cv2.imshow('Noisy', noisy)
cv2.imshow('Median Blur', median)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 1.4 양방향 필터 (Bilateral Filter)

경계선은 보존하면서 노이즈만 제거하는 고급 필터입니다.

```
import cv2

img = cv2.imread('image.jpg')

# 양방향 필터
# d: 필터 크기, sigmaColor: 색상 공간 시그마, sigmaSpace: 좌표 공간 시그마
bilateral = cv2.bilateralFilter(img, d=9, sigmaColor=75, sigmaSpace=75)

# 가우시안과 비교
gaussian = cv2.GaussianBlur(img, (9, 9), 0)

cv2.imshow('Original', img)
cv2.imshow('Bilateral', bilateral)
cv2.imshow('Gaussian', gaussian)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 2. 경계선 검출 (Edge Detection)

### 2.1 Sobel 필터

1차 미분을 이용한 경계선 검출 방법입니다.

```
import cv2
import numpy as np

img = cv2.imread('image.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# X 방향 경계선
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
sobelx = np.absolute(sobelx)
sobelx = np.uint8(sobelx)

# Y 방향 경계선
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)
sobely = np.absolute(sobely)
sobely = np.uint8(sobely)

# 두 방향 합성
sobel_combined = cv2.bitwise_or(sobelx, sobely)

cv2.imshow('Original', gray)
cv2.imshow('Sobel X', sobelx)
cv2.imshow('Sobel Y', sobely)
cv2.imshow('Sobel Combined', sobel_combined)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

### 2.2 Canny 경계선 검출

가장 널리 사용되는 경계선 검출 알고리즘입니다.

```
import cv2

img = cv2.imread('image.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Canny 경계선 검출
# threshold1: 최소 임계값, threshold2: 최대 임계값
edges = cv2.Canny(gray, threshold1=100, threshold2=200)

# 다양한 임계값 비교
edges1 = cv2.Canny(gray, 50, 100)
edges2 = cv2.Canny(gray, 100, 200)
edges3 = cv2.Canny(gray, 200, 300)

cv2.imshow('Original', gray)
cv2.imshow('Canny 50-100', edges1)
cv2.imshow('Canny 100-200', edges2)
cv2.imshow('Canny 200-300', edges3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

#### Canny 알고리즘의 단계:

1. 가우시안 블러로 노이즈 제거
2. Sobel 필터로 경계선 강도와 방향 계산
3. 비최대 억제 (Non-maximum Suppression)
4. 이중 임계값을 이용한 경계선 추출
5. 히스테리시스를 이용한 경계선 연결

## 3. 컨투어 (Contours)

### 3.1 컨투어 찾기

컨투어는 동일한 색상이나 강도를 가진 연속적인 점들의 곡선입니다.

```
import cv2
import numpy as np

img = cv2.imread('shapes.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 이진화
ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# 컨투어 찾기
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)

# 컨투어 그리기
img_contours = img.copy()
cv2.drawContours(img_contours, contours, -1, (0, 255, 0), 2)

print(f"발견된 컨투어 수: {len(contours)}")

cv2.imshow('Original', img)
cv2.imshow('Threshold', thresh)
cv2.imshow('Contours', img_contours)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

### 3.2 컨투어 특징

```
import cv2

img = cv2.imread('shapes.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)

for i, cnt in enumerate(contours):
    # 면적
    area = cv2.contourArea(cnt)

    # 둘레
    perimeter = cv2.arcLength(cnt, True)

    # 중심점
    M = cv2.moments(cnt)
    if M['m00'] != 0:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])

    # 경계 사각형
    x, y, w, h = cv2.boundingRect(cnt)

    print(f"컨투어 {i}: 면적={area:.1f}, 둘레={perimeter:.1f}")

    # 경계 사각형 그리기
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('Bounding Rectangles', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

### 3.3 컨투어 근사

```
import cv2

img = cv2.imread('shapes.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(thresh, cv2.RETR_TREE,
                               cv2.CHAIN_APPROX_SIMPLE)

for cnt in contours:
```

```
# 컨투어 근사
epsilon = 0.02 * cv2.arcLength(cnt, True)
approx = cv2.approxPolyDP(cnt, epsilon, True)

# 꼭짓점 수로 도형 판별
vertices = len(approx)

if vertices == 3:
    shape = "Triangle"
elif vertices == 4:
    shape = "Rectangle/Square"
elif vertices > 4:
    shape = "Circle"
else:
    shape = "Unknown"

# 텍스트 표시
M = cv2.moments(cnt)
if M['m00'] != 0:
    cx = int(M['m10']/M['m00'])
    cy = int(M['m01']/M['m00'])
    cv2.putText(img, shape, (cx-50, cy),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

cv2.imshow('Shape Detection', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 4. 얼굴 인식 기초

### 4.1 Haar Cascade 분류기

OpenCV에서 제공하는 사전 학습된 얼굴 검출 모델입니다.

```
import cv2

# Haar Cascade 분류기 로드
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
)

# 이미지 읽기
img = cv2.imread('people.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 얼굴 검출
faces = face_cascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30)
)

print(f"검출된 얼굴 수: {len(faces)}")

# 얼굴에 사각형 그리기
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow('Face Detection', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

### 4.2 눈 검출

```
import cv2

# Cascade 분류기 로드
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
)
eye_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_eye.xml'
)

img = cv2.imread('people.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 얼굴 검출
faces = face_cascade.detectMultiScale(gray, 1.1, 5)

for (x, y, w, h) in faces:
    # 얼굴 영역
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # 얼굴 영역 내에서 눈 검출
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray, 1.1, 3)
    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)

cv2.imshow('Face and Eye Detection', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

### 4.3 웹캠에서 실시간 얼굴 인식

```
import cv2

# Cascade 분류기 로드
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
)
```

```

# 웹캠 시작
cap = cv2.VideoCapture(0)

while True:
    # 프레임 읽기
    ret, frame = cap.read()
    if not ret:
        break

    # 그레이스케일 변환
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 얼굴 검출
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    # 얼굴에 사각형 그리기
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        cv2.putText(frame, 'Face', (x, y-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)

    # 화면 표시
    cv2.imshow('Face Detection', frame)

    # 'q' 키로 종료
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# 정리
cap.release()
cv2.destroyAllWindows()

```

### Haar Cascade 파라미터:

- **scaleFactor:** 이미지 스케일 감소 비율 (1.1 ~ 1.5 추천)
- **minNeighbors:** 최소 이웃 수 (높을수록 정확, 낮을수록 민감)
- **minSize:** 최소 검출 크기
- **maxSize:** 최대 검출 크기 (선택사항)

## 5. 실습 예제

### 예제 1: 문서 스캔 효과

```
import cv2
import numpy as np

img = cv2.imread('document.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 가우시안 블러로 노이즈 제거
blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# Canny 경계선 검출
edges = cv2.Canny(blurred, 50, 150)

# 컨투어 찾기
contours, _ = cv2.findContours(edges, cv2.RETR_LIST,
                                cv2.CHAIN_APPROX_SIMPLE)

# 가장 큰 사각형 컨투어 찾기
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:5]

for cnt in contours:
    epsilon = 0.02 * cv2.arcLength(cnt, True)
    approx = cv2.approxPolyDP(cnt, epsilon, True)

    # 사각형 (4개 꼭짓점)
    if len(approx) == 4:
        cv2.drawContours(img, [approx], -1, (0, 255, 0), 3)
        break

cv2.imshow('Document Detection', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

### 예제 2: 동전 카운터

```
import cv2
import numpy as np

img = cv2.imread('coins.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 가우시안 블러
blurred = cv2.GaussianBlur(gray, (11, 11), 0)

# 원 검출 (Hough Circle Transform)
circles = cv2.HoughCircles(
    blurred,
    cv2.HOUGH_GRADIENT,
    dp=1,
    minDist=50,
    param1=100,
    param2=30,
    minRadius=20,
    maxRadius=100
)

if circles is not None:
    circles = np.uint16(np.around(circles))

    for i in circles[0, :]:
        # 원의 외곽
        cv2.circle(img, (i[0], i[1]), i[2], (0, 255, 0), 2)
        # 원의 중심
        cv2.circle(img, (i[0], i[1]), 2, (0, 0, 255), 3)

    print(f"동전 개수: {len(circles[0])}")

cv2.imshow('Coin Counter', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 6. 연습 문제

### 문제 1: 노이즈 제거 비교

노이즈가 있는 이미지에 대해 가우시안 블러, 미디언 블러, 양방향 필터를 각각 적용하고, 결과를 비교하는 프로그램을 작성하세요.

### 문제 2: 도형 인식

이미지에서 삼각형, 사각형, 원을 검출하고, 각 도형의 개수를 세는 프로그램을 작성하세요.

### 문제 3: 얼굴 모자이크

이미지에서 얼굴을 검출하고, 검출된 얼굴 영역에 모자이크 효과를 적용하는 프로그램을 작성하세요.

### 문제 4: 차선 검출

도로 이미지에서 Canny 경계선 검출과 Hough Line Transform을 이용하여 차선을 검출하세요.

### 문제 5: 객체 추적

특정 색상의 객체를 HSV 색상 공간을 이용하여 추적하고, 그 중심점에 원을 그리는 프로그램을 작성하세요.

### 힌트

- 문제 1: `cv2.GaussianBlur()`, `cv2.medianBlur()`, `cv2.bilateralFilter()` 비교
- 문제 2: `cv2.findContours()`와 `cv2.approxPolyDP()` 사용
- 문제 3: 얼굴 영역을 작게 리사이즈 후 다시 확대하여 모자이크 효과
- 문제 4: `cv2.HoughLinesP()` 함수 사용
- 문제 5: `cv2.inRange()`로 마스크 생성 후 `cv2.moments()` 사용

수고하셨습니다!

중급 과정을 완료하셨습니다.

다음 고급 과정에서는 영상 처리, 객체 추적, 고급 얼굴 인식 등을 배우게 됩니다.