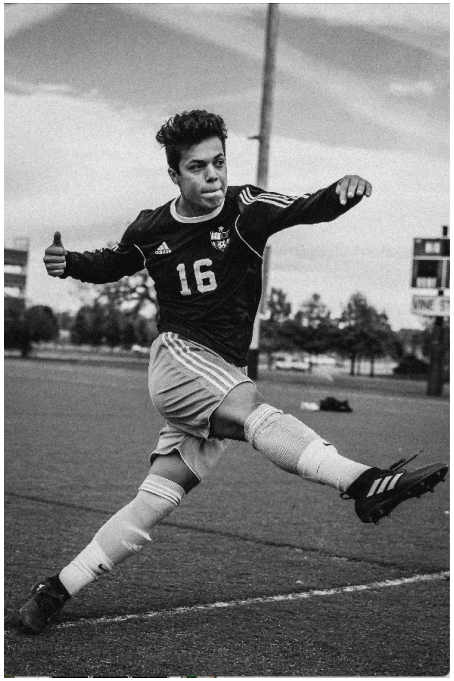
파이썬으로 배우는 딥러닝

이미지 전처리 (OpenCV)

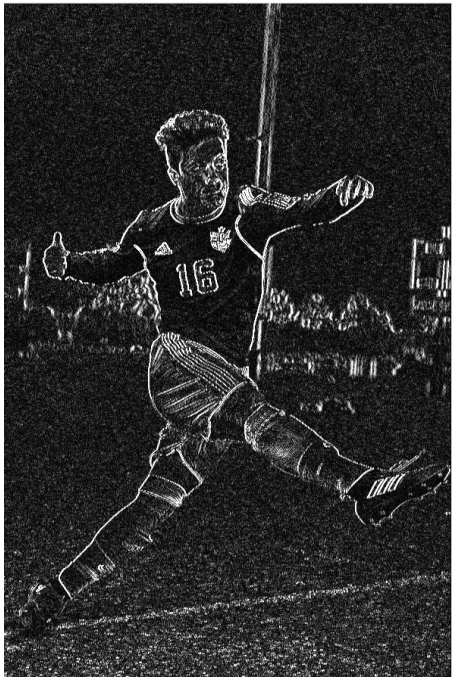
Ch.04 OpenCV 엣지와 영역 검출

Sobel 연산자를 활용한 엣지 검출

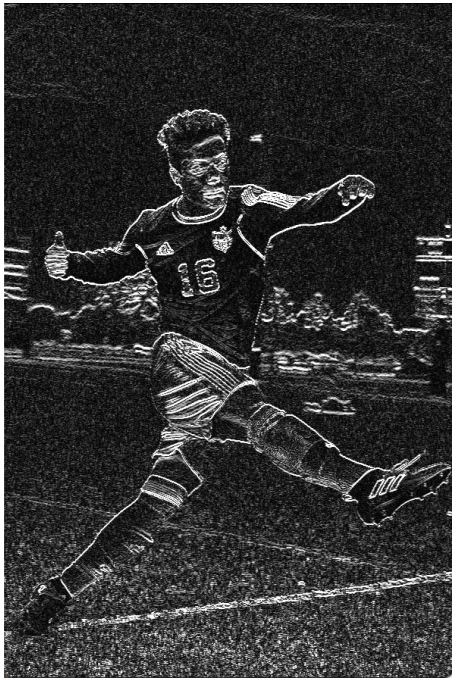
◆ 엣지 검출 결과



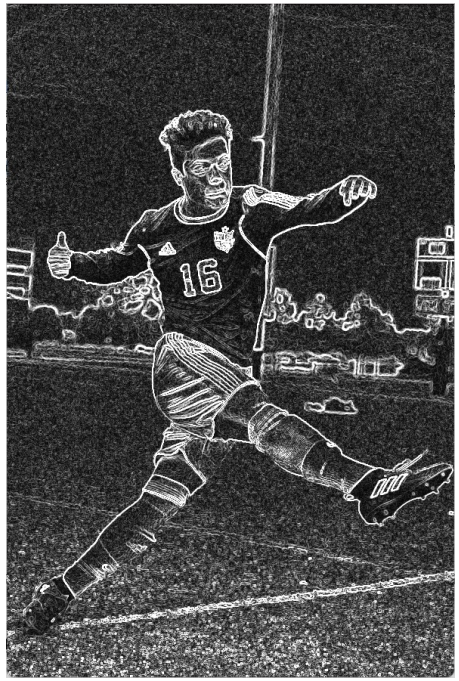
Original



sobel_x



sobel_y



strength

[OpenCV 예제1] Sobel 연산자를 활용한 엣지 검출

◆ **Ch.04₩4-1.py에 아래 코드 작성해 넣기**

```python
import cv2 as cv
import numpy as np

# 이미지 불러오기 (그레이스케일)
img = cv.imread('Ch.04/soccer.jpg', cv.IMREAD_GRAYSCALE)

# Sobel 연산자 적용 (64F로 계산 후 절댓값, uint8 변환)
sobel_x = cv.Sobel(img, cv.CV_64F, 1, 0, ksize=3)
sobel_y = cv.Sobel(img, cv.CV_64F, 0, 1, ksize=3)

# 엣지 강도 계산: sqrt(sobel_x^2 + sobel_y^2)
magnitude = np.sqrt(sobel_x**2 + sobel_y**2)

# 정수형으로 변환
abs_sobel_x = cv.convertScaleAbs(sobel_x)
abs_sobel_y = cv.convertScaleAbs(sobel_y)
edge_strength = cv.convertScaleAbs(magnitude)

# 출력
cv.imshow('Original', img)
cv.imshow('Sobel X', abs_sobel_x)
cv.imshow('Sobel Y', abs_sobel_y)
cv.imshow('Edge Strength (Magnitude)', edge_strength)

cv.waitKey(0)
cv.destroyAllWindows()
```

Sobel 연산자를 활용한 엣지 검출

◆ **sobel_x = cv.Sobel(img, cv.CV_64F, 1, 0, ksize=3)**

- OpenCV에서 소벨(Sobel) 연산자를 사용하여 이미지의 수평 방향(가로 방향) 경계(엣지)를 계산하는 코드. 소벨 연산은 미분을 통해 이미지에서 경계를 추출하는데, 주로 엣지 검출에 사용됨.
- gray: 입력 이미지로, 보통 그레이스케일 이미지에서 경계를 추출하기 위해 사용됨.
- cv.CV_32F: 출력 이미지의 데이터 타입을 지정. 소벨 연산 결과는 음수 값도 포함될 수 있으므로, 8비트 형식 대신 부동소수점 형식을 사용함.
- 1, 0: 소벨 연산의 미분 차수를 나타냄. 1은 x방향 1차 미분, 0는 y방향 미분 미수행
- ksize=3: 커널 크기를 설정합니다. 여기서는 3x3 크기의 커널을 사용. 일반적으로 커널 크기는 3, 5, 7 등의 홀수 값으로 설정되며, 커널 크기가 커질수록 경계 검출이 더 부드럽고 넓은 영역에서 이루어짐.

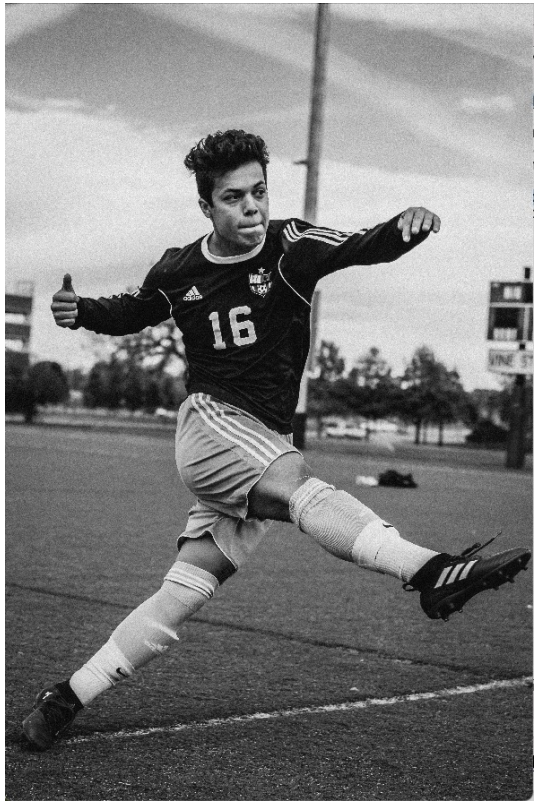
Sobel 연산자를 활용한 엣지 검출

◆ **magnitude = np.sqrt(sobel_x**2 + sobel_y**2)**

- x 방향과 y 방향의 소벨(Sobel) 연산 결과를 가중합(weighted sum)하여 **에지 강도(edge strength)**를 계산하는 코드
- sobel_x: x 방향으로 소벨 연산을 수행한 결과 이미지. 이는 수평 방향의 경계(수직 엣지)를 강조
- sobel_y: y 방향으로 소벨 연산을 수행한 결과 이미지. 이는 수직 방향의 경계(수평 엣지)를 강조

캐니를 활용한 엣지 검출

◆ 캐니 에지 검출 결과 – 의미적 검출



Original

Canny

[OpenCV 예제2] 캐니를 활용한 엣지 검출

◆ **Ch.04₩4-2.py에 아래 코드 작성해 넣기**

```python
import cv2 as cv

# 이미지 불러오기 (그레이스케일)
img = cv.imread('Ch.04/soccer.jpg', cv.IMREAD_GRAYSCALE)

# 가우시안 블러 적용 (노이즈 제거)
blurred = cv.GaussianBlur(img, (5, 5), 1.4)

# 캐니 엣지 검출 (임계값1, 임계값2)
edges = cv.Canny(blurred, 100, 200)

# 결과 출력
cv.imshow('Original', img)
cv.imshow('Canny Edge', edges)

cv.waitKey(0)
cv.destroyAllWindows()
```

캐니를 활용한 엣지 검출

◆ **edges = cv.Canny(blurred, 100, 200)**

- OpenCV를 사용하여 **Canny 에지 검출기(Canny Edge Detector)**를 적용하는 코드
- Tlow=100, Thigh=200으로 설정되어 있으며, 이 값들은 에지 검출을 위한 임계값. Canny 에지 검출은 이미지에서 중요한 에지를 찾는 매우 효과적인 방법임.
- 비최대 억제(Non-Maximum Suppression) : 경계가 존재하는 것으로 판단되는 곳 중에서 실제로 경계일 가능성이 높은 곳만 남기고, 나머지는 억제
- 100: 낮은 임계값 Tlow. 그레이디언트 크기가 이 값보다 작으면 에지로 간주되지 않음
- 200: 높은 임계값 Thigh. 그레이디언트 크기가 이 값보다 크면 강한 에지로 간주.

경계선 찾기

◆ **canny edge를 이용하여 경계선 찾기**

[OpenCV 예제3] 경계선 찾기

◆ **Ch.04₩4-3.py에 아래 코드 작성해 넣기**

```python
import cv2 as cv
import numpy as np

# 이미지 읽기
image = cv.imread('Ch.04/soccer.jpg')  # 컬러 이미지
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

cv.imshow("aaa", image)

# Canny 엣지 검출
edges = cv.Canny(gray, 100, 200)

# 윤곽선 찾기
contours, _ = cv.findContours(edges, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

# 윤곽선 길이가 50 이상인 것만 원본 이미지에 그리기
for contour in contours:
    length = cv.arcLength(contour, closed=True)
    if length >= 50:
        cv.drawContours(image, [contour], -1, (0, 255, 0), 1)  # 초록색, 두께 1

# 결과 이미지 출력
cv.imshow("Contours over Original", image)
cv.waitKey(0)
cv.destroyAllWindows()
```

경계선 찾기

◆ **contours, _ = cv.findContours(edges, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)**

- OpenCV를 사용하여 **이미지에서 경계선(Contour)**를 찾는 코드
- edges : 입력 이미지로, 보통 에지 검출 결과나 이진화된 이미지가 사용됨. 여기서는 Canny 에지 검출을 통해 얻은 결과 이미지(canny)가 입력으로 사용됨.
- cv.RETR_LIST: 윤곽선을 검색하는 방식. RETR_LIST는 모든 윤곽선을 계층 구조와 상관없이 단순 리스트 형태로 반환하는 모드. 윤곽선 간의 계층 관계(부모-자식 관계)를 무시하고 모든 윤곽선을 찾음.
- cv.CHAIN_APPROX_NONE: 윤곽선을 근사하는 방식. CHAIN_APPROX_NONE은 모든 경계점을 저장.

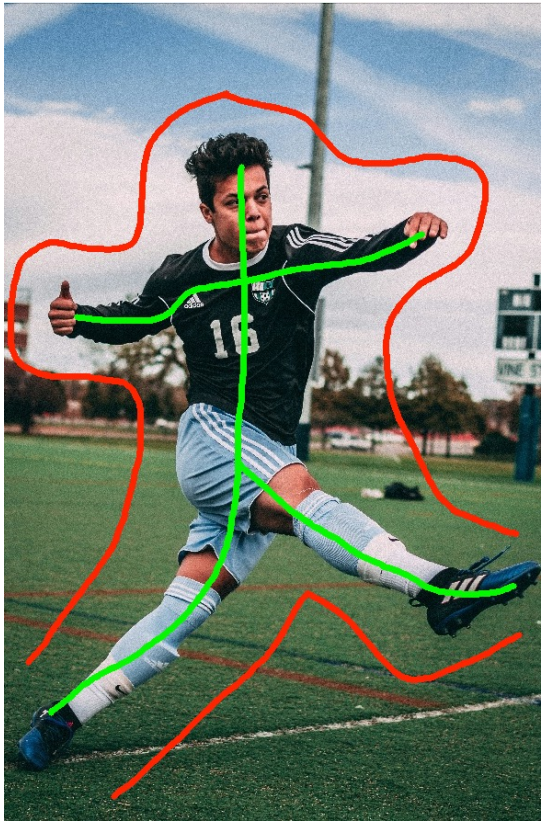
경계선 찾기

◆ **cv.drawContours(image, [contour], -1, (0, 255, 0), 1) # 초록색, 두께 1**

- OpenCV를 사용하여 **윤곽선(Contours)**을 이미지 위에 그리는 코드
- image : 윤곽선을 그릴 대상 이미지
- [contour]: 그릴 윤곽선의 리스트. 이 리스트는 cv.findContours() 함수로 얻은 윤곽선 데이터이며, 각 윤곽선은 경계를 이루는 좌표 값으로 구성된 배열
- -1: 그릴 윤곽선의 인덱스. -1로 설정하면 모든 윤곽선을 그림.
- (0, 255, 0): 윤곽선을 그릴 색상. OpenCV에서 색상은 BGR 형식으로 지정되므로, (0, 255, 0)은 초록색을 의미함.
- 1: 윤곽선의 두께를 나타냄.. 여기서는 3픽셀 두께로 그려짐.

GrabCut – 배경 제거하기

◆ **사용자 상호작용을 통해 배경 제거하기**



[OpenCV 예제4] GrabCut – 배경 제거하기

◆ **Ch.04₩4-4.py에 아래 코드 작성해 넣기**

```python
import cv2 as cv
import numpy as np

# 초기 설정
drawing = False
value = cv.GC_PR_FGD  # 기본은 왼쪽 클릭 → 물체
ix, iy = -1, -1

# 마우스 이벤트 콜백 함수
def draw_mask(event, x, y, flags, param):
    global drawing, ix, iy, value, mask, image_display

    if event == cv.EVENT_LBUTTONDOWN:
        drawing = True
        value = cv.GC_FGD  # 확실한 물체
        ix, iy = x, y

    elif event == cv.EVENT_RBUTTONDOWN:
        drawing = True
        value = cv.GC_BGD  # 확실한 배경
        ix, iy = x, y

    elif event == cv.EVENT_MOUSEMOVE:
        if drawing:
            cv.line(mask, (ix, iy), (x, y), value, 5)
            color = (0, 0, 255) if value == cv.GC_BGD else (0, 255, 0)
            cv.line(image_display, (ix, iy), (x, y), color, 5)
            ix, iy = x, y

    elif event == cv.EVENT_LBUTTONUP or event == cv.EVENT_RBUTTONUP:
        drawing = False
```

```python
# 이미지 읽기 및 초기화
image = cv.imread('Ch.04/soccer.jpg')
image_display = image.copy()
mask = np.full(image.shape[:2], cv.GC_PR_BGD, dtype=np.uint8)  # 기본은 불확실 배경

cv.namedWindow('Input')
cv.setMouseCallback('Input', draw_mask)

print("왼쪽 클릭 → 물체, 오른쪽 클릭 → 배경, 그린 후 Enter 누르면 GrabCut 수행")

while True:
    cv.imshow('Input', image_display)
    key = cv.waitKey(1)

    if key == 13:  # Enter 키
        break
    elif key == 27:  # ESC 키
        cv.destroyAllWindows()
        exit()

# GrabCut 초기화용 배열
bgdModel = np.zeros((1, 65), np.float64)
fgdModel = np.zeros((1, 65), np.float64)

# GrabCut 적용
cv.grabCut(image, mask, None, bgdModel, fgdModel, 5, cv.GC_INIT_WITH_MASK)

# 결과 마스크 생성: 0,2=배경 / 1,3=물체
result_mask = np.where((mask == cv.GC_FGD) | (mask == cv.GC_PR_FGD), 255, 0).astype('uint8')

# 결과 적용
output = cv.bitwise_and(image, image, mask=result_mask)

# 결과 보기
cv.imshow('Extracted Object', output)
cv.waitKey(0)
cv.destroyAllWindows()
```

GrabCut – 배경 제거하기

◆ **cv.grabCut(image, mask, None, bgdModel, fgdModel, 5, cv.GC_INIT_WITH_MASK)**

- OpenCV에서 제공하는 반자동 전경-배경 분리(segmentation) 알고리즘으로 이미지와 초기 마스크, 또는 사각형 정보를 기반으로 물체를 분리하는 데 사용됨.
- img : 입력이미지
- mask : 전경/배경 정보를 가진 마스크
- rect : 전경이 포함된 사각형 영역
- bgdModel : 내부적으로 사용하는 배경 모델 배열
- fgdModel : 내부적으로 사용하는 전경 모델 배열
- 5(iterCount) : 반복 횟수
- cv.GC_INIT_WITH_MASK : 초기화 모드