Python OpenCV

초급 교육 자료

이미지 처리 기초부터 시작하기

1. OpenCV 소개

OpenCV(Open Source Computer Vision Library)란?

OpenCV는 실시간 컴퓨터 비전을 목적으로 한 오픈소스 라이브러리입니다. 이미지 처리, 비디오 분석, 객체 인식, 얼굴 인식 등 다양한 기능을 제공합니다.

주요 특징

- 무료 오픈소스 라이브러리
- 2500개 이상의 최적화된 알고리즘 제공
- Python, C++, Java 등 다양한 언어 지원
- 실시간 처리에 최적화되어 있음
- 크로스 플랫폼 (Windows, Linux, Mac, Android, iOS)

활용 분야

- 얼굴 인식 및 객체 검출
- 의료 영상 분석
- 자율주행 자동차
- 증강 현실(AR)
- 보안 감시 시스템
- 로봇 비전

2. 설치 및 환경 설정

OpenCV 설치하기

pip를 사용하여 간단하게 설치할 수 있습니다:

```
pip install opencv-python
pip install opencv-contrib-python  # 추가 모듈 포함
```

설치 확인하기

```
import cv2
print(cv2.__version__)  # OpenCV 버전 확인
```

필요한 추가 라이브러리

- **numpy**: 배열 및 수학 연산
- **matplotlib**: 이미지 시각화
- **PIL/Pillow**: 이미지 파일 처리 보조

```
pip install numpy matplotlib pillow
```

3. 이미지 읽기, 쓰기, 보기

3.1 이미지 읽기 (cv2.imread)

`cv2.imread()` 함수는 이미지 파일을 읽어 NumPy 배열로 반환합니다.

```
import cv2

# 컬러 이미지로 읽기 (기본값)
img = cv2.imread('image.jpg')

# 흑백 이미지로 읽기
img_gray = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# 알파 채널 포함하여 읽기
img_rgba = cv2.imread('image.png', cv2.IMREAD_UNCHANGED)
```

imread 플래그

- `cv2.IMREAD_COLOR` : 컬러 이미지로 읽기 (기본값)
- `cv2.IMREAD_GRAYSCALE` : 흑백 이미지로 읽기
- `cv2.IMREAD_UNCHANGED` : 알파 채널 포함 원본 그대로

3.2 이미지 보기 (cv2.imshow)

`cv2.imshow()` 함수는 창을 띄워 이미지를 표시합니다.

```
import cv2

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

# 이미지 표시
cv2.imshow('Image Window', img)

# 키 입력 대기 (0은 무한 대기, 숫자는 밀리초)
cv2.waitKey(0)

# 모든 창 닫기
cv2.destroyAllWindows()
```

3.3 이미지 저장 (cv2.imwrite)

```
import cv2

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

# 이미지 저장
cv2.imwrite('output.jpg', img)

# 압축률 조정하여 저장 (JPEG)
cv2.imwrite('output.jpg', img, [cv2.IMWRITE_JPEG_QUALITY, 90])

# PNG 압축 레벨 지정 (0-9)
cv2.imwrite('output.png', img, [cv2.IMWRITE_PNG_COMPRESSION, 9])
```

4. 이미지 기본 처리

4.1 이미지 크기 조정 (cv2.resize)

```
import cv2

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

# 절대 크기로 조정
resized = cv2.resize(img, (300, 200))  # (너비, 높이)

# 비율로 조정
resized = cv2.resize(img, None, fx=0.5, fy=0.5)  # 50%로 축소

# 보간법 지정
resized = cv2.resize(img, (300, 200), interpolation=cv2.INTER_LINEAR)
```

보간법 종류

- `cv2.INTER_NEAREST` : 최근접 이웃 (가장 빠름)
- `cv2.INTER_LINEAR` : 선형 보간 (기본값)
- `cv2.INTER_CUBIC` : 3차 보간 (고품질)
- `cv2.INTER_AREA` : 영역 보간 (축소 시 추천)

4.2 이미지 자르기 (슬라이싱)

```
import cv2

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

# NumPy 배열 슬라이싱 사용
# img[y:y+h, x:x+w]
cropped = img[100:400, 200:500]  # y:100-400, x:200-500

cv2.imshow('Cropped', cropped)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4.3 이미지 회전

```
import cv2

img = cv2.imread('image.jpg')
height, width = img.shape[:2]

# 회전 중심점 (보통 이미지 중앙)
center = (width // 2, height // 2)

# 회전 행렬 생성 (중심점, 각도, 스케일)
matrix = cv2.getRotationMatrix2D(center, 45, 1.0)  # 45도 회전

# 회전 적용
rotated = cv2.warpAffine(img, matrix, (width, height))

cv2.imshow('Rotated', rotated)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4.4 이미지 뒤집기 (cv2.flip)

```
import cv2

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

```
# 좌우 반전
flipped_h = cv2.flip(img, 1)

# 상하 반전
flipped_v = cv2.flip(img, 0)

# 상하좌우 반전
flipped_both = cv2.flip(img, -1)
```

flip 파라미터

- **1**: 좌우 반전 (수평)
- **0**: 상하 반전 (수직)
- **-1**: 상하좌우 모두 반전

5. 색상 공간 변환

5.1 BGR과 RGB

OpenCV는 기본적으로 BGR(Blue-Green-Red) 순서로 색상을 저장합니다. 이는 일반적인 RGB와 반대입니다.

```
import cv2

img = cv2.imread('image.jpg')  # BGR 형식

# BGR을 RGB로 변환
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Matplotlib로 표시할 때는 RGB 변환 필요
import matplotlib.pyplot as plt
plt.imshow(img_rgb)
plt.show()
```

5.2 그레이스케일 변환

```
import cv2

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

# BGR을 그레이스케일로 변환
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imshow('Grayscale', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5.3 HSV 색상 공간

HSV(Hue-Saturation-Value)는 색상 기반 작업에 유용합니다. 특정 색상 범위를 쉽게 추출할 수 있습니다.

```
import cv2
import numpy as np

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

# BGR을 HSV로 변환
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# 특정 색상 범위 추출 (예: 파란색)
lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])

# 마스크 생성
mask = cv2.inRange(hsv, lower_blue, upper_blue)

# 원본 이미지에 마스크 적용
result = cv2.bitwise_and(img, img, mask=mask)

cv2.imshow('Original', img)
cv2.imshow('Mask', mask)
cv2.imshow('Result', result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

주요 색상 변환 코드

- `cv2.COLOR_BGR2RGB` : BGR → RGB
- `cv2.COLOR_BGR2GRAY` : BGR → 그레이스케일
- `cv2.COLOR_BGR2HSV` : BGR → HSV
- `cv2.COLOR_GRAY2BGR` : 그레이스케일 → BGR

6. 실습 예제

예제 1: 이미지 썸네일 만들기

```
import cv2

# 이미지 읽기
img = cv2.imread('photo.jpg')

# 썸네일 크기로 조정 (너비 200px)
height, width = img.shape[:2]
aspect_ratio = height / width
new_width = 200
new_height = int(new_width * aspect_ratio)

thumbnail = cv2.resize(img, (new_width, new_height))

# 저장
cv2.imwrite('thumbnail.jpg', thumbnail)
print(f"썸네일 생성: {new_width}x{new_height}")
```

예제 2: 이미지 밝기 조절

```
import cv2
import numpy as np

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

# 방법 1: 직접 값 더하기
brightness = 50
bright_img = cv2.add(img, np.array([brightness]))

# 방법 2: convertScaleAbs 사용
alpha = 1.0  # 대비 (1.0 = 원본)
beta = 50    # 밝기
adjusted = cv2.convertScaleAbs(img, alpha=alpha, beta=beta)

cv2.imshow('Original', img)
cv2.imshow('Brighter', bright_img)
cv2.imshow('Adjusted', adjusted)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

예제 3: 특정 색상 추출

```
import cv2
import numpy as np

img = cv2.imread('photo.jpg')
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# 빨간색 범위 (HSV)
lower_red1 = np.array([0, 50, 50])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 50, 50])
upper_red2 = np.array([180, 255, 255])

mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
mask = cv2.bitwise_or(mask1, mask2)

result = cv2.bitwise_and(img, img, mask=mask)
cv2.imshow('Red Objects', result)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

7. 연습 문제

문제 1: 기본 이미지 처리

이미지를 읽어서 다음 작업을 수행하는 프로그램을 작성하세요:

- 이미지를 50%로 축소
- 좌우 반전
- 그레이스케일로 변환
- 결과를 'result1.jpg'로 저장

문제 2: 색상 추출

이미지에서 빨간색 영역만 추출하여 표시하는 프로그램을 작성하세요.
(힌트: HSV 색상 공간 사용)

문제 3: 이미지 합성

두 개의 이미지를 읽어서 50:50 비율로 블렌딩한 후 저장하세요.

문제 4: 이미지 회전

이미지를 중심점을 기준으로 90도 회전시키고, 회전된 이미지가 잘리지 않도록 캔버스 크기를 조정하세요.

문제 5: 이미지 모자이크

4개의 이미지를 2x2 격자로 배치한 하나의 이미지를 만드세요.
(힌트: NumPy의 hstack, vstack 사용)

힌트

- **문제 1**: cv2.resize(), cv2.flip(), cv2.cvtColor() 함수 사용
- **문제 2**: HSV에서 빨간색은 두 범위로 나뉩니다 (0-10, 170-180)
- **문제 3**: cv2.addWeighted() 함수 사용
- **문제 4**: 회전 후 이미지 크기는 대각선 길이를 고려해야 합니다
- **문제 5**: 모든 이미지를 같은 크기로 조정한 후 배치

수고하셨습니다!

초급 과정을 완료하셨습니다.
다음 중급 과정에서는 이미지 필터링, 경계선 검출, 얼굴 인식 등을 배우게 됩니다.

연습 문제를 충분히 풀어보시고, 코드를 직접 작성하며 익히시기 바랍니다.