동양미래대학교

파이썬을 활용한 컴퓨터 비전 입문

Chapter 03. OpenCV 주요 행렬 연산

동양미래대학교
인공지능소프트웨어학과
권 범

본 강의자료는 길벗 출판사의『OpenCV 4로 배우는 컴퓨터 비전과 머신 러닝』
교재 내용을 토대로 작성되었습니다.

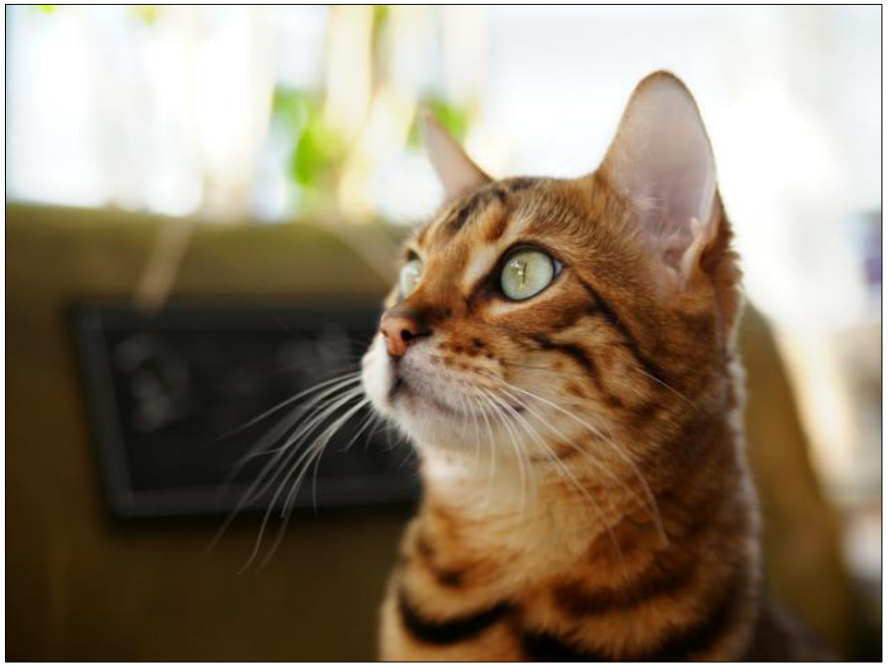
목차

3.1 행렬의 복사

3.2 부분 행렬 추출

❖ 영상 데이터의 유형과 모양 (1/3)

- `imread()` 함수로 불러들인 영상 데이터의 유형(Type)과 모양(Shape)을 살펴보자



**Python에서 불러들인
영상 데이터의 유형과 모양은?**

❖ 영상 데이터의 유형과 모양 (2/3)

코드 3-8 영상 데이터의 유형과 모양 확인 (matrix.py)

```
1   import cv2
2
3   def type_shape():
4       img = cv2.imread('cat.bmp', cv2.IMREAD_GRAYSCALE)  # cv2.IMREAD_COLOR
5
6       if img is None:
7           print('Image load failed!')
8           return
9
10      print('type(img):', type(img))
11      print('img.shape:', img.shape)
12
13      if len(img.shape) == 2:
14          print('img is a grayscale image')
15      elif len(img1.shape) == 3:
16          print('img is a truecolor image')
17
18      cv2.imshow('img', img)
19      cv2.waitKey()
20      cv2.destroyAllWindows()
21
22  if __name__ == '__main__':
23      type_shape()
```

❖ 영상 데이터의 유형과 모양 (3/3)

- matrix.py 소스 코드 실행 결과

cv2.IMREAD_GRAYSCALE

```
type(img1): <class 'numpy.ndarray'>
img1.shape: (480, 640)
img1 is a grayscale image
```

cv2.IMREAD_COLOR

```
type(img1): <class 'numpy.ndarray'>
img1.shape: (480, 640, 3)
img1 is a truecolor image
```

❖ 행렬의 복사 (1/6)

- `imread()` 함수로 불러들인 영상 데이터는 NumPy ndarray 타입으로 저장되는 것을 확인하였다
- 이번에는 NumPy ndarray 타입의 변수에 저장된 행렬 객체를 다른 행렬 객체에 대입하거나 복사하는 방법에 대해 알아보자

❖ 행렬의 복사 (2/6)

- 먼저 강아지 사진이 담겨 있는 dog.bmp 파일을 불러와서 변수 `img1`에 저장하고, 이를 이용하여 다양한 예제 코드를 만들어 보자

```
img1 = imread('dog.bmp')
```

- NumPy ndarray 객체에 저장된 영상 또는 행렬을 복사하는 가장 간단한 방법은 대입 연산자를 사용하는 것이다
- 대입 연산자를 이용하여 `img1` 영상을 새로운 변수 `img2`에 복사하려면 다음과 같이 코드를 작성함

```
img2 = img1
```

❖ 행렬의 복사 (3/6)

- 만약 복사본 영상을 새로 생성할 때, 픽셀 데이터를 공유하는 것이 아니라 메모리 공간을 새로 할당하여 픽셀 데이터 전체를 복사하고 싶다면 `copy()` 메서드를 사용해야 함
- 앞에서 dog.bmp 강아지 영상을 저장하고 있던 `img1` 영상을 `copy()` 메서드를 이용하여 새로운 영상에 복사하려면 다음과 같이 코드를 작성함

```
img3 = img1.copy()
```

- 대입 연산자를 이용하는 행렬의 복사와 `copy()` 메서드를 이용한 행렬의 복사 차이를 직관적으로 이해할 수 있는 예제 코드를 살펴보자

❖ 행렬의 복사 (4/6)

코드 3-8 행렬의 다양한 복사 방법 예제 (matrix.py)

```
1   import cv2
2
3   def copy_method():
4       img1 = cv2.imread('dog.bmp')
5
6       img2 = img1
7       img3 = img1.copy()
8
9       img1[:, :] = (0, 255, 255)  # yellow
10
11      cv2.imshow('img1', img1)
12      cv2.imshow('img2', img2)
13      cv2.imshow('img3', img3)
14      cv2.waitKey()
15      cv2.destroyAllWindows()
16
17  if __name__ == '__main__':
18      copy_method()
```

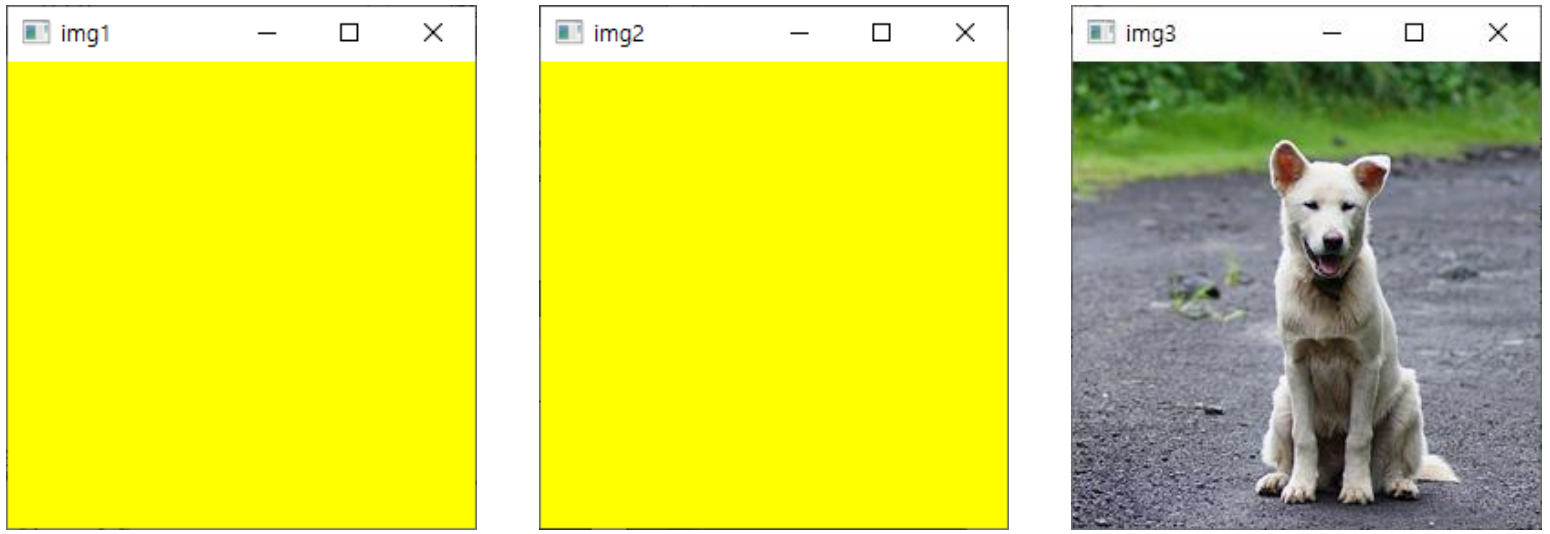
❖ 행렬의 복사 (5/6)

- matrix.py 소스 코드 설명
  - 4행 dog.bmp 강아지 영상을 불러와서 `img1`에 저장합니다.
  - 6행 대입 연산자를 이용하여 `img1`의 복사본 영상 `img2`를 생성합니다.
  - 7행 `copy()` 메서드를 이용하여 `img1`의 복사본 영상 `img3`를 생성합니다.
  - 9행 `img1` 영상의 모든 픽셀을 (0, 255, 255)에 해당하는 노란색으로 설정합니다.
  - 11~13행 `img1~img3` 영상을 모두 새 창으로 출력합니다.

❖ 행렬의 복사 (6/6)

- 9행에서 `img1` 영상만 노란색으로 설정하였지만 출력 결과를 보면 `img1`, `img2` 영상이 모두 노란색으로 바뀐 것을 확인할 수 있음
- img2 영상이 `img1`의 픽셀 데이터를 공유하기 때문에 나타난 결과임
- 반면에 `img3` 영상은 강아지 영상을 그대로 간직하고 있음

▼ **그림 3-3 행렬의 다양한 복사 방법 예제 실행 결과**

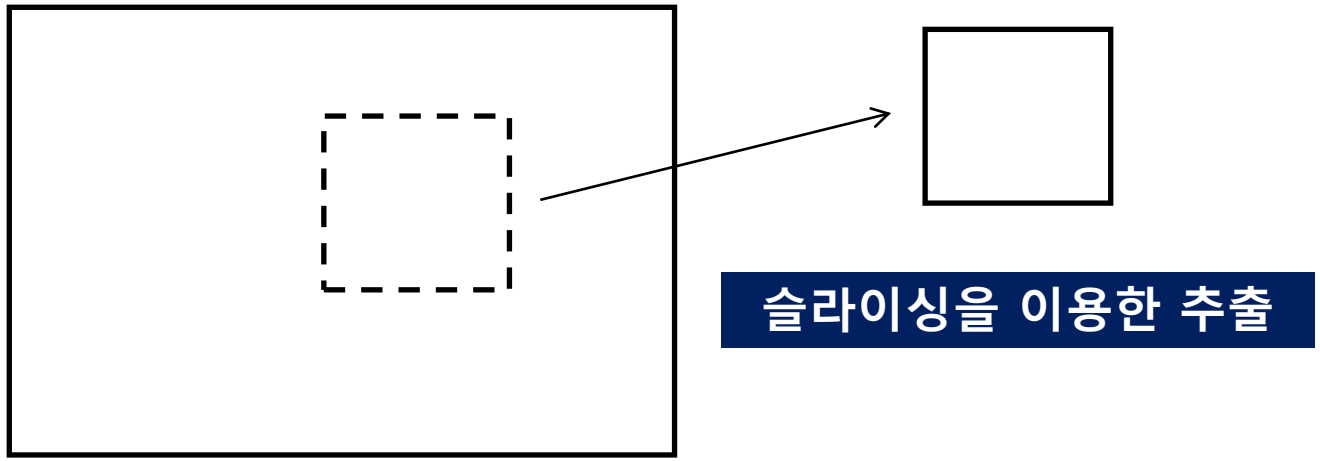


3.2 부분 행렬 추출

3.1 행렬의 복사

❖ 부분 행렬 추출 (1/9)

- NumPy ndarray로 정의된 행렬에서 특정 사각형 영역의 부분 행렬을 추출하고 싶을 때에는, 슬라이싱(Slicing)을 사용함



❖ 부분 행렬 추출 (2/9)

- 다음 예제 코드는 lenna.bmp 파일에 저장된 영상을 불러와서 얼굴 주변의 부분 영상을 추출하는 예제 코드임

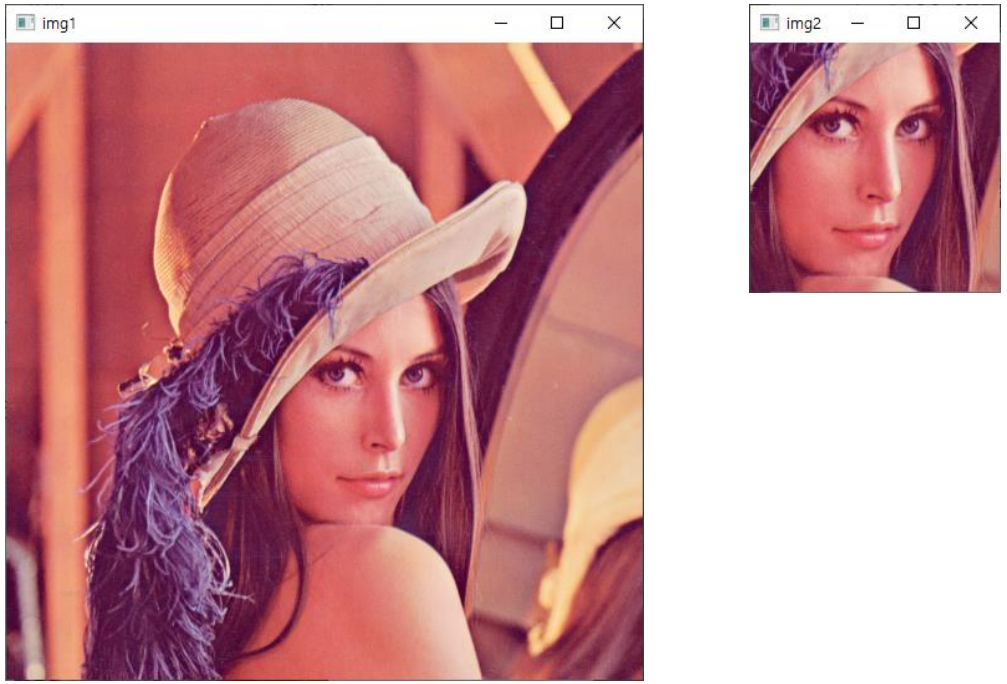
```
img1 = cv2.imread('lenna.bmp', cv2.IMREAD_COLOR)
img2 = img1[200:400, 200:400, :]
```

- 이 예제 코드의 첫 번째 행에서는 lenna.bmp 영상을 3채널 컬러 영상 형태로 불러와서 `img1` 변수에 저장함
- 두 번째 행에서 `img1` 변수 이름 바로 뒤에 대괄호`[]`를 붙여서 사용하였는데, 이 부분이 슬라이싱 동작을 수행함
- `img1[200:400, 200:400, :]` 코드는 `img1` 영상의 (200, 200) 좌표부터 200×200 크기만큼의 사각형 부분 영상을 추출하는 코드임
- 추출한 부분 영상은 `img2` 변수에 저장함

❖ 부분 행렬 추출 (3/9)

- `img1`과 `img2` 영상을 `imshow()` 함수를 사용하여 화면에 출력하면 그림 3-4와 같이 나타남
- `img2` 창에 나타난 부분 영상의 가로와 세로의 크기는 각각 200 픽셀임

▼ **그림 3-4 부분 영상 추출 결과**



❖ 부분 행렬 추출 (4/9)

- 부분 영상을 추출할 때 주의할 점은 대입 연산자를 이용하여 얻은 부분 영상은 픽셀 데이터를 공유하는 형식이라는 점임
- 부분 영상을 추출한 후 부분 영상의 픽셀 값을 변경하면 추출한 부분 영상뿐만 아니라 원본 영상의 픽셀 값도 함께 변경됨
- 부분 영상 추출 시 픽셀 데이터를 공유한다는 특성을 이용하면 입력 영상의 일부분에만 특정한 영상 처리를 수행할 수 있음

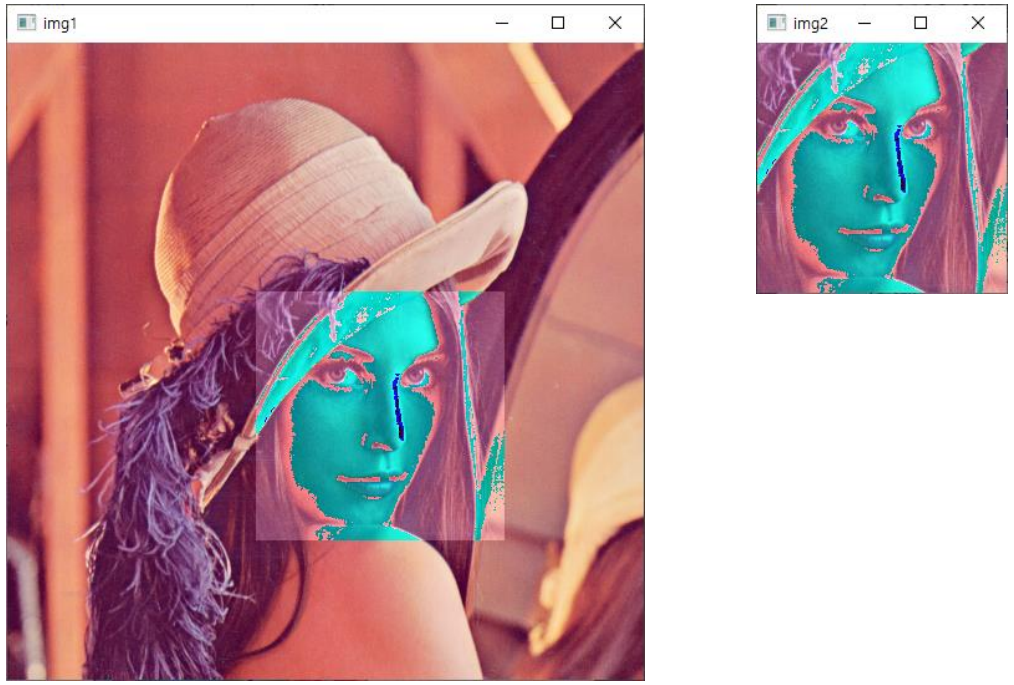
❖ 부분 행렬 추출 (5/9)

- 예를 들어, 부분 영상 `img2`에 +50을 수행하기 위해 다음과 같이 코드를 작성함

```
img2 += 50
```

- `img2` 영상 전체가 +50이 되었고, 더불어 `img1` 영상에서 얼굴 주변의 부분 영상만 +50이 되어 나타나는 것을 확인할 수 있음

▼ **그림 3-5 부분 영상 추출 후 50을 더한 결과**



❖ 부분 행렬 추출 (6/9)

- 부분 영상 참조 기능은 입력 영상에 사각형 모양의 **관심 영역**(**ROI**, **Region Of Interest**)을 설정하는 용도로 사용할 수 있음
- ROI는 영상의 전체 영역 중에서 특정 영역에 대해서만 영상 처리를 수행할 때 설정하는 영역을 의미함
- 사각형이 아닌 임의의 모양의 ROI를 설정하고 싶은 경우에는 마스크 연산을 응용함

❖ 부분 행렬 추출 (7/9)

- 만약 독립된 메모리 영역을 확보하여 부분 영상을 추출하고자 한다면, 대괄호[] 뒤에 copy() 메서드를 함께 사용해야 함
- 대괄호[] 바로 뒤에 .copy()를 붙여서 사용하면 독립된 복사본의 부분 영상을 만들 수 있음

```
img3 = img1[200:400, 200:400, :].copy()
```

- 위와 같이 코드를 작성하면 img1 영상과 img3 영상은 서로 다른 메모리 공간을 사용함
- 추후 img3 영상의 픽셀 값을 변경해도 img1 영상은 변경되지 않음

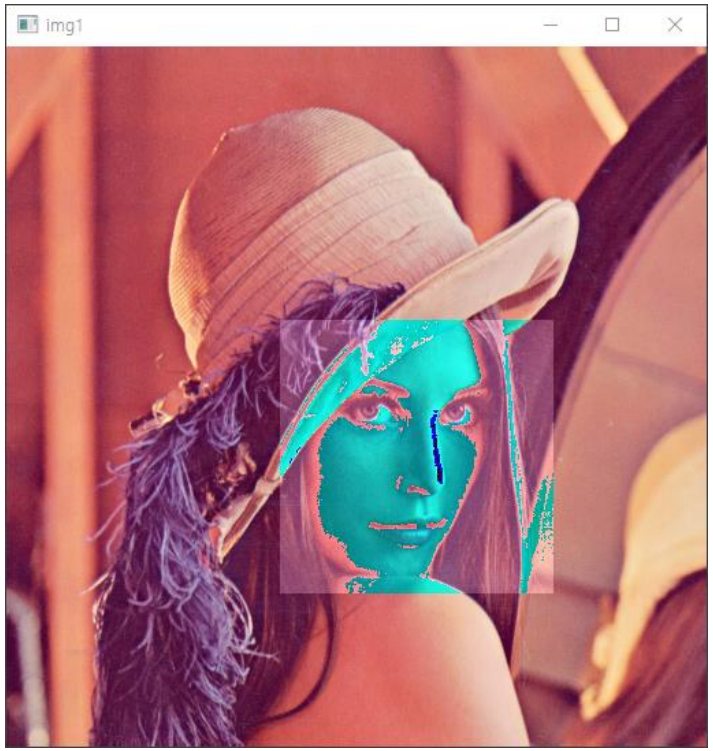
❖ 부분 행렬 추출 (8/9)

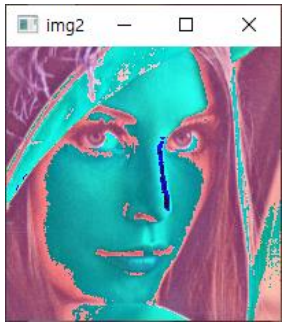
코드 3-9 영상의 부분 영상 처리하기 (matrix.py)

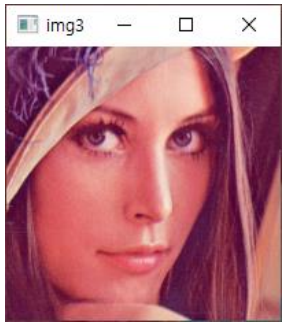
```
1  import cv2
2
3  def partial_extraction():
4      img1 = cv2.imread('lenna.bmp', cv2.IMREAD_COLOR)
5
6      img2 = img1[200:400, 200:400, :]
7      img3 = img1[200:400, 200:400, :].copy()
8
9      img2 += 50
10
11     cv2.imshow('img1', img1)
12     cv2.imshow('img2', img2)
13     cv2.imshow('img3', img3)
14     cv2.waitKey()
15     cv2.destroyAllWindows()
16
17 if __name__ == '__main__':
18     partial_extraction()
```

❖ 부분 행렬 추출 (9/9)

- `img3`은 `copy()` 메서드를 통해서 독립된 복사본으로 만들어졌기 때문에, `img1`과는 서로 다른 메모리 공간을 사용함







THANK YOU!
Q & A

- Name: 권범
- Office: 동양미래대학교 2호관 704호 (02-2610-5238)
- E-mail: bkwon@dongyang.ac.kr
- Homepage: https://sites.google.com/view/beomkwon/home