동양미래대학교

파이썬을 활용한 컴퓨터 비전 입문

Chapter 06. 영상의 산술 및 논리 연산

동양미래대학교
인공지능소프트웨어학과
권 범

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

목차

6.1 영상의 산술 연산

6.2 영상의 논리 연산

❖ 영상의 산술 연산

- 영상은 일종의 2차원 행렬이기 때문에 행렬의 **산술 연산**(**arithmetic operation**)을 그대로 적용할 수 있음
- 두 개의 영상을 서로 더하거나 빼는 연산을 수행함으로써 새로운 결과 영상을 생성 할 수 있음
- 다만 영상을 서로 곱하거나 나누는 연산은 거의 사용하지 않음

❖ 영상의 산술 연산: ① add( ) 함수 (1/5)

- 영상의 덧셈 연산은 두 개의 입력 영상에서 같은 위치 픽셀 값을 서로 더하여 결과 영상 픽셀 값으로 설정하는 연산임
- 영상의 덧셈 연산을 수식으로 표현하면 다음과 같음

$$dst(x, y) = src1(x, y) + src2(x, y)$$

- 위 수식에서 src1과 src2는 입력 영상이고, dst는 덧셈 연산의 결과 영상임
- 영상의 덧셈 연산을 수행하면 그 결괏값이 그레이스케일 최댓값인 255보다 커지는 경우가 발생할 수 있음
- 이러한 경우에는 결과 영상 픽셀 값을 255로 설정하는 **포화 연산**도 함께 수행해야 함
- 영상의 덧셈 연산 수식을 좀 더 정확하게 표현하면 다음과 같음

$$dst(x, y) = saturate(src1(x, y) + src2(x, y))$$

❖ 영상의 산술 연산: ① add() 함수 (2/5)

- OpenCV에서는 add() 함수를 사용하여 영상의 덧셈을 수행할 수 있음

```
dst = cv2.add(src1, src2)
```

src1	첫 번째 입력 행렬
src2	두 번째 입력 행렬
dst	입력 행렬과 같은 크기, 같은 채널 수를 갖는 출력 행렬

- add() 함수는 두 개의 행렬 또는 영상을 입력으로 받고, 하나의 행렬 또는 영상을 출력으로 생성함
- src1과 src2 인자에는 NumPy ndarray 객체 또는 정수, 실수 자료형 등을 전달할 수 있음

❖ 영상의 산술 연산: ① add() 함수 (3/5)

- src1과 src2가 모두 영상처럼 2차원 행렬을 나타내는 NumPy ndarray 객체라면 일반적인 행렬의 덧셈 연산을 수행함
- 만약 src1이 NumPy ndarray 객체이고 src2는 정수, 실수라면 src1 행렬의 모든 픽셀 값에 src2 값을 더하여 결과 영상을 생성함
- 덧셈 결과가 dst 객체가 표현할 수 있는 자료형 범위를 벗어나면 자동으로 포화 연산을 수행함

❖ 영상의 산술 연산: ① add() 함수 (4/5)

- add() 함수를 사용하여 두 개의 영상을 더하는 코드는 다음과 같이 작성할 수 있음

```
dst = cv2.add(src1, src2)
```

- aero2.bmp 파일과 camera.bmp 파일은 모두 256×256 크기의 grayscale 영상임
- 두 입력 영상의 dtype은 모두 uint8임
- 그 결과로 생성되는 dst 영상의 타입은 두 입력 영상의 dtype과 같은 uint8로 설정됨

aero2.bmp

camera.bmp

❖ 영상의 산술 연산: ① add( ) 함수 (5/5)

- 그림 6-1에서 왼쪽 영상은 aero2.bmp 파일이고, 가운데 영상은 camera.bmp 파일임
- 덧셈 연산의 결과 영상은 그림 6-1에서 맨 오른쪽에 나타냄
- 덧셈 연산의 결과 영상은 두 입력 영상의 윤곽을 조금씩 포함하고 있고, 전반적으로 밝게 포화되는 부분이 많다는 특징이 있음

▼ **그림 6-1 영상의 덧셈 연산**

aero2.bmp　　camera.bmp

❖ 영상의 산술 연산: ② addWeighted() 함수 (1/5)

- 두 영상을 더할 때 각 영상에 가중치를 부여하여 덧셈 연산을 할 수도 있음
- 두 개의 행렬에 각각 가중치를 부여하여 덧셈하는 연산을 수식으로 표현하면 다음과 같음

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\alpha \cdot \mathrm{src1}(x, y) + \beta \cdot \mathrm{src2}(x, y))$$

- 위 수식에서 α와 β는 각각 src1과 src2 영상의 가중치를 의미하는 실수임

❖ 영상의 산술 연산: ② addWeighted() 함수 (2/5)

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\alpha \cdot \mathrm{src1}(x, y) + \beta \cdot \mathrm{src2}(x, y))$$

- 보통 $\alpha + \beta = 1$이 되도록 가중치를 설정하는 경우가 많으며, $\alpha + \beta = 1$이면 결과 영상에서 포화되는 픽셀이 발생하지 않음
- 만약 $\alpha = 0.1, \beta = 0.9$로 설정하면 src1 영상의 윤곽은 조금만 나타나고 src2영상의 윤곽은 많이 나타나는 결과 영상이 생성됨
- 만약 $\alpha = \beta = 0.5$로 설정하면 두 입력 영상의 윤곽을 골고루 가지는 평균 영상이 생성됨
- 만약 $\alpha + \beta > 1$이면 결과 영상이 두 입력 영상보다 밝아지게 되고, 덧셈의 결과가 255보다 커지는 포화 현상이 발생할 수 있음
- 만약 $\alpha + \beta < 1$이면 dst 영상은 두 입력 영상의 평균 밝기보다 어두운 결과 영상이 생성됨

❖ 영상의 산술 연산: ② addWeighted() 함수 (3/5)

- OpenCV에서 두 영상의 가중치 합을 구하려면 addWeighted() 함수를 사용함

```
dst = cv2.addWeighted(src1, alpha, src2, beta, gamma)
```

src1	첫 번째 입력 행렬
alpha	src1 행렬의 가중치
src2	두 번째 입력 행렬. src1과 크기와 채널 수가 같아야 합니다.
beta	src2 행렬의 가중치
gamma	가중합 결과에 추가적으로 더할 값
dst	출력 행렬. 입력 행렬과 같은 크기, 같은 채널 수의 행렬이 생성됩니다.

❖ 영상의 산술 연산: ② addWeighted() 함수 (4/5)

- addWeighted() 함수는 gamma 인자를 통해 가중치의 합에 추가적인 덧셈을 한꺼번에 수행할 수 있음
- addWeighted() 함수에 의해 생성되는 dst는 다음과 같이 나타낼 수 있음

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\mathrm{src1}(x, y) * \mathrm{alpha} + \mathrm{src2}(x, y) * \mathrm{beta} + \mathrm{gamma})$$

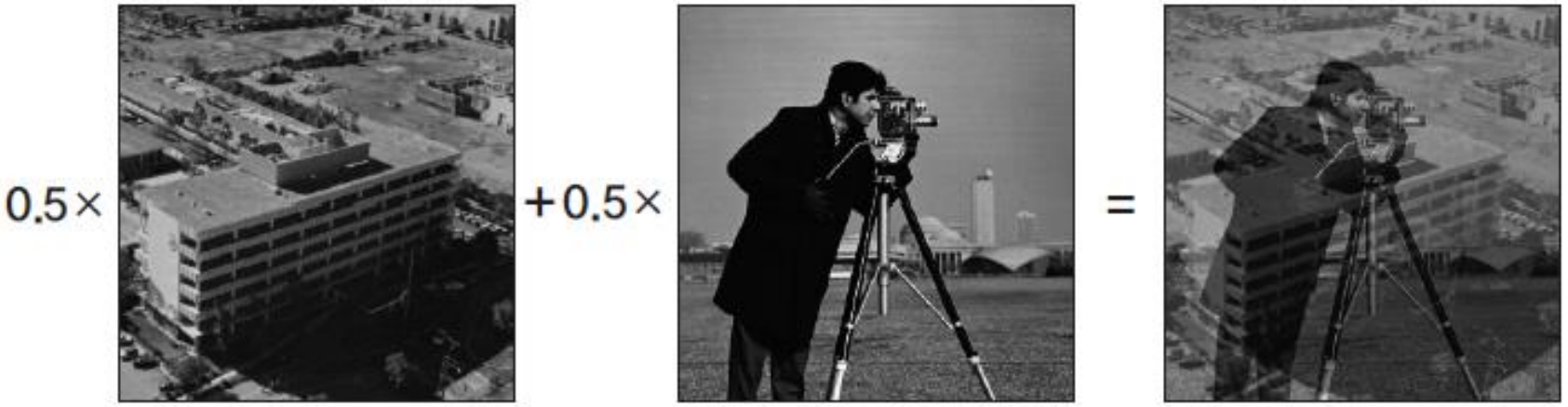
❖ 영상의 산술 연산: ② addWeighted() 함수 (5/5)

- addWeighted() 함수를 이용하여 두 입력 영상의 평균 영상을 생성하려면 다음과 같이 코드를 작성함

```
dst = cv2.addWeighted(src1, 0.5, src2, 0.5, 0)
```

- 평균 연산에 의한 결과 영상이 두 입력 영상의 윤곽을 골고루 포함하고 있고 평균 밝기가 그대로 유지되는 것을 확인할 수 있음

▼ 그림 6-2 영상의 평균 연산



❖ 영상의 산술 연산: ③ subtract() 함수 (1/3)

- 덧셈 연산과 마찬가지로 두 개의 영상에 대하여 뺄셈 연산도 수행할 수 있음
- 두 영상의 뺄셈 연산을 수식으로 표현하면 다음과 같음

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\mathrm{src1}(x, y) - \mathrm{src2}(x, y))$$

- 뺄셈 연산은 두 영상에서 같은 위치에 있는 픽셀끼리 빼기 연산을 수행하는 것임
- 뺄셈의 결과가 0보다 작아지면 결과 영상의 픽셀 값을 0으로 설정하는 **포화 연산**을 수행해야 함

❖ 영상의 산술 연산: ③ subtract() 함수 (2/3)

- OpenCV에서는 subtract() 함수를 통해 두 영상의 뺄셈 연산을 수행할 수 있음
- subtract() 함수의 인자 구성과 설명은 add() 함수와 동일함

```
dst = cv2.subtract(src1, src2)
```

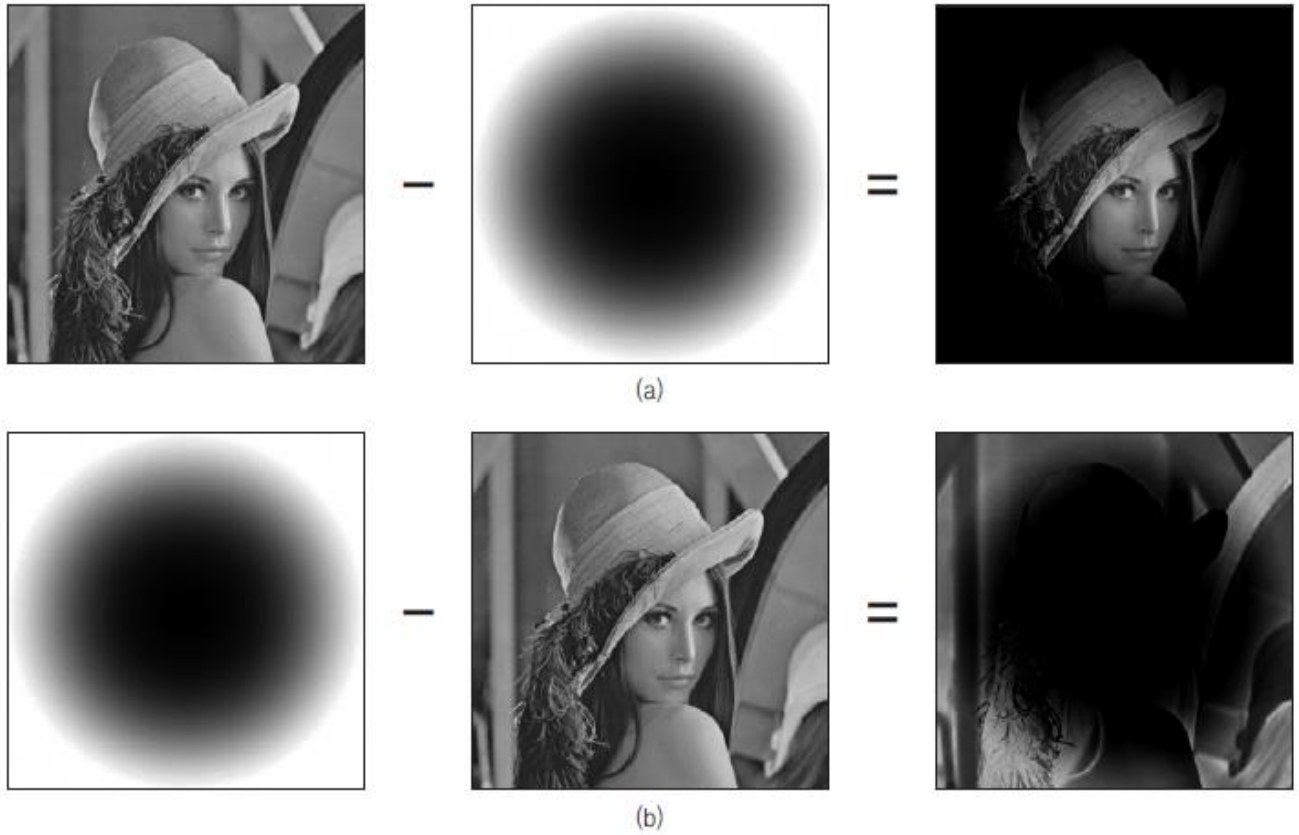
src1	첫 번째 입력 행렬
src2	두 번째 입력 행렬
dst	입력 행렬과 같은 크기, 같은 채널 수를 갖는 출력 행렬

- 덧셈 연산과 달리 뺄셈 연산은 **뺄셈의 대상이 되는 영상 순서에 따라 결과가 달라짐**

❖ 영상의 산술 연산: ③ subtract() 함수 (3/3)

- 영상 순서에 따라 결과 영상이 달라지는 것을 확인할 수 있음

▼ 그림 6-3 영상의 뺄셈 연산



❖ 영상의 산술 연산: ④ absdiff() 함수 (1/3)

- 만약 두 영상의 뺄셈 순서에 상관없이 픽셀 값 차이가 큰 영역을 두드러지게 나타내고 싶다면 차이 연산을 수행할 수 있음
- 차이 연산은 **뺄셈 연산 결과에 절댓값을 취하는 연산임**
- 차이 연산으로 구한 결과 영상을 **차영상**(**difference image**)이라고 함
- 차이 연산을 수식으로 표현하면 다음과 같음

$$\mathrm{dst}(x, y) = |\mathrm{src1}(x, y) - \mathrm{src2}(x, y)|$$

❖ 영상의 산술 연산: ④ absdiff() 함수 (2/3)

- OpenCV에서는 absdiff() 함수를 이용하여 차영상을 구할 수 있음

```
dst = cv2.absdiff(src1, src2)
```

src1	첫 번째 입력 행렬
src2	두 번째 입력 행렬
dst	입력 행렬과 같은 크기, 같은 채널 수를 갖는 출력 행렬

❖ 영상의 산술 연산: ④ absdiff() 함수 (3/3)

- 차이 연산을 이용하면 두 개의 영상에서 **변화가 있는 영역을 쉽게 찾을 수 있음**
- 그림 6-4의 왼쪽 영상은 움직임이 없는 정적인 배경 영상이고, 가운데 영상은 한 대의 자동차가 지나가고 있을 때 촬영된 영상임
- 이 두 영상에 대해 차이 연산을 수행하면 그림 6-4의 오른쪽에 나타난 영상처럼 움직이는 자동차 영역에서만 픽셀 값 차이가 두드러지게 나타남
- 두 입력 영상에서 큰 변화가 없는 영역은 픽셀 값이 0에 가까운 검은색으로 채워지게 됨

▼ **그림 6-4 영상의 차이 연산**

❖ 영상의 산술 연산: ⑤ multply() 함수 & divide() 함수 (1/2)

- 영상도 일종의 행렬이므로 두 입력 영상을 행렬로 생각하여 행렬의 곱셈을 수행할 수도 있음
- 영상을 이용하여 행렬의 곱셈을 수행하는 경우는 거의 없음
- 두 영상에서 같은 위치에 있는 픽셀 값끼리 서로 곱하거나 나누는 연산을 수행할 수 있음
- OpenCV에서는 `multiply()` 함수와 `divide()` 함수를 제공함

```
dst = cv2.multiply(src1, src2, scale=1)
```

src1	첫 번째 입력 행렬
src2	두 번째 입력 행렬. src1과 크기와 타입이 같아야 합니다.
dst	출력 행렬. src1과 같은 크기, 같은 타입 dst(x, y) = saturate(scale·src1(x, y)·src2(x, y))
scale	추가적으로 확대/축소할 비율

❖ 영상의 산술 연산: ⑤ multply() 함수 & divide() 함수 (2/2)

```
dst = cv2.divide(src1, src2, scale=1)
```

src1	첫 번째 입력 행렬
src2	두 번째 입력 행렬. src1과 크기와 타입이 같아야 합니다.
dst	출력 행렬. src1과 같은 크기, 같은 타입 dst(x, y) = saturate(scale·src1(x, y)/src2(x, y))
scale	추가적으로 확대/축소할 비율

❖ 영상의 산술 연산 예제 프로그램 (1/4)

- 코드 6-1은 lenna.bmp 파일과 square.bmp 파일을 이용하여 덧셈, 뺄셈, 평균, 차이 영상을 생성하고 그 결과를 화면에 출력함

코드 6-1 영상의 산술 연산 예제 프로그램 (arithmetic.py)

```
1   import sys
2   import cv2
3   from matplotlib import pyplot as plt
4
5   src1 = cv2.imread('lenna256.bmp', cv2.IMREAD_GRAYSCALE)
6   src2 = cv2.imread('square.bmp', cv2.IMREAD_GRAYSCALE)
7
8   if src1 is None or src2 is None:
9       print('Image load failed!')
10      sys.exit()
11
```

❖ 영상의 산술 연산 예제 프로그램 (2/4)

코드 6-1 영상의 산술 연산 예제 프로그램 (arithmetic.py)

```
12  dst1 = cv2.add(src1, src2)
13  dst2 = cv2.addWeighted(src1, 0.5, src2, 0.5, 0.0)
14  dst3 = cv2.subtract(src1, src2)
15  dst4 = cv2.absdiff(src1, src2)
16
17  plt.subplot(231), plt.axis('off'), plt.imshow(src1, 'gray'), plt.title('src1')
18  plt.subplot(232), plt.axis('off'), plt.imshow(src2, 'gray'), plt.title('src2')
19  plt.subplot(233), plt.axis('off'), plt.imshow(dst1, 'gray'), plt.title('add')
20  plt.subplot(234), plt.axis('off'), plt.imshow(dst2, 'gray'), plt.title('addWeighted')
21  plt.subplot(235), plt.axis('off'), plt.imshow(dst3, 'gray'), plt.title('subtract')
22  plt.subplot(236), plt.axis('off'), plt.imshow(dst4, 'gray'), plt.title('absdiff')
23  plt.show()
```

❖ 영상의 산술 연산 예제 프로그램 (3/4)

- arithmetic.py 소스 코드 설명

  - 5~6행 영상의 산술 입력으로 사용할 영상을 grayscale 형식으로 불러와서 `src1`과 `src2` 변수에 저장합니다.

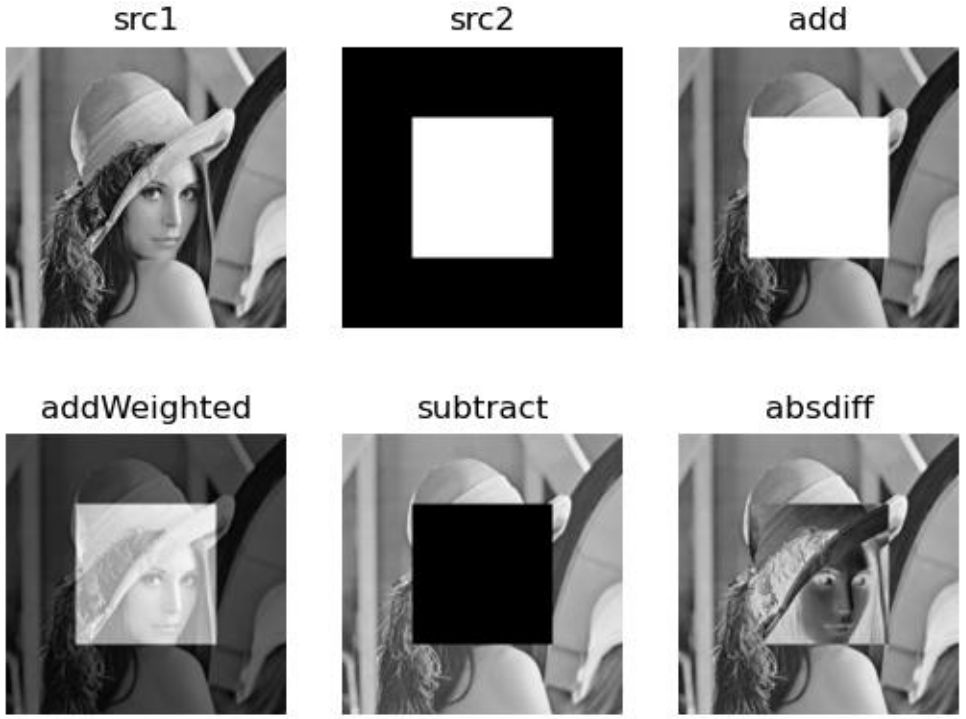
  - 12~15행 `src1`과 `src2` 영상을 이용하여 덧셈, 평균, 뺄셈, 차이 연산을 수행하고, 그 결과 영상을 각각 `dst1`, `dst2`, `dst3`, `dst4` 변수에 저장합니다.

  - 17~23행 덧셈, 평균, 뺄셈, 차이 연산의 결과 영상을 하나의 창으로 나타냅니다.

❖ 영상의 산술 연산 예제 프로그램 (4/4)

- square.bmp 파일은 0 또는 255의 픽셀 값으로 구성된 간단한 영상임
- 각 산술 연산 결과가 머릿속으로 가늠한 것과 동일하게 나타나는지를 꼭 확인 해 보기 바람

▼ **그림 6-5 영상의 산술 연산 예제 프로그램 실행 화면**



6.2 영상의 논리 연산

6.1 영상의 산술 연산

❖ 영상의 논리 연산 (1/6)

- 영상의 **논리 연산**(**logical operation**)은 픽셀 값을 이진수로 표현하여 각 비트(bit) 단위 논리 연산을 수행하는 것을 의미함
- OpenCV에서는 다양한 논리 연산 중에서 논리곱(AND), 논리합(OR),배타적 논리합(XOR), 부정(NOT) 연산을 지원함

❖ 영상의 논리 연산 (2/6)

- 비트 단위 **논리곱**은 두 개의 입력 비트가 모두 1인 경우에 결과가 1이 되는 연산임
- 비트 단위 **논리합**은 두 개의 입력 비트 중 하나라도 1이 있으면 결과가 1이 됨
- 비트 단위 **배타적 논리합**은 두 개의 입력 비트 중 오직 하나만 1인 경우에 결과가 1이 되고, 입력 비트가 모두 0이거나 모두 1이면 결과가 0이 됨
- 비트 단위 **부정**은 하나의 입력 영상에 대해 동작하며 입력 비트가 0이면 결과가 1이 되고 입력 비트가 1이면 결과가 0이 됨

❖ 영상의 논리 연산 (3/6)

- OpenCV에서 제공하는 논리 연산의 종류와 동작 방식을 하나의 진리표로 정리하여 표 6-1에 나타냄

▼ **표 6-1 OpenCV에서 제공하는 논리 연산 진리표**

입력 비트		논리 연산 결과			
a	b	a AND b	a OR b	a XOR b	NOT a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

❖ 영상의 논리 연산 (4/6)

- 영상의 논리 연산은 각 픽셀 값에 대하여 비트 단위로 이루어짐
- grayscale 영상의 경우, 한 픽셀을 구성하는 여덟 개의 비트에 모두 논리 연산이 이루어짐
- 예를 들어 두 개의 입력 영상에서 특정 좌표에 있는 픽셀의 grayscale 값이 각각 110과 200인 경우, 이 두 값에 대하여 논리곱(AND), 논리합(OR), 배타적 논리합(XOR), 부정(NOT) 연산을 수행하면 다음과 같이 계산됨

$$110 = 01101110_{(2)}$$
$$200 = 11001000_{(2)}$$

$$110 \text{ AND } 200 = 01001000_{(2)} = 72$$
$$110 \text{ OR } 200 = 11101110_{(2)} = 238$$
$$110 \text{ XOR } 200 = 10100110_{(2)} = 166$$
$$\text{NOT } 110 = 10010001_{(2)} = 145$$

❖ 영상의 논리 연산 (5/6)

- bitwise_and() 함수는 비트 단위 논리곱, bitwise_or() 함수는 비트 단위 논리합, bitwise_xor() 함수는 비트 단위 배타적 논리합, bitwise_not() 함수는 비트 단위 부정 연산을 수행함

```
dst = cv2.bitwise_and(src1, src2)
dst = cv2.bitwise_or(src1, src2)
dst = cv2.bitwise_xor(src1, src2)
dst = cv2.bitwise_not(src1)
```

src1	첫 번째 입력 행렬
src2	두 번째 입력 행렬. src1과 크기와 타입이 같아야 합니다.
dst	출력 행렬. src1과 같은 크기, 같은 타입으로 생성됩니다. dst 행렬 원소 값은 논리 연산 종류에 의해 각각 다르게 결정됩니다.

❖ 영상의 논리 연산 (6/6)

- 앞서 나열된 네 개의 비트 단위 논리 연산 함수들 중에서 `bitwise_and()`, `bitwise_or()`, `bitwise_xor()` 함수는 두 개의 영상을 입력으로 받음
- `bitwise_not()` 함수는 하나의 영상을 입력으로 받음

❖ 영상의 논리 연산 예제 프로그램 (1/6)

- `bitwise_and()`, `bitwise_or()`, `bitwise_xor()`, `bitwise_not()` 함수를 이용하여 영상의 논리 연산을 수행하는 예제 프로젝트 소스 코드를 코드 6-2에 나타냄
- 코드 6-2는 lenna.bmp 파일과 square.bmp 파일을 이용하여 논리곱, 논리합, 배타적 논리합 연산을 수행함
- lenna.bmp 영상에 대해서 부정 연산을 수행한 후 그 결과를 화면에 출력함

❖ 영상의 논리 연산 예제 프로그램 (2/6)

코드 6-2 영상의 논리 연산 예제 프로그램 (logical.py)

```
1  import sys
2  import cv2
3  from matplotlib import pyplot as plt
4
5  src1 = cv2.imread('lenna256.bmp', cv2.IMREAD_GRAYSCALE)
6  src2 = cv2.imread('square.bmp', cv2.IMREAD_GRAYSCALE)
7
8  if src1 is None or src2 is None:
9      print('Image load failed!')
10     sys.exit()
11
```

❖ 영상의 논리 연산 예제 프로그램 (3/6)

코드 6-2 영상의 논리 연산 예제 프로그램 (logical.py)

```
12  dst1 = cv2.bitwise_and(src1, src2)
13  dst2 = cv2.bitwise_or(src1, src2)
14  dst3 = cv2.bitwise_xor(src1, src2)
15  dst4 = cv2.bitwise_not(src1)
16
17  plt.subplot(231), plt.axis('off'), plt.imshow(src1, 'gray'), plt.title('src1')
18  plt.subplot(232), plt.axis('off'), plt.imshow(src2, 'gray'), plt.title('src2')
19  plt.subplot(233), plt.axis('off'), plt.imshow(dst1, 'gray'), plt.title('bitwise_and')
20  plt.subplot(234), plt.axis('off'), plt.imshow(dst2, 'gray'), plt.title('bitwise_or')
21  plt.subplot(235), plt.axis('off'), plt.imshow(dst3, 'gray'), plt.title('bitwise_xor')
22  plt.subplot(236), plt.axis('off'), plt.imshow(dst4, 'gray'), plt.title('bitwise_not')
23  plt.show()
```

❖ 영상의 논리 연산 예제 프로그램 (4/6)

- logical.py 소스 코드 설명

  - 5~6행 영상의 비트 단위 논리 연산 입력으로 사용할 영상을 grayscale 형식으로 불러와서 `src1`과 `src2` 변수에 저장합니다.

  - 12~15행 `src1`과 `src2` 영상을 이용하여 논리곱, 논리합, 배타적 논리합, (`src1` 영상의) 부정을 구하고, 그 결과 영상을 각각 `dst1`, `dst2`, `dst3`, `dst4` 변수에 저장합니다.

  - 17~23행 논리곱, 논리합, 배타적 논리합, (`src1` 영상의) 부정 연산의 결과 영상을 하나의 창으로 나타냅니다.

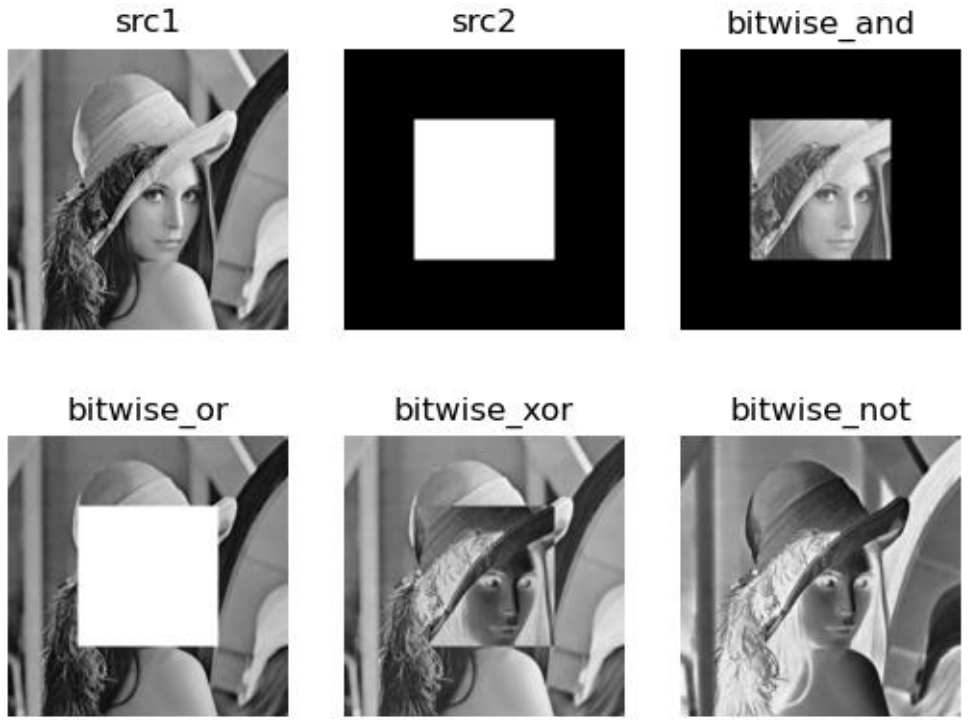
❖ 영상의 논리 연산 예제 프로그램 (5/6)

- 그림 6-6에서 src1창에 나타난 영상은 256×256 크기의 lenna256.bmp 파일이고, src2 창 영상은 같은 크기의 square.bmp 파일임
- square.bmp 영상에서 가운데 사각형 영역의 픽셀 값은 255이고, 이를 이진수로 표현하면 모든 비트가 1로 설정된 11111111(2)임
- square.bmp 영상에서 사각형 바깥 영역의 픽셀 값은 0이고, 이를 이진수로 표현하면 00000000(2)임
- 이 두 개의 영상에 대한 비트 단위 논리곱, 비트 단위 논리합, 비트 단위 배타적 논리합, 비트 단위 부정 연산을 수행한 결과 영상을 각각 dst1, dst2, dst3, dst4 창에 표시함

❖ 영상의 논리 연산 예제 프로그램 (6/6)

- square.bmp 영상의 픽셀 값 비트 구성에 따라 각각의 논리 연산 결과가 부합되게 나타나는지 확인해 보자

▼ 그림 6-6 영상의 논리 연산 예제 프로그램 실행 화면



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