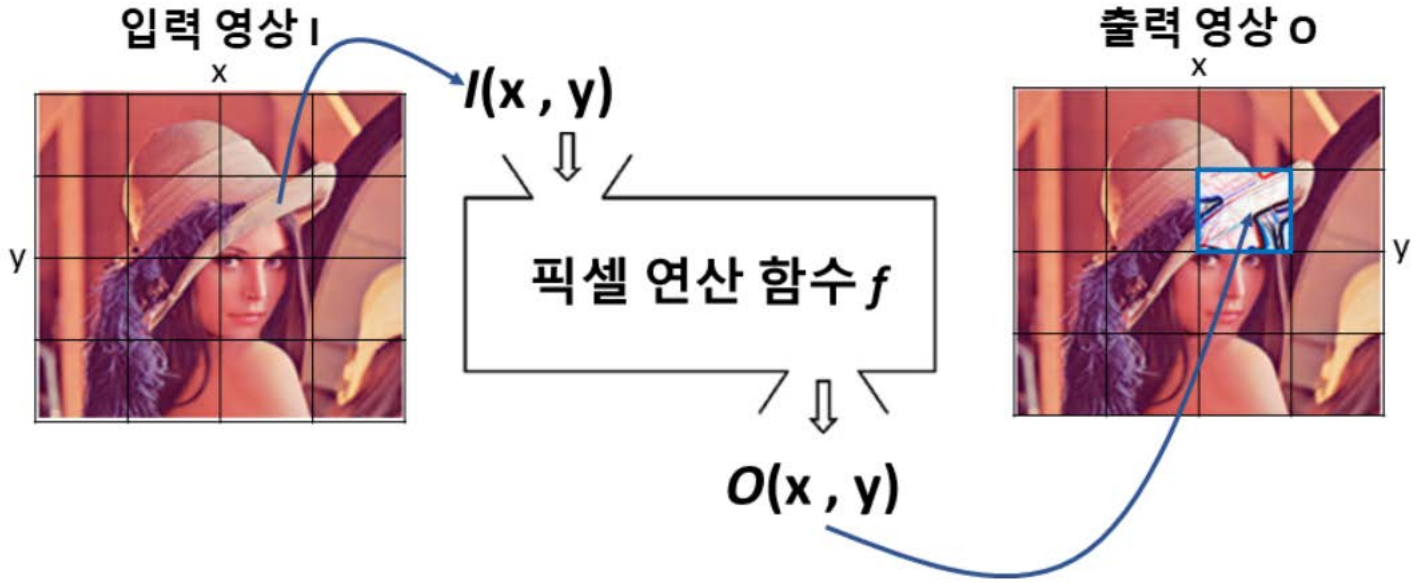
# 제3장 영상의 산술 및 논리 연산

한밭대학교 전자공학과
곽수영

# 픽셀 연산(pixel point processing)

- 입력 영상의 한 픽셀 I(x, y)를 매핑 함수 f에 적용하여 출력 영상의 한 픽셀 O(x, y)를 얻을 수 있음

$$O(x, y) = f[I(x, y)]$$




(그림 3-1) 픽셀 연산 과정

# 픽셀 연산(pixel point processing)

- 화소의 밝기값과 명암 대비
  - 화소의 밝기값
    - 밝기의 단계 수는 화소를 표현하는 양자화 비트 수가 결정
    - 그레이 레벨 영상에서는 색은 없고 밝기만 있음.
    - 보통, 화소는 밝기를 나타내는데, 주로 양자화 비트 수를 8비트로 표현
  - 화소의 명암대비
    - 대비(Contrast): 영상 내에 있는 가장 밝은 값과 가장 어두운 값의 차이로, 영상의 품질을 결정하는 중요한 요소임.
    - 높은 대비를 보이는 디지털 영상: 어두운 명도와 밝은 명도의 차이가 너무 커서 시각적으로 좀더 명확하게 보임.
    - 낮은 대비를 보이는 디지털 영상: 밝기의 차이가 크지 않아 시각적으로 명확하지 못함.

# 3.1 픽셀 더하기

# 덧셈/뺄셈 연산

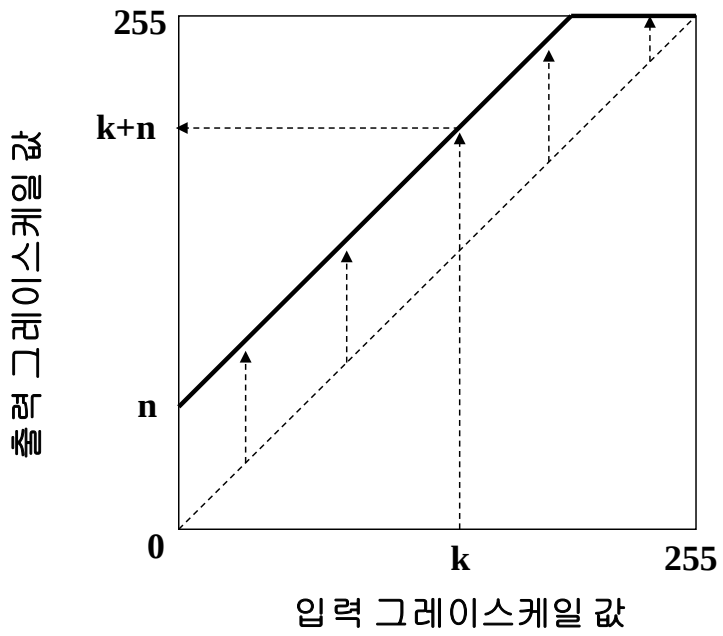
- 화소값의 덧셈연산 (C: 상수값)
  - 화소의 밝기 값에 특정한 상수 값을 더해 화소의 밝기 값을 증가시켜 영상을 밝게 처리함

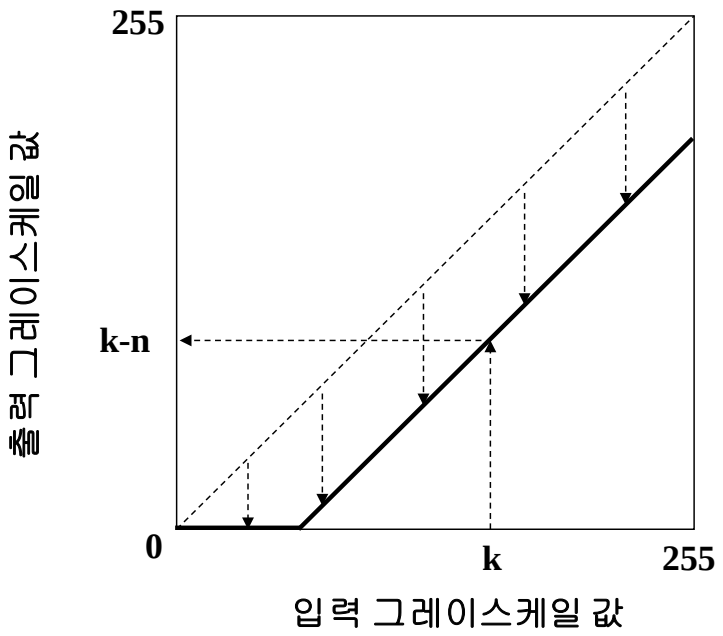
$$O(x, y) = P(x, y) + C$$

- 화소값의 뺄셈연산 (C: 상수값)
  - 화소의 밝기 값에 특정한 상수 값을 빼 화소의 밝기 값을 감소시켜 영상의 밝기를 어둡게 처리함

$$O(x, y) = P(x, y) - C$$

# 덧셈/뺄셈 연산


255
k+n
n
0
k
255
출력 그레이스케일 값
입력 그레이스케일 값


255
k-n
0
k
255
출력 그레이스케일 값
입력 그레이스케일 값

# 덧셈/뺄셈 연산

원본 영상

밝기 64 감소

밝기 64 증가

# 덧셈/뺄셈 연산

- 영상의 밝기 조절 시 주의할 점
  - 픽셀 값을 증가시켜 밝게 만들 경우, 그 결과 값이 255보다 커지지 않도록 주의해야 함.
  - 밝기 값을 감소시킬 때에는?

결과 픽셀 값이 255보다 커지는 경우, unsigned char 자료형이 표현할 수 있는 값의 범위를 벗어나서 원치 않는 결과가 발생함

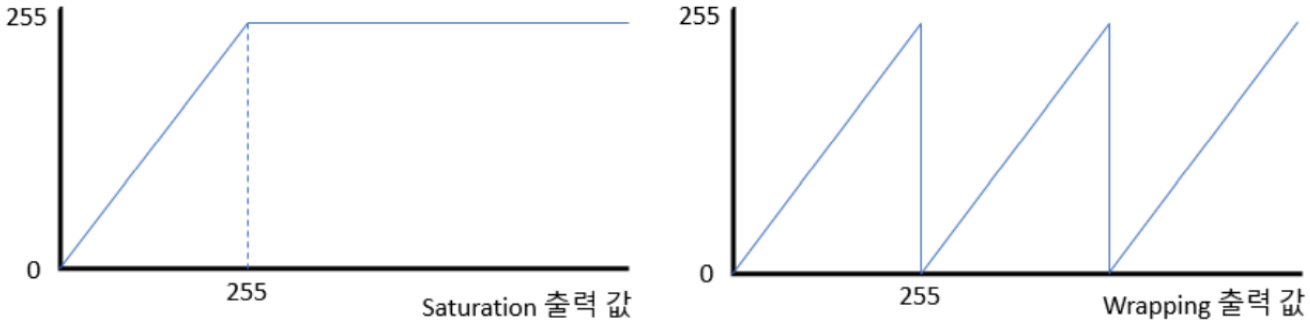
결과 영상에서 255보다 큰 그레이스케일 값을 255로 설정해준 경우

# 덧셈/뺄셈 연산

- 더하기 연산 시 연산값이 픽셀의 최대값인 255를 초과하는 오버플로우(overflow)가 발생할 수 있음
  - Saturation
    - 픽셀값이 설정된 최대값을 초과할 경우 초과된 값을 설정된 최대값으로 교체
    - 예) 255가 넘는 값은 모두 255를 갖는다

$$\text{saturate}(x) = \begin{cases} 0 & x < 0 \text{ 일 때} \\ 255 & x > 255 \text{ 일 때} \\ x & \text{그 외} \end{cases}$$

  - Wrapping
    - 픽셀값이 설정된 최대값을 초과할 경우 초과된 값에서 최대값+1을 뺀 값으로 픽셀값을 교체
    - 예) 최소값이 0이고 최대값이 255라면 256값은 0으로, 257값은 1을 갖는다




(그림 3-2) Saturation(왼쪽)과 Wrapping(오른쪽)

# 덧셈/뺄셈 연산

- 최대-최소 정규화 방법
  - 픽셀 값(I(x,y))을 미리 설정된 최대값(max)과 최소값(min) 사이 (예를 들어 0과 255사이) 값으로 재계산하는 방법

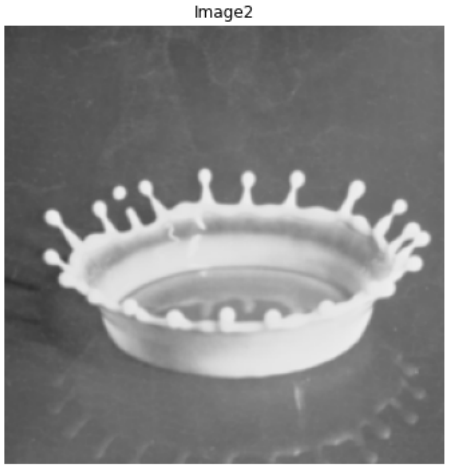
$$O(x,y) = \frac{I(x,y) - \min}{\max - \min} \times \max$$

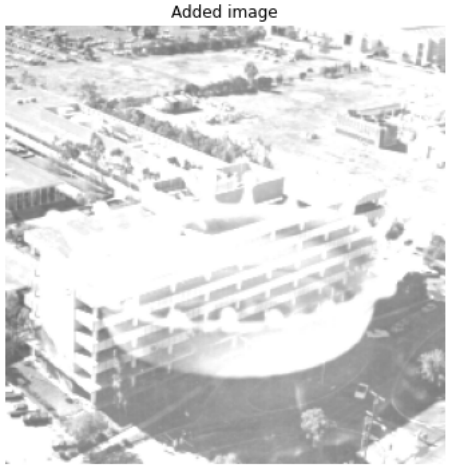
# 덧셈/뺄셈 연산

- 두 영상 사이의 덧셈 연산
  - 동일한 크기의 두 영상으로부터 동일한 좌표의 각 픽셀들을 더하여 결과 영상에 입력하는 연산

$$O(x,y) = P(x,y) + Q(x,y)$$

  - 덧셈 결과가 255보다 크면 픽셀값을 255로 설정

# 연습 3-2) 픽셀 더하기 OpenCV 함수 이용하기

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/gdrive')
#영상 불러오기
img1 = cv2.imread('/content/gdrive/My Drive/Image_Processing/aero2.bmp')
img2 = cv2.imread('/content/gdrive/My Drive/Image_Processing/milkdrop.bmp')
img_plus = img1+img2 #일반적인 더하기 연산
img_add=cv2.add(img1,img2) #OpenCV의 add()함수 이용하기
#그림을 화면에 출력
plt.figure(figsize=(20,20))# 영상의 크기를 키워주자
plt.subplot(1,2,1)# 1행 4열에서 1번째 열
plt.title("Plus image")
plt.imshow(img_plus)
plt.axis("off")
plt.subplot(1,2,2)# 1행 4열에서 1번째 열
plt.title("Saturation")
plt.imshow(img_add)
plt.axis("off")
plt.show()
```

# 연습 3-2) 픽셀 더하기 OpenCV 함수 이용하기

**코드설명**

**img_plus = img1+img2:** 두 영상의 픽셀끼리 더하기 연산을 한다. 255를 넘는 값은 따로 처리하지 않기 때문에 결과 왼쪽 결과 그림과 같이 255가 넘을 경우 잘못된 값이 들어간다.

**img_add=cv2.add(img1,img2):** 더하기 연산과 동시에 saturation기능을 같이 한다. 오른쪽 결과그림과 같이 255가 넘는 부분은 255 픽셀로 변경되었다.


# 3.2 픽셀 빼기

# 뺄셈 연산

- 두 영상 사이의 뺄셈 연산
  - 동일한 크기의 두 영상으로부터 동일한 좌표의 각 픽셀들을 뺀 값을 결과 영상에 입력하는 연산

$$O(x,y) = P(x,y) - Q(x,y)$$

  - 뺄셈 결과가 0보다 작으면 픽셀값은 0으로 설정
  - 결과값이 음수가 될 수 있으므로 뺄셈 연산에 절대값을 취하여 입력할 수도 있음

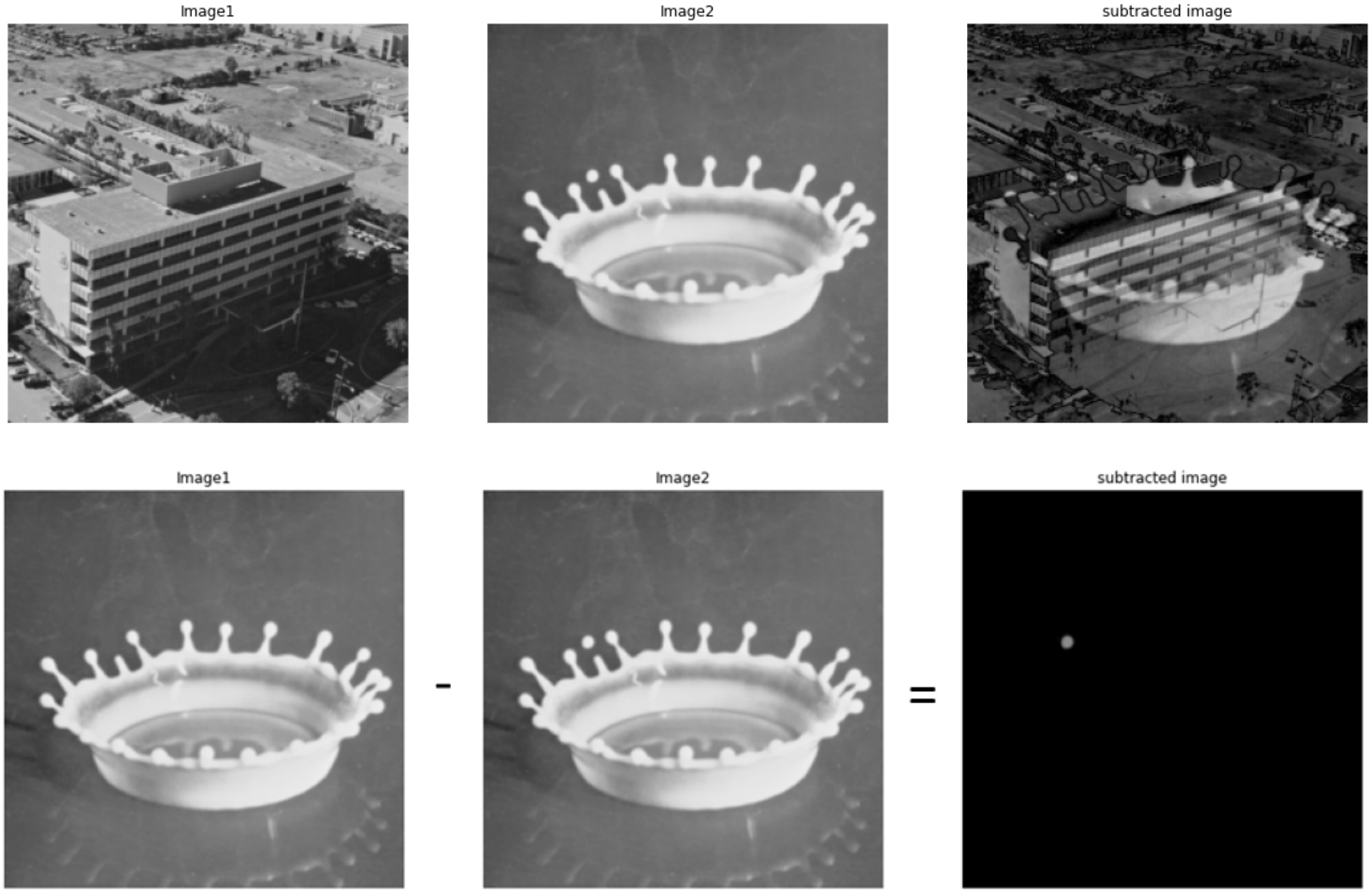
$$O(x,y) = \left| P(x,y) - Q(x,y) \right|$$

# 연습 3-3) 픽셀 빼기 연산하기

```
import cv2
from google.colab.patches import cv2_imshow
path1='/content/drive/MyDrive/Image_Processing/aero2.bmp'
path2='/content/drive/MyDrive/Image_Processing/milkdrop.bmp'

img1=cv2.imread(path1)
img2=cv2.imread(path2)
img_sub=cv2.subtract(img1,img2)
img_sub2=cv2.absdiff(img1,img2)
cv2_imshow(img_sub)
cv2_imshow(img_sub2)
```

# 뺄셈 연산

- 두 영상의 뺄셈 연산 결과



# 3.3 픽셀 곱하기

# 곱셈 연산

- 상수값에 의한 곱셈연산
  - 화소의 밝기 값에 특정 상수 값을 곱해 전체적으로 화소의 밝기 값이 증가해 더 밝아짐.

$$O(x, y) = P(x, y) \times C \quad (C >= 0)$$

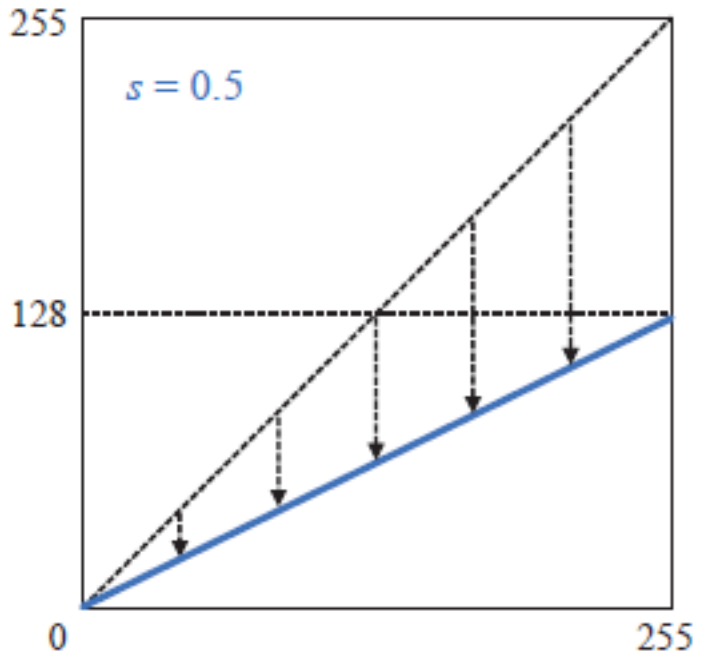
  - 밝은 부분은 더욱 밝아지고, 어두운 부분은 약간 밝아져 영상 내의 밝기에 커다란 차이가 생기는 것
  - 밝기의 차이가 커지므로 영상의 선명도 증가함.
  - 컬러 영상의 경우 목적에 따라 R, G, B 각 채널에 서로 다른 상수값을 곱하여 출력 영상의 색상을 조절함
  - 연산 시 오버플로우가 발생할 수 있는데 이런 경우 Saturation, Wrapping, 최대-최소 정규화 방법을 사용

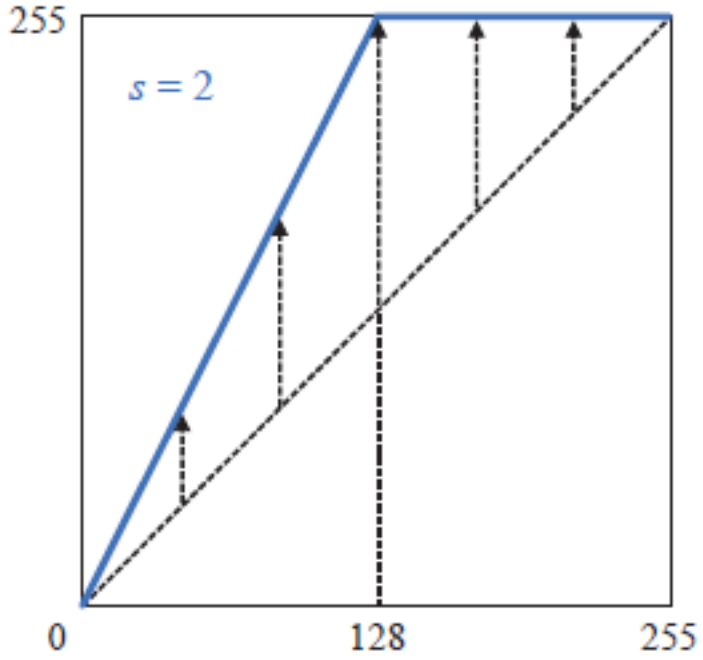
# 명암비 조절 방법

- 명암비 = 명암대비
  - 그레이 스케일 값의 밝은 부분과 어두운 부분의 차이
  - 영상이 전체적으로 밝거나 어둡기만 하면 우리는 그 영상을 명암비가 작다고 표현하며 이와 반대로 영상의 특정 부분은 밝고 다른 부분은 어두워서 두 부분의 차이가 잘 느껴지는 영상은 명암비가 높다고 한다.
  - 명암비가 높을수록 눈으로 보기에 훨씬 더 선명한 느낌

# 명암비 조절 방법

- 기본적인 명암비 조절방법
  - s = 0.5인 경우와 s = 2인 경우의 그래프



# 곱셉 연산

- 동일한 크기의 두 영상으로부터 동일한 좌표의 각 픽셀 곱을 결과 영상에 입력하는 연산

$$O(x, y) = P(x, y) \times Q(x, y)$$

# 3.4 픽셀 나누기

# 나눗셈 연산

- 상수값에 의한 나눗셈 연산
- 동일한 크기의 두 영상으로부터 동일한 좌표의 각 픽셀들을 나눈 값을 결과 영상에 입력하는 연산

$$O(x, y) = P(x, y) \div Q(x, y)$$

$$O(x, y) = P(x, y) \div C$$

- 연산된 값이 부동소수점 값이 될 수 있는데, 출력 영상에서는 정수로 입력되어야 하므로 소수점 이하를 버림 혹은 반올림을 취함


input
output
+100
input
output
-100


input
output
*1.3
input
output
*0.5

# 3.5 픽셀 결합

# 픽셀 결합

- 두 개의 동일한 크기의 입력 영상을 결합하는 방법으로, 대응되는 두 영상의 픽셀들을 선형 결합(linear combination) 방식에 의해 결합하여 출력 영상을 생성하는 연산

$$O(x,y) = W \times P(x,y) + (1-W) \times Q(x,y)$$

  - 가중치에 해당하는 W는 $0 \le W \le 1$의 범위의 실수값을 가지며 두 영상을 결합하기 전에 사용자에 의해 조절될 수 있음
  - 가중치의 크기에 따라 입력영상이 결과영상에 미치는 중요도를 조절할 수 있음

# 픽셀 결합

- 두영상 가중치의 덧셈: 합성한 효과

$$O(x,y) = \frac{1}{2}P(x,y) + \frac{1}{2}Q(x,y)$$

  - 덧셈 연산: 결과 영상이 전체적으로 밝아짐
  - 평균 연산: 입력 영상의 밝기 정도를 그대로 유지

# 픽셀 결합

```
import cv2
from google.colab.patches import cv2_imshow
path1='/content/drive/MyDrive/Image_Processing/aero2.bmp'
path2='/content/drive/MyDrive/Image_Processing/milkdrop.bmp'

img1=cv2.imread(path1)
img2=cv2.imread(path2)
result=cv2.addWeighted(img1,0.5,img2,0.5,0)
cv2_imshow(result)
```

cv2.addWeighted(img1,0.5,img2,0.5,0)

cv2.addWeighted(img1,0.2,img2,0.8,0)

# 3.6 픽셀 비트 연산

# 디지털 영상의 논리연산

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

$$
\begin{array}{rl}
 & 110 = 01101110_{(2)} \\
 & 200 = 11001000_{(2)} \\
\hline
110 \text{ AND } 200 = 01001000_{(2)} & = 72 \\
110 \text{ OR } 200 = 11101110_{(2)} & = 238 \\
110 \text{ XOR } 200 = 10100110_{(2)} & = 166 \\
\text{NOT } 110 = 10010001_{(2)} & = 145
\end{array}
$$

# 픽셀 비트 연산

- 화소 값의 AND/OR 연산
  - 영상의 픽셀 값을 2진수로 표현한 후, 각 비트마다 AND/ OR 연산을 수행함
  - AND 연산: 원하는 비트를 선택적으로 0(검정)으로 만드는 기능이 있어 마스크(mask) 연산이라고도 함
  - OR연산: 특정 비트를 선택적으로 1(흰색)로 구성할 수 있어 선택적-세트(selective-set)연산이라고도 함

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

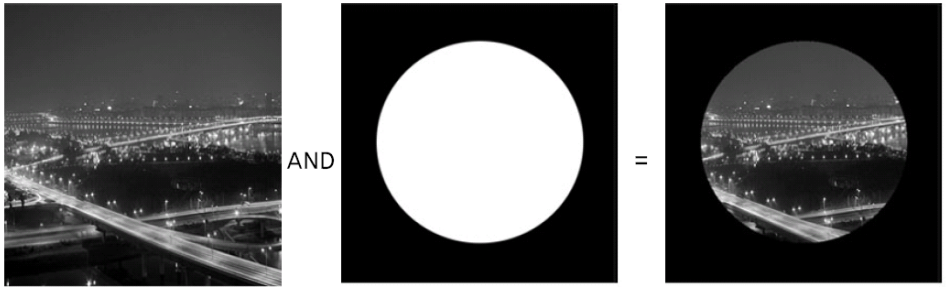
AND 연산 진리 테이블

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

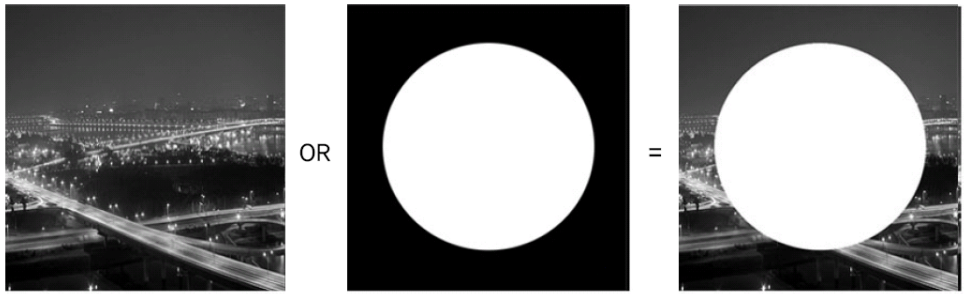
OR연산 진리 테이블

# 픽셀 비트 연산

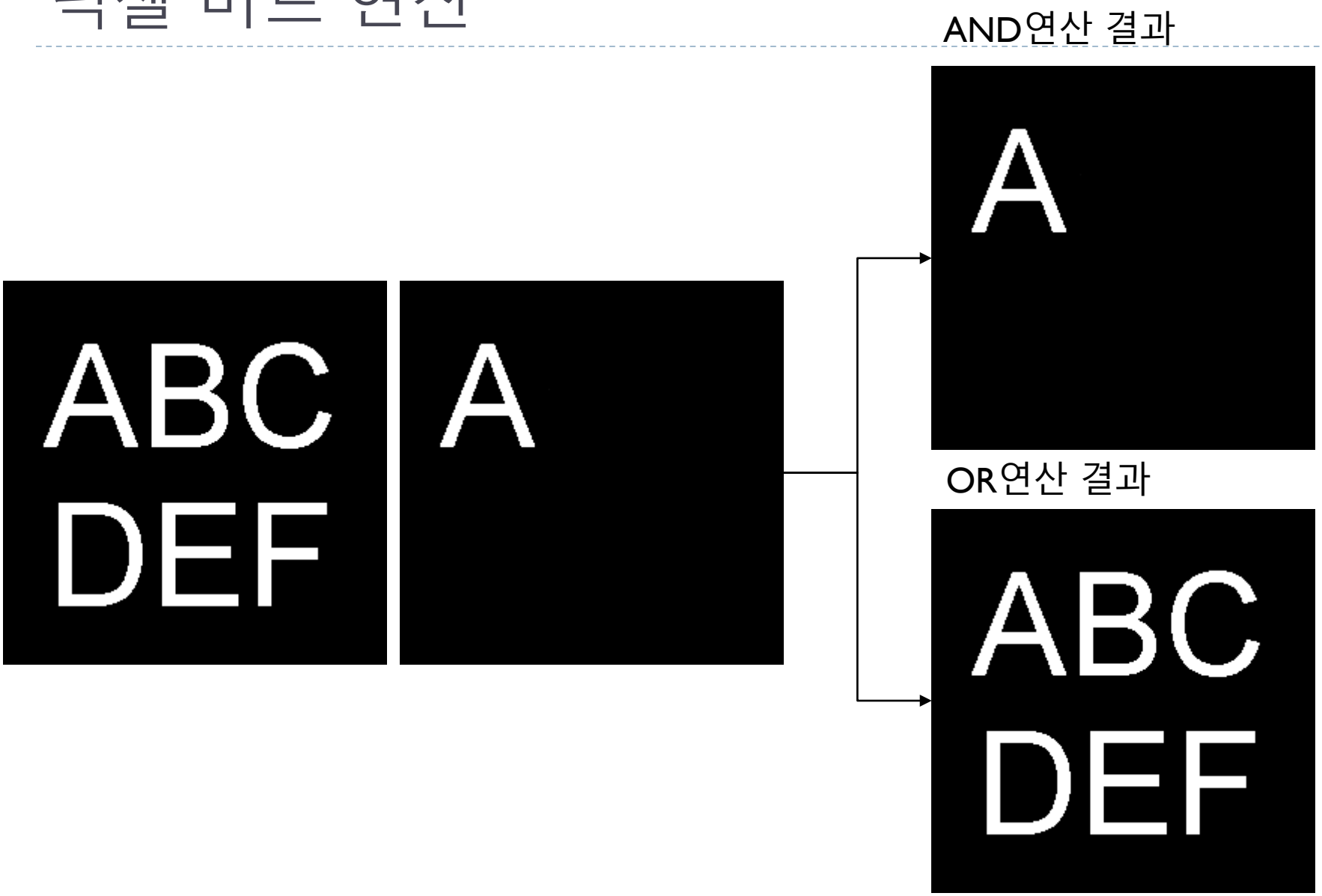
화소 비트 : 1 0 1 1 0 1 0 1 연산 전
이진 데이터 : 0 0 0 0 1 1 1 1 마스크(AND) 연산
마스크 결과 : 0 0 0 0 0 1 0 1 연산 후




화소 비트 : 1 0 0 1 0 0 1 0 연산 전
이진 데이터 : 0 0 0 0 1 1 1 1 선택적-세트(OR)연산
선택적 세트 결과 : 1 0 0 1 1 1 1 1 연산 후



# 픽셀 비트 연산



# 실습) 픽셀 비트 연산

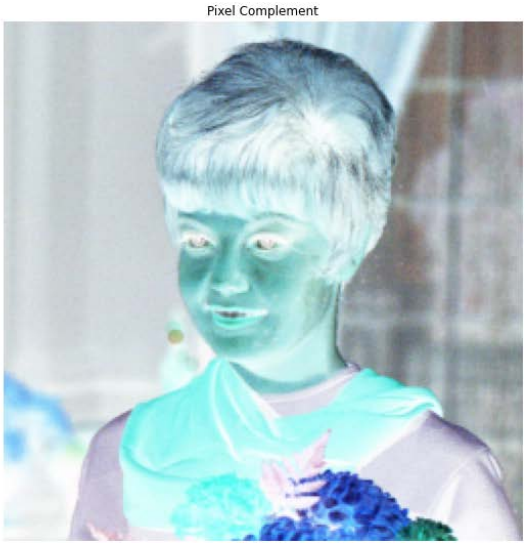
```
from google.colab.patches import cv2_imshow
import cv2
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/gdrive')
#영상 불러오기
img1 = cv2.imread('/content/gdrive/My Drive/Image_Processing/abcdef.bmp')
img2 = cv2.imread('/content/gdrive/My Drive/Image_Processing/a.bmp')
bitwise_and=cv2.bitwise_and(img1,img2)
bitwise_or=cv2.bitwise_or(img1,img2)
cv2_imshow(bitwise_and)
cv2_imshow(bitwise_or)
```

# 3.7 픽셀 반전 연산하기

# 픽셀 반전 연산(Pixel complement)

- 이진 영상의 경우 검은색 픽셀은 흰색으로, 흰색픽셀은 검은색으로 매핑
- 컬러영상의 경우 각 픽셀값의 보수로 매핑되는 연산

$$O(x, y) = 255 - P(x, y)$$

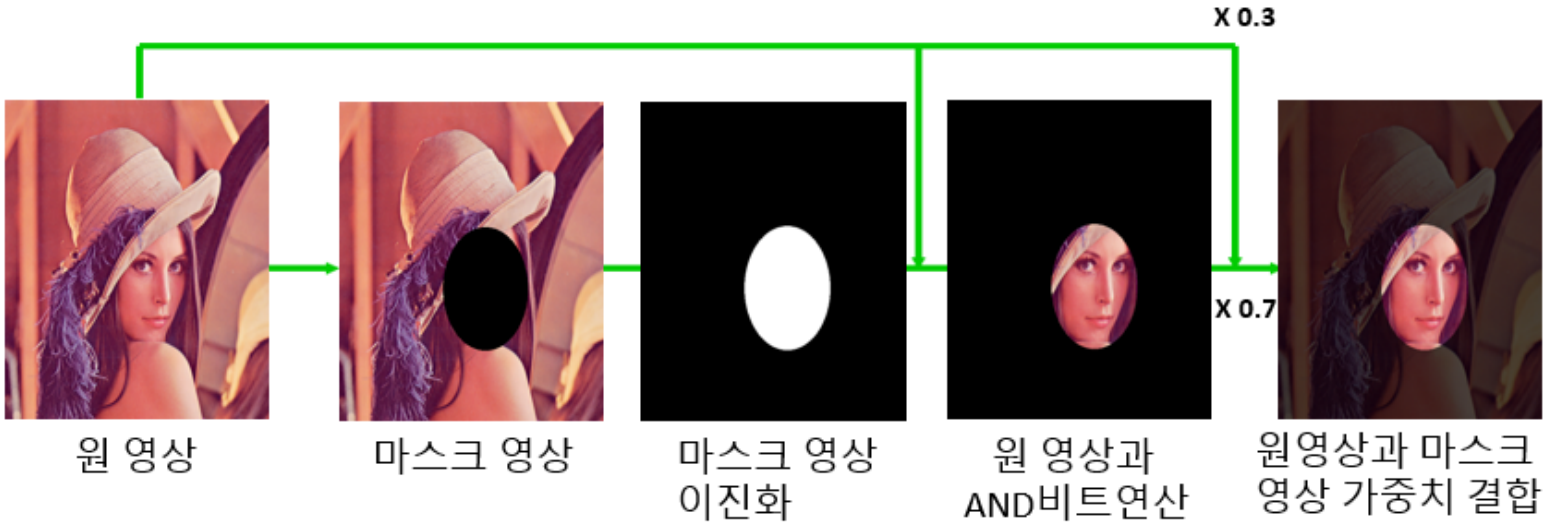



```
img1 = cv2.imread('/content/gdrive/My Drive/Image_Processing/girl.jpg')
negative=cv2.bitwise_not(img1)
cv2_imshow(negative)
```

# 3. 픽셀 산술 연산의 응용

# 픽셀 산술 연산의 응용

1. 입력 영상에 대해 마스크로 사용될 영역을 선택하여 검게 표시함
2. 검게 표시된 영상에 대해 이진화를 수행
3. 이진화가 수행된 영상과 원 영상에 대해 논리 AND 연산을 수행
4. 마스크 영역에 대해서는 0.7의 가중치를 부여하고 원 영상에 대해서는 0.3의 가중치를 부여하여 결합 가중치 연산을 수행



# 픽셀 산술 연산의 응용

- 연습 3-12 (Image Blending)