Image Thresholding and Blending

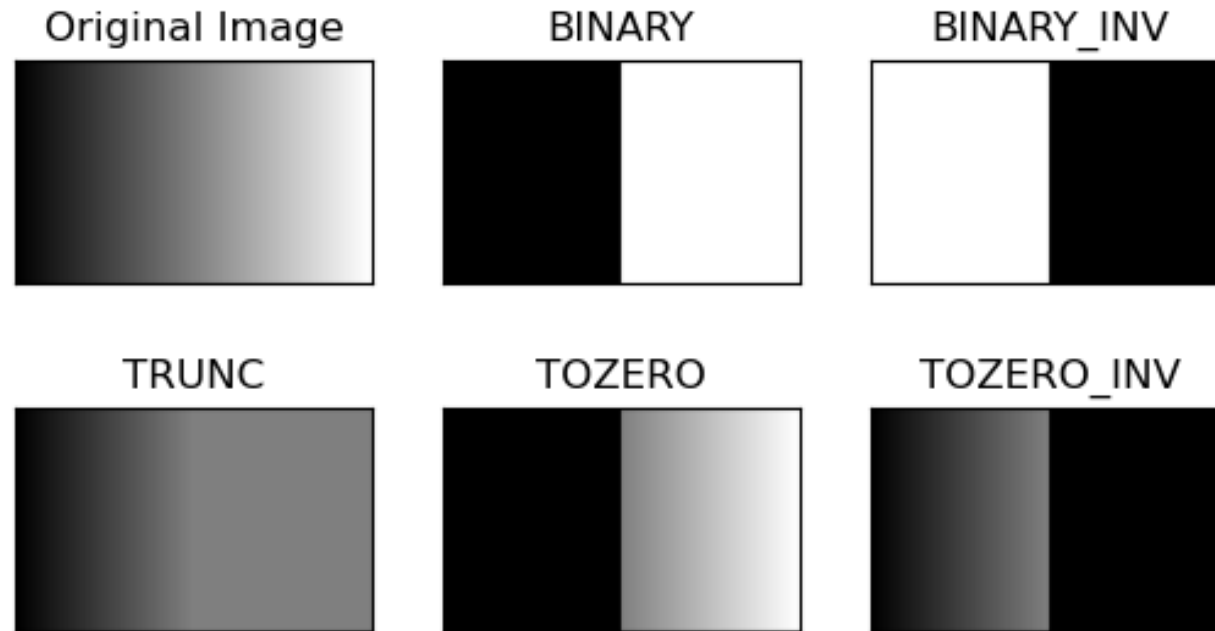김수환

학습목표

학습목표

1. Image Processing
    1. Image Thresholding
    2. Image Blending

Image Thresholding

Image Thresholding: Example

- Grayscale 이미지를 binary 이미지로 이진화 (binarization)
- 임계값(threshold)을 기준으로 흰색과 검은색 영역으로 분리

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

Image Thresholding: Code

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Create a gradient image
img = np.zeros([160, 256], np.uint8)
for i in range(256):
    img[:, i] = i

# Image thresholding
thld = 127
max_v = 255
ret, img1 = cv2.threshold(img, thld, max_v, cv2.THRESH_BINARY)
ret, img2 = cv2.threshold(img, thld, max_v, cv2.THRESH_BINARY_INV)
ret, img3 = cv2.threshold(img, thld, 0,     cv2.THRESH_TRUNC)
ret, img4 = cv2.threshold(img, thld, 0,     cv2.THRESH_TOZERO)
ret, img5 = cv2.threshold(img, thld, 0,     cv2.THRESH_TOZERO_INV)

# Draw results
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
```

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

컴퓨터비전 - 김수환

`cv.threshold()`[1]

```
retval, dst = cv.threshold(src, thresh, maxval, type[, dst])
```

- 고정된 임계점을 기준으로 각 픽셀을 이진화한다.
  - `src`: input array (multiple-channel, 8-bit or 32-bit floating point)
  - `dst`: output array of the same same and type and the same number of channels
  - `thresh`: threshold value
  - `maxval`: maximum value to use with the `THRESH_BINARY` and `THRESH_BINARY_INV` types
  - `type`: thresholding type

type[2]	
cv.THRESH_BINARY	cv.THRESH_BINARY_INV
cv.THRESH_TRUNC	
cv.THRESH_TOZERO	cv.THRESH_TOZERO_INV

1. https://docs.opencv.org/4.4.0/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57
2. https://docs.opencv.org/4.4.0/d7/d1b/group__imgproc__misc.html#gaa9e58d2860d4afa658ef70a9b1115576

Threshold Types

- `cv.THRESH_BINARY`

$$\mathrm{dst}(x,y) = \begin{cases} \mathrm{maxval}, & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ 0, & \text{otherwise} \end{cases}$$

- `cv.THRESH_BINARY_INV`

$$\mathrm{dst}(x,y) = \begin{cases} 0, & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ \mathrm{maxval}, & \text{otherwise} \end{cases}$$
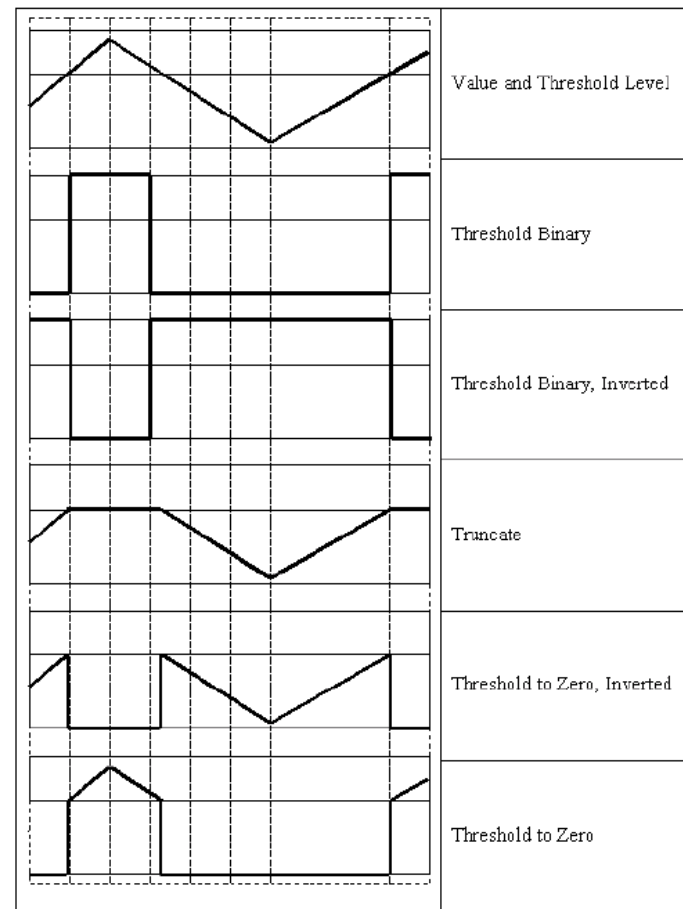
- `cv.THRESH_TRUNC`

$$\mathrm{dst}(x,y) = \begin{cases} \mathrm{threshold}, & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ \mathrm{src}(x,y), & \text{otherwise} \end{cases}$$

- `cv.THRESH_TOZERO`

$$\mathrm{dst}(x,y) = \begin{cases} \mathrm{src}(x,y), & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ 0, & \text{otherwise} \end{cases}$$

- `cv.THRESH_TOZERO_INV`

$$\mathrm{dst}(x,y) = \begin{cases} 0, & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ \mathrm{src}(x,y), & \text{otherwise} \end{cases}$$

Value and Threshold Level

Threshold Binary

Threshold Binary, Inverted

Truncate

Threshold to Zero, Inverted

Threshold to Zero

https://docs.opencv.org/4.4.0/d7/d1b/group__imgproc__misc.html#gaa9e58d2860d4afa658ef70a9b1115576

컴퓨터비전 - 김수환

Otsu's Binarization

- Thresholding
  - Threshold value: manual selection
  - Threshold 값을 알고 있을 때 사용

- Otsu's Binarization: automatic selection
  - Otsu's Image Segmentation
  - Histogram의 peak가 두 개인 bimodal image일 때 사용
  - 자동으로 두 peak 사이의 중간값을 찾아서 사용
  - `retval`: optimal threshold

Otsu's Binarization: Example and Result
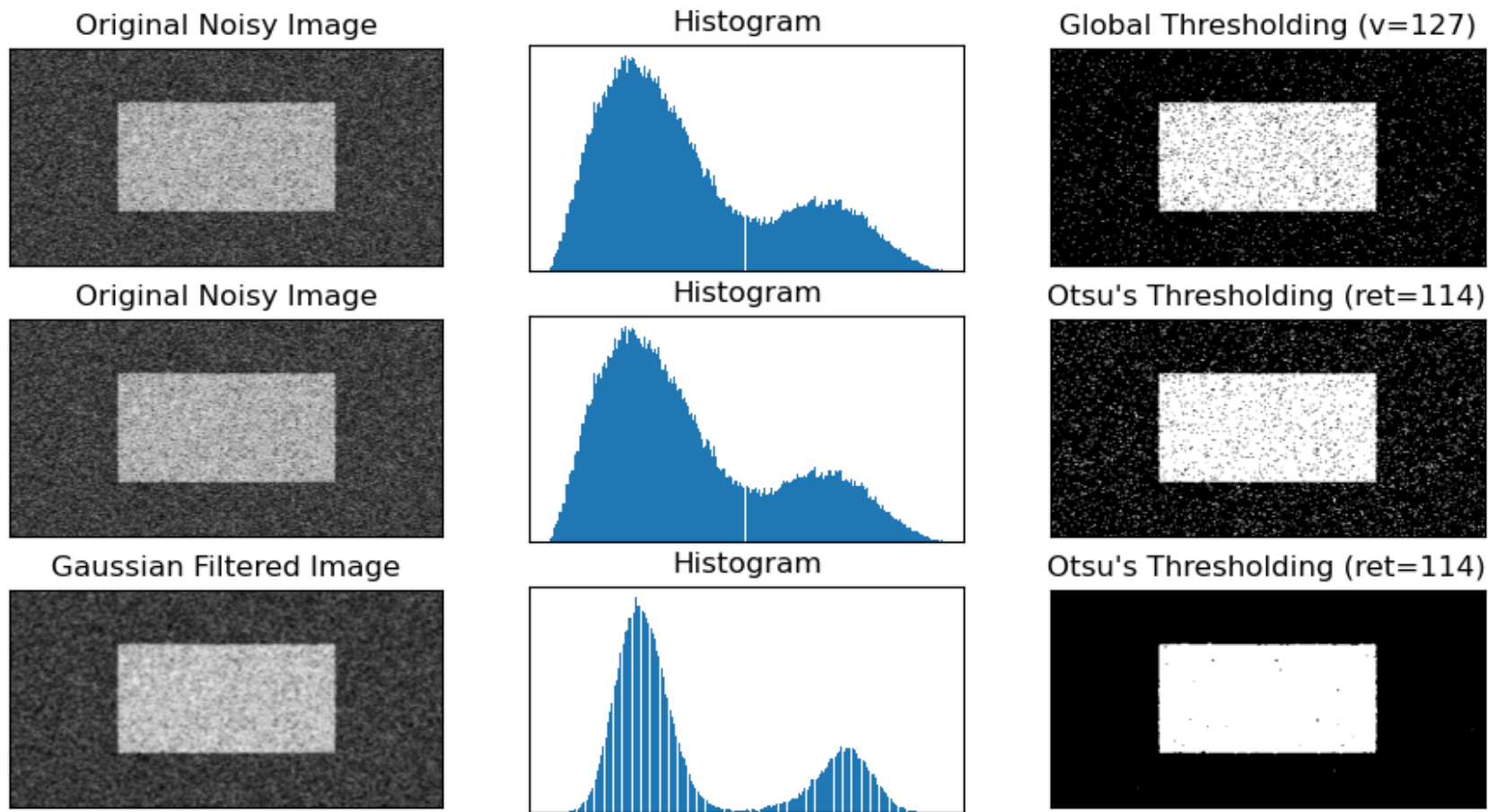
Otsu's Binarization: Code

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load a grayscale image
img = cv2.imread('noisy2.png', 0)

# global thresholding
ret1, th1 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# Otsu's thresholding
ret2,th2=cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Otsu's thresholding after Gaussian filtering
blur = cv2.GaussianBlur(img, (5,5), 0)
ret3,th3=cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# plot all the images and their histograms
images = [img, 0, th1,
```

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#otsus-binarization

컴퓨터비전 - 김수환

Otsu's Binarization: Under the Hood

- Optimal threshold value t
  - Minimizing the weighted within-class variance

$$\sigma_w^2 = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

where
  - Cumulative probabilities for two classes

$$q_1(t) = \sum_{i=1}^{t} P(i), \quad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

  - Means

$$\mu_1 = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)}, \quad \mu_2 = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

  - Variances

$$\sigma_1^2 = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}, \quad \sigma_2^2 = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

Otsu's Binarization: Under the Hood

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load a grayscale image
img = cv2.imread('noisy2.png', 0)

# Blur the image
blur = cv2.GaussianBlur(img, (5,5), 0)

# Find normalized_histogram, and its cumulative distribution
hist = cv2.calcHist([blur], [0], None, [256], [0,256])
hist_norm = hist.ravel()/hist.max()
Q = hist_norm.cumsum()

# Find probabilities
bins = np.arange(256)

fn_min = np.inf
```

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#otsus-binarization

컴퓨터비전 - 김수환

Adaptive Thresholding

Adaptive Thresholding

- Thresholding
  - Threshold value: global
  - Image 모든 영역에서 lighting condition이 동일할 때 사용

- Adaptive Thresholding
  - Threshold value: local
  - Image 각 영역의 lighting condition이 다를 때 사용

Adaptive Thresholding: Example 1

Original Image | Global Thresholding (v = 127)

Adaptive Mean Thresholding | Adaptive Gaussian Thresholding

컴퓨터비전 - 김수환

Adaptive Thresholding: Code 1

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load a grayscale image
img_gray = cv2.imread('sudoku.png', 0)

# Blur the image
img_blurred = cv2.medianBlur(img_gray, 5)
cv2.imshow('Original vs Blurred', cv2.hconcat([img_gray, img_blurred]))
cv2.waitKey(0)
cv2.destroyAllWindows()

# Threshold the image
ret, img1 = cv2.threshold(img_blurred, 127, 255, cv2.THRESH_BINARY)
img2=cv2.adaptiveThreshold(img_blurred,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2)
img3=cv2.adaptiveThreshold(img_blurred,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,

# Draw results
titles = ['Original Image', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding',
'Adaptive Gaussian Thresholding']
```

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

cv.adaptiveThreshold()[1]

```
dst = cv.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst])
```

- Array에 adaptive threshold를 적용한다.
  - `src`: input 8-bit single-channel image
  - `dst`: output image of the same size and the same type as input
  - `maxval`: non-zero value assigned to the pixels for which the condition is satisfied
  - `adaptiveMethod`: adaptive thresholding algorithm to use
  - `thresholdType`: thresholding type that must be either `THRESH_BINARY` or `THRESH_BINARY_INV`
  - `blockSize`: Size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on.
  - `C`: constant subtracted from the mean or weighted mean. Normally, it is positive but may be zero or negative as well.

1. https://docs.opencv.org/4.4.0/d7/d1b/group__imgproc__misc.html#ga72b913f352e4a1b1b397736707afcde3

Threshold Types

- `cv.THRESH_BINARY`

$$\mathrm{dst}(x, y) = \begin{cases} \mathrm{maxval}, & \text{if } \mathrm{src}(x, y) > T(x, y) \\ 0, & \text{otherwise} \end{cases}$$

- `cv.THRESH_BINARY_INV`

$$\mathrm{dst}(x, y) = \begin{cases} 0, & \text{if } \mathrm{src}(x, y) > T(x, y) \\ \mathrm{maxval}, & \text{otherwise} \end{cases}$$

컴퓨터비전 - 김수환

AdaptiveThresholdTypes[1]

AdaptiveThresholdTypes[2]	
`cv.ADAPTIVE_THRESH_MEAN_C`	The threshold value T(x,y) is a mean of the `blockSize×blockSize` neighborhood of (x,y) minus `C`
`cv.ADAPTIVE_THRESH_GAUSSIAN_C`	The threshold value T(x,y) is a weighted sum (cross-correlation with a Gaussian window) of the `blockSize×blockSize` neighborhood of (x,y) minus `C` . The default sigma (standard deviation) is used for the specified blockSize

1. https://docs.opencv.org/4.4.0/d7/d1b/group__imgproc__misc.html#gaa42a3e6ef26247da787bf34030ed772c

컴퓨터비전 - 김수환

Adaptive Thresholding: Example 2

Adaptive Thresholding: Result 2

Original Image
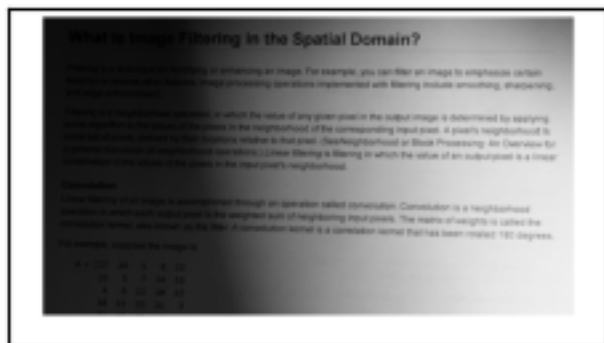
Global Thresholding (v = 127)

Adaptive Mean Thresholding

Adaptive Gaussian Thresholding

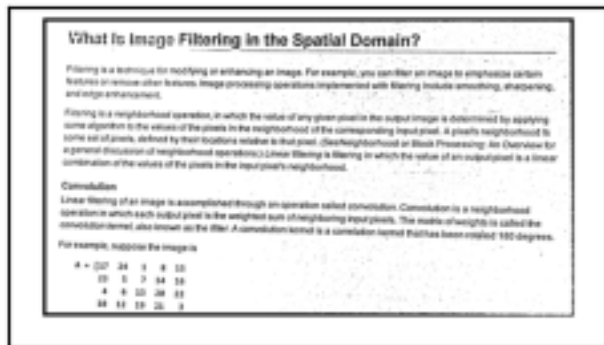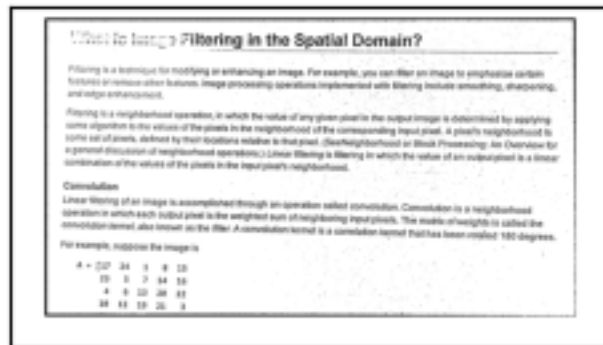# Adaptive Thresholding: Code 2

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load a grayscale image
img = cv2.imread('book-scan.png', 0)

# Thresholding
ret, img1 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
img2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,21,5)
img3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,21,5)

# Draw results
titles = ['Original Image', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding',
'Adaptive Gaussian Thresholding']
images = [img, img1, img2, img3]

for i in range(4):
    plt.subplot(2, 2, i+1)
    plt.imshow(images[i], 'gray')
```

1. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html
2. https://webnautes.tistory.com/1254

🧬 Image Blending

Image Addition

- Numpy
  - `res = img1 + img2`
  - Modulo operation

- OpenCV:
  - `res = cv2.add(img1, img2)`
  - Saturated operation

```
>>> x = np.uint8([250])
>>> y = np.uint8([10])

>>> print cv2.add(x,y)  # 250+10 = 260 => 255
[[255]]

>>> print x+y           # 250+10 = 260 % 256 = 4
[4]
```

Use OpenCV for Image Addition!

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_core/py_image_arithmetics/py_image_arithmetics.html

컴퓨터비전 - 김수환

Image Blending

Image Blending

- Weighted sum

$$z = (1 - \alpha)x + \alpha y$$

- `cv.addWeighted()`

$$\mathrm{dst}(I) = \mathrm{saturate}(\alpha \cdot \mathrm{src1}(I) + \beta \cdot \mathrm{src2}(I) + \gamma)$$

Image Blending

```python
import numpy as np
import cv2

# Load two images
img_park = cv2.imread('park.png')
img_tom = cv2.imread('tom.png')

# Blend those two images
cv2.imshow('Image Blening', cv2.hconcat([img_park, img_park, img_tom]))
cv2.waitKey(0)
for i in np.linspace(0, 1, 100):
    dst = cv2.addWeighted(img_park, 1-i, img_tom, i, 0)
    cv2.imshow('Image Blening', cv2.hconcat([img_park, dst, img_tom]))
    cv2.waitKey(30)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_core/py_image_arithmetics/py_image_arithmetics.html#image-blending

`cv.addWeighted()`[1]

```
dst = cv.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]])
```

- 두 array의 weighted sum을 계산한다.
    - `src1`: first input array
    - `alpha`: weight of the first array elements
    - `src2`: second input array
    - `beta`: weight of the second array elements
    - `gamma`: scalar added to each sum
    - `dst`: output array that has the same size and number of channels as the input arrays
    - `dtype`: optional depth of the output array

$$\mathrm{dst}(I) = \mathrm{saturate}(\alpha \cdot \mathrm{src1}(I) + \beta \cdot \mathrm{src2}(I) + \gamma)$$

1. https://docs.opencv.org/4.4.0/d2/de8/group__core__array.html#gafafb2513349db3bcff51f54ee5592a19
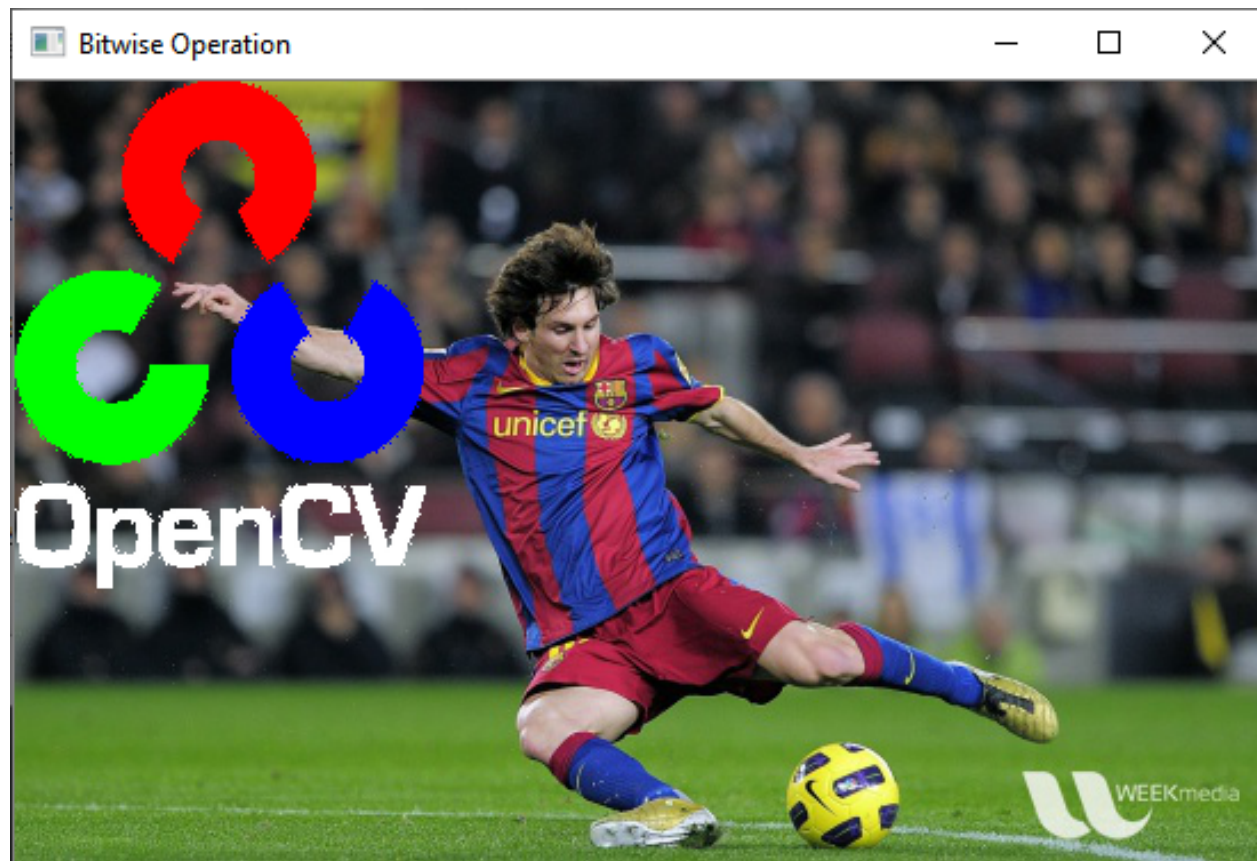
🧬 Bitwise Operations

Bitwise Operations

- AND, OR, NOT, XOR
- Image에서 특정 영역을 추출할 때 사용
- 사각형 이외의 특이한 ROI가 필요할 때 사용

Bitwise Operations: Example

Bitwise Operations: Code

```python
import numpy as np
import cv2
from matplotlib import pyplot as plt

# Load two images
img_messi = cv2.imread('messi5.jpg')
img_logo = cv2.imread('opencv-logo-white.png')

# Now create a mask of logo and its inverse
img_logo_gray = cv2.cvtColor(img_logo, cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img_logo_gray,10,255,cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)

# I want to put logo on top-left corner, So I create a ROI
rows, cols, channels = img_logo.shape
roi = img_messi[0:rows, 0:cols]

# Take only region of logo from logo image
img_logo_fg = cv2.bitwise_and(img_logo, img_logo, mask = mask)

# Now black-out the area of logo in ROI
```
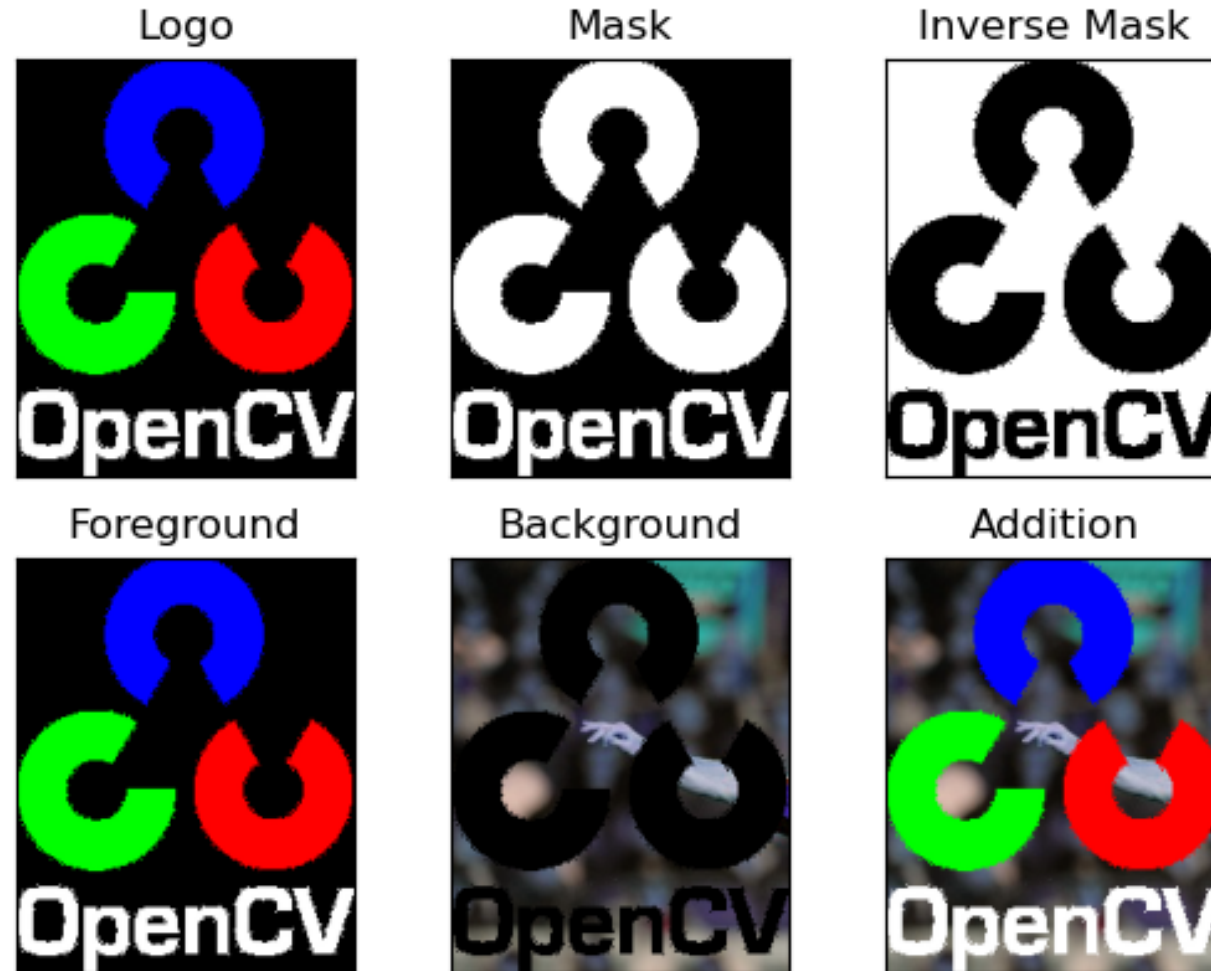
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_core/py_image_arithmetics/py_image_arithmetics.html#bitwise-operations

컴퓨터비전 - 김수환

Bitwise Operations: Result

`cv.bitwise_and()`[1]

```
dst = cv.bitwise_and(src1, src2[, dst[, mask]])
```

- Bit-wise AND 연산
  - `src1`: first input array or a scalar
  - `src2`: second input array or a scalar
  - `dst`: output array that has the same size and type as the input arrays
  - `mask`: optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed

$$\mathrm{dst}(I) = \mathrm{src1}(I) \ \wedge \ \mathrm{src2}(I), \quad \text{if } \mathrm{mask}(I) \neq 0$$
$$\mathrm{dst}(I) = \mathrm{src1}(I) \ \wedge \ \mathrm{src2}, \qquad \text{if } \mathrm{mask}(I) \neq 0$$
$$\mathrm{dst}(I) = \mathrm{src1} \qquad \wedge \ \mathrm{src2}(I), \quad \text{if } \mathrm{mask}(I) \neq 0$$

1. https://docs.opencv.org/4.4.0/d2/de8/group__core__array.html#ga60b4d04b251ba5eb1392c34425497e14

cv.bitwise_or()[1]

```
dst = cv.bitwise_or(src1, src2[, dst[, mask]])
```

- Bit-wise OR 연산
  - `src1`: first input array or a scalar
  - `src2`: second input array or a scalar
  - `dst`: output array that has the same size and type as the input arrays
  - `mask`: optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed

$$\mathrm{dst}(I) = \mathrm{src1}(I) \ \vee \ \mathrm{src2}(I), \quad \text{if mask}(I) \neq 0$$
$$\mathrm{dst}(I) = \mathrm{src1}(I) \ \vee \ \mathrm{src2}, \qquad \text{if mask}(I) \neq 0$$
$$\mathrm{dst}(I) = \mathrm{src1} \qquad \vee \ \mathrm{src2}(I), \quad \text{if mask}(I) \neq 0$$

1. https://docs.opencv.org/4.4.0/d2/de8/group__core__array.html#gab85523db362a4e26ff0c703793a719b4

`cv.bitwise_not()`[1]

```
dst = cv.bitwise_not(src[, dst[, mask]])
```

- Bit-wise NOT 연산
    - `src`: input array or a scalar
    - `dst`: output array that has the same size and type as the input arrays
    - `mask`: optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed

$$\text{dst}(I) = \neg\text{src}(I), \ \ \text{if mask}(I) \neq 0$$

`cv.bitwise_xor()`[1]

```
dst = cv.bitwise_xor(src1, src2[, dst[, mask]])
```

- Bit-wise XOR 연산
  - `src1`: first input array or a scalar
  - `src2`: second input array or a scalar
  - `dst`: output array that has the same size and type as the input arrays
  - `mask`: optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed

$$\mathrm{dst}(I) = \mathrm{src1}(I) \ \oplus \ \mathrm{src2}(I), \quad \text{if } \mathrm{mask}(I) \neq 0$$
$$\mathrm{dst}(I) = \mathrm{src1}(I) \ \oplus \ \mathrm{src2}, \qquad \text{if } \mathrm{mask}(I) \neq 0$$
$$\mathrm{dst}(I) = \mathrm{src1} \qquad \oplus \ \mathrm{src2}(I), \quad \text{if } \mathrm{mask}(I) \neq 0$$

1. https://docs.opencv.org/4.4.0/d2/de8/group__core__array.html#ga84b2d8188ce506593dcc3f8cd00e8e2c

컴퓨터비전 - 김수환

Push Code to GitHub

References

References

- OpenCV Python Tutorials
  - Core Operations
    - Basic Operations on Images
    - Arithmetic Operations on Images
    - Image Thresholding