Slides and example codes are available:
https://github.com/mint-lab/cv_tutorial

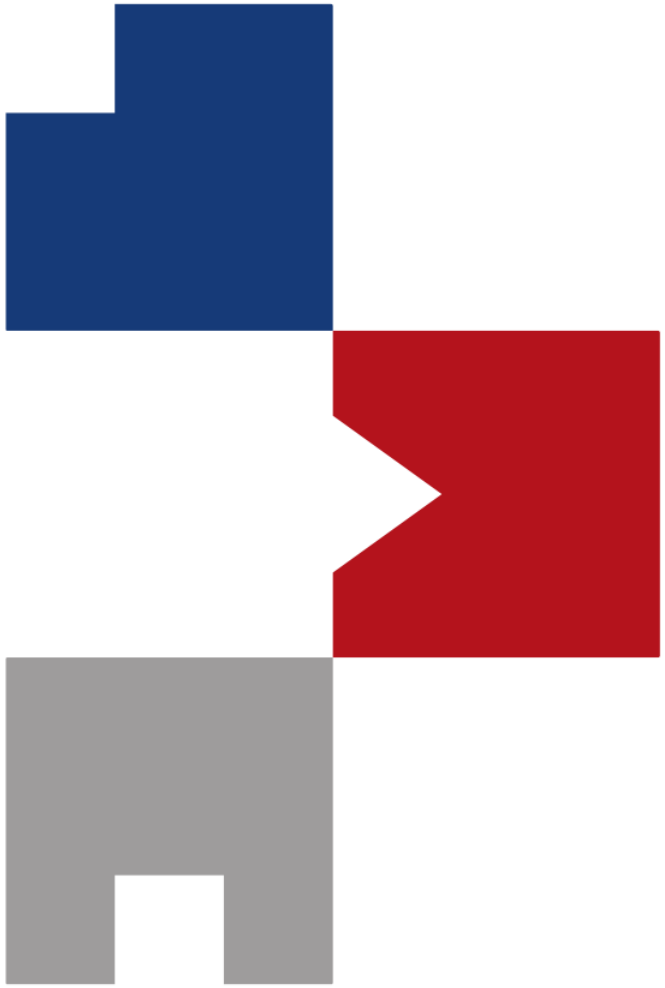


Image Editing:
Learning OpenCV

Sunglok Choi, Assistant Professor, Ph.D.
Computer Science and Engineering Department, SEOULTECH
sunglok@seoultech.ac.kr | https://mint-lab.github.io/

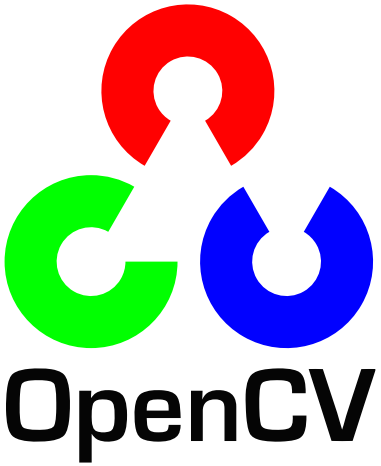
OpenCV: Open-source Computer Vision Library

- OpenCV is the most popular open-source library for real-time computer vision.
  - It includes many basic functionalities (e.g. `cv::Mat`, `cv::imread()`, `cv::VideoCapture`, `cv::FileStorage`) and useful algorithms (e.g. `cv::solvePnP()`, `cv::SIFT`, `cv::ml::SVM`, `cv::dnn::DetectionModel`)
  - Its codes are written in C++ and highly optimized (for real-time operation) with Intel IPP, Intel TBB, NVIDIA CUDA, OpenCL, OpenMP, and more.
  - Cross-platform: C++, Python, Java, MATLAB @ Windows, Linux, MacOS, iOS, Android
  - License: Apache 2 (free for commercial use)
- References: **Documentation** (tutorials and APIs), Github, Cheatsheet (by Antonio Anjos)
- Installation in Python
  - Installation

    ```
    pip install opencv-python opencv-contrib-python
    ```

  - Check the installation and its version

    ```
    import cv2 as cv
    print(cv.__version__) # 4.7.0
    ```



OpenCV Documentation

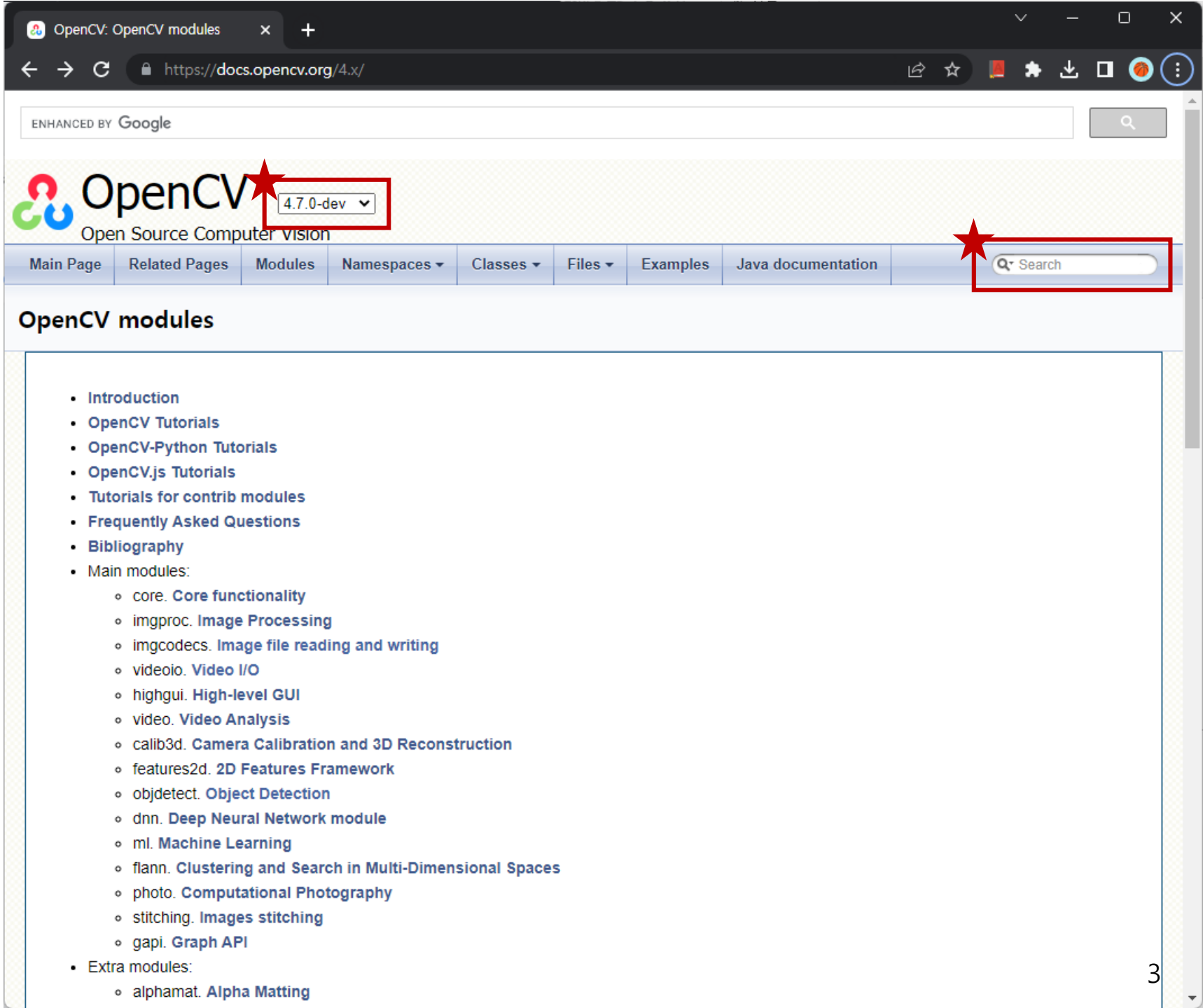
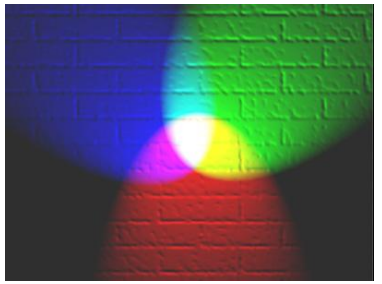


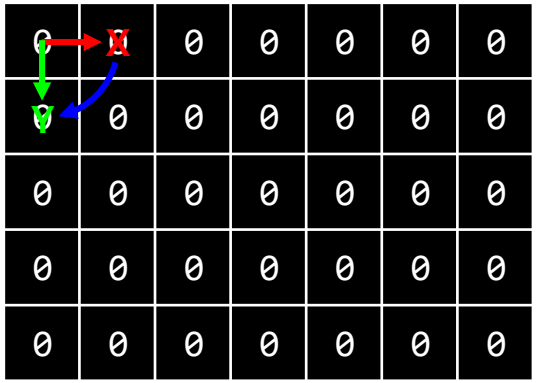
Table of Contents

OpenCV Image Representation

- An image is represented by the **multi-dimensional array of NumPy** (`np.array`) in Python.
  - Note) An image is represented by the `cv::Mat` class in C++.
  - The shape of NumPy array is `(height, width, channel)`, not `(width, height, channel)`.
    - A pixel at (x, y) is accessed through `image[y,x]`, not `image[x,y]`.
    - The channel of a **gray** image: **1**
    - The channel of a **color** image: **3** (or 4 with alpha channel)
      - The order of color channels is **Blue-Green-Red (BGR)**, not RGB.
  - The data type of NumPy array is `np.uint8` (size: 8 bits) whose range is 0 to 255.
  - Examples) 8-by-5 images (width: 8, height: 5)

Why? Additive color





255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

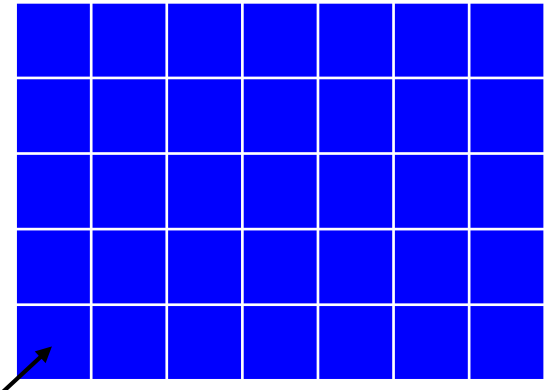
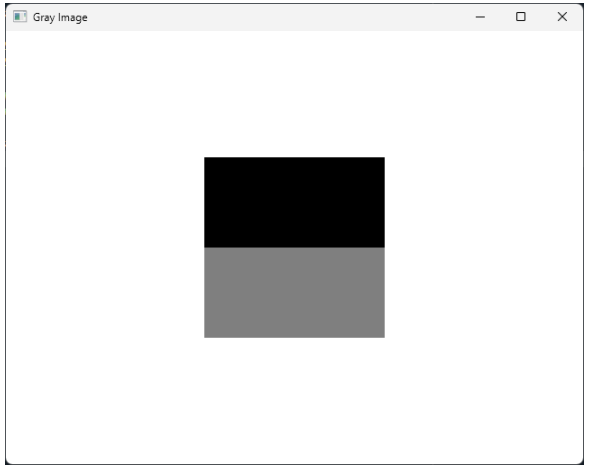


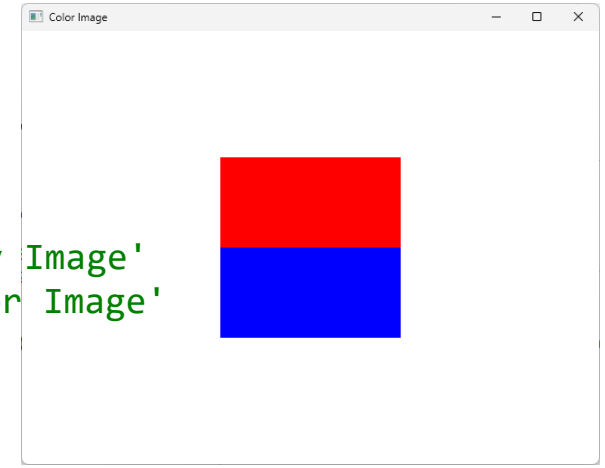
Image: Wikipedia

OpenCV Image Representation

- Example) Image creation

```
import numpy as np
import cv2 as cv

img_gray = np.full((480, 640), 255, dtype=np.uint8) # Create a gray image (white)
img_gray[140:240, 220:420] = 0                      # Draw the black box
img_gray[240:340, 220:420] = 127                    # Draw the gray box

img_color = np.zeros((480, 640, 3), dtype=np.uint8) # Create a color image (black)
img_color[:] = 255                                  # Make the color image white
img_color[140:240, 220:420, :] = (0, 0, 255)        # Draw the red box
img_color[240:340, 220:420, :] = (255, 0, 0)        # Draw the blue box

cv.imshow('Gray Image',  img_gray)  # Show 'img_gray'  on a new window named as 'Gray Image'
cv.imshow('Color Image', img_color) # Show 'img_color' on a new window named as 'Color Image'
cv.waitKey()                        # Wait until a user press any key
cv.destroyAllWindows()              # It is necessary only for Spyder IDE.
```





OpenCV Image Representation

- Note) Image visualization functions in OpenCV High-level GUI module
  - `cv.imshow(winname, mat) → None`
    - Displays an image (`mat`) in the specified window.
  - `cv.waitKey([, delay]) → keycode or -1`
    - Waits for a pressed key and return the key code.
      - `delay`: Delay in milliseconds. 0 is the special value that means "forever".
      - `keycode`: ASCII code of the pressed key (e.g. 27 for ESC key code)
    - Note) `cv.waitKeyEx()` can return full key code including arrow keys.
      - However, its additional key codes are different according to its GUI backend (e.g. Win32/QT/GTK).
  - `cv.destroyAllWindows() → None`
    - Destroys all of the HighGUI windows.
    - Note) `cv.destroyWindow(winname)` can destroy the specific window.

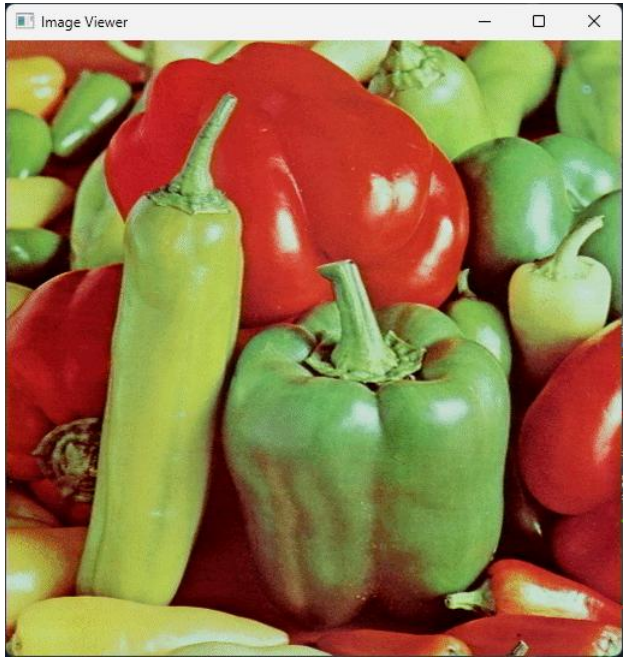
OpenCV Image and Video Input/Output

- **Image files**
  - `cv.imread(filename[, flags]) → image or None`
    - Loads an image from a file.
  - `cv.imwrite(filename, img[, params]) → retval`
    - Saves an image to a specified file.
- **Video files and cameras**
  - `cv.VideoCapture`
    - A class for reading images from video files, image sequences, cameras, and RTSP.
    - The member function, `get()` and `set()`, can access and modify video properties.
      - Note) Flags for Video I/O (e.g. `cv.CAP_PROP_FPS`)
  - `cv.VideoWriter`
    - A class for video writing.

OpenCV Image Input/Output

- Example) Image file viewer

```
import cv2 as cv

img_file = 'data/peppers_color.tif'

# Read the given image file
img = cv.imread(img_file)

# Check whether the image is valid or not
if img is not None:
    # Show the image
    cv.imshow('Image Viewer', img)
    cv.waitKey()
    cv.destroyAllWindows()
```



OpenCV Image Input/Output

- Example) Image format converter

```
import cv2 as cv

img_file = 'data/peppers_color.tif'
target_format = 'png'

# Read the given image file
img = cv.imread(img_file)

# Check whether the image is valid or not
if img is not None:
    # Write 'img' as a file named 'target_file'
    target_file = img_file[:img_file.rfind('.')] + '.' + target_format
    cv.imwrite(target_file, img)
```

OpenCV Video Input/Output

- Example) Video file player

```
video_file = 'data/PETS09-S2L1-raw.webm'

# Read the given video file
# Note) Additional argument examples
# - Image sequence: video_file = 'data/PETS09-S2L1-raw_%04d.png'
# - Camera        : video_file = 0 (Note: The camera index)
video = cv.VideoCapture(video_file)

if video.isOpened():
    # Get FPS and calculate the waiting time in millisecond
    fps = video.get(cv.CAP_PROP_FPS)
    wait_msec = int(1 / fps * 1000)

    while True:
        # Read an image from 'video'
        valid, img = video.read()
        if not valid:
            break

        # Show the image
        cv.imshow('Video Player', img)

        # Terminate if the given key is ESC
        key = cv.waitKey(wait_msec)
        if key == 27: # ESC
            break

    cv.destroyAllWindows()
```

OpenCV Video Input/Output

- Example) Video format converter

```
video_file = 'data/PETS09-S2L1-raw.webm'
target_format = 'avi'
target_fourcc = 'XVID' # Note) FourCC: https://learn.microsoft.com/en-us/windows/win32/medfound/video-fourccs

# Read the given video file
video = cv.VideoCapture(video_file)

if video.isOpened():
    target = cv.VideoWriter()
    while True:
        # Get an image from 'video'
        valid, img = video.read()
        if not valid:
            break

        if not target.isOpened():
            # Open the target video file
            target_file = video_file[:video_file.rfind('.')] + '.' + target_format
            fps = video.get(cv.CAP_PROP_FPS)
            h, w, *_ = img.shape
            is_color = (img.ndim > 2) and (img.shape[2] > 1)
            target.open(target_file, cv.VideoWriter_fourcc(*target_fourcc), fps, (w, h), is_color)

        # Add the image to 'target'
        target.write(img)

    target.release()
```

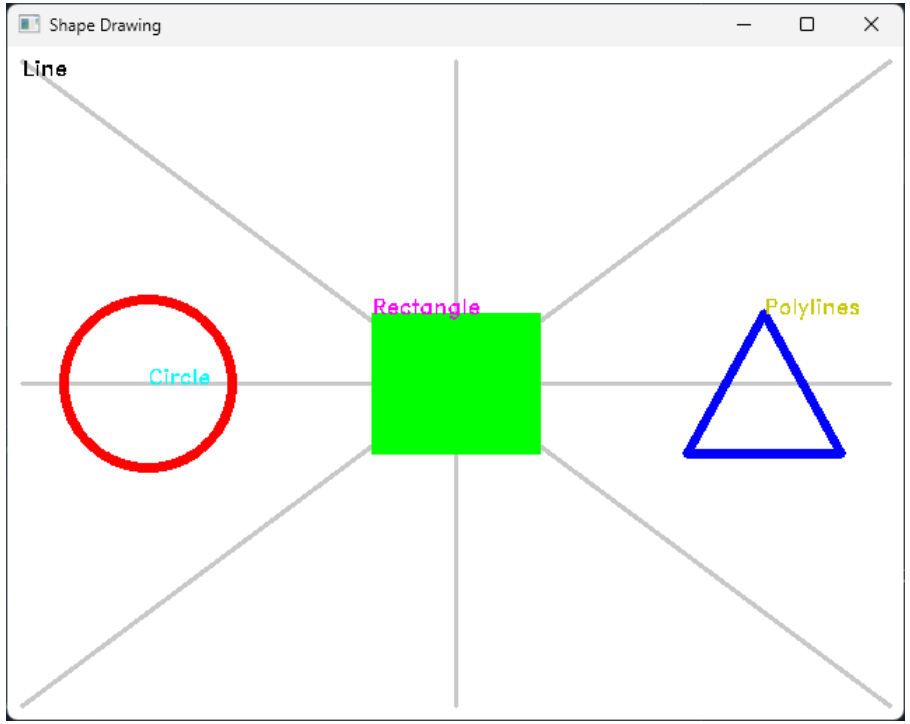
OpenCV Drawing Functions

- Drawing functions in OpenCV Image Processing module
  - `cv.line(img, pt1, pt2, color, [thickness, …]) → img`
    - Draw a line segment connecting two points.
    - Note) `img` is modified as call-by-reference. `pt1` and `pt2` should be integers, not real numbers.
  - `cv.circle(img, center, radius, color [, thickness, …]) → img`
    - Draw a circle.
    - Note) Negative values of `thickness` draws a circle filled with the given `color`.
  - `cv.rectangle(img, pt1, pt2, color [, thickness, …]) → img`
    - Draw a rectangle.
  - `cv.polylines(img, pts, isClosed, color [, thickness, …]) → img`
    - Draw a multiple connected lines or a polygon (if `isClosed` is True).
  - `cv.fillPoly(img, pts, color [, …]) → img`
    - Draw a polygon filled with the given `color`.
  - `cv.putText(img, text, org, fontFace, fontScale, color [, thickness, …]) → img`
    - Draw a `text` at the given point, `org`. (Note: The top left of the `text` is aligned at `org` by default.)

OpenCV Drawing Functions

- Examples) Shape drawing

```
# Prepare a canvas
canvas = np.full((480, 640, 3), 255, dtype=np.uint8)

# Draw lines with its label
cv.line(canvas, (10, 10), (640-10, 480-10), color=(200, 200, 200), thickness=2)
cv.line(canvas, (640-10, 10), (10, 480-10), color=(200, 200, 200), thickness=2)
cv.line(canvas, (320, 10), (320, 480-10), color=(200, 200, 200), thickness=2)
cv.line(canvas, (10, 240), (640-10, 240), color=(200, 200, 200), thickness=2)
cv.putText(canvas, 'Line', (10, 20), cv.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 0))

# Draw a circle with its label
center = (100, 240)
cv.circle(canvas, center, radius=60, color=(0, 0, 255), thickness=5)
cv.putText(canvas, 'Circle', center, cv.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 0))

# Draw a rectangle with its label
pt1, pt2 = (320-60, 240-50), (320+60, 240+50)
cv.rectangle(canvas, pt1, pt2, color=(0, 255, 0), thickness=-1)
cv.putText(canvas, 'Rectangle', pt1, cv.FONT_HERSHEY_DUPLEX, 0.5, (255, 0, 255))

# Draw a polygon (triangle) with its label
pts = np.array([(540, 240-50), (540-55, 240+50), (540+55, 240+50)])
cv.polylines(canvas, [pts], True, color=(255, 0, 0), thickness=5)
cv.putText(canvas, 'Polylines', pts[0].flatten(), cv.FONT_HERSHEY_DUPLEX, 0.5, (0, 200, 200))

# Show the canvas
cv.imshow('Shape Drawing', canvas)
cv.waitKey()
cv.destroyAllWindows()
```



OpenCV High-level GUI

- OpenCV high-level GUI provides **common interface** for image visualization and minimal user interaction regardless of operating systems and GUI backends (e.g. Win32, Qt, GTK).
  - `cv.namedWindow(winname[, flags]) → None`
    - Creates a window.
  - `cv.setMouseCallback(winname, event_handler, userdata) → None`
    - Sets mouse handler for the specified window.
      - `event_handler`: Callback function for mouse events.
      - `userdata`: The optional parameter passed to the callback (for information sharing).
  - `cv.imshow(winname, mat) → None`
    - Displays an image (`mat`) in the specified window.
  - `cv.waitKey([, delay]) → keycode or -1`
    - Waits for a pressed key and return the key code.
  - `cv.destroyAllWindows() → None`
    - Destroys all of the HighGUI windows.

OpenCV High-level GUI: Handling Keyboard Events

- Example) Video file player with frame navigation

```
if video.isOpened():
    # Configure the frame navigation
    frame_total = int(video.get(cv.CAP_PROP_FRAME_COUNT))
    frame_shift = 10
    speed_table = [1/10, 1/8, 1/4, 1/2, 1, 2, 3, 4, 5, 8, 10]
    speed_index = 4

    while True:
        # Get an image from 'video'
        ...

        # Show the image
        ...

        # Process the key event
        key = cv.waitKey(max(int(wait_msec / speed_table[speed_index]), 1))
        if key == ord(' '):
            key = cv.waitKey()
        if key == 27: # ESC
            break
        elif key == ord('\t'):
            speed_index = 4
        elif key == ord('>') or key == ord('.'):
            speed_index = min(speed_index + 1, len(speed_table) - 1)
        elif key == ord('<') or key == ord(','):
            speed_index = max(speed_index - 1, 0)
        elif key == ord(']') or key == ord('}'):
            video.set(cv.CAP_PROP_POS_FRAMES, frame + frame_shift)
        elif key == ord('[') or key == ord('{'):
            video.set(cv.CAP_PROP_POS_FRAMES, max(frame - frame_shift, 0))
```

OpenCV High-level GUI: Handling Mouse Events

- Example) Free drawing

```
def mouse_event_handler(event, x, y, flags, param):
    # Change 'mouse_state' (given as 'param') according to the mouse 'event'
    if event == cv.EVENT_LBUTTONDOWN:
        param[0] = True
        param[1] = (x, y)
    elif event == cv.EVENT_LBUTTONUP:
        param[0] = False
    elif event == cv.EVENT_MOUSEMOVE and param[0]:
        param[1] = (x, y)

def free_drawing(canvas_width=640, canvas_height=480, init_brush_radius=3):
    # Prepare a canvas and palette
    canvas = np.full((canvas_height, canvas_width, 3), 255, dtype=np.uint8)
    palette = [(0, 0, 0), (255, 255, 255), (0, 0, 255), (0, 255, 0), (255, 0, 0), (255, 255, 0), (255, 0, 255), (0, 255, 255)]

    # Initialize drawing states
    mouse_state = [False, (-1, -1)] # Note) [mouse_left_button_click, mouse_xy]
    brush_color = 0
    brush_radius = init_brush_radius

    # Instantiate a window and register the mouse callback function
    cv.namedWindow('Free Drawing')
    cv.setMouseCallback('Free Drawing', mouse_event_handler, mouse_state)

    while True:
        # Draw a point if necessary
        mouse_left_button_click, mouse_xy = mouse_state
        if mouse_left_button_click:
            cv.circle(canvas, mouse_xy, brush_radius, palette[brush_color], -1)

        # Show the canvas
```

OpenCV High-level GUI: Handling Mouse Events

- Example) Free drawing

```
def free_drawing(canvas_width=640, canvas_height=480, init_brush_radius=3):
    ...
    while True:
        # Draw a point if necessary
        mouse_left_button_click, mouse_xy = mouse_state
        if mouse_left_button_click:
            cv.circle(canvas, mouse_xy, brush_radius, palette[brush_color], -1)

        # Show the canvas
        canvas_copy = canvas.copy()
        info = f'Brush Radius: {brush_radius}'
        cv.putText(canvas_copy, info, (10, 25), cv.FONT_HERSHEY_DUPLEX, 0.6, (127, 127, 127), thickness=2)
        cv.putText(canvas_copy, info, (10, 25), cv.FONT_HERSHEY_DUPLEX, 0.6, palette[brush_color])
        cv.imshow('Free Drawing', canvas_copy)

        # Process the key event
        key = cv.waitKey(1)
        if key == 27: # ESC
            break
        elif key == ord('\t'):
            brush_color = (brush_color + 1) % len(palette)
        elif key == ord('+') or key == ord('='):
            brush_radius += 1
        elif key == ord('-') or key == ord('_'):
            brush_radius = max(brush_radius - 1, 1)

    cv.destroyAllWindows()

if __name__ == '__main__':
    free_drawing()
```

Table of Contents

Image Editing

- Mission) Learn and implement common image editing techniques available in many applications

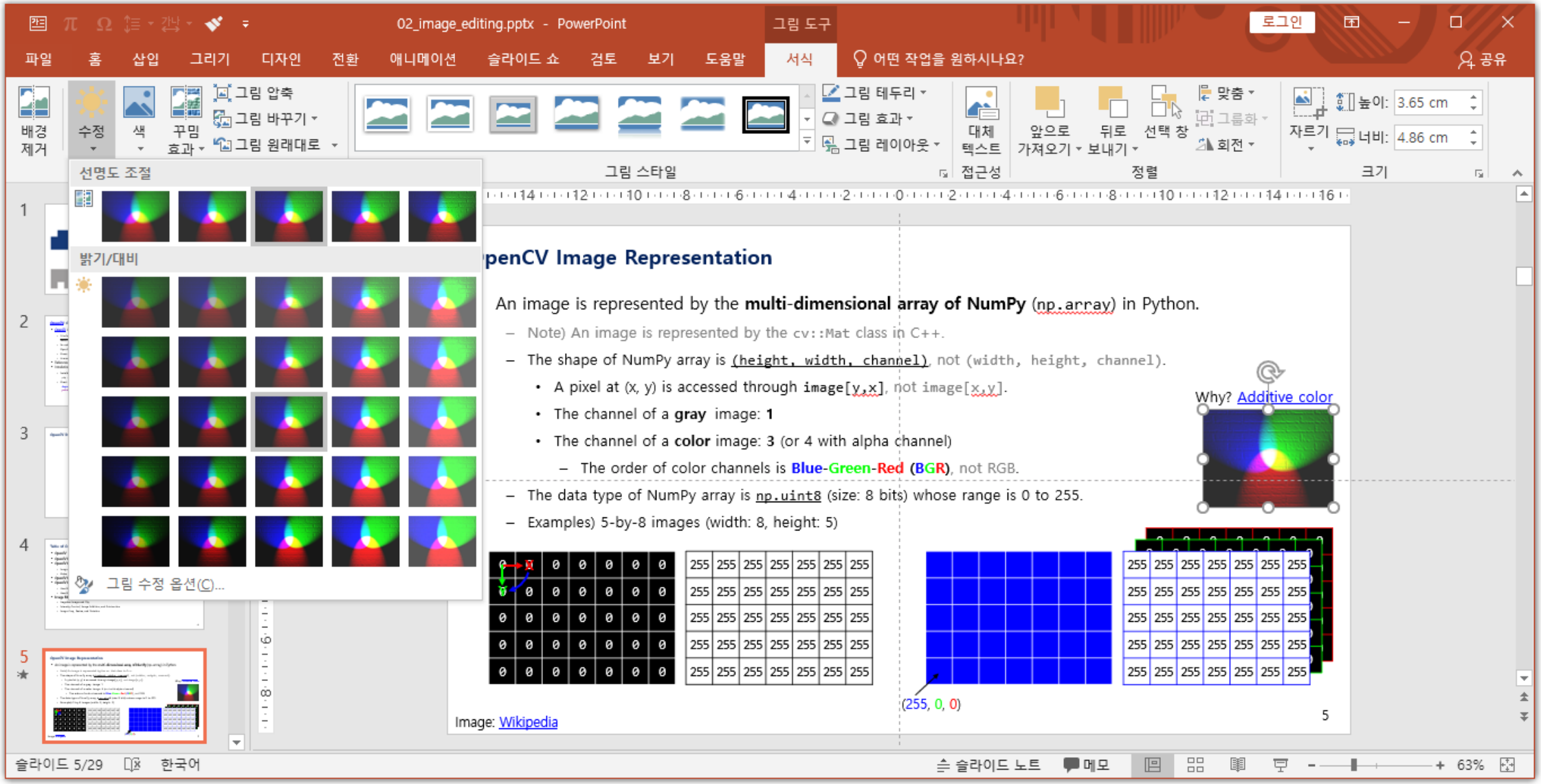
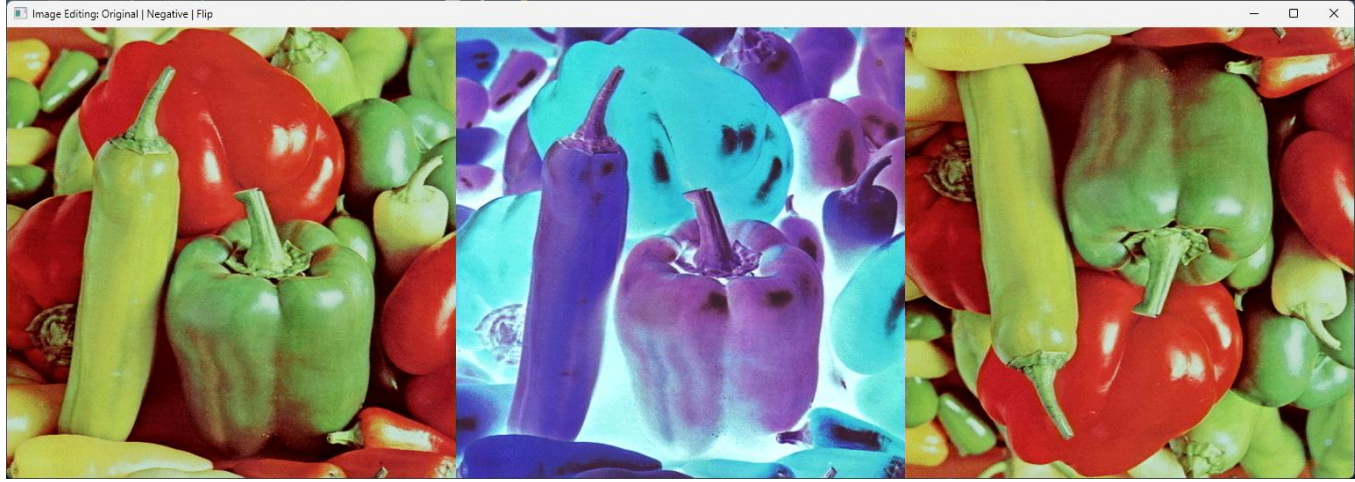


Image Editing: Negative Image and Flip

- Example) Negative image and vertical flip



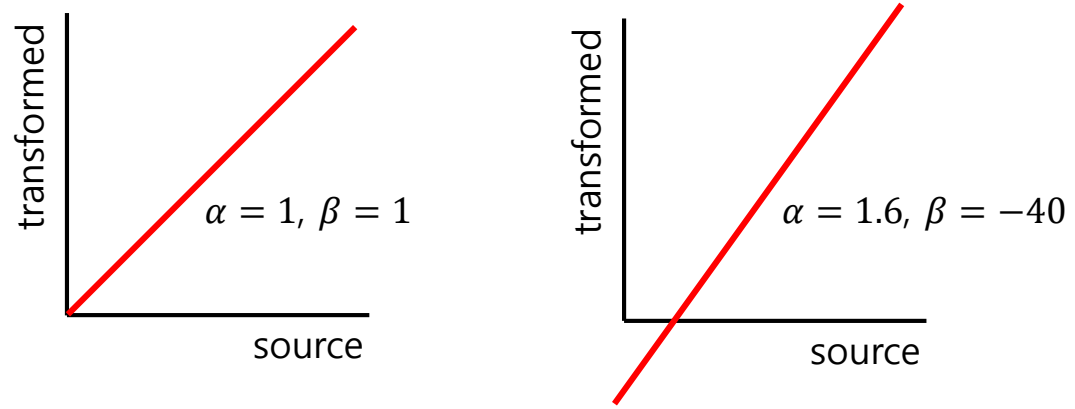
Photometric editing
(e.g. negative image)

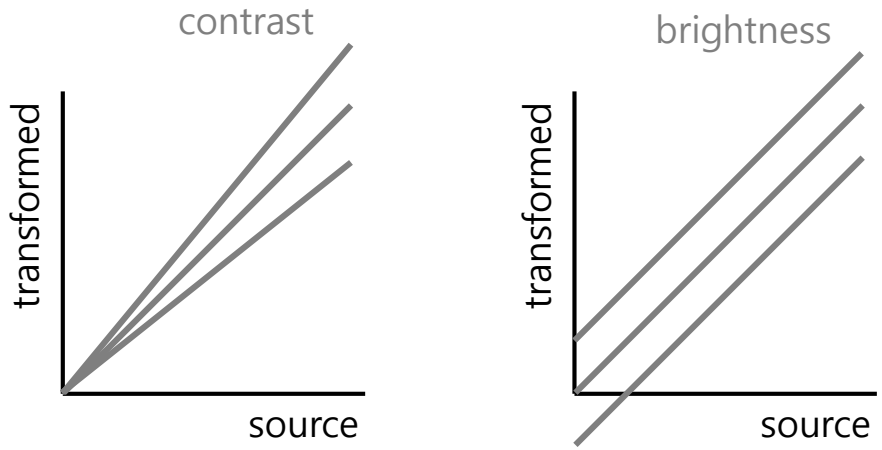
Geometric editing
(e.g. flipped image)

```
import cv2 as cv
import numpy as np

# Read the given image
img = cv.imread('data/peppers_color.tif')

if img is not None:
    # Get its negative image
    img_nega = 255 - img                          # Alternative) cv.bitwise_xor()

    # Get its vertically flipped image
    img_flip = img[::-1,:,:]                      # Alternative) cv.flip()

    # Show all images
    merge = np.hstack((img, img_nega, img_flip)) # Alternative) cv.hconcat()
    cv.imshow('Image Editing: Original | Negative | Flip', merge)
    cv.waitKey()
    cv.destroyAllWindows()
```

Image Editing: Intensity Transformation

- Example) Intensity transformation with contrast and brightness
  - *Contrast* (대비 in Korean) is the property that makes an object (or its representation in an image or display) distinguishable.
  - *Brightness* (명암 in Korean) is the strength of overall luminance.
  - A simple formulation: $I' = \alpha I + \beta$
    - α: contrast (slope)
    - β: brightness (Y intercept)





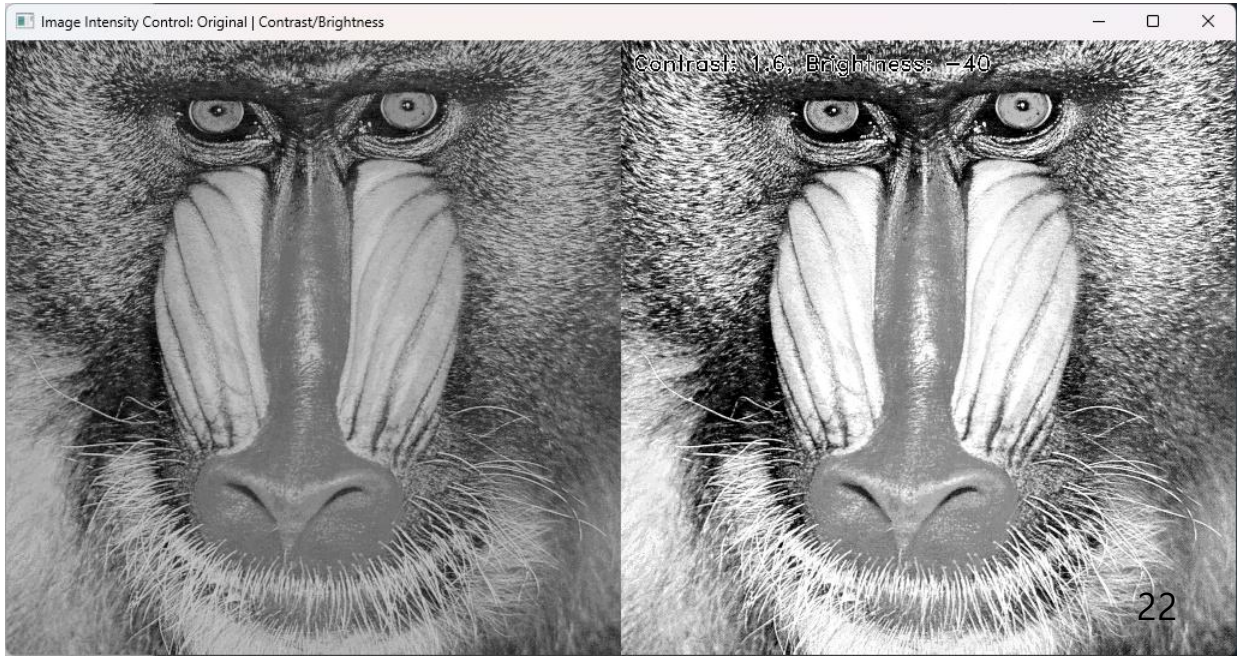


Image Editing: Intensity Transformation

- Example) Intensity transformation with contrast and brightness

```
# Read the given image as gray scale
img = cv.imread('data/mandril_color.tif', cv.IMREAD_GRAYSCALE)

if img is not None:
    # Configure the intensity control
    contrast = 1.6
    contrast_step = 0.1
    brightness = -40
    brightness_step = 1

    while True:
        # Apply contrast and brightness
        img_tran = contrast * img + brightness # Alternative) cv.equalizeHist(), cv.intensity_transform
        img_tran[img_tran < 0] = 0
        img_tran[img_tran > 255] = 255         # Saturate values
        img_tran = img_tran.astype(np.uint8)

        # Show all images
        ...

        # Process the key event
        key = cv.waitKey()
        if key == 27: # ESC
            break
        elif key == ord('+') or key == ord('='):
            contrast += contrast_step
        elif key == ord('-') or key == ord('_'):
            contrast -= contrast_step
        elif key == ord(']') or key == ord('}'):
            brightness += brightness_step
        elif key == ord('[') or key == ord('{'):
```

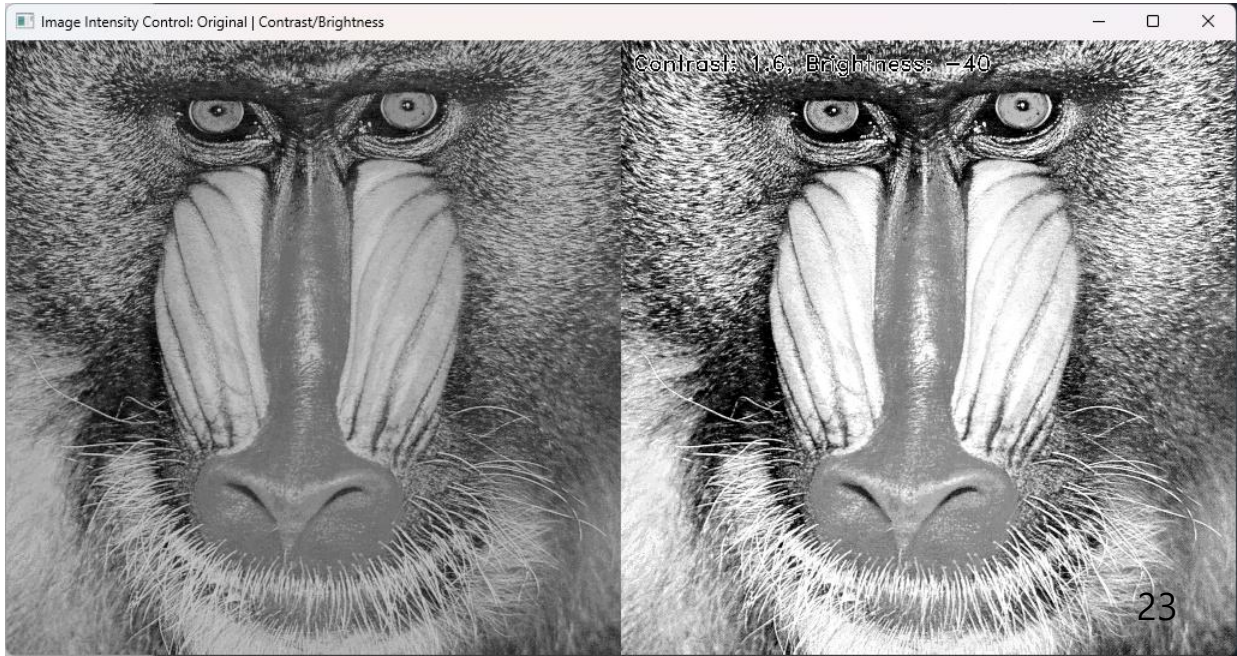


Image Editing: Image Addition

- Example) Alpha blending: $I_b = \alpha I_1 + (1 - \alpha) I_2$

```
# Read the given images
img1 = cv.imread('data/mandril_color.tif')
img2 = cv.imread('data/peppers_color.tif')

if img1 is not None and img2 is not None:
    alpha = 0.5

    while True:
        # Apply alpha blending
        blend = (alpha * img1 + (1 - alpha) * img2).astype(np.uint8) # Alternative) cv.addWeighted()

        # Show all images
        info = f'Alpha: {alpha:.1f}'
        cv.putText(blend, info, (10, 25), cv.FONT_HERSHEY_DUPLEX, 0.6, (255, 255, 255), thickness=2)
        cv.putText(blend, info, (10, 25), cv.FONT_HERSHEY_DUPLEX, 0.6, (0, 0, 0))
        merge = np.hstack((img1, img2, blend))
        cv.imshow('Image Blending: Image1 | Image2 | Blended', merge)

        # Process the key event
        key = cv.waitKey()
        if key == 27: # ESC
            break
        elif key == ord('+') or key == ord('='):
            alpha = min(alpha + 0.1, 1)
        elif key == ord('-') or key == ord('_'):
            alpha = max(alpha - 0.1, 0)

    cv.destroyAllWindows()
```

Transparent effect

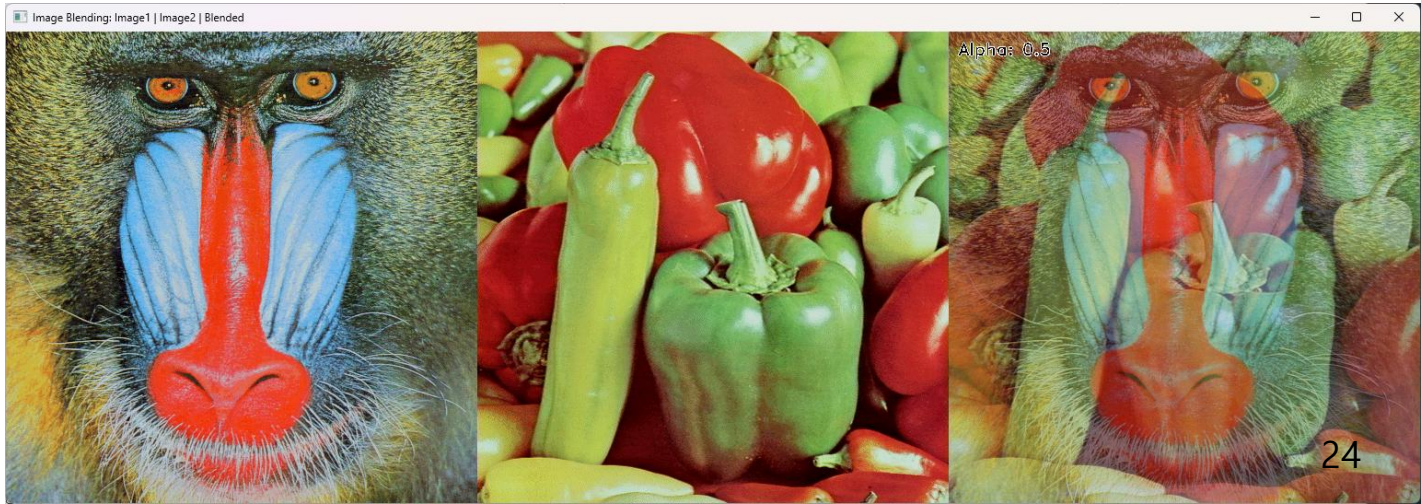


Image Editing: Image Addition

- Example) Background extraction: $\mathrm{I}_b = \frac{1}{N}\sum_{i=1}^{N} \mathrm{I}_i$

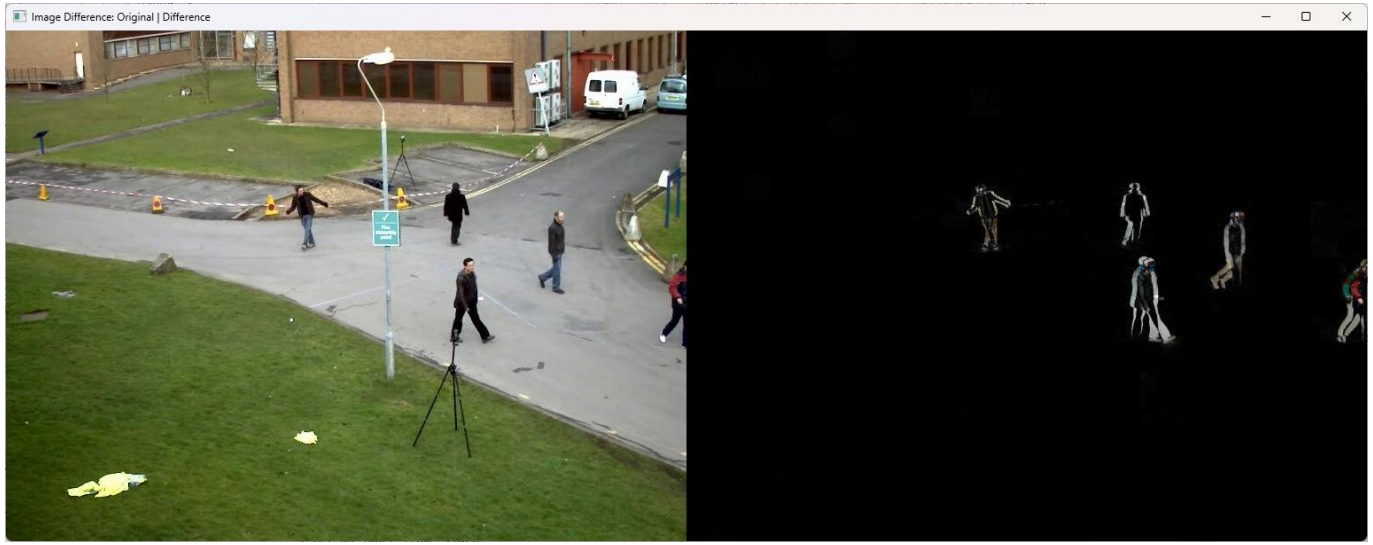
```
# Read the given video
video = cv.VideoCapture('../data/PETS09-S2L1-raw.webm')

frame_count = 0
img_back = None
while True:
    # Get an image from 'video'
    valid, img = video.read()
    if not valid:
        break
    frame_count += 1

    # Show progress
    if frame_count % 100 == 0:
        print(f'Frame: {frame_count}')

    # Add the image to the averaged image (the background image)
    # Alternative) cv.createBackgroundSubtractorMOG2(), cv::bgsegm
    if img_back is None:
        img_back = np.zeros_like(img, dtype=np.float64)
    img_back += img.astype(np.float64)
img_back = img_back / frame_count
img_back = img_back.astype(np.uint8)

# Save and show the background image
cv.imwrite('../data/PETS09-S2L1-raw_back.png', img_back)
cv.imshow('Background Extraction', img_back)
cv.waitKey()
cv.destroyAllWindows()
```

Averaging

Image Editing: Image Subtraction

- Example) Image difference: $I_d = |I_t - I_{t-1}|$

```
# Read the given video
video = cv.VideoCapture('data/PETS09-S2L1-raw.webm')

if video.isOpened():
    img_prev = None
    while True:
        # Get an image from 'video'
        valid, img = video.read()
        if not valid:
            break

        # Get the image difference
        if img_prev is None:
            img_prev = img.copy()
            continue
        img_diff = np.abs(img.astype(np.int32) - img_prev).astype(np.uint8) # Alternative) cv.absdiff()
        img_prev = img.copy()

        # Show all images
        merge = np.hstack((img, img_diff))
        cv.imshow('Image Difference: Original | Difference', merge)

        # Process the key event
        key = cv.waitKey(1)
        if key == ord(' '):
            key = cv.waitKey()
        if key == 27: # ESC
            break

cv.destroyAllWindows()
```



Change detection

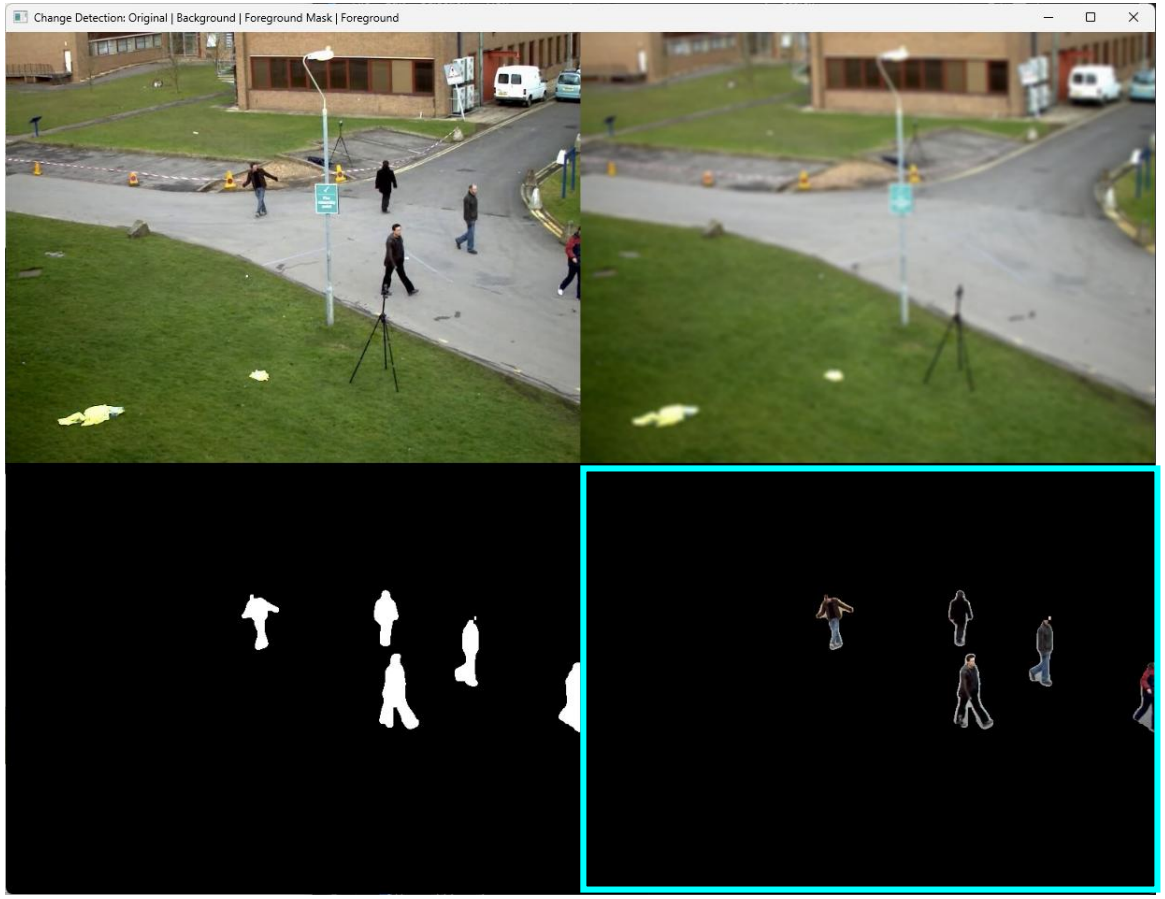
Image Editing: Image Subtraction

- Example) Background subtraction: $\mathrm{I}_d = \|\mathrm{I}_t - \mathrm{I}_b\|_2$

```python
# Read the given video
video = cv.VideoCapture('../data/PETS09-S2L1-raw.webm')

# Initialize control parameters
blur_ksize = (9, 9)
blur_sigma = 3
diff_threshold = 50
bg_update_rate = 0.05
fg_update_rate = 0.001
zoom_level = 0.8

# Read the background image
img_back = cv.imread('../data/PETS09-S2L1-raw_back.png')
img_back = cv.GaussianBlur(img_back, blur_ksize, blur_sigma).astype(

box = lambda ksize: np.ones((ksize, ksize), dtype=np.uint8)
while True:
    # Get an image from 'video'
    valid, img = video.read()
    if not valid:
        break

    # Get the difference between the current image and background
    img_blur = cv.GaussianBlur(img, blur_ksize, blur_sigma)
    img_diff = img_blur - img_back

    # Apply thresholding
    img_norm = np.linalg.norm(img_diff, axis=2)
    img_bin = np.zeros_like(img_norm, dtype=np.uint8)
    img_bin[img_norm > diff_threshold] = 255
```

Image Editing: Image Subtraction

- Example) Background subtraction: $\mathrm{I}_d = \|\mathrm{I}_t - \mathrm{I}_b\|_2$




```
# Initialize control parameters
...
bg_update_rate = 0.05
fg_update_rate = 0.001
...

while True:
    # Get the difference between the current image and background
    img_blur = cv.GaussianBlur(img, blur_ksize, blur_sigma)
    img_diff = img_blur - img_back

    # Apply thresholding
    img_norm = np.linalg.norm(img_diff, axis=2)
    img_bin = np.zeros_like(img_norm, dtype=np.uint8)
    img_bin[img_norm > diff_threshold] = 255

    # Apply morphological operations
    img_mask = img_bin.copy()
    ...
    fg = img_mask == 255                                # Keep the (thick) foreground mask

    # Update the background
    # Alternative) cv.createBackgroundSubtractorMOG2(), cv.bgsegm
    bg = ~fg
    img_back[bg] = (bg_update_rate * img_blur[bg] + (1 - bg_update_rate) * img_back[bg]) # With the higher weight
    img_back[fg] = (fg_update_rate * img_blur[fg] + (1 - fg_update_rate) * img_back[fg]) # With the lower weight

    # Get the foreground image
    img_fore = np.zeros_like(img)
    img_fore[fg] = img[fg]
```

Image Editing: Image Crop

- Example) Image file viewer with the zoom window

```
def mouse_event_handler(event, x, y, flags, param):
    # Catch the mouse position when it moves
    if event == cv.EVENT_MOUSEMOVE:
        param[0] = x # Note) Please do not use 'param = [x, y]'
        param[1] = y

def image_viewer(img_file, zoom_level=10, zoom_box_radius=5, zoom_box_margin=10):
    # Read the given image file
    img = cv.imread(img_file)
    if img is None:
        return False
    img_height, img_width, *_ = img.shape

    # Instantiate a window and register the mouse callback function
    cv.namedWindow('Image Viewer')
    mouse_xy = [-1, -1]
    cv.setMouseCallback('Image Viewer', mouse_event_handler, mouse_xy)

    while True:
        # Paste 'zoom_box' on 'img_copy'
        img_copy = img.copy()
        if mouse_xy[0] >= zoom_box_radius and mouse_xy[0] < (img_width  - zoom_box_radius) and \
           mouse_xy[1] >= zoom_box_radius and mouse_xy[1] < (img_height - zoom_box_radius):
            # Crop the target region
            img_crop = img[mouse_xy[1]-zoom_box_radius:mouse_xy[1]+zoom_box_radius, \
                           mouse_xy[0]-zoom_box_radius:mouse_xy[0]+zoom_box_radius, :]

            # Get the zoomed (resized) image
            zoom_box = cv.resize(img_crop, None, None, zoom_level, zoom_level)

            # Paste the zoomed image on 'img_copy'
```

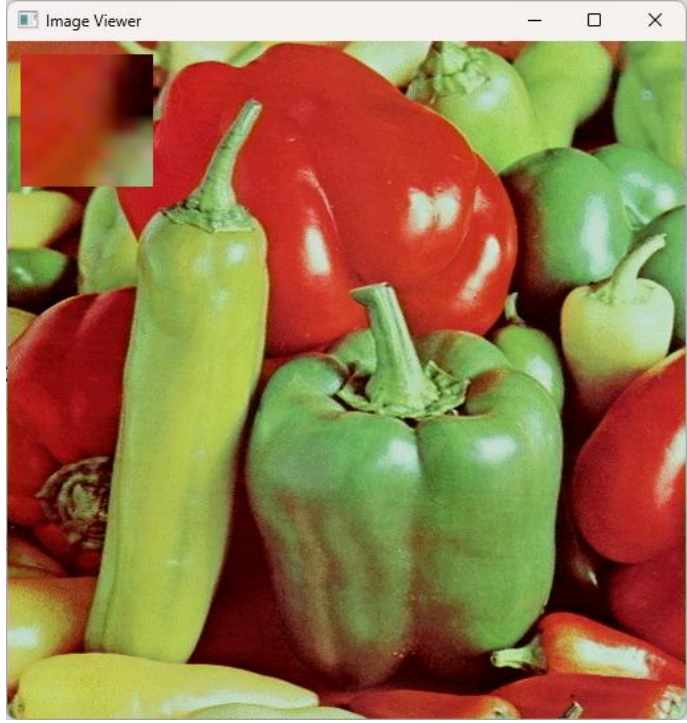


Image Editing: Image Crop

- Example) Image file viewer with the zoom window

```
def image_viewer(img_file, zoom_level=10, zoom_box_radius=5, zoom_box_margin=10):
    ...
    while True:
        # Paste 'zoom_box' on 'img_copy'
        img_copy = img.copy()
        if mouse_xy[0] >= zoom_box_radius and mouse_xy[0] < (img_width  - zoom_box_radius) and \
           mouse_xy[1] >= zoom_box_radius and mouse_xy[1] < (img_height - zoom_box_radius):
            # Crop the target region
            img_crop = img[mouse_xy[1]-zoom_box_radius:mouse_xy[1]+zoom_box_radius, \
                           mouse_xy[0]-zoom_box_radius:mouse_xy[0]+zoom_box_radius, :]

            # Get the zoomed (resized) image
            zoom_box = cv.resize(img_crop, None, None, zoom_level, zoom_level)

            # Paste the zoomed image on 'img_copy'
            s = zoom_box_margin
            e = zoom_box_margin + len(zoom_box)
            img_copy[s:e,s:e,:] = zoom_box

        # Show the image with the zoom
        cv.imshow('Image Viewer', img_copy)
        ...

    cv.destroyAllWindows()
    return True

if __name__ == '__main__':
    img_file = 'data/peppers_color.tif'
    if not image_viewer(img_file):
        print(f'Cannot open the given file, {img_file}.')
```

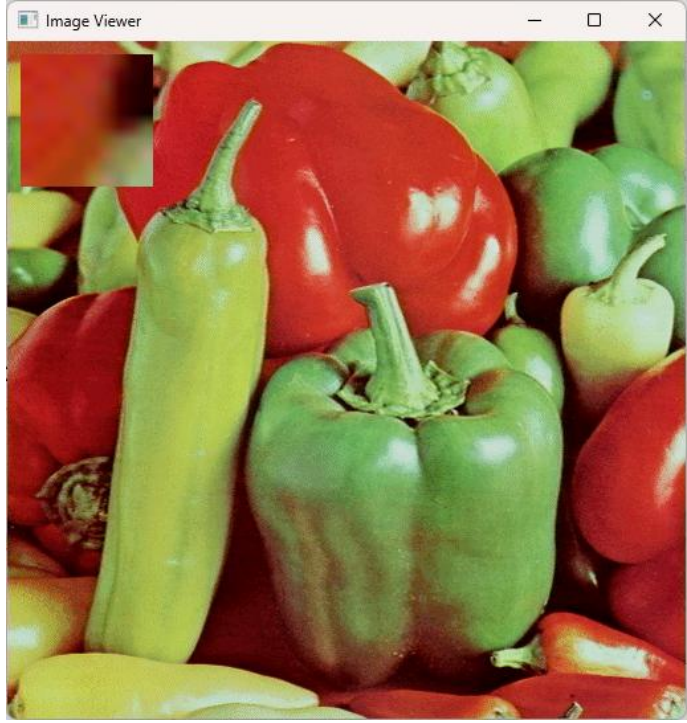
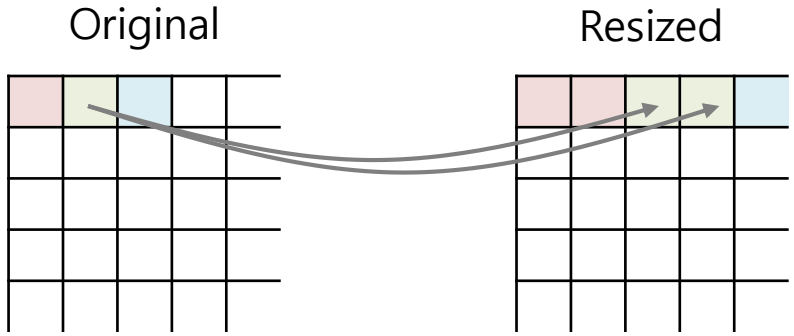


Image Editing: Image Resize

- Image resize
  - **Forward value copy**
    - For every (x, y) @ the original image:
      - Find its **all** (x_r, y_r) @ the resized image
      - For every (x_r, y_r) @ the resized image:
        - $\mathrm{I}_r(x_r, y_r) = \mathrm{I}(x, y)$
    - Issue) How about resize scale of 0.5 or 3.29? The target pixels are not clear!



  - **Backward value copy**
    - For every (x_r, y_r) @ the resized image:
      - Find its **single** (x, y) @ the original image
      - $\mathrm{I}_r(x_r, y_r) = \mathrm{I}(x, y)$
    - Issue) (x, y) can be real numbers, not integer numbers.
      - Solutions) The nearest pixel (rounding), (bi)linear **interpolation**, …

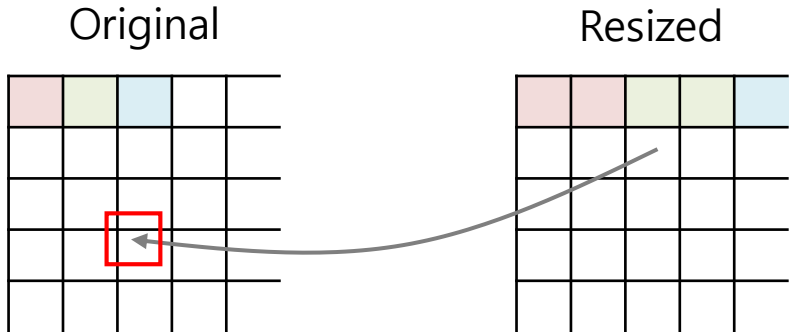
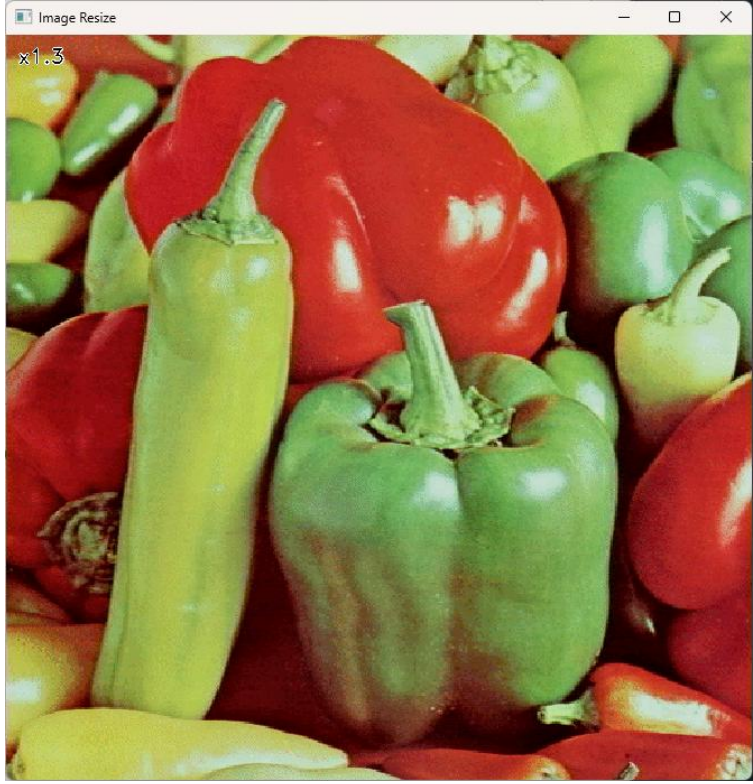


Image Editing: Image Resize



- Example) Image resize with backward value copy

```
def resize(img, scale):
    # Prepare the (empty) resized image
    img_shape = list(img.shape)
    img_shape[0] = int(img_shape[0] * scale)
    img_shape[1] = int(img_shape[1] * scale)
    img_resize = np.zeros(img_shape, dtype=np.uint8)

    # Copy each pixel from the given image
    for ry in range(img_shape[0]):
        y = ry / scale
        for rx in range(img_shape[1]):
            x = rx / scale
            img_resize[ry, rx, :] = img[int(y+0.5), int(x+0.5), :] # Note) Rounding: int(x+0.5)
    return img_resize
```

Discussion) Why is resizing process quite slow?

Image Editing: Image Rotation

- Image rotation
  - **Forward value copy**
    - For every (x, y) @ the original image:
      - Find its **rotated point** (x_r, y_r) @ the resized image
        - $\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \mathrm{R}(\theta) \begin{bmatrix} x \\ y \end{bmatrix}$ where $\mathrm{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
      - $\mathrm{I}_r(x_r, y_r) = \mathrm{I}(x, y)$
  - **Backward value copy**
    - For every (x_r, y_r) @ the resized image:
      - Find its **original point** (x, y) @ the original image
        - $\begin{bmatrix} x \\ y \end{bmatrix} = \mathrm{R}^{-1}(\theta) \begin{bmatrix} x_r \\ y_r \end{bmatrix}$ where $\mathrm{R}^{-1}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$
      - $\mathrm{I}_r(x_r, y_r) = \mathrm{I}(x, y)$

Discussion) What is better? Why?

Image Editing: Image Rotation

- Example) Image rotation with backward value copy

```
def rotate(img, degree):
    # Prepare the (empty) rotated image
    img_rotate = np.zeros(img.shape, dtype=np.uint8)

    # Prepare the inverse transformation
    c, s = np.cos(np.deg2rad(degree)), np.sin(np.deg2rad(degree))
    R = np.array([[c, -s], [s, c]]).T # Note) Transpose is the inverse.
    h, w, *_ = img.shape
    cx = (w - 1) / 2
    cy = (h - 1) / 2

    # Copy each pixel from the given image (backward mapping)
    for ry in range(h):
        for rx in range(w):
            dx, dy = R @ [rx - cx, ry - cy]
            x, y = int(dx + cx + 0.5), int(dy + cy + 0.5) # The nearest pixel
            if x >= 0 and y >= 0 and x < w and y < h:
                img_rotate[ry, rx, :] = img[y, x, :]
    return img_rotate

def rotate_forward(img, degree):
    # Prepare the (empty) rotated image
    img_rotate = np.zeros(img.shape, dtype=np.uint8)

    # Prepare the forward transformation
    c, s = np.cos(np.deg2rad(degree)), np.sin(np.deg2rad(degree))
```

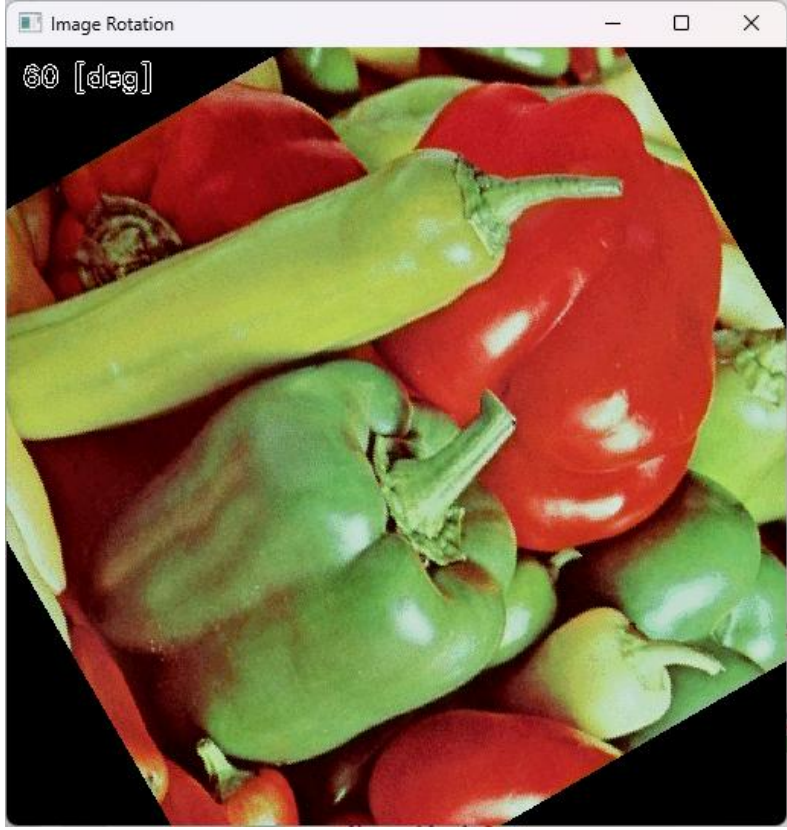
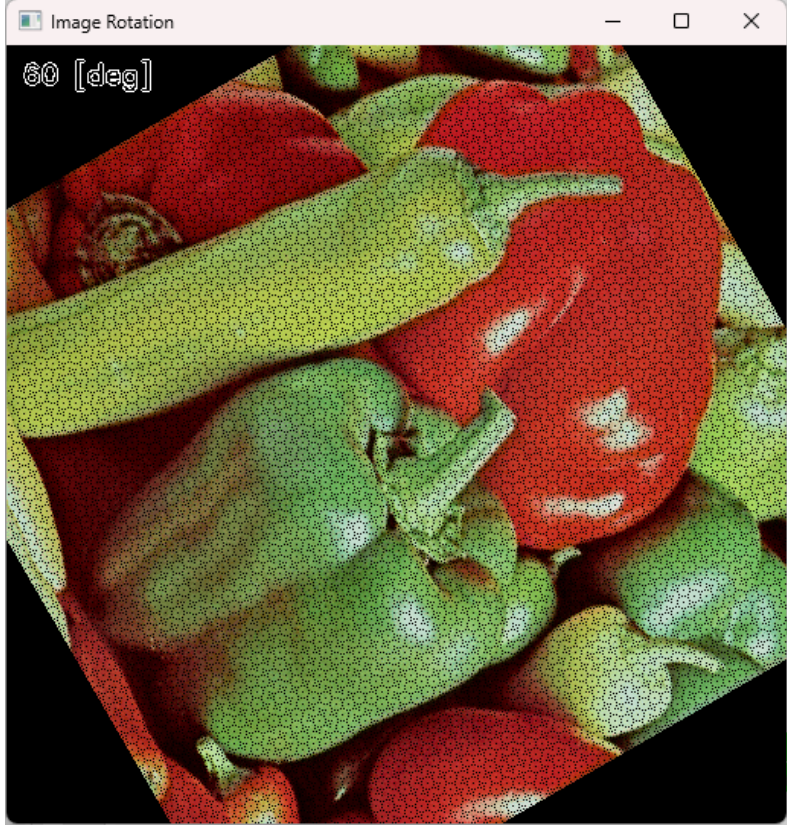


Image Editing: Image Rotation

- Example) Image rotation with backward/forward value copy

```
def rotate(img, degree):
    ...
    R = np.array([[c, -s], [s, c]]).T # Note) Transpose is the inverse.
    ...
    for ry in range(h):
        for rx in range(w):
            dx, dy = R @ [rx - cx, ry - cy]
            x, y = int(dx + cx + 0.5), int(dy + cy + 0.5) # The nearest pixel
            ...

def rotate_forward(img, degree):
    ...
    R = np.array([[c, -s], [s, c]])
    h, w, *_ = img.shape
    cx = (w - 1) / 2
    cy = (h - 1) / 2

    # Copy each pixel from the given image (forward mapping)
    for y in range(h):
        for x in range(w):
            dx, dy = R @ [x - cx, y - cy]
            rx, ry = int(dx + cx + 0.5), int(dy + cy + 0.5) # The nearest pixel
            if rx >= 0 and ry >= 0 and rx < w and ry < h:
                img_rotate[ry, rx, :] = img[y, x, :]
    return img_rotate
```



Discussion) Why does forward copy generate black dots?

Summary

- **OpenCV Image Representation**
  - Be careful
    - OpenCV: (x, y) vs. NumPy: [y, x]
    - OpenCV: (width, height) vs. NumPy: (height, width)
    - OpenCV: BGR vs. Others: RGB
- **OpenCV Image and Video Input/Output**
  - Image files / video files and cameras
- **OpenCV Drawing Functions**
  - Common errors (mistakes) of OpenCV in Python
    - Argument types: e.g. `list/tuple` or `np.array` or …, `np.int32` (`CV_32S`) or `np.float64` (`CV_64F`) or …
    - Argument shapes: e.g. 2D points `(n, 2)` or `(n, 1, 2)`
- **OpenCV High-level GUI**
  - Handling keyboard/mouse events
- **Image Editing**
  - Photometric: Negative image, intensity transformation, image addition, and subtraction
  - Geometric: Flip, image crop, resize, and rotation