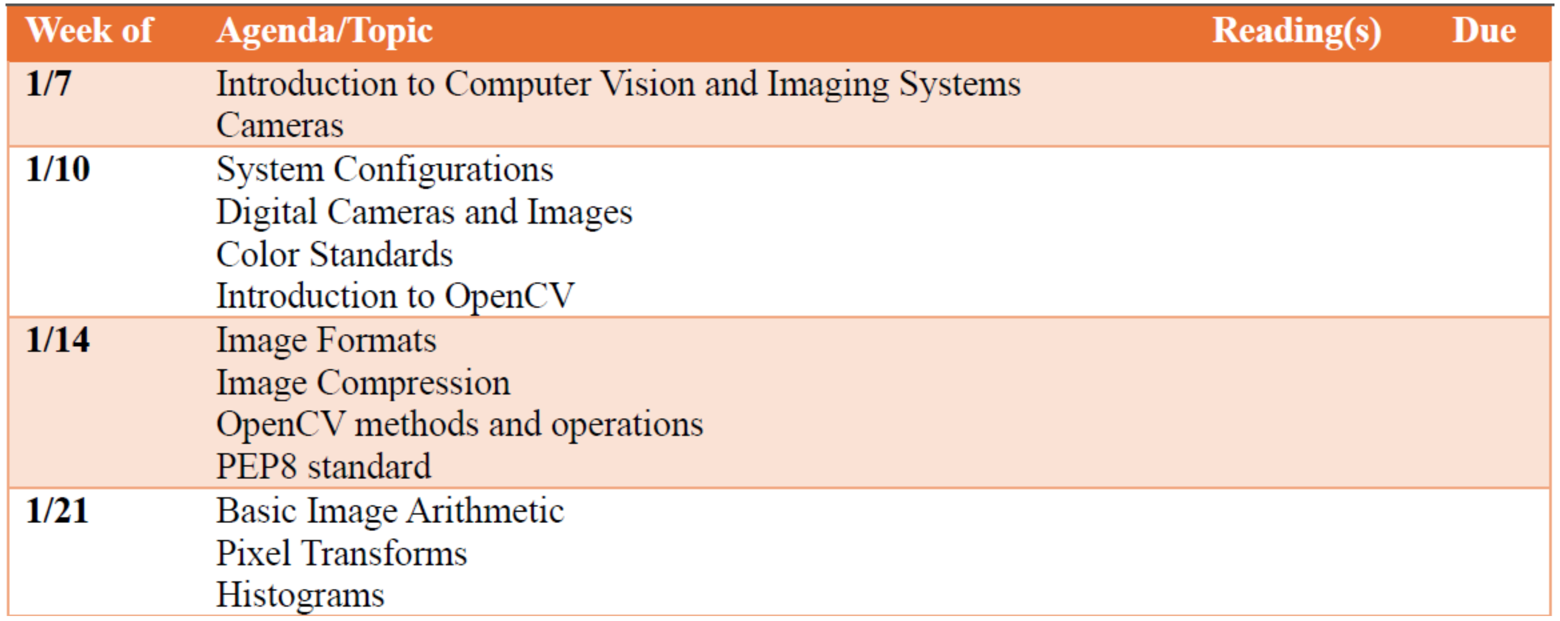
Computer Vision

CVI620

Session 4
01/2025

Review



Week of	Agenda/Topic	Reading(s)	Due
1/7	Introduction to Computer Vision and Imaging Systems Cameras		
1/10	System Configurations Digital Cameras and Images Color Standards Introduction to OpenCV		
1/14	Image Formats Image Compression OpenCV methods and operations PEP8 standard		
1/21	Basic Image Arithmetic Pixel Transforms Histograms		

Review

Min & Max

ROI

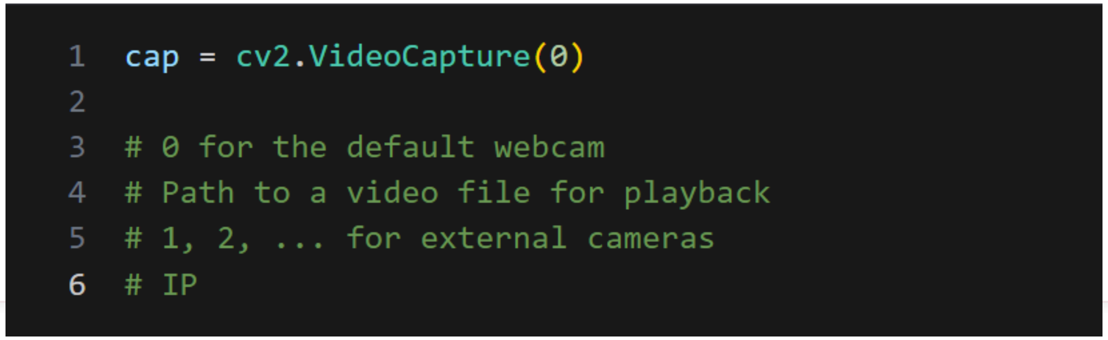
Split and Merge

Padding

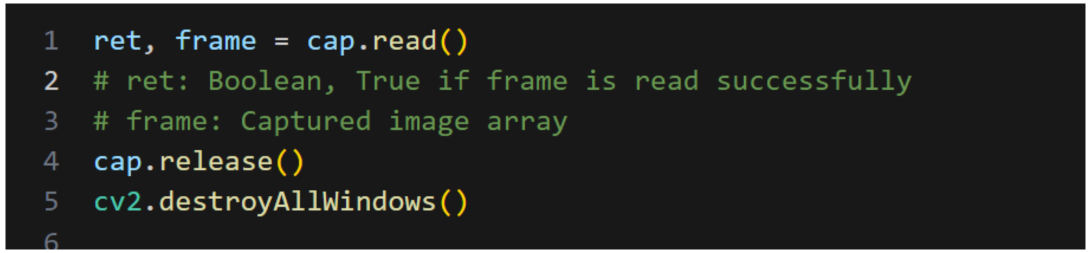
Video

- Videos are sequences of images
- A class to capture video streams from:
  - Webcam
  - Video files (e.g., .mp4, .avi)
  - IP cameras or other sources.

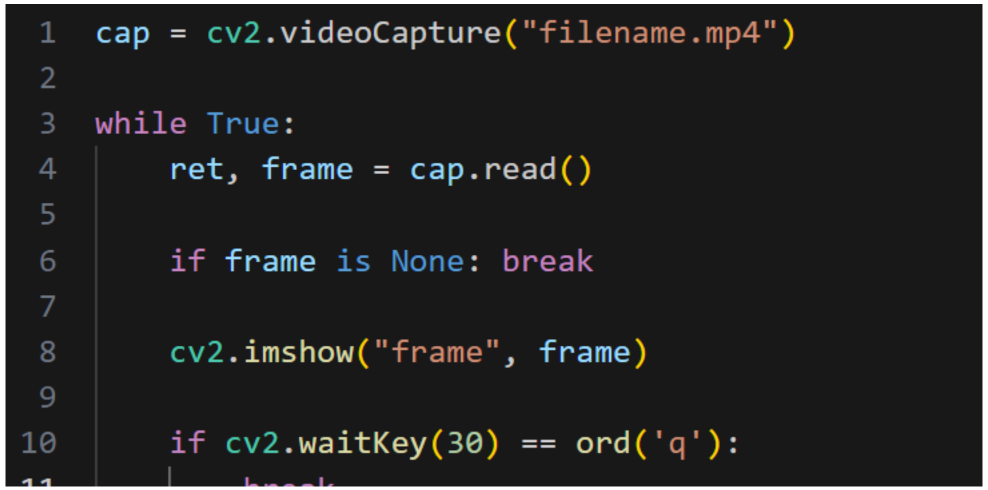
- object being created
- waitKey for speed



```python
cap = cv2.VideoCapture(0)

# 0 for the default webcam
# Path to a video file for playback
# 1, 2, ... for external cameras
# IP
```

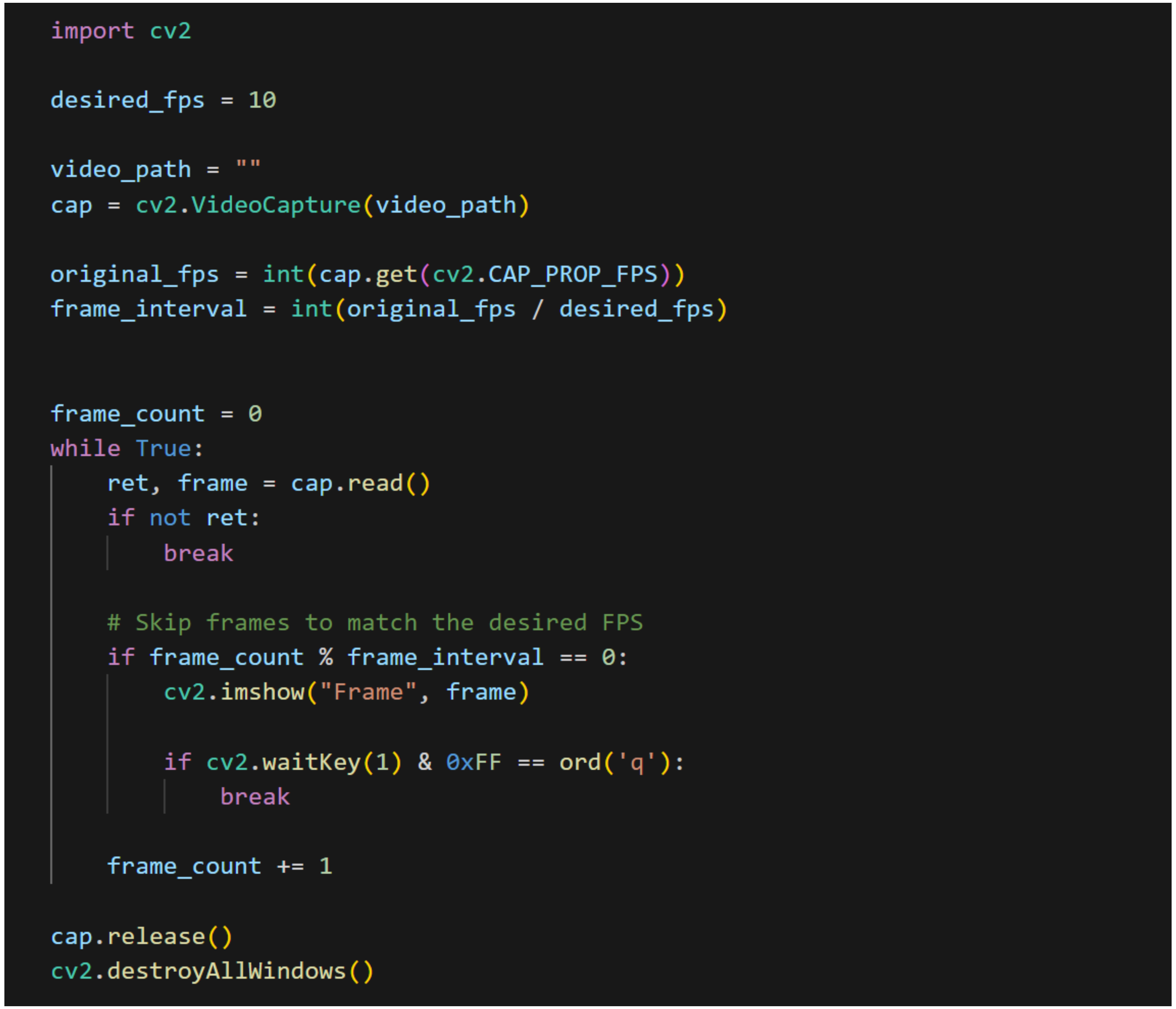


```python
ret, frame = cap.read()
# ret: Boolean, True if frame is read successfully
# frame: Captured image array
cap.release()
cv2.destroyAllWindows()
```



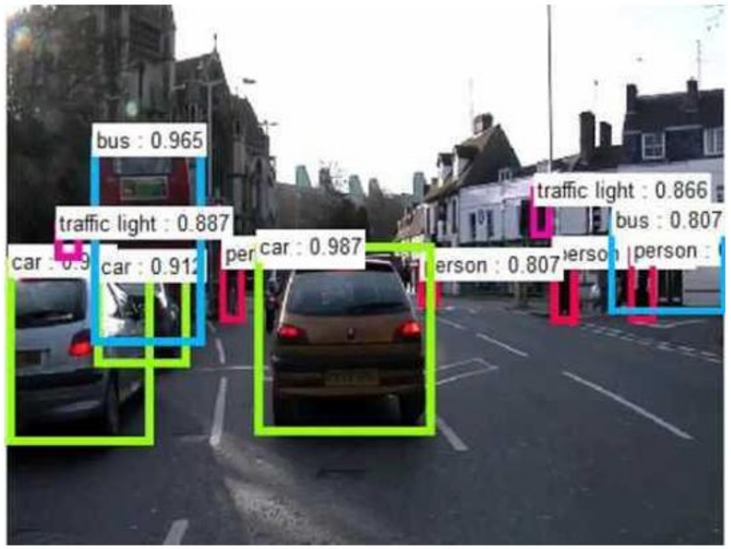
```python
cap = cv2.videoCapture("filename.mp4")

while True:
    ret, frame = cap.read()

    if frame is None: break

    cv2.imshow("frame", frame)

    if cv2.waitKey(30) == ord('q'):
        break
```

FPS



```python
import cv2

desired_fps = 10

video_path = ""
cap = cv2.VideoCapture(video_path)

original_fps = int(cap.get(cv2.CAP_PROP_FPS))
frame_interval = int(original_fps / desired_fps)


frame_count = 0
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Skip frames to match the desired FPS
    if frame_count % frame_interval == 0:
        cv2.imshow("Frame", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    frame_count += 1

cap.release()
cv2.destroyAllWindows()
```

Drawing Shapes

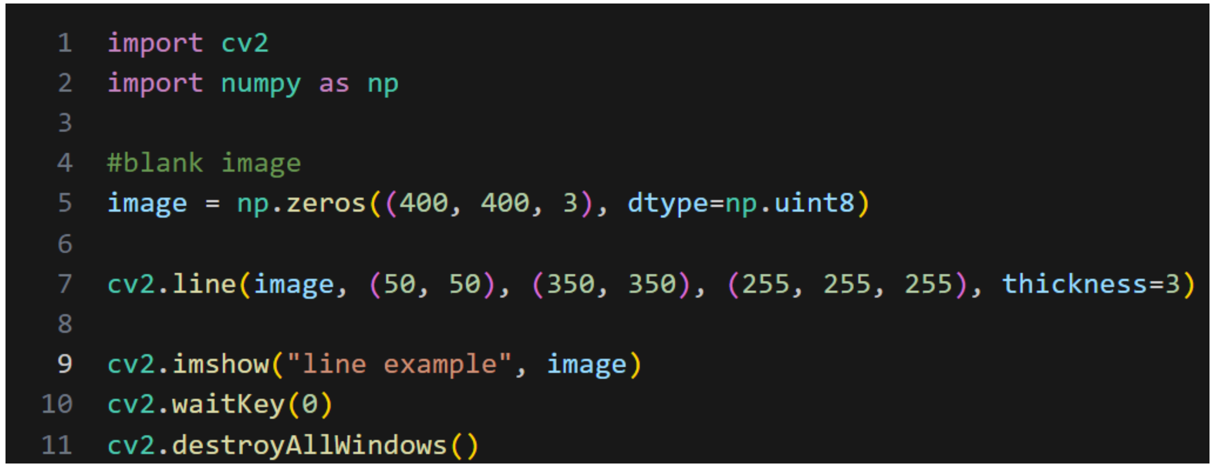


Line

- draw a straight line on an image

cv2.line(image, pt1, pt2, color, thickness)

image: Input image where the line will be drawn
pt1: starting point (x1, y1) (W, H)
pt2: ending point (x2, y2)
color: line color in BGR format (e.g., (255, 0, 0) for blue)
thickness: line thickness (integer)



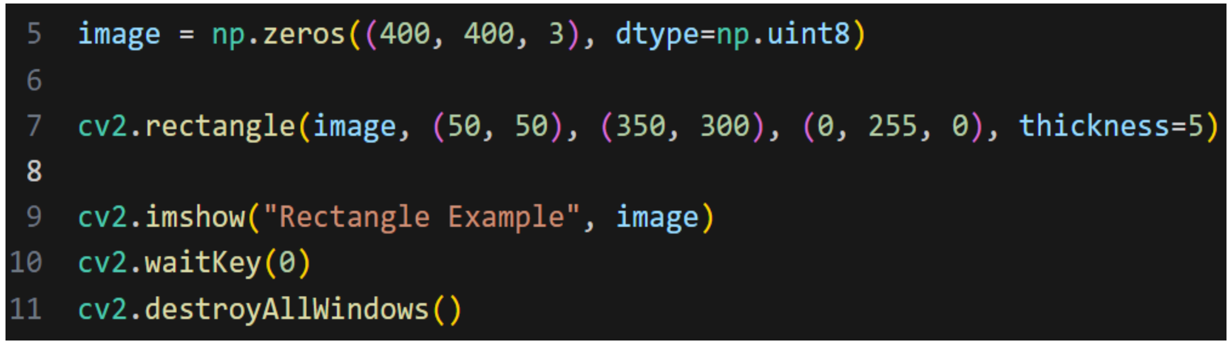
```python
import cv2
import numpy as np

#blank image
image = np.zeros((400, 400, 3), dtype=np.uint8)

cv2.line(image, (50, 50), (350, 350), (255, 255, 255), thickness=3)

cv2.imshow("line example", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Rectangle

- Draw rectangle on an image

cv2.rectangle(image, pt1, pt2, color, thickness)

image: Input image where the rectangle will be drawn.
pt1: Top-left corner (x1, y1).
pt2: Bottom-right corner (x2, y2).
color: Rectangle color in BGR format (e.g., (0, 255, 0) for green).
thickness: Border thickness (integer). Use -1 to fill the rectangle.



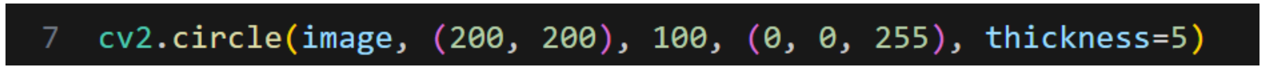
```
image = np.zeros((400, 400, 3), dtype=np.uint8)

cv2.rectangle(image, (50, 50), (350, 300), (0, 255, 0), thickness=5)

cv2.imshow("Rectangle Example", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Circle

- Draw circle

cv2.circle(image, center, radius, color, thickness)

image: input image where the circle will be drawn
center: center of the circle (x, y)
radius: radius of the circle (integer)
color: circle color in BGR format (e.g., (0, 0, 255) for red)
thickness: circle thickness (integer). Use -1 for a filled circle



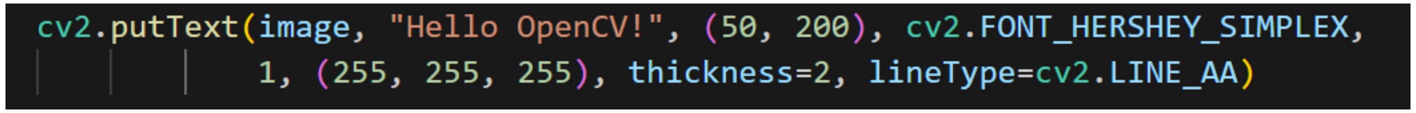
```python
cv2.circle(image, (200, 200), 100, (0, 0, 255), thickness=5)
```

Text

- Add text on an image

cv2.putText(image, text, org, font, font_scale, color, thickness, line_type)

image: Input image where text will be added.
text: The string to display.
org: Bottom-left corner of the text (x, y).
font: Font type (e.g., cv2.FONT_HERSHEY_SIMPLEX).
font_scale: Scale factor for font size.
color: Text color in BGR format (e.g., (255, 255, 255) for white).
thickness: Thickness of the text stroke.
line_type: Type of line for the text (e.g., cv2.LINE_AA).



```
cv2.putText(image, "Hello OpenCV!", (50, 200), cv2.FONT_HERSHEY_SIMPLEX,
            1, (255, 255, 255), thickness=2, lineType=cv2.LINE_AA)
```

More shapes

- cv2.line()
- cv2.rectangle()
- cv2.circle()
- cv2.ellipse()
- cv2.polylines()
- cv2.fillPoly()
- cv2.putText()
- cv2.arrowedLine()
- cv2.drawMarker()

Operators

Point operators

Neighborhood operators

Point Operators

- The value of each pixel in the output depends only on the value of the same pixel in the input (and possibly some global information or some parameters)
- Each pixel is processed independently.
- No influence from neighboring pixels.
- Example:
  - Contrast Adjustment: Stretching or compressing intensity values.
  - Thresholding: Converting grayscale images to binary by setting a threshold.
  - Brightness Adjustment: Adding or subtracting a constant to pixel values.

Pixel Transformers

Assuming one color channel for simplicity

Examples:

- Addition with a constant - Brightness adjustment
- Multiplication with a constant – Contrast adjustment

Addition and Subtraction

- Using arithmetic operations in Numpy does not work when values can go above 255 or lower than 0
- OpenCV's saturated arithmetic ensures that values above 255 are set to 255, and values below 0 are clamped to 0
- Use OpenCV functions instead

```
image = cv2.imread("Lucy.jpg")
img2 = cv2.add(image, np.ones(image.shape, dtype="uint8") *50)
img3 = cv2.subtract(image, np.ones(image.shape, dtype="uint8") *100)
```

Multiply and Divide

- Make bright colors brighter - use scale > 1

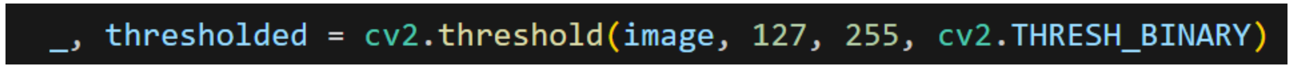
```
img2 = cv2.multiply(image, np.ones(image.shape, dtype="uint8"), scale=1.6)
```

- Make dark colors darker - use scale < 1

```
img2 = cv2.multiply(image, np.ones(image.shape, dtype="uint8"), scale=0.5)
```

Thresholding

- A technique to segment an image by converting it into binary form.
- Separates foreground (object) from background based on intensity values.



```
_, thresholded = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
```

- Object detection and segmentation.
- Preprocessing for OCR and feature extraction.

Linear Blend

weighted image addition

Two input images, img1 and img2

- $dst = \alpha . img1 + \beta . img2 + \gamma$
- img1 = cv2.imread("Trillium.jpg")
- img2 = cv2.imread("flower.jpg")
- img2 = cv2.resize(img2, (img1.shape[1], img1.shape[0]))
- img3 = cv2.addWeighted(img1, 0.6 , img2, 0.4, 0)

Neighborhood Operators

- The value of each pixel in the output depends on the value of the pixel and the value of its neighbors in the input
- Example: Smoothing or blurring
- Will continue the discussion in filtering

Color Pixel Extraction

- Feature extraction from images
- Edge or color detection
- A kind of masking (like segmentation)