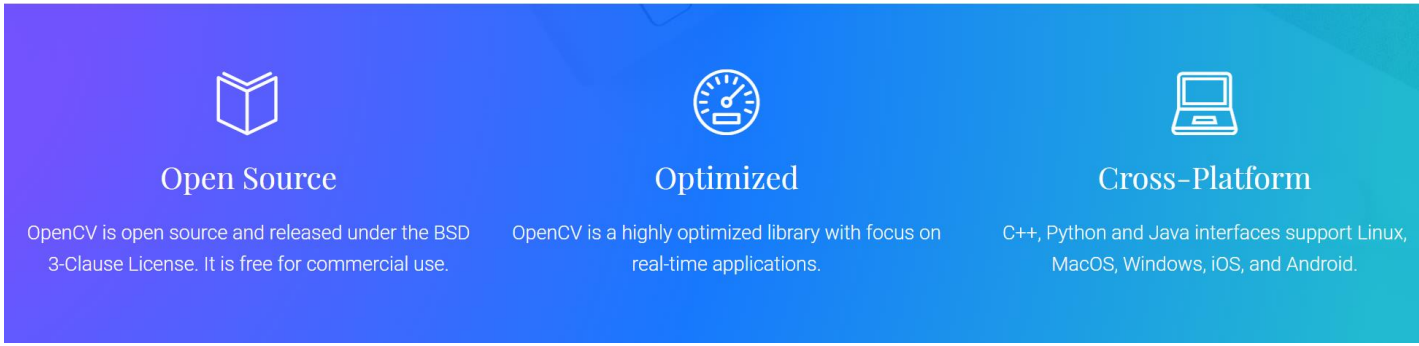
SIT32004
ICT Application Development

Image Processing and Class

Prof. Changbeom Choi

OpenCV[1]

- OpenCV is …
  - An open source computer vision and machine learning software library
  - OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
  - OpenCV was started at Intel in 1999
  - 2500+ optimized computer vision and machine learning algorithms



Requirements

- OpenCV-python:
  - **Unofficial** pre-built OpenCV packages for Python.
  - Version: 4.1.0.25

- NumPy
  - NumPy is the fundamental package for scientific computing with Python
  - Version: 1.16.3

- Matplotlib
  - Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms
  - Version: 3.0.3

Images

- Read images
  - Use function cv2.imread()v

```
retval=cv.imread(filename[, flags])
```

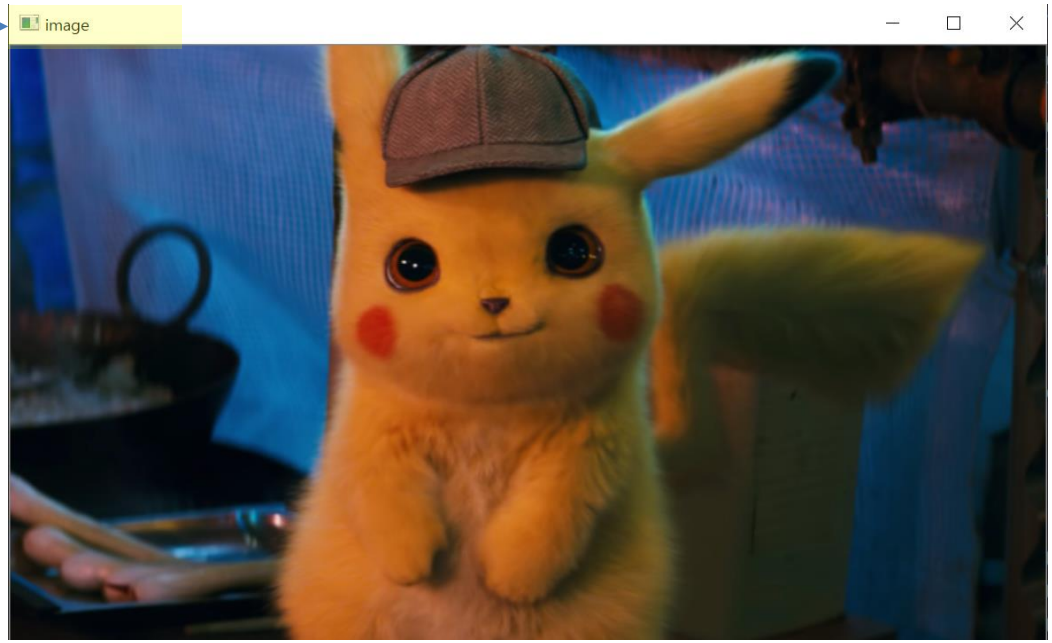
Enumerator	
IMREAD_UNCHANGED Python: cv.IMREAD_UNCHANGED	If set, return the loaded image as is (with alpha channel, otherwise it gets cropped).
IMREAD_GRAYSCALE Python: cv.IMREAD_GRAYSCALE	If set, always convert image to the single channel grayscale image (codec internal conversion).
IMREAD_COLOR Python: cv.IMREAD_COLOR	If set, always convert image to the 3 channel BGR color image.
IMREAD_ANYDEPTH Python: cv.IMREAD_ANYDEPTH	If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.
IMREAD_ANYCOLOR Python: cv.IMREAD_ANYCOLOR	If set, the image is read in any possible color format.
IMREAD_LOAD_GDAL Python: cv.IMREAD_LOAD_GDAL	If set, use the gdal driver for loading the image.
IMREAD_REDUCED_GRAYSCALE_2 Python: cv.IMREAD_REDUCED_GRAYSCALE_2	If set, always convert image to the single channel grayscale image and the image size reduced 1/2.
IMREAD_REDUCED_COLOR_2 Python: cv.IMREAD_REDUCED_COLOR_2	If set, always convert image to the 3 channel BGR color image and the image size reduced 1/2.
IMREAD_REDUCED_GRAYSCALE_4 Python: cv.IMREAD_REDUCED_GRAYSCALE_4	If set, always convert image to the single channel grayscale image and the image size reduced 1/4.
IMREAD_REDUCED_COLOR_4 Python: cv.IMREAD_REDUCED_COLOR_4	If set, always convert image to the 3 channel BGR color image and the image size reduced 1/4.
IMREAD_REDUCED_GRAYSCALE_8 Python: cv.IMREAD_REDUCED_GRAYSCALE_8	If set, always convert image to the single channel grayscale image and the image size reduced 1/8.
IMREAD_REDUCED_COLOR_8 Python: cv.IMREAD_REDUCED_COLOR_8	If set, always convert image to the 3 channel BGR color image and the image size reduced 1/8.
IMREAD_IGNORE_ORIENTATION Python: cv.IMREAD_IGNORE_ORIENTATION	If set, do not rotate the image according to EXIF's orientation flag.

Images Example

```
01: import cv2
02:
03: img = cv2.imread('pikachu1.png')
04:
05: cv2.imshow('image',img)
06: cv2.waitKey(0)# wait
07: cv2.destroyAllWindows()
```

* **cv2.waitKey()** is a keyboard binding function
If **0** is passed, it waits indefinitely for a key stroke.

cv2.destroyAllWindows() simply destroys all the windows we created.



Images Example

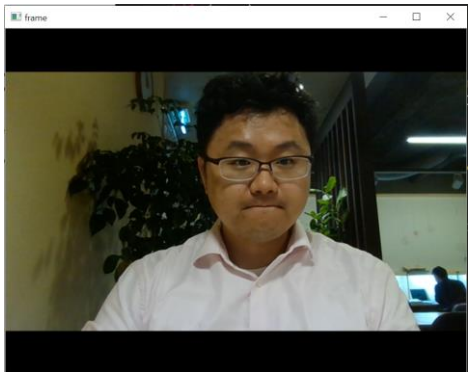
- Use the function **cv2.imwrite()** to save an image.

```
01: import cv2
02:
03: import cv2
04:
05: img = cv2.imread('pikachu1.png', cv2.IMREAD_GRAYSCALE)
06:
07: cv2.imshow('image',img)
08:
09: cv2.imwrite('pikachu1_gray.jpg', img)
10: cv2.waitKey(0)
11: cv2.destroyAllWindows()
```

Video

- OpenCV provides a very simple interface to capture livestream with camera
- VideoCapture object
  - Argument: device index or the name of a video file

```
01: import numpy as np
02: import cv2
03:
04: cap = cv2.VideoCapture(0)
05:
06: while(True):
07:     # Capture frame-by-frame
08:     ret, frame = cap.read()
09:
10:     # Our operations on the frame come here
11:     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2BGRA)
12:
13:     # Display the resulting frame
14:     cv2.imshow('frame',gray)
15:     if cv2.waitKey(1) & 0xFF == ord('q'):
16:         break
17:
18: # When everything done, release the capture
19: cap.release()
20: cv2.destroyAllWindows()
```



Video Capture

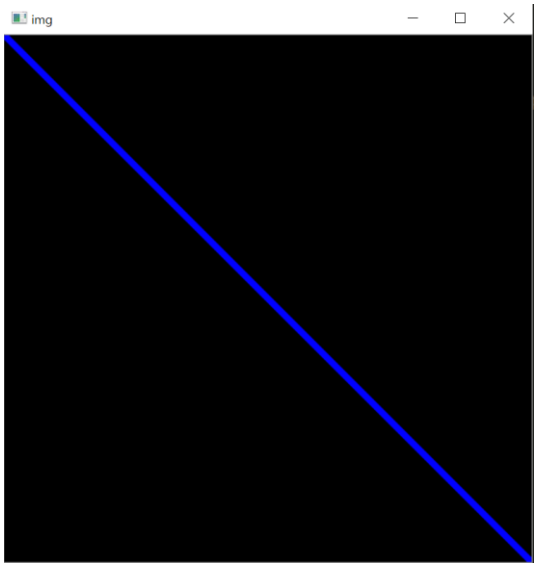
- Use VideoWriter object
  - You should specify the output filename and the CODEC(FourCC)
  - FourCC is a 4-byte code used to specify the video codec

```
01: import numpy as np
02: import cv2
03:
04: cap = cv2.VideoCapture(0)
05:
06: # Define the codec and create VideoWriter object
07: fourcc = cv2.VideoWriter_fourcc(*'XVID')
08: out = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))
09:
10: while(cap.isOpened()):
11:     ret, frame = cap.read()
12:     if ret==True:
13:         frame = cv2.flip(frame,0) # flip the video source
14:
15:         # write the flipped frame
16:         out.write(frame)
17:
18:         cv2.imshow('frame',frame)
19:         if cv2.waitKey(1) & 0xFF == ord('q'):
20:             break
21:     else:
22:         break
23:
24: # Release everything if job is finished
25: cap.release()
26: out.release()
27: cv2.destroyAllWindows()
```

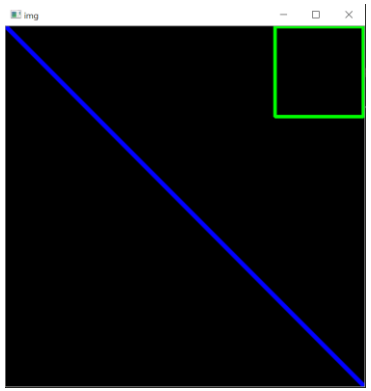
Drawing Functions in OpenCV

- **Drawing Line**

```
01: import numpy as np
02: import cv2
03:
04: # Create a black image
05: img = np.zeros((512,512,3), np.uint8)
06:
07: # Draw a diagonal blue line with
thickness of 5 px
08: img =
cv2.line(img,(0,0),(511,511),(255,0,0),5)
```



- **Drawing Rectangle**
  - `img = cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)`



Mouse Event Handling in OpenCV

- Register callback functions
  - cv2.setMouseCallback(*window_name, callback_fn, param*)
    - » window_name: window that cv2 uses
      - Ex) cv2.namedWindow('frame’)
    - » callback_fn: callback function
      - Callback function’s argument should be (event,x,y,flags,param)
      - Event list
        - 'EVENT_FLAG_ALTKEY', 'EVENT_FLAG_CTRLKEY', 'EVENT_FLAG_LBUTTON', 'EVENT_FLAG_MBUTTON', 'EVENT_FLAG_RBUTTON', 'EVENT_FLAG_SHIFTKEY', 'EVENT_LBUTTONDBLCLK', 'EVENT_LBUTTONDOWN', 'EVENT_LBUTTONUP', 'EVENT_MBUTTONDBLCLK', 'EVENT_MBUTTONDOWN', 'EVENT_MBUTTONUP', 'EVENT_MOUSEHWHEEL', 'EVENT_MOUSEMOVE', 'EVENT_MOUSEWHEEL', 'EVENT_RBUTTONDBLCLK', 'EVENT_RBUTTONDOWN', 'EVENT_RBUTTONUP'
    - » param: additional parameters which you may pass

Mouse Event Handling in OpenCV

- Register callback functions
  - cv2.setMouseCallback(*window_name, callback_fn, param*)

```
01: import numpy as np
02: import cv2
03:
04: def draw_circle(event,x,y,flags,param):
05:     if event == cv2.EVENT_LBUTTONDOWN:
06:             cv2.circle(img,(x,y), 100,(0,255,255),-1)
07:             cv2.imshow('img', img)
08:             print(x, y)
09:
10: # Create a black image
11: img = np.zeros((768,1024,3), np.uint8)
12:
13: cv2.namedWindow('img')
14: cv2.setMouseCallback('img', draw_circle, None)
15:
16: cv2.imshow('img',img)
17: cv2.waitKey(0)
```

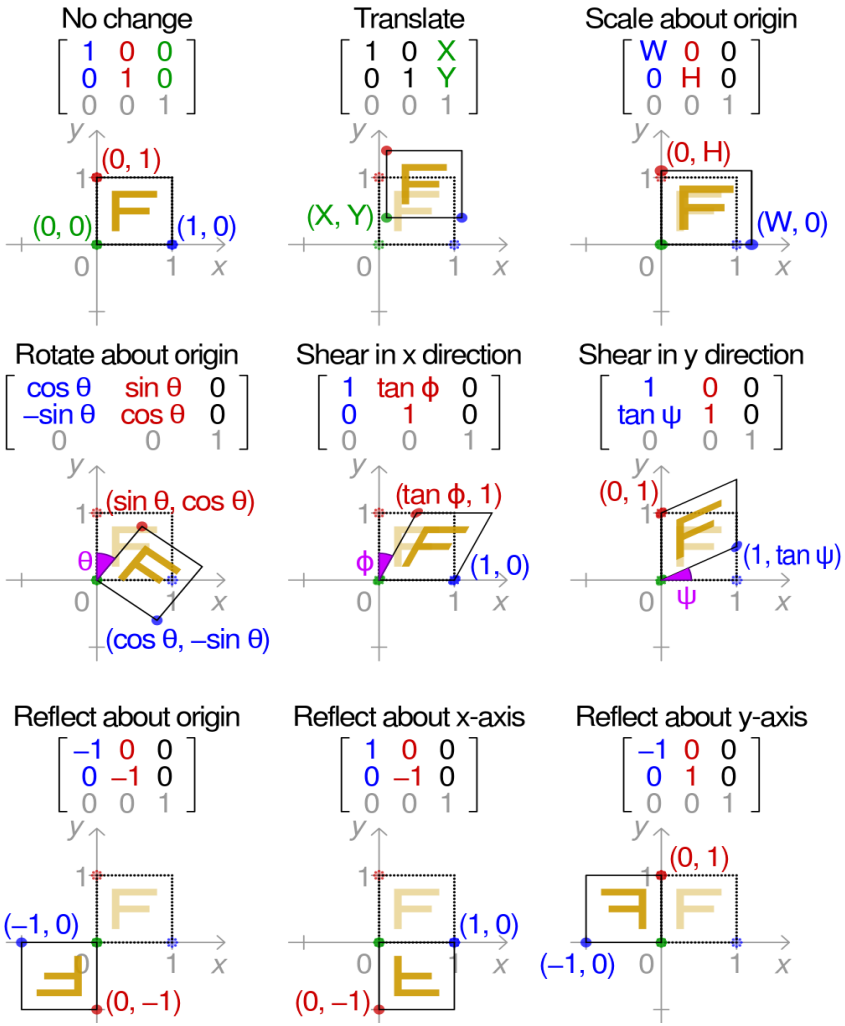
Image Processing

- **Scaling**
  - Scaling is just resizing of the image
  - OpenCV provides **cv2.resize()**
    - » The size of the image can be specified manually, or you can specify the scaling factor.
    - » Different interpolation methods may be used
      - **cv2.INTER_AREA:** Suitable for shrinking
      - **cv2.INTER_LINEAR**: Suitable for zooming
      - **cv2.INTER_CUBIC:** Suitable for zooming but slow

```
01: import cv2
02: import numpy as np
03:
04: img = cv2.imread('pikachu1.png')
05:
06: res = cv2.resize(img,None,fx=2, fy=2, interpolation = cv2.INTER_CUBIC)
07: cv2.imshow('zoom', res)
08:
09: res = cv2.resize(img,None,fx=0.5, fy=0.5, interpolation = cv2.INTER_AREA)
10: cv2.imshow('shrink', res)
11:
12: cv2.waitKey(0)
```

Image Processing

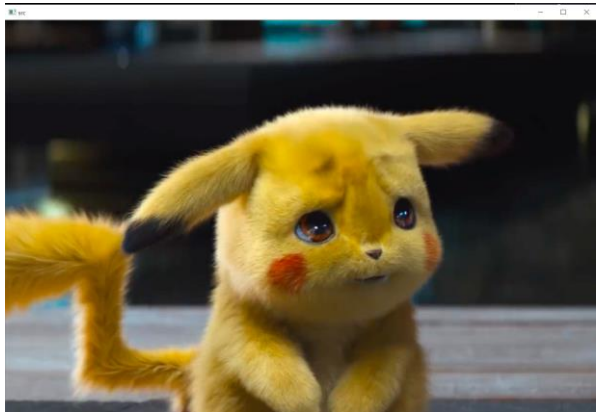
- **Transformation Matrix[2]**



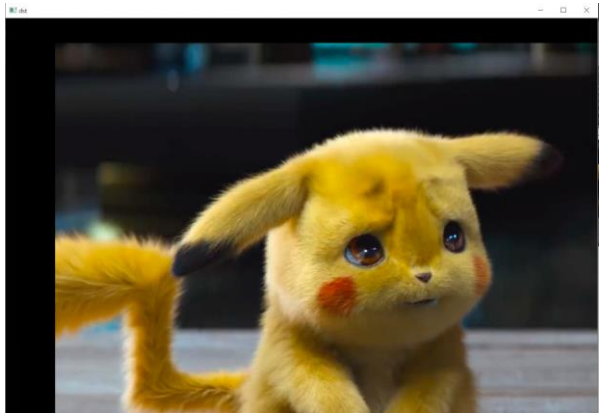
- **Translation**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

```
01: import cv2
02: import numpy as np
03:
04: img = cv2.imread('pikachu2.jpg', cv2.IMREAD_COLOR)
05: print(img.shape)
06: rows,cols,plane = img.shape
07:
08: cv2.imshow('src',img)
09:
10: M = np.float32([[1,0,100],[0,1,50]])
11: dst = cv2.warpAffine(img,M,(cols,rows))
12:
13: cv2.imshow('dst',dst)
14: cv2.waitKey(0)
15: cv2.destroyAllWindows()
```



Before



After

Image Processing

- **Rotation**
  - Rotation of an image for an angle \theta is achieved by the transformation matrix of the form

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - OpenCV provides scaled rotation with adjustable center of rotation so that you can rotate at any location you prefer. Modified transformation matrix is given by

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

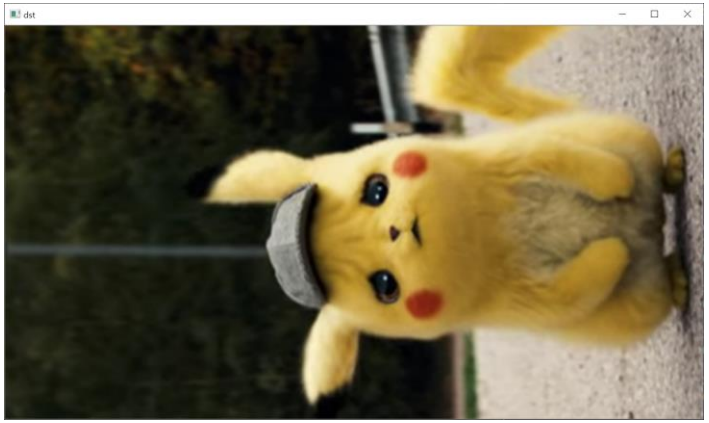
Image Processing: Rotation

```
01: import cv2
02:
03: img = cv2.imread('pikachu3.jpg', cv2.IMREAD_COLOR)
04: rows,cols,plane = img.shape
05:
06: M = cv2.getRotationMatrix2D((cols/2,rows/2),90,2)
07: dst = cv2.warpAffine(img,M,(cols,rows))
08:
09: cv2.imshow('src', img)
10: cv2.imshow('dst', dst)
11:
12: cv2.waitKey(0)
13: cv2.destroyAllWindows()
```

scale

Center points

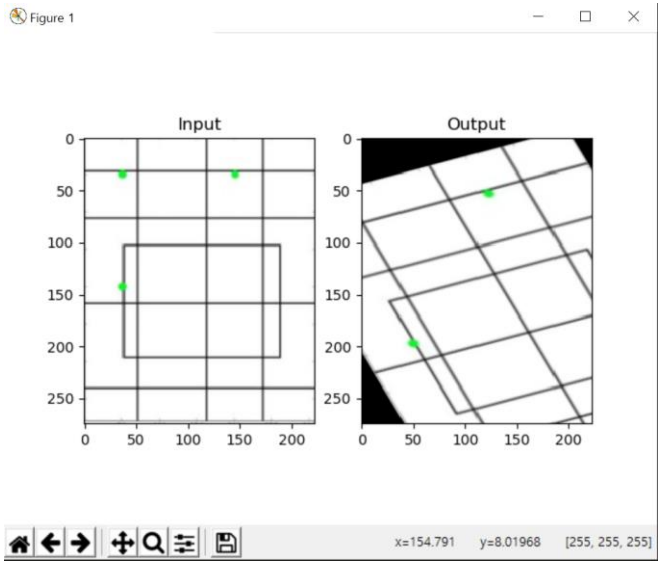
Before



After

- **Affine Transformation for OpenCV**

```
01: import cv2
02: import numpy as np
03: import matplotlib.pyplot as plt
04:
05: img = cv2.imread('drawing.png')
06: rows,cols,ch = img.shape
07:
08: pts1 = np.float32([[50,50],[200,50],[50,200]])
09: pts2 = np.float32([[10,100],[200,50],[100,250]])
10:
11: M = cv2.getAffineTransform(pts1,pts2)
12:
13: dst = cv2.warpAffine(img,M,(cols,rows))
14:
15: plt.subplot(121),plt.imshow(img),plt.title('Input')
16: plt.subplot(122),plt.imshow(dst),plt.title('Output')
17: plt.show()
```



- Perspective Transformation

```
01: import cv2
02: import numpy as np
03: import matplotlib.pyplot as plt
04:
05: img = cv2.imread('sudokusmall.png')
06: rows,cols,ch = img.shape
07:
08: pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
09: pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]])
10:
11: M = cv2.getPerspectiveTransform(pts1,pts2)
12:
13: dst = cv2.warpPerspective(img,M,(300,300))
14:
15: plt.subplot(121),plt.imshow(img),plt.title('Input')
16: plt.subplot(122),plt.imshow(dst),plt.title('Output')
17: plt.show()
```

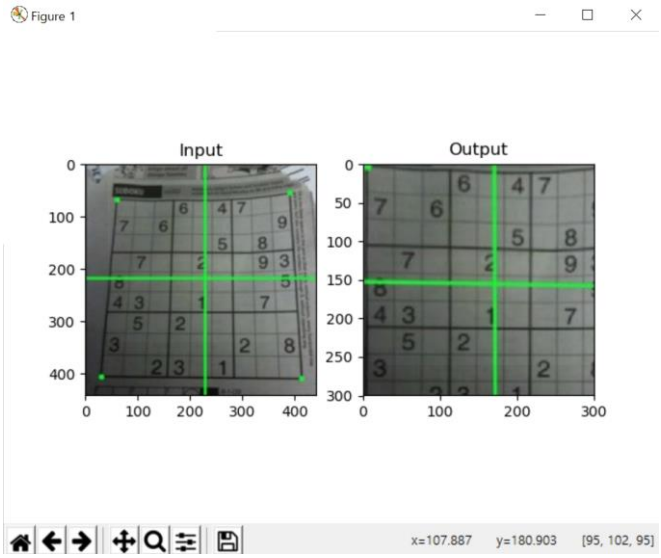
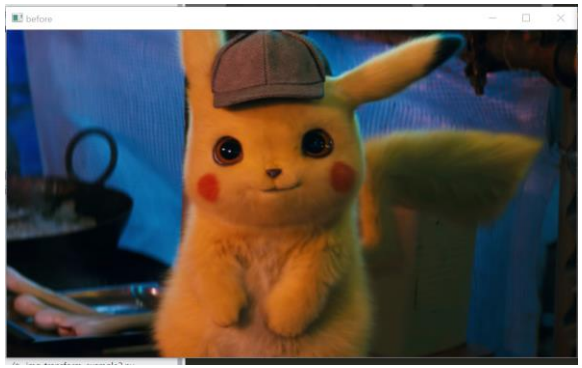
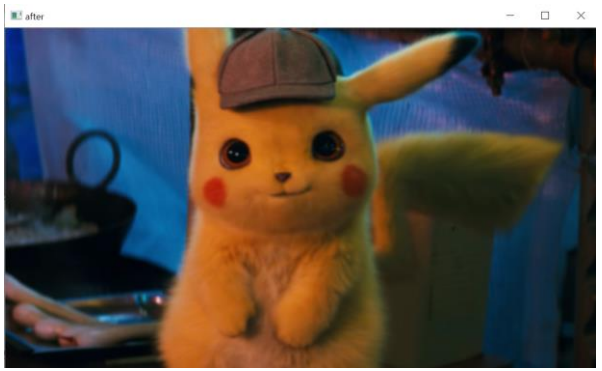


Image Processing

- Image Smoothing
  - **Averaging**
    - » cv2.blur(src, ksize)
      - src: source image
      - ksize: Kernel Size
  - **Gaussian**
    - » cv2.GaussianBlur(img, ksize, sigmaX)
      - Img: source image
      - ksize – (width, height), should be positive odd number
  - **Median**
    - » cv2.medianBlur(src, ksize)
      - src: source image
      - ksize: Kernel Size, an odd number greater than 1



Before



After: Gaussian Blur 5 X 5

Reference

- OpenCV, https://opencv.org/about/
- Affine Transformation, https://en.wikipedia.org/wiki/Transformation_matrix

SIT32004
ICT Application Development

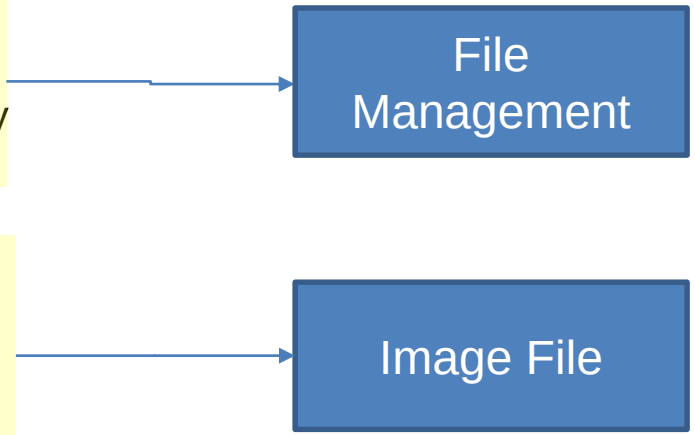
Practice

Prof. Changbeom Choi

Integrating OpenCV and Your Own Application

- Designing Class
  - Identify the requirements
    - » List the image files
    - » Save/Load image file
    - » Rename image file with given data
    - » Resize an image
    - » Rotate an image
    - » Show an image
    - » Upload the image file to given image directory
  - Classify the requirements
    - » List the image files
    - » Rename image file with given data
    - » Upload the image file to given image directory
    - » Save/Load image file

    - » Show an image
    - » Resize an image
    - » Rotate an image
    - » Rename image file with given data



Listing the files

- Considerations
  - What information does we need to implement the list operation?
    - » Path to the files
    - » How to discriminate the image file from other files?
  - Path to the files and discriminate the image files
    - » We use ‘pathlib’

Directory Management [3]

- Object-oriented file system paths: pathlib
  - This module offers classes representing filesystem paths with semantics appropriate for different operating systems.
  - Basic use
    - » Importing
      - from pathlib import Path
    - » Listing subdirectories
      - p = Path('.')
        [x for x in p.iterdir() if x.is_dir()]
    - » Listing Python source files in this directory tree:
      - list(p.glob('**/*.py'))
    - » Navigating inside a directory tree:
      - p = Path('/etc')
        q = p / 'init.d' / 'reboot'
    - » Retrieve absolute path
      - q.resolve()

* Slides from previous lecture

Design Image Handler Class

- class ImageHandler(object)
  - File management
    - » def LoadImageFile(self, _path):
    - » def SaveImageFile(self):
    - » def SaveAsImageFile(self, _path):
  - Image management
    - » def ResizeImageFile(self, _fx, _fy):
    - » def RotateImageFile(self, angle, scale):
    - » def ShowImageFile(self):
  - Others
    - » def __init__(self, _path):

Checking Validity

- If you are developing an image management software, you may consider the validity of the image file. Sometimes image file may be not exist at the position. Therefore, you may check validity of the image path.
  - Method01: if-else

```
01:     def LoadImageFile(self, _path):
02:             if not self._img_path and not self._img:
03:                     print("Load Image File First!")
04:             else:
05:                     self._img = cv2.imread(_path, cv2.IMREAD_COLOR)
06:                     self._img_path = _path
```

  - Method02: using decorator

```
01:     def validity_check(fn):
02:             @wraps(fn)
03:             def wrap_fn(self, *args, **kwargs):
04:                     if not self._img_path and not self._img:
05:                             print("Load Image File First!")
06:                     else:
07:                             fn(self, *args, **kwargs)
08:
09:             return wrap_fn
```

Integrate application with other open source

- Develop an Image Telegram Bot
  - Refer image_telegram_bot.py