LAB # 2

DIGITAL IMAGE PROCESSING

Mentor: Dr. Irfan Ali, PhD
Assistant Professor (AI & MMG)

Aror University of Art, Architechture, Design & Heritage, Sukkur

Digital image processing.

Processing digital images using algorithms/Operations.

Involves:

- Image acquisition, Enhancement, Restoration, Compression
- Segmentation & Object Recognition
- Goal: Extract meaningful information from images.

- What is OpenCV ?.

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.

It includes several hundreds of computer vision algorithms.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.

OpenCV is written natively in C++.

Initial release June 2000

- Essential Libraries for DIP.

- NumPy – Numerical computing
- Matplotlib – Visualization
- OpenCV – Image/Video processing
- Pillow (PIL) – Image editing
- scikit-image – Image processing algorithms
- TensorFlow / PyTorch – Deep learning with images.

- OpenCV Task.

  - Object classification
  - Object identification
  - Object tracking
  - Image restoration
  - Feature matching
  - Video motion analysis.

Object/Task

How to read and show image

(Python/jupyter notebook).

Read and show image for 1 second.

```
[*]: import cv2
```

```
[ ]: img = cv2.imread("image1.jpg")
     cv2.imshow("Ali" ,img)
     cv2.waitKey(10000)
     cv2.destroyAllWindows()
```

Read and show multiple (mimic) images.

```
[ ]: img = cv2.imread("image1.jpg")
     cv2.imshow("Ali" ,img)
     cv2.imshow("Ali 1" ,img)
     cv2.waitKey(10000)
     cv2.destroyAllWindows()
```

Read and show multiple (mimic)

Images and close particular window.

```
[ ]: img = cv2.imread("image1.jpg")
     cv2.imshow("Ali" ,img)
     cv2.imshow("Ali 1" ,img)
     cv2.waitKey(10000)
     cv2.destroyAllWindows()
     cv2.destroyWindow("Ali 1")
```

Make channels through numpy.

```
[*]: v= np.array([[1,2,3,1,2,3],[1,2,3,1,2,3]])
     v
```

Show multiple images in one frame using stack function.

```
[ ]: img = cv2.imread(r"C:\Users\HP\Desktop\J1/image2.jpg")
     resize_img = cv2.resize(img, (500,30))
     h = np.hstack((re_img,re_img))
     cv2.imshow("Ali" ,h)
     cv2.waitKey(0)
     cv2.destroyAllWindows()
     #cv2.destroyWindow("Ali 1")
```

Put text on image in python

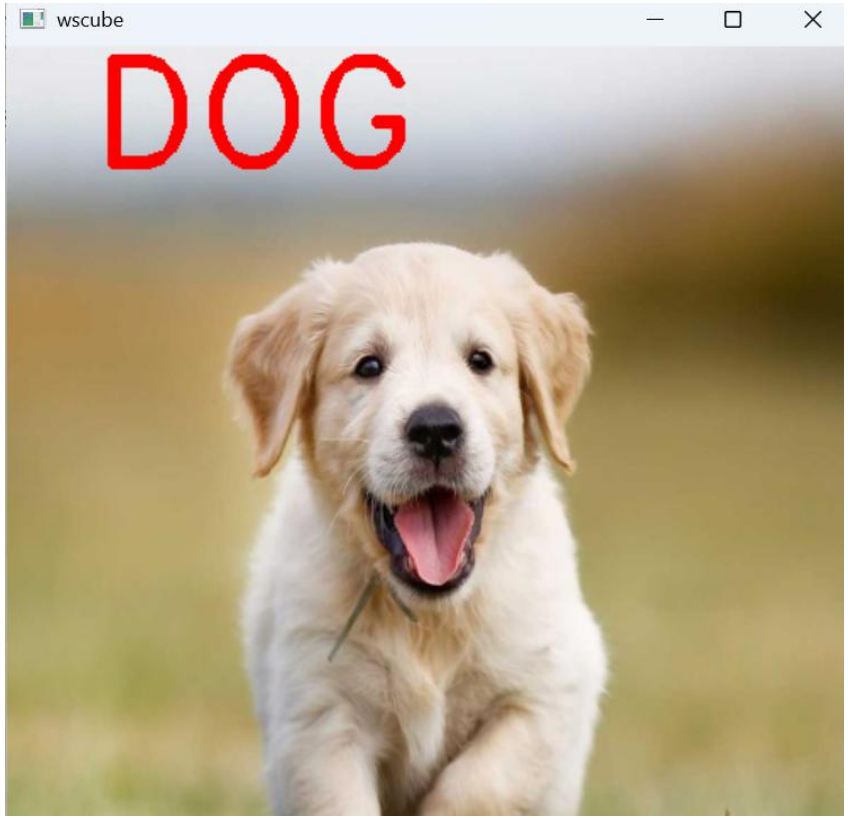
[18]:

```
img_get = cv2.imread(r"C:\Users\HP\Desktop\J1\image2.jpg")
img_get = cv2.resize(img_get, (500, 600))

txt = cv2.putText(
    img_get,
    text="DOG",
    org=(50, 70),
    fontFace=cv2.FONT_HERSHEY_DUPLEX,
    fontScale=3,
    color=(0, 0, 255),
    thickness=3,
    lineType=cv2.LINE_8,
    bottomLeftOrigin=False
)

cv2.imshow("wscube", img_get)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Put text on image in python

- **img_get** → the image where text will be written
- **text=“DOG"** → the actual text to display
- **org=(50, 70)** → starting position of text (x=50, y=70)
- **fontFace=cv2.FONT_HERSHEY_DUPLEX** → font style
- **fontScale=3** → size of the text (bigger number = bigger text)
- **color=(0, 0, 255)** → text color in BGR (here it’s red)
- **thickness=3** → thickness/boldness of text
- **lineType=cv2.LINE_8** → how the edges of text are drawn (normal, smooth, etc.)
- **bottomLeftOrigin=False** → text is drawn normally (if True, it flips vertically)

Put text on image in python



Handling/ Open an image (works for JPG, PNG , BMP, TIFF)

[1]:
```
import cv2
```

[2]:
```
from PIL import Image
```

[3]:
```
import matplotlib.pyplot as plt
```

[4]:
```
img = Image.open("image2.jpg")
plt.imshow(img)
plt.axis("off")  # Hide axes
plt.show()
```

Handling/ Open an image (works for JPG, PNG , BMP, TIFF)

[5]:
```
img.save("output.png")   # Convert JPG → PNG
img.save("output.bmp")   # Convert JPG → BMP
img.save("output.tiff")  # Convert JPG → TIFF
```

Handling/ Open an image (works for JPG, PNG , BMP, TIFF)

Format	Compression	Quality	File Size	Special Feature	Common Use
JPG	Lossy	Medium–High	Small	Good for photos	Everyday photos
PNG	Lossless	High	Larger	Transparency	Logos, graphics
BMP	None	Very High	Very Large	Simple raw pixels	Rare today
TIFF	Lossless (or lossy)	Very High	Very Large	Multi-page support	Printing, scanning

Image rotation, flipping using matplotlib)

Flip = like flipping a paper in front of a mirror (horizontal) or flipping it upside down (vertical).

Rotate = like turning the paper by 90° or 180°.

- `cv2.flip(img, 1)` → horizontal flip
- `cv2.flip(img, 0)` → vertical flip
- `cv2.flip(img, -1)` → both
- `cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)` → rotates

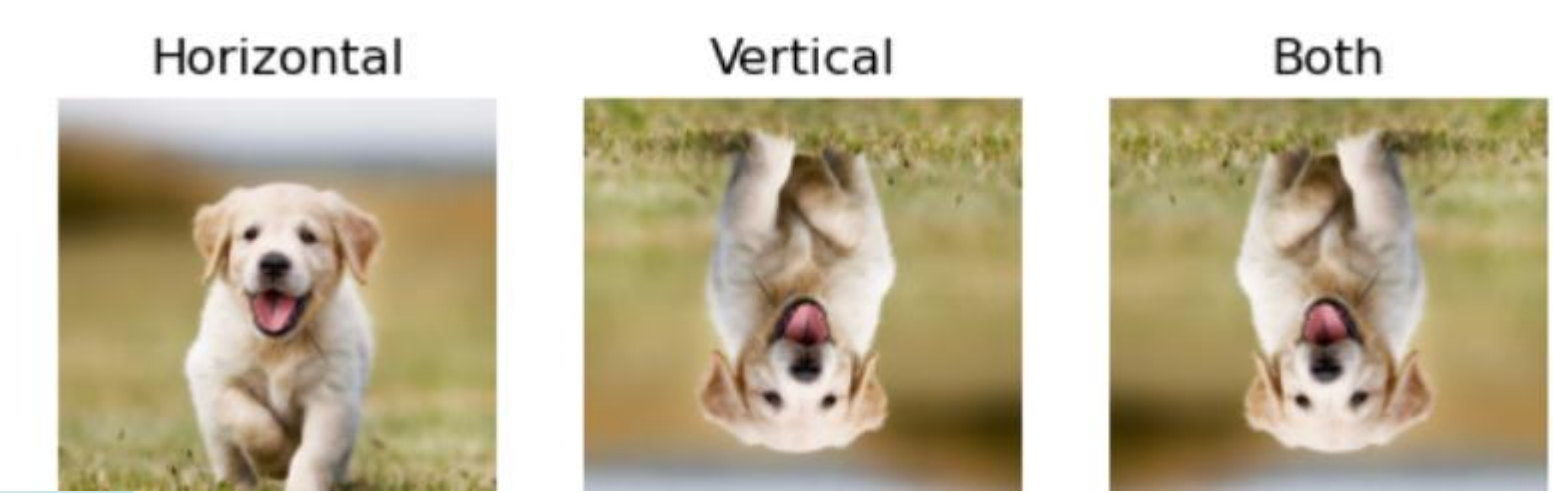
Image rotation, flipping using matplotlib

[9]:
```
#Read image
img = cv2.imread("image2.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

[10]:
```
# Flip horizontally
flip_h = cv2.flip(img, 1)

# Flip vertically
flip_v = cv2.flip(img, 0)

# Flip both (horizontal + vertical)
flip_hv = cv2.flip(img, -1)
```

Image rotation, flipping using matplotlib

```
# Show results
plt.subplot(1,3,1); plt.imshow(cv2.cvtColor(flip_h, cv2.COLOR_BGR2RGB)); plt.title("Horizontal"); plt.axis("off")
plt.subplot(1,3,2); plt.imshow(cv2.cvtColor(flip_v, cv2.COLOR_BGR2RGB)); plt.title("Vertical"); plt.axis("off")
plt.subplot(1,3,3); plt.imshow(cv2.cvtColor(flip_hv, cv2.COLOR_BGR2RGB)); plt.title("Both"); plt.axis("off")
plt.show()
```



Convert image into grayscale/ Black & White

[11]:

```
# Read the image
img = cv2.imread("image2.jpg")

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Show the grayscale image
plt.imshow(gray, cmap="gray")
plt.axis("off")
plt.show()

# Save the grayscale image
cv2.imwrite("output_gray.jpg", gray)
```

Convert image into grayscale/ Black & White

Cropping images with NumPy slicing

[12]:
```
# Read image
img = cv2.imread("image2.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Crop: img[y1:y2, x1:x2]
crop = img_rgb[50:200, 100:300]   # (row range, column range)

# Show cropped image
plt.imshow(crop)
plt.axis("off")
plt.show()
```

Cropping images with NumPy slicing