National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Ma'am Neelma Naz **Date:** September 25, 2025

Semester: 7th **Group:** 02

CS471 Machine Learning

Lab 3: Introduction to OpenCV and PIL

		PLO4	PLO5	PLO5	PLO8	PLO9
		CLO4	CLO5	CLO5	CLO6	CLO7
Student Name	**Reg. No**	**Viva / Quiz / Demo** 5 Marks	**Analysis of Data in Report** 5 Marks	**Modern Tool Usage** 5 Marks	**Ethics** 5 Marks	**Individual and Teamwork** 5 Marks
Hanzla Sajjad	403214					
Irfa Farooq	412564					

Introduction

This laboratory exercise will introduce OpenCV which is a popular and widely used library for image processing and computer vision applications. The field of computer vision overlaps extensively with the field of Machine Learning particularly deep learning. A number of image-based applications such as face detection, image classification, object tracking and pose estimation rely on machine learning techniques. In such cases, images will be the dataset and so, image processing becomes a prerequisite for applying machine learning to images. It is important to familiarize yourself with the basics of image processing which is the subject of this lab. You will also learn PIL which is another imaging library native to python.

Objectives

- Load, save and display image data using Python
- Access and modify pixels as well as ROIs in images
- Place lines, rectangles, circles and text in images
- Resize and rotate images at various scales/angles

Lab Conduct

- Respect faculty and peers through speech and actions
- The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
- In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
- Use the tab key to provide the indentation in python.

Theory

OpenCV is a library that focuses on image processing and computer vision. An image is an array of colored square called pixels. Each pixel has a certain location in the array and color values in BGR format. By referring to the array indices, the individual pixels or a range of pixels can be accessed and modified. OpenCV provides many functions for resizing, rotating, and placing objects in images. Rotation involves computing a 2-D rotation matrix which is applied for the transformation of the image. **PIL** (Python Imaging Library) is another imaging

library native to python. The extension of PIL for Python 3 is called "Pillow". Unlike OpenCV which uses a BGR format for image data, PIL uses the RGB format.

A brief summary of the relevant keywords and functions in python is provided below:

print()	output text on console
input()	get input from user on console
range()	create a sequence of numbers
len()	gives the number of characters in a string
if	contains code that executes depending on a logical condition
else	connects with **if** and **elif**, executes when conditions are not met
elif	equivalent to **else if**
while	loops code as long as a condition is true
for	loops code through a sequence of items in an iterable object
break	exit loop immediately
continue	jump to the next iteration of the loop
def	used to define a function

In this lab, you will use 4 different image files in the upcoming tasks. 3 of these image files you must download on your own. One of these downloaded images must contain at least 20 subjects (humans, animals, cars, objects etc.). The last image will be provided to you by the lab instructor.

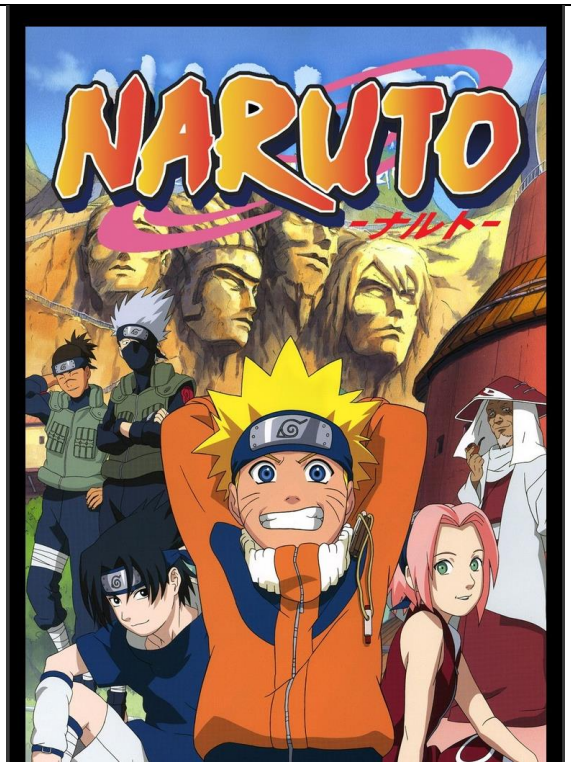
Lab Task 1 – Load and Display Images _____

Write a python script in which you will load the 4 images from disk. Then, display the images in different windows at the same time. You will need to provide the code and a single screenshot which shows all 4 windows at the same time.

Code

```
#Task 1
import cv2
from google.colab.patches import cv2_imshow

# 0 = greyscale, 1 = colored imaged BGR
list_of_images = ["anime1.jpg", "anime2.jpg", "anime3.jpg", "anime4.jpg"]

for i in range(len(list_of_images)):
  img = cv2.imread(list_of_images[i], 1)
  cv2_imshow(img)
```

Console Output


NARUTO
-ナルト-


DRAGON BALL Z

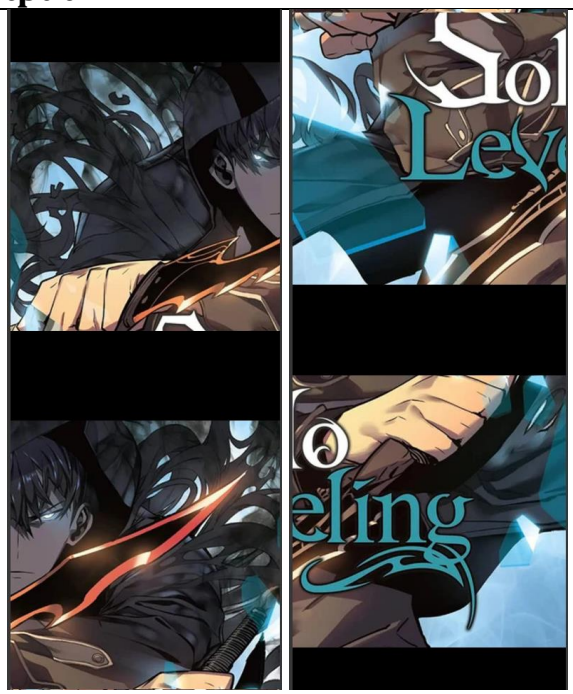
Lab Task 2 – Cropping Images _____

Write code to load all the image files. Using the slice operation, crop out the four quadrants of the image and display them in separate windows. The code must be generic enough to take the image size into account. For submission, provide the code and a single screenshot showing all 4 windows.

Code

```
#Task 2
import cv2
from google.colab.patches import cv2_imshow

# 0 = greyscale, 1 = colored imaged BGR
list_of_images = ["anime1.jpg", " anime2.jpg", " anime3.jpg", " anime4.jpg"]

for i in range(len(list_of_images)):
  img = cv2.imread(list_of_images[i], 1)
  rows = img.shape[0]
  cols = img.shape[1]

  top_left = img[0:rows//2, 0:cols//2]
  top_right = img[0:rows//2, cols//2:cols]
  bottom_left = img[rows//2:rows, 0:cols//2]
  bottom_right = img[rows//2:rows, cols//2:cols]

  cv2_imshow(top_left)
  cv2_imshow(top_right)
  cv2_imshow(bottom_left)
  cv2_imshow(bottom_right)
```

Console Output

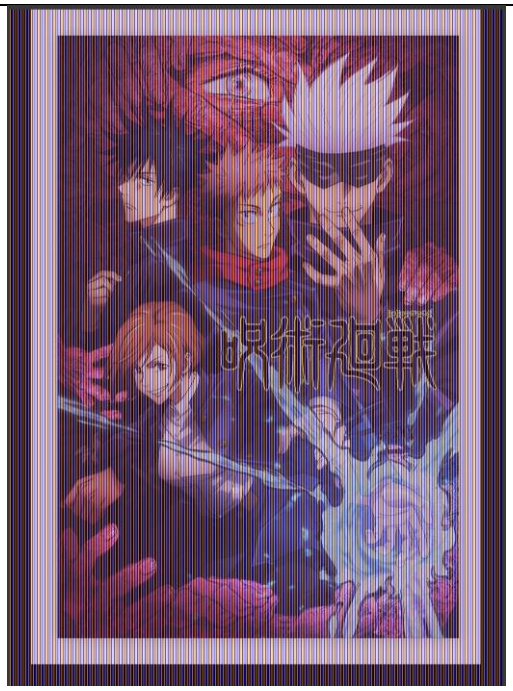
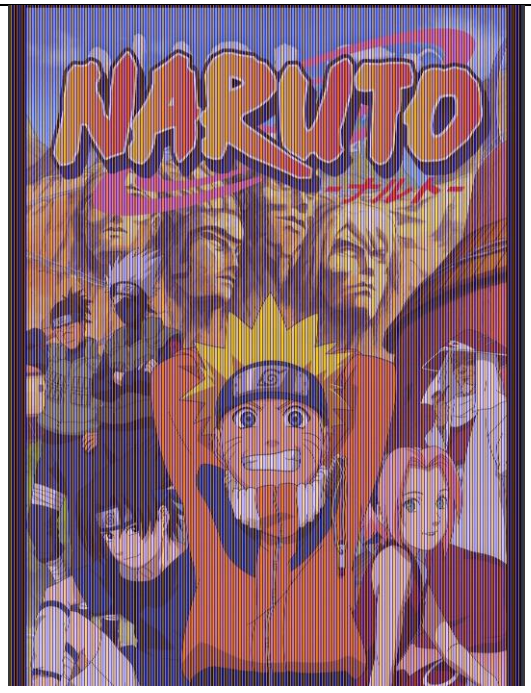


Lab Task 3 – Modifying Pixel Data _____

Write code to load the image files and place alternating blue and orange vertical lines in the images. Do **NOT** use the line function (cv2.line). You need to do this by changing the pixel colors. Each line must be 1-pixel thick. The lines are also spaced apart by a 1-pixel wide gap. Thus, the image will have one blue line, then one line of image pixels, then one orange line, then another line of image pixels and so on. Provide the code and screenshot for the submission.

Code

```
#Task 3
import cv2
from google.colab.patches import cv2_imshow

list_of_images = ["flower1.jpg", "flower2.jpg", " anime.jpg", "anime4.jpg"]

for i in range(len(list_of_images)):
  img = cv2.imread(list_of_images[i], 1)
  rows = img.shape[0]
  cols = img.shape[1]

  for i in range(cols):
    if(i % 2 == 0):
      if(i % 4 == 0):
        for j in range(rows):
          img[j][i] = [0, 165, 255] # Setting whole line to orange
      else:
        for j in range(rows):
          img[j][i] = [255, 0, 0] # Setting whole line to blue
  cv2_imshow(img)
```

Console Output	
	
	

Lab Task 4 – Placing Shapes and Text _____

Load all of the downloaded images and place a line, a rectangle, a circle and some text using the inbuilt functions in OpenCV on each of the image. Each of the placed objects must have a different color. The text must contain the name of at least one member of your group. Provide the code and screenshot of the image.

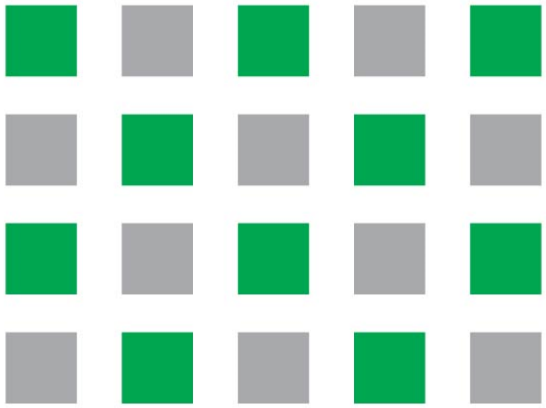
Code

```
#Task 4
import cv2
from google.colab.patches import cv2_imshow

img1 = cv2.imread("flower1.jpg", 1)
img2 = cv2.imread("flower2.jpg", 1)

#Using arbitrary ending values keeping note of the total pixels in rows and
columns
img1 = cv2.line(img1, (0, 0), (80, 200), (255, 0, 0), 1)
img2 = cv2.line(img2, (0, 0), (80, 200), (255, 0, 0), 1)
img1 = cv2.rectangle(img1, (1, 1), (255, 255), (0, 255, 0), 1)
img2 = cv2.rectangle(img2, (1, 1), (255, 255), (0, 255, 0), 1)
img1 = cv2.circle(img1, (80, 200), 30, (255, 0, 255), 1)
img2 = cv2.circle(img2, (80, 200), 30, (255, 0, 255), 1)
img1 = cv2.putText(img1, 'Hanzla', (50, 350), cv2.FONT_HERSHEY_SIMPLEX, 3, (255,
255, 255), 5, cv2.LINE_AA)
img2 = cv2.putText(img2, 'Irfa', (50, 350), cv2.FONT_HERSHEY_SIMPLEX, 3, (255,
255, 255), 5, cv2.LINE_AA)

cv2_imshow(img1)
cv2_imshow(img2)
```

Console Output

Lab Task 5 – Placing Shapes II _____

In this task, you will place solid squares of alternating colors similar to those shown in the figure.



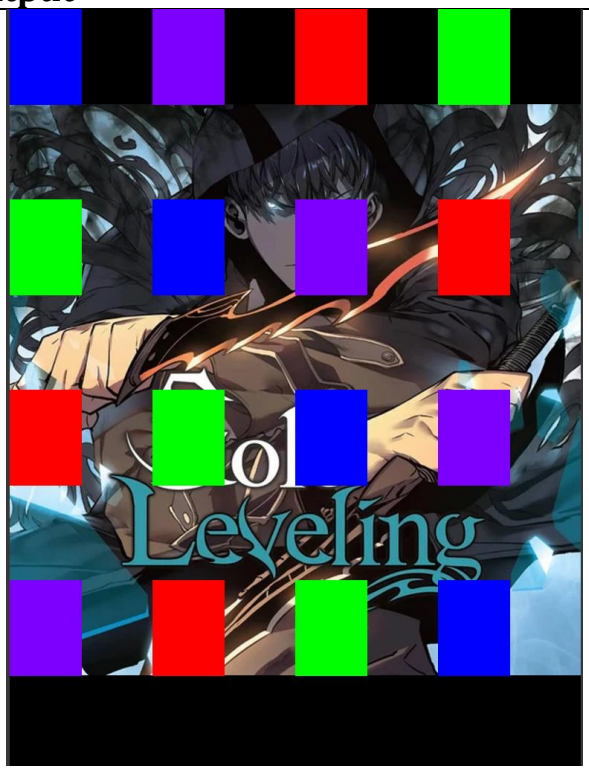
To make the square solid, the thickness argument should be set to -1 in the cv2.rectangle function. The above shown pattern must be placed on all of the downloaded images. It is up to you to choose the size of the squares as well as their colors. At least, 4 different colors must be used. Provide the code and screenshot of the final result.

Code

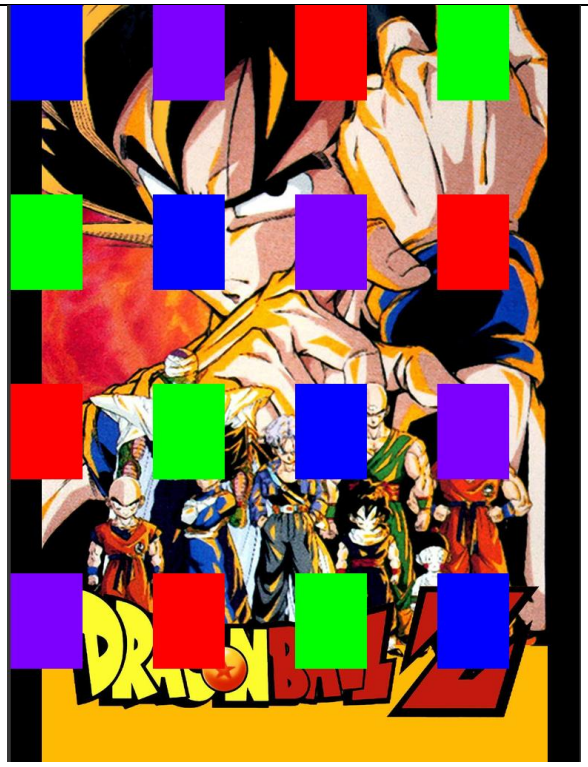
```
# Task: Pattern of colored blocks
import cv2
from google.colab.patches import cv2_imshow

list_of_images = ["flower1.jpg", "flower2.jpg", "flower3.jpg", "flower4.jpg"]
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 0, 125)]

for i in range(len(list_of_images)):
  img = cv2.imread(list_of_images[i], 1)
  rows = img.shape[0]
  cols = img.shape[1]

  # Diving the columns and rows in 8 equal parts
  rows_idx = rows//8
  cols_idx = cols//8

  for j in range(0, cols, cols_idx*2):
    temp_colors = [(), (), (), ()]
```

```
    for k, l in zip(range(0, rows, rows_idx*2), range(len(colors))):
      img = cv2.rectangle(img, (j, k), (j + cols_idx, k + rows_idx), colors[l],
-1)

    for j in range(len(colors)):

      if(j == 0):
        temp_colors[j] = colors[len(colors) - 1]
      else:
        temp_colors[j] = colors[j - 1]

    colors = temp_colors

  # Show final patterned image
  cv2_imshow(img)
```

Console Output




NARUTO


DRAGON BALL Z

Lab Task 6 – Bounding Boxes _____

For detection of faces, people, objects etc. in images, the result of the detection is depicted as a rectangular box around the detection. Load an image containing at least 20 subjects (humans, animals, cars, objects etc.) and use the cv2.rectangle function to place bounding boxes around the subjects. Each bounding box must be of a different color. Provide the code and screenshot of the final result.

Code

```
!pip install -q ultralytics

from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt
from google.colab import files
import random

uploaded = files.upload()

# Get the list of uploaded file names
uploaded_filenames = list(uploaded.keys())

# If no files were uploaded, raise an error
if not uploaded_filenames:
    raise FileNotFoundError("No files uploaded.")

# Let the user select an image from the uploaded files
print("Please select an image to process:")
for i, filename in enumerate(uploaded_filenames):
    print(f"{i+1}: {filename}")

while True:
    try:
        selection = int(input("Enter the number of the image you want to use: "))
        if 1 <= selection <= len(uploaded_filenames):
            selected_filename = uploaded_filenames[selection - 1]
            break
```

```
            else:
                print("Invalid selection. Please enter a valid number.")
        except ValueError:
            print("Invalid input. Please enter a number.")



img = cv2.imread(selected_filename)
if img is None:
    raise FileNotFoundError(f"Error loading image: {selected_filename}. Please
check the file.")

model = YOLO("yolov8n.pt")

results = model(img)

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

for result in results:
    boxes = result.boxes
    for box in boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        conf = box.conf[0]
        cls = int(box.cls[0])
        label = model.names[cls]
        color = (random.randint(0, 255), random.randint(0, 255), random.randint(0,
255))
        cv2.rectangle(img_rgb, (x1, y1), (x2, y2), color, 2)
        cv2.putText(img_rgb, f"{label} {conf:.2f}", (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

plt.figure(figsize=(12, 8))
plt.imshow(img_rgb)
plt.axis("off")
plt.title("Detected Objects with Random Colors")
plt.show()
```

Console Output

```
Enter the number of the image you want to use: yes
Invalid input. Please enter a number.
Enter the number of the image you want to use: 1

0: 480x640 7 persons, 274.0ms
Speed: 9.9ms preprocess, 274.0ms inference, 14.6ms postprocess per image at shape (1, 3, 480, 640)
```

Detected Objects with Random Colors

Lab Task 7 – Resizing Images _____

Load the images and use the resize function to make copies of each image at different sizes. Try changing the aspect ratio for some results. Display at least 3 different sizes in separate windows and take the screenshot. Provide the code and screenshot for the submission.

Code

```
#Task 7
import cv2
from google.colab.patches import cv2_imshow

img = cv2.imread("anime1.jpg", 1)
imgResize1 = cv2.resize(img, None, fx = 0.5, fy = 0.5, interpolation =
cv2.INTER_CUBIC)
imgResize2 = cv2.resize(img, None, fx = 0.75, fy = 0.5, interpolation =
cv2.INTER_CUBIC)
imgResize3 = cv2.resize(img, None, fx = 0.8, fy = 0.3, interpolation =
cv2.INTER_CUBIC)

cv2_imshow(img)
cv2_imshow(imgResize1)
cv2_imshow(imgResize2)
cv2_imshow(imgResize3)
```

Console Output

Figure 1: Original Image

Figure 2: Resized samples

Lab Task 8 – Rotating Images _____

Load the images and use the rotate function to rotate at three different angles. For each rotated image, you need to manually adjust the scale factor (in get2DRotationMatrix function) so that the entire image is shown in the window. The rotated image's border/corner must touch the window's border. Show all 3 windows in the screenshot. Provide the code and screenshot for the submission.

Code

```
#Task 8
import cv2
from google.colab.patches import cv2_imshow

img = cv2.imread("anime2.jpg", 1)
rows = img.shape[0]
cols = img.shape[1]

M1 = cv2.getRotationMatrix2D((cols/2,rows/2), 45, 0.7)
rot1 = cv2.warpAffine(img, M1, (cols,rows))
M2 = cv2.getRotationMatrix2D((cols/2,rows/2), 30, -0.7)
rot2 = cv2.warpAffine(img, M2, (cols,rows))
M3 = cv2.getRotationMatrix2D((cols/2,rows/2), 60, 0.7)
rot3 = cv2.warpAffine(img, M2, (cols,rows))

cv2_imshow(img)
cv2_imshow(rot1)
cv2_imshow(rot2)
cv2_imshow(rot3)
```

Console Output

Lab Task 9 – PIL _____

Import the Python Imaging Library (PIL) and use it to load all of the image files. For each image, stretch it by changing the scale of the image and place different shapes and text in the resulting image. Also, rotate the image by 30 degrees. Provide the code and screenshots of the final output. You will need to learn to use PIL on your own for this task.

Code

```
from google.colab import files
from PIL import Image, ImageDraw, ImageFont
import matplotlib.pyplot as plt
print("Please upload 4 images from your Downloads folder.")
uploaded = files.upload()
image_files = list(uploaded.keys())[:4] # only take first 4
processed_images = []
try:
    font = ImageFont.truetype("DejaVuSans-Bold.ttf", 50)
except:
    font = ImageFont.load_default()
for idx, fname in enumerate(image_files):
    # Load image
    img = Image.open(fname).convert("RGB")
    w, h = img.size
    # Step 2: Stretch differently
    if idx == 0:
        stretched = img.resize((w*2, h)) # stretch width
    elif idx == 1:
        stretched = img.resize((w, h*2)) # stretch height
    elif idx == 2:
        stretched = img.resize((w//2, h//2)) # shrink
    else:
        stretched = img.resize((int(w*1.5), int(h*1.5))) # 150% both
    sw, sh = stretched.size
    draw = ImageDraw.Draw(stretched)
    # Rectangle (inside border)
    rect_w, rect_h = sw//4, sh//6
    draw.rectangle([10, 10, 10+rect_w, 10+rect_h], outline="red", width=8)
    # Ellipse (inside border)
    ell_x1, ell_y1 = sw//3, sh//4
```

```
        ell_x2, ell_y2 = ell_x1 + sw//4, ell_y1 + sh//4
        draw.ellipse([ell_x1, ell_y1, ell_x2, ell_y2], outline="blue", width=8)
        # Line (diagonal, inside border)
        draw.line([sw//10, sh//2, sw//2, sh - 20], fill="green", width=12)
        # Big text (inside safe zone)
        draw.text((20, sh - 80), f"Image {idx+1}", font=font, fill="purple")
        # Step 3: Rotate all by 30 degrees
        rotated = stretched.rotate(30, expand=True)
        processed_images.append((fname, rotated))
    # Step 4: Display results
    fig, axes = plt.subplots(1, 4, figsize=(24, 6))
    for ax, (fname, img) in zip(axes, processed_images):
        ax.imshow(img)
        ax.set_title(f"{fname}\nProcessed", fontsize=12)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

Console Output