**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Mohsin Kamal **Dated:** 28/09/2023

**Semester:** 7th **Section:** BEE 12C

**CS-477 Computer Vision**

Lab 3: Introduction to OpenCV

**Group Members**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4 - CLO4** | | **PLO5 -CLO5** | **PLO8 -CLO6** | **PLO9 -CLO7** |
| **Name** | **Reg. No** | **Viva / Quiz / Lab Performance** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Teamwork** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |  |  |
| Muhammad Umer | 345834 |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Table of Contents**

[2 Introduction to OpenCV 3](#_Toc146833662)

[2.1 Introduction 3](#_Toc146833663)

[2.2 Objectives 3](#_Toc146833664)

[2.3 Software 3](#_Toc146833665)

[3 Lab Tasks 4](#_Toc146833666)

[3.1 Task 1 4](#_Toc146833667)

[3.2 Task 2 4](#_Toc146833668)

[3.3 Task 3 5](#_Toc146833669)

[3.4 Task 4 6](#_Toc146833670)

[3.5 Task 5 7](#_Toc146833671)

[3.6 Task 6 8](#_Toc146833672)

[3.7 Task 7 10](#_Toc146833673)

[4 Conclusion 11](#_Toc146833674)

# Introduction to OpenCV

## Introduction

This laboratory exercise will introduce OpenCV which is a popular and widely used library for image processing and computer vision applications. Many algorithms used in robotics employ aspects of vision such as in feature extraction, image stitching, stereovision, visual serving, and structure from motion etc. Before starting computer vision, it is important to familiarize yourself with the basics of image processing which is the subject of this lab.

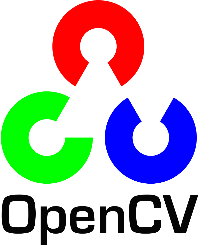
## Objectives

The following are the main objectives of this lab:

* 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 image at various scales
* Rotate image at various angles

## Software

OpenCV is a library that focuses on image processing and computer vision. An image is an array of colored squares 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.



# Lab Tasks

## Task 1

Write a python script in which you will load 3 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 3 windows.

### Task 1 Code Starts Here ###

img1 = cv2.imread(path\_img1)

img2 = cv2.imread(path\_img2)

img3 = cv2.imread(path\_img3)

cv2.imshow("img1", img1)

cv2.imshow("img2", img2)

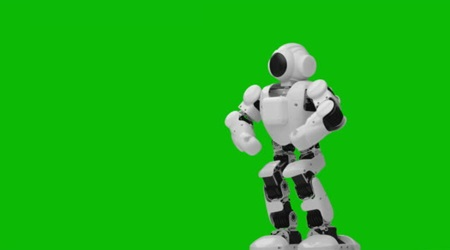
cv2.imshow("img3", img3)

cv2.waitKey(0)

cv2.destroyAllWindows()

### Task 1 Code Ends Here ###

### Task 1 Screenshots Starts Here ###



### Task 1 Screenshots Ends Here ###

## Task 2

Write code to load the walle.jpg file. 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.

### Task 2 Code Starts Here ###

img2 = cv2.imread(path\_img2)

height, width, channels = img2.shape

img2\_1 = img2[0 : *int*(height / 2), 0 : *int*(width / 2)]

img2\_2 = img2[0 : *int*(height / 2), *int*(width / 2) : width]

img2\_3 = img2[*int*(height / 2) : height, 0 : *int*(width / 2)]

img2\_4 = img2[*int*(height / 2) : height, *int*(width / 2) : width]

cv2.imshow("img2\_1", img2\_1)

cv2.imshow("img2\_2", img2\_2)

cv2.imshow("img2\_3", img2\_3)

cv2.imshow("img2\_4", img2\_4)

cv2.waitKey(0)

cv2.destroyAllWindows()

### Task 2 Code Ends Here ###

### Task 2 Screenshot Starts Here ###

### Task 2 Screenshot Ends Here ###

## Task 3

Write code to load the walle.jpg file and place alternating green and white horizontal lines in the image. 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 green line, then one line of image pixels, then one white line, then another line of image pixels and so on. Provide the code and screenshot for the submission.

### Task 3 Code Starts Here ###

img2 = cv2.imread(path\_img2)

for i in range(0, height, 3):

    img2[i, :, :] = [0, 255, 0]

    img2[i + 2, :, :] = [255, 255, 255]

cv2.imshow("img2", img2)

cv2.waitKey(0)

cv2.destroyAllWindows()

### Task 3 Code Ends Here ###

### Task 3 Screenshot Starts Here ###



### Task 3 Screenshot Ends Here ###

## Task 4

Load any one of the provided images and place a line, rectangle, circle and text using the inbuilt functions in OpenCV. 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.

### Task 4 Code Starts Here ###

img2 = cv2.imread(path\_img2)

height, width, channels = img2.shape

img2 = cv2.line(img2, (0, 0), (width, height), (255, 0, 0), 5)

img2 = cv2.rectangle(img2, (30, 30), (width // 4, height // 4), (0, 255, 0), 5)

img2 = cv2.circle(

    img2, (*int*(width / 3), *int*(height / 3)), *int*(height / 3), (0, 0, 255), 5

)

img2 = cv2.putText(

    img2,

    "Muhammad Umer",

    (10, *int*(height) - *int*(height / 50)),

    cv2.FONT\_HERSHEY\_SIMPLEX,

    1,

    (0, 0, 0),

    2,

)

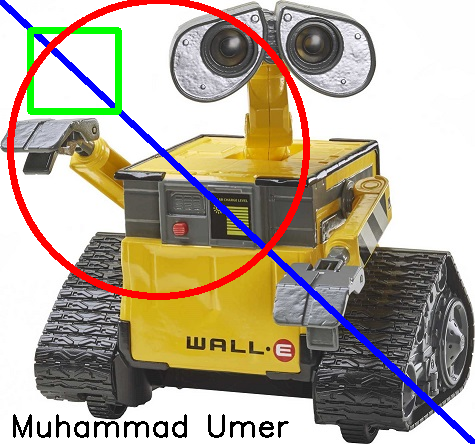
cv2.imshow("img2", img2)

cv2.waitKey(0)

cv2.destroyAllWindows()

### Task 4 Code Ends Here ###

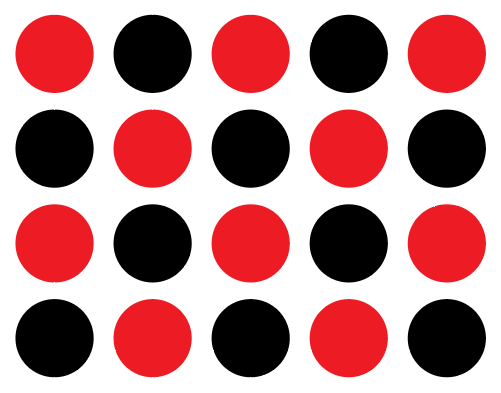
### Task 4 Screenshot Starts Here ###



### Task 4 Screenshot Ends Here ###

## Task 5

In this task, you will place solid circles of alternating colors similar to those shown in the figure. To make the circle solid, the thickness argument should be set to -1 in the cv2.circle function. The above shown pattern must be placed on one of the provided images. It is up to you to choose the radius of the circles as well as their colors. At least, 2 colors must be used. Provide the code and screenshot of the result.



### Task 5 Code Starts Here ###

img2 = cv2.imread(path\_img2)

height, width, channels = img2.shape

color\_tracker = 0

for i in range(0, width, 125):

    for j in range(0, height, 115):

        if (i // 125) % 2 == 0:

            if color\_tracker % 2 == 0:

                cv2.circle(img2, (i + 50, j + 50), 50, (0, 0, 255), -1)

            else:

                cv2.circle(img2, (i + 50, j + 50), 50, (0, 0, 0), -1)

        else:

            if color\_tracker % 2 == 0:

                cv2.circle(img2, (i + 50, j + 50), 50, (0, 0, 0), -1)

            else:

                cv2.circle(img2, (i + 50, j + 50), 50, (0, 0, 255), -1)

        color\_tracker += 1

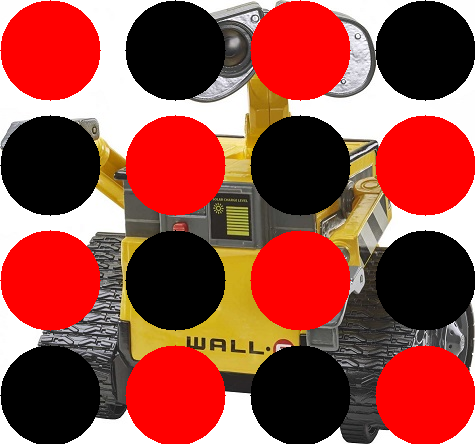
cv2.imshow("img2", img2)

cv2.waitKey(0)

cv2.destroyAllWindows()

### Task 5 Code Ends Here ###

### Task 5 Screenshot Starts Here ###



### Task 5 Screenshot Ends Here ###

## Task 6

Load any one of the provided images and use the resize function to make copies of the image at different sizes. Display at least 3 different sizes in separate windows and take the screenshot. Provide the code and screenshot for the submission.

### Task 6 Code Starts Here ###

img3 = cv2.imread(path\_img3)

height, width, channels = img3.shape

img3\_1 = cv2.resize(img3, (*int*(width / 2), *int*(height / 2)))

img3\_2 = cv2.resize(img3, (*int*(width / 4), *int*(height / 4)))

img3\_3 = cv2.resize(img3, (*int*(width \* 2), *int*(height \* 2)))

cv2.imshow("img3\_1", img3\_1)

cv2.imshow("img3\_2", img3\_2)

cv2.imshow("img3\_3", img3\_3)

cv2.waitKey(0)

cv2.destroyAllWindows()

### Task 6 Code Ends Here ###

### Task 6 Screenshots Starts Here ###







### Task 6 Screenshots Ends Here ###

## Task 7

Load any one of the provided images and use the rotate function to rotate the image at angles of 30, 60 and 90 degrees. 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.

### Task 7 Code Starts Here ###

img3 = cv2.imread(path\_img3)

height, width, channels = img3.shape

img3\_1 = cv2.getRotationMatrix2D((width / 2, height / 2), 30, 0.56)

img3\_1 = cv2.warpAffine(img3, img3\_1, (width, height))

img3\_2 = cv2.getRotationMatrix2D((width / 2, height / 2), 60, 0.47)

img3\_2 = cv2.warpAffine(img3, img3\_2, (width, height))

img3\_3 = cv2.getRotationMatrix2D((width / 2, height / 2), 90, 0.56)

img3\_3 = cv2.warpAffine(img3, img3\_3, (width, height))

cv2.imshow("img3\_1", img3\_1)

cv2.imshow("img3\_2", img3\_2)

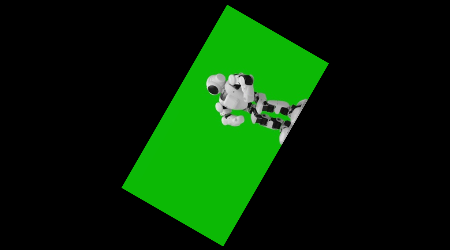
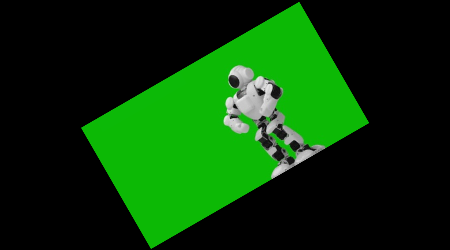
cv2.imshow("img3\_3", img3\_3)

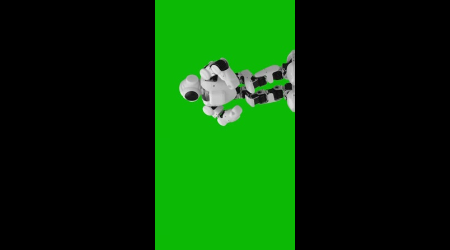
cv2.waitKey(0)

cv2.destroyAllWindows()

### Task 7 Code Ends Here ###

### Task 7 Screenshots Starts Here ###



### Task 7 Screenshots Ends Here ###

# Conclusion

In this laboratory exercise, we were introduced to OpenCV, a popular and widely used library for image processing and computer vision applications. We learned about the basic concepts of image processing, such as reading and writing images, converting images between different color spaces, and performing basic image operations such as resizing, rotating, and cropping images.