National University of Computer & Emerging Sciences, Karachi
Artificial Intelligence-School of Computing
Fall 2024, Lab Manual - 01

Course Code:	Course: Computer Vision Lab
Instructor(s):	Muhammad Khalid

Objectives:

1. Introduction to Computer Vision & Computer Vision Applications
2. Essential Libraries for CV
3. Digital Image & Image Coordinates
4. Digital Image Processing System
5. Image Operations

Introduction to Computer Vision

Computer vision is a multidisciplinary field of study and technology that focuses on enabling computers to interpret, analyze, and understand visual information from the world, much like humans do. It involves the development of algorithms, models, and techniques that allow computers to extract meaningful information from images or videos, and then make decisions or take actions based on that information.

Technically, computer vision involves a range of tasks, including image and video processing, feature extraction, pattern recognition, object detection, tracking, image segmentation, scene understanding, and more. It often utilizes techniques from various domains such as machine learning, image processing, artificial intelligence, and statistical modeling to interpret visual data and derive useful insights or actions from it.

In essence, computer vision aims to bridge the gap between the visual information captured by cameras or sensors and the understanding and decision-making capabilities of computers. This technology finds applications in various fields, including autonomous vehicles, medical imaging, surveillance, robotics, augmented reality, and many others.

Basic Terminology

1. **Image:** A 2D array of pixels representing visual information.
2. **Pixel:** Short for "picture element," it's the smallest unit of an image. Each pixel carries color and intensity information.
3. **Resolution:** The dimensions of an image, typically expressed in pixels (e.g., 1920x1080).
4. **Grayscale:** An image where each pixel represents only intensity, typically ranging from 0 (black) to 255 (white).
5. **RGB:** Stands for Red, Green, Blue. It's a common color representation where each pixel is composed of values for these three primary colors.
6. **Feature:** A distinctive part of an image, used for identifying and distinguishing objects.
7. **Feature Extraction:** The process of identifying and isolating relevant features from an image, often using filters or algorithms.
8. **Edge Detection:** Identifying boundaries or transitions between different regions in an image.
9. **Object Detection:** Identifying and localizing specific objects within an image or video frame.
10. **Segmentation:** Dividing an image into meaningful segments or regions, often used for separating objects from the background.
11. **Image Processing:** Manipulating and enhancing images to improve their quality or extract useful information.
12. **Convolutional Neural Network (CNN):** A type of deep learning model designed to process grid-structured data, like images. CNNs use convolutional layers to automatically learn hierarchical features.
13. **Deep Learning:** A subset of machine learning that uses neural networks with multiple layers to model and solve complex tasks.
14. **Feature Map:** Output maps generated by applying convolutional filters to an image.
15. **Object Recognition:** Identifying and classifying objects within an image or video.
16. **Classification:** Assigning a label or category to an input image, such as identifying whether an image contains a cat or a dog.
17. **Image Registration:** Aligning multiple images of the same scene or object to a common coordinate system.
18. **Optical Flow:** Estimating the motion of objects between consecutive frames in a video.
19. **Tracking:** Following the movement of objects across multiple frames in a video.
20. **Homography:** A transformation that relates two images of the same planar surface.
21. **Feature Descriptor:** A compact representation of a feature that can be used for matching or recognition.
22. **Histogram:** A representation of the distribution of pixel intensities in an image.

23. **SIFT (Scale-Invariant Feature Transform):** An algorithm for detecting and describing local features in images.
24. **SURF (Speeded-Up Robust Features):** A feature detection and description algorithm similar to SIFT but faster.
25. **HOG (Histogram of Oriented Gradients):** A feature descriptor used for object detection and recognition.

Applications of Computer Vision

1. **Autonomous Vehicles:** Computer vision enables self-driving cars to perceive their environment, recognize obstacles, pedestrians, traffic signs, and lane markings, and make real-time driving decisions.
2. **Medical Imaging:** Computer vision assists in medical diagnosis through techniques like image segmentation, tumor detection, and anomaly identification in various medical imaging modalities, including X-rays, MRI, CT scans, and more.
3. **Robotics:** Robots can use computer vision to navigate and interact with their surroundings, grasp objects, and perform tasks that require visual understanding.
4. **Object Detection and Recognition:** Computer vision is used to identify and categorize objects within images or videos, enabling applications like security surveillance, retail analytics, and inventory management.
5. **Facial Recognition:** This technology is employed for authentication, security, and human-computer interaction, such as unlocking devices or verifying identities.
6. **Augmented Reality (AR) and Virtual Reality (VR):** Computer vision enhances AR and VR experiences by overlaying digital content onto the real world and enabling interactions with virtual environments.

Essential Libraries for Computer Vision

- ✓ OpenCV
- ✓ Scikit-Image
- ✓ Pillow (PIL Fork)
- ✓ TorchVision
- ✓ TensorFlow
- ✓ Keras
- ✓ OpenVINO
- ✓ PyTorch
- ✓ Hugging Face
- ✓ Caffe
- ✓ Detectron2

Reading: https://www.superannotate.com/blog/computer-vision-libraries

Tabular comparison of the libraries for computer vision:

Library	Main Focus	Framework	Language	Level of Abstraction	Community Support	Integration with Deep Learning Frameworks
OpenCV	General CV tasks	Independent	C++, Python	High	Strong	Integration via Python bindings
Scikit-Image	Image processing	Independent	Python	Medium	Moderate	Integration with NumPy, Matplotlib
Pillow	Image processing	Independent	Python	Low	Moderate	Basic integration, image processing
TorchVision	CV for PyTorch	PyTorch	Python	High	Strong	Native integration with PyTorch
TensorFlow	General ML, incl. CV	TensorFlow	Python	High	Strong	Native integration with TensorFlow
Keras	High-level neural networks	TensorFlow, Theano (legacy)	Python	High	Strong (for TensorFlow backend)	High-level API, now part of TensorFlow
OpenVINO	CV inference optimization	Intel	C++, Python	High	Moderate	Integration with various frameworks
PyTorch	General ML, incl. CV	PyTorch	Python	High	Strong	Native integration

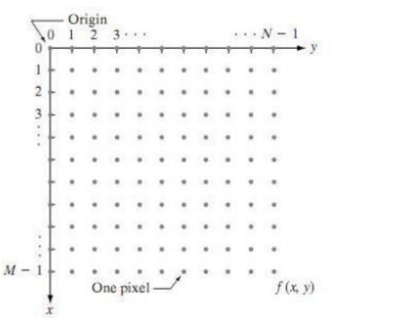
Hugging Face	NLP and CV models	PyTorch, TensorFlow	Python	High	Strong	Pretrained models, NLP focus
Caffe	Deep learning framework	Caffe	C++	Medium	Moderate	Focus on CNNs
Detectron2	Object detection	PyTorch	Python	High	Strong	Specialized in object detection

Digital Image and Image Coordinates

Digital Image

An image may be defined as a two-dimensional function, f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of **f** at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a digital image.

Image Coordinates



1 Coordinate convention used to represent digital images

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0, N-1) \\ f(1,0) & f(1,1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1, N-1) \end{bmatrix}.$$

Digital Image Processing System

In computer science, digital image processing uses algorithms to perform image processing on digital images to extract some useful information. Digital image processing has many advantages as compared to analog image processing. Wide range of algorithms can be applied to input data which can avoid problems such as noise and signal distortion during processing. As we know, images are defined in two dimensions, so DIP can be modeled in multidimensional systems.

Purpose of Image processing

The main purpose of the DIP is divided into following 5 groups:

1. Visualization: The objects which are not visible, they are observed.
2. Image sharpening and restoration: It is used for better image resolution.
3. Image retrieval: An image of interest can be seen
4. Measurement of pattern: In an image, all the objects are measured.
5. Image Recognition: Each object in an image can be distinguished.

Fundamental Steps of Digital Image Processing:

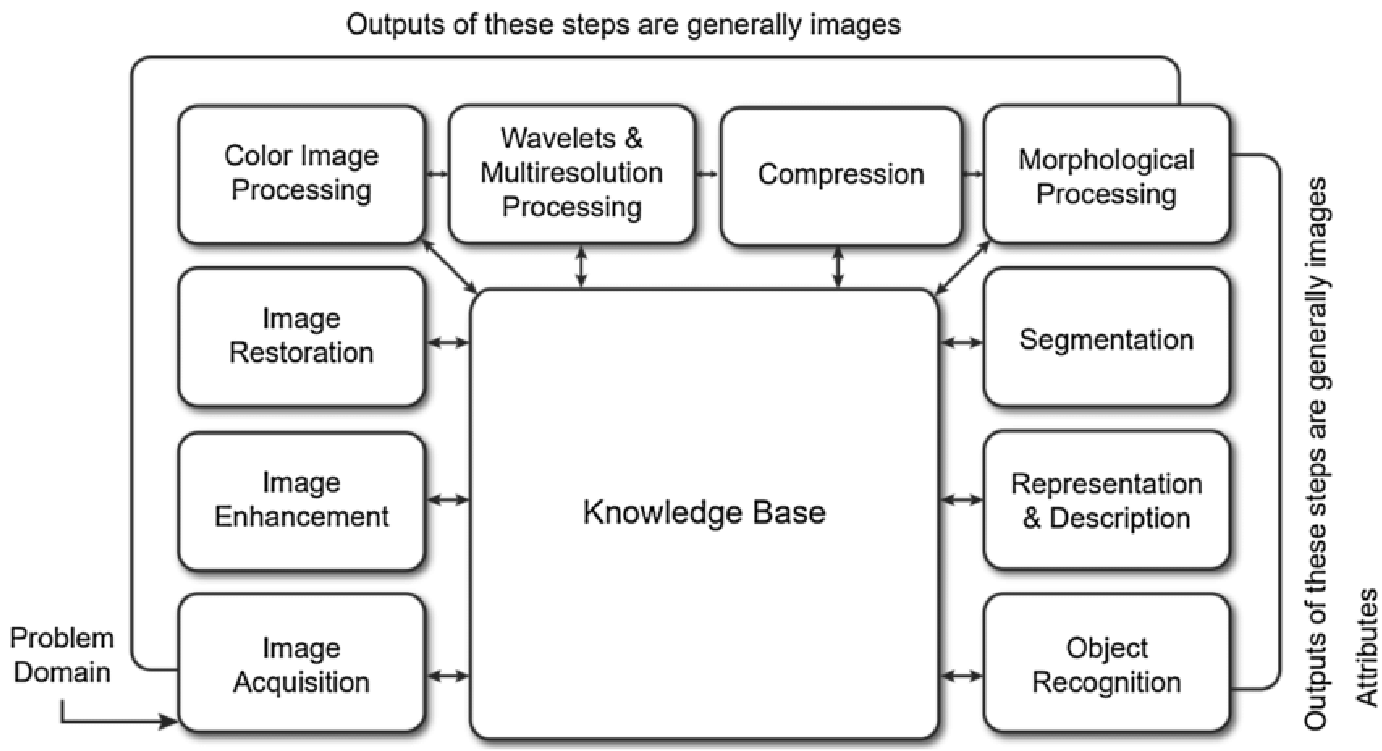


Figure 1 Digital Image Processing System

1. **Image Acquisition**
   Image acquisition is the first step of the fundamental steps of DIP. In this stage, an image is given in the digital form. Generally, in this stage, pre-processing such as scaling is done.

2. **Image Enhancement**
   Image enhancement is the simplest and most attractive area of DIP. In this stage details which are not known, or we can say that interesting features of an image is highlighted. Such as brightness, contrast, etc...

3. **Image Restoration**
   Image restoration is the stage in which the appearance of an image is improved.

4. **Color Image Processing**
   Color image processing is a famous area because it has increased the use of digital images on the internet. This includes color modeling, processing in a digital domain, etc....

5. **Wavelets and Multi-Resolution Processing**
   In this stage, an image is represented in various degrees of resolution. Image is divided into smaller regions for data compression and for the pyramidal representation.

6. **Compression**
   Compression is a technique which is used for reducing the requirement of storing an image. It is a very important stage because it is very necessary to compress data for internet use.

7. **Morphological Processing**
This stage deals with tools which are used for extracting the components of the image, which is useful in the representation and description of shape.

8. **Segmentation**
In this stage, an image is a partitioned into its objects. Segmentation is the most difficult tasks in DIP. It is a process which takes a lot of time for the successful solution of imaging problems which requires objects to identify individually.

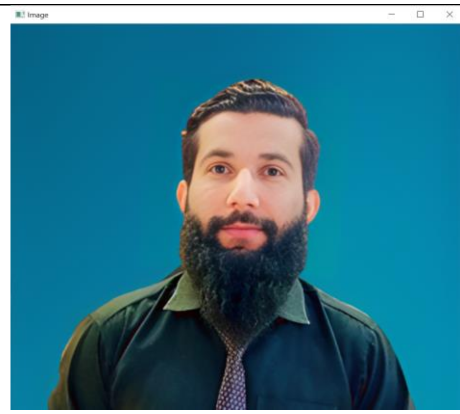
9. **Representation and Description**
Representation and description follow the output of the segmentation stage. The output is a raw pixel data which has all points of the region itself. To transform the raw data, representation is the only solution. Whereas description is used for extracting information's to differentiate one class of objects from another.

10. **Object recognition**
In this stage, the label is assigned to the object, which is based on descriptors.

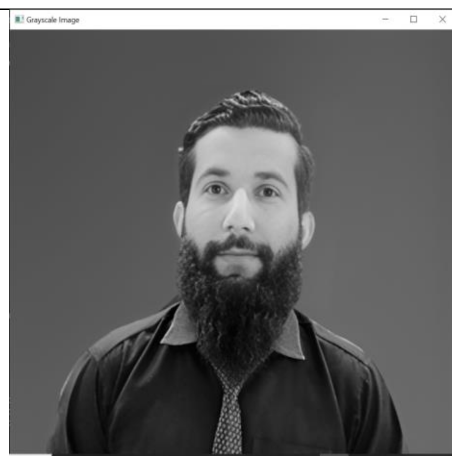
11. **Knowledge Base**
Knowledge is the last stage in DIP. In this stage, important information of the image is located, which limits the searching processes. The knowledge base is very complex when the image database has a high-resolution satellite.

Basic Image Operations:

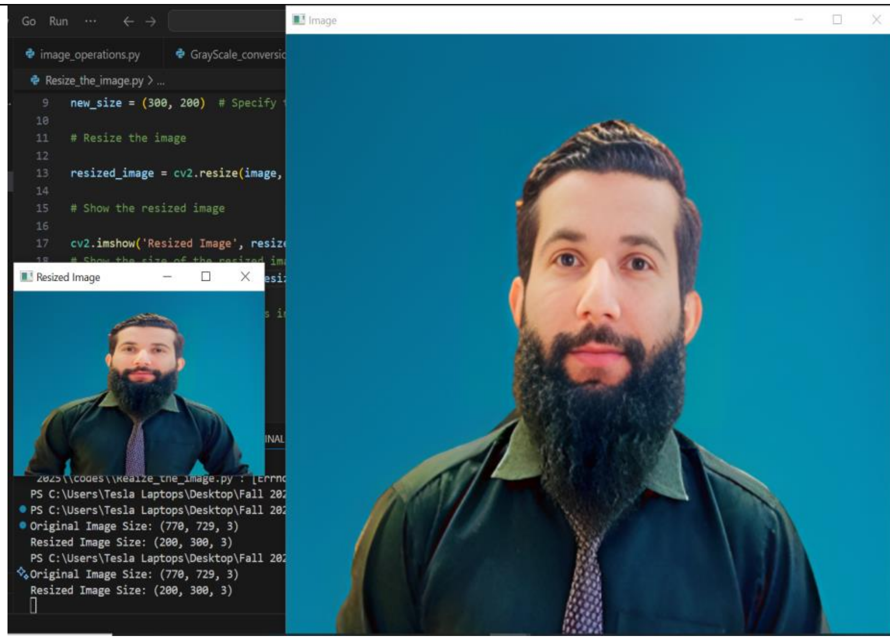
Reading and Displaying an Image

import cv2
image = cv2.imread('image.png')
 	
cv2.imshow('Image', image)
 	
# Wait for a key press (0 means indefinitely) 	
cv2.waitKey(0)
 	
# Close all OpenCV windows 	
cv2.destroyAllWindows()
		Description:
		This code uses OpenCV to read an image from a file called **'image.png'** and displays it in a window. **cv2.imshow()** displays the image, and **cv2.waitKey(0)** waits for a key press before closing the window.

Grayscale Conversion

import cv2
image = cv2.imread('image.jpg')
 	
# Convert to grayscale 	
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
 	
cv2.imshow('Grayscale Image', gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
		Description:
		Description: This code converts the loaded color image to grayscale using **cv2.cvtColor().** The parameter **cv2.COLOR_BGR2GRAY** specifies the color conversion type.

Resizing an Image

# Resize the image 	
new_size = (300, 200)
resized_image = cv2.resize(image, new_size)
		Description:
		This code resizes the image to a new size of (300, 200) pixels using cv2.resize().
		
		.

Drawing Shapes on an Image

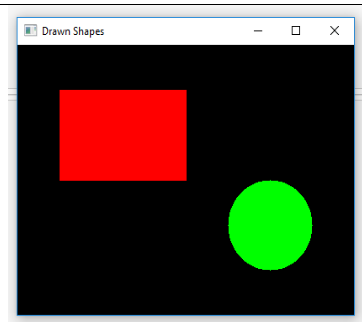
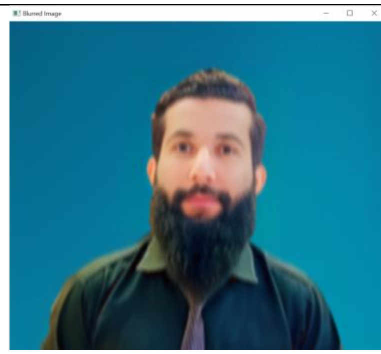
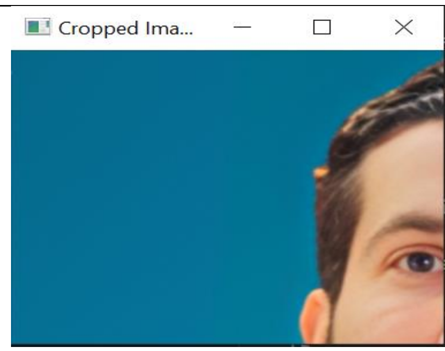
# Create a blank image 	
image = np.zeros((300, 400, 3), dtype=np.uint8)
 	
# Draw a red rectangle 	
cv2.rectangle(image, (50, 50), (200, 150), (0, 0, 255), -1)
# Draw a green circle 	
cv2.circle(image, (300, 200), 50, (0, 255, 0), -1)
		Description:
		This code creates a blank image and uses cv2.rectangle() and cv2.circle() to draw a red rectangle and a green circle on it.
		.

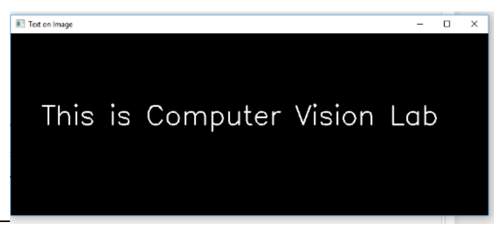
Image Filtering (Blur)

# Apply Gaussian blur 	
blurred_image = cv2.GaussianBlur(image, (5, 5), 0)
		Description:
		This code applies Gaussian blur to the image using cv2.GaussianBlur() to reduce noise and smooth the image. The (5, 5) parameter is the kernel size, and 0 is the standard deviation.

Cropping an Image

import cv2
import numpy as np
 	
image = cv2.imread('image.jpg')
# Crop a region of interest (ROI) 	
roi = image[100:300, 150:350]
cv2.imshow('Cropped Image', roi)
		Description:
		This code uses NumPy array slicing to crop a region of interest (ROI) from the image.

Adding Text to an Image

import cv2
import numpy as np
#image = np.zeros((300, 800, 3), dtype=np.uint8)
image = cv2.imread('image.png')
 	
# Set the target width and height 	
target_width = 800
target_height = 600
# Resize the image 	
resized_image = cv2.resize(image, (target_width, target_height))
# Add text 	
text = "This is Computer Vision Lab"
font = cv2.FONT_HERSHEY_SIMPLEX
	
	
		Description:
		This code adds text to a blank image using cv2.putText(). You can specify the font, text position, size, color, and thickness.
		.

cv2.putText(resized_image, text, (50, 150), font, 1.5, (0, 0, 255), 2) cv2.imshow('Text on Image', resized_image) cv2.waitKey(0) cv2.destroyAllWindows()		

Analyzing Image Pixel Values with Pandas

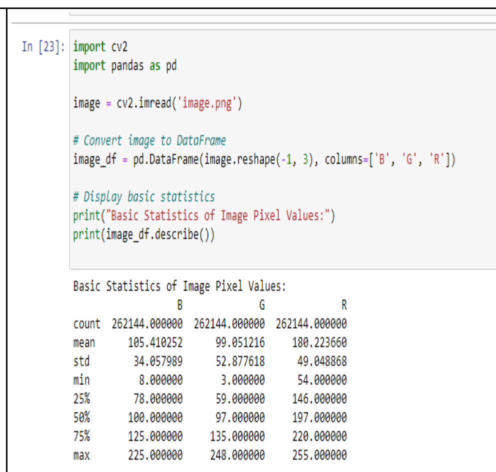
import cv2
import pandas as pd
 	
image = cv2.imread('image.jpg')
 	
# Convert image to DataFrame 	
image_df = pd.DataFrame(image.reshape(-1, 3), columns=['B', 'G', 'R'])
 	
# Display basic statistics 	
print("Basic Statistics of Image Pixel Values:")
print(image_df.describe())
	
	In [23]: import cv2
	import pandas as pd
	
	image = cv2.imread('image.png')
	
	# Convert image to DataFrame
	image_df = pd.DataFrame(image.reshape(-1, 3), columns=['B', 'G', 'R'])
	
	# Display basic statistics
	print("Basic Statistics of Image Pixel Values:")
	print(image_df.describe())
	
	Basic Statistics of Image Pixel Values:
	B G R
	count 262144.000000 262144.000000 262144.000000
	mean 105.410252 99.051216 180.223660
	std 34.057989 52.877618 49.048868
	min 8.000000 3.000000 54.000000
	25% 78.000000 59.000000 146.000000
	50% 100.000000 97.000000 197.000000
	75% 125.000000 135.000000 220.000000
	max 225.000000 248.000000 255.000000
		Description:
		This code converts the image pixel values into a Pandas DataFrame and calculates basic statistics (mean, std, min, max, etc.) of the color channels.

Image Thresholding

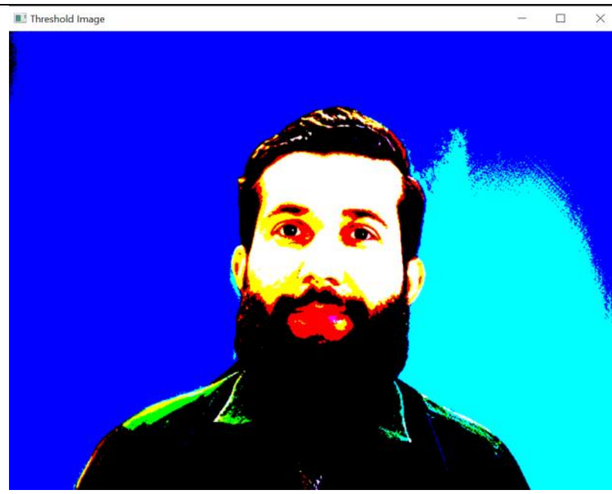
import cv2 import pandas as pd image = cv2.imread('image.jpg') ret, thresholded_image = cv2.threshold(image, 150, 255, cv2.THRESH_BINARY) cv2.imshow('Threshold Image', thresholded_image) cv2.waitKey(0) cv2.destroyAllWindows()		Description: This code converts a grayscale image into a binary image using thresholding. Pixels with values above 127 become white (255), and those below become black (0).

Image Rotation

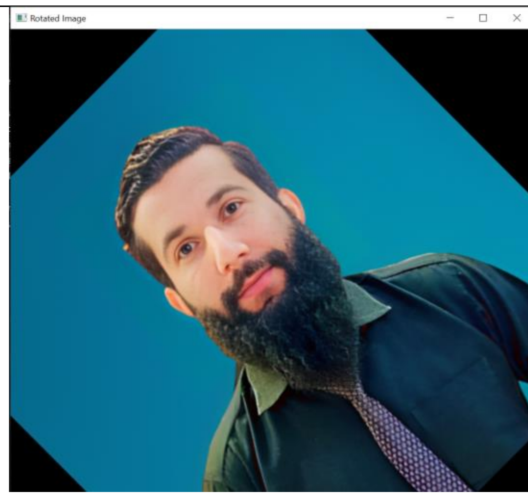
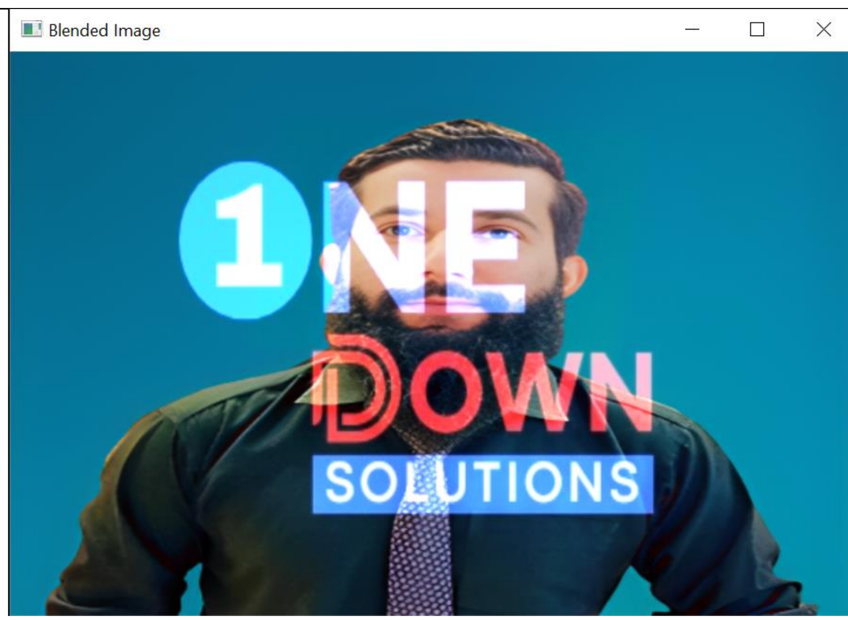
import cv2
 	
image = cv2.imread('image.png')
# Rotate the image 	
rotation_matrix = cv2.getRotationMatrix2D((image.shape[1] / 2, image.shape[0] / 2), 45, 1)
rotated_image = cv2.warpAffine(image, rotation_matrix, (image.shape[1], image.shape[0]))
cv2.imshow('Rotated Image', rotated_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
		Description:
		This code rotates the image by 45 degrees using an affine transformation.

Image Blending (Addition)

import cv2
import numpy as np
# Load the two images 	
image1 = cv2.imread('image1.jpg')
image2 = cv2.imread('image2.jpg')
# Ensure both images are the same size 	
image1 = cv2.resize(image1, (image2.shape[1], image2.shape[0]))
# Blend the images using the addition method 	
blended_image = cv2.add(image1, image2)
#display the image
		Description:
		This code blends two images by linearly combining them with specified weights.

Histogram Equalization

import cv2
 	
image = cv2.imread('image.jpg',
cv2.IMREAD_GRAYSCALE)
 	
# Apply histogram equalization 	
equalized_image = cv2.equalizeHist(image)
 	
cv2.imshow('Equalized Image',
equalized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
		Description:
		This code enhances the contrast of a grayscale image using histogram equalization..

Bitwise Operations

import cv2
import numpy as np
 	
# Load the original image 	
image = cv2.imread('image.png')
height, width, _ = image.shape
 	
# Create a binary mask image (white rectangle on a black background) 	
mask = np.zeros((height, width), dtype=np.uint8)
cv2.rectangle(mask, (100, 100), (400, 300), 255, -1)
 	
# Perform bitwise operations 	
bitwise_and = cv2.bitwise_and(image, image,
mask=mask)
bitwise_or = cv2.bitwise_or(image, image,
mask=mask)
bitwise_xor = cv2.bitwise_xor(image, image,
mask=mask)
bitwise_not = cv2.bitwise_not(image, mask=mask)
 	
# Display the original image and the results 	
cv2.imshow('Original Image', image)
cv2.imshow('Mask', mask)
cv2.imshow('Bitwise AND', bitwise_and)
cv2.imshow('Bitwise OR', bitwise_or)
cv2.imshow('Bitwise XOR', bitwise_xor)
cv2.imshow('Bitwise NOT', bitwise_not)
 	
cv2.waitKey(0)
cv2.destroyAllWindows()
	Description:
	In this code, we first load the original image "image.png". We then create a binary mask image with a white rectangle on a black background using np.zeros() and cv2.rectangle(). The mask defines the region where the bitwise operations will be applied.
	The code performs bitwise AND, OR, XOR, and NOT operations on the original image using the mask. The results are displayed using cv2.imshow().
	We create a binary mask image using np.zeros() with the same dimensions as the loaded image. This creates a black image with the same height and width. Then, we draw a white rectangle on this mask using cv2.rectangle(). The (100, 100) coordinate is the top-left corner of the rectangle, (400, 300) is the bottom-right corner, and 255 is the color value for white. The -1 parameter means we want the rectangle to be filled.

Class Quiz

Instructions

- You have **only 30 minutes.**
- This Quiz has just one question worth **50 marks.**
- It is an **open book exam.**
- **Internet is available**, and you may use your **PC, gadgets, Python libraries, and documentation**.
- **You may consult a teacher if needed, but no discussion with other students is allowed.**
- This is **strictly individual work**.
- Your project idea must be **original**. Ready-made code, GitHub repositories, or direct copy-paste from the internet will not be accepted.
- Marks will depend on **creativity, scope, implementation quality, and how well you manage your time.**

Question (50 Marks)

In the given time, design and implement a **Python-based project** of your choice that belongs to the field of **Computer Science**. The project must be large and meaningful enough to **justify 50 marks** — not a small script or trivial automation.

At the end, you should provide:

1. The **working code** of your project.
2. A **short explanation** describing the project's scope, purpose, and importance.
3. A confirmation that your project is **your own work** and not copied from any online source.

⚡ Think fast, code smart, and prove your skills.

Lab Tasks:

1. **Grocery List:** Your task is to create a simple program to manage a grocery list. Your program should allow the user to add items to the list, remove items, and display the current list of items. Write a Python program that implements this grocery list management system.
2. **Student Record System**: Your task is to build a simple student record system. Each student is identified by their student ID, and you need to store their names and corresponding grades. Your program should allow adding students, updating their grades, and displaying the student records. Write a Python program that implements this student record system using a dictionary.
3. Load an image from file and display it, convert to grayscale, resize it to specific size using OpenCV.
4. Create a blank image and draw basic shapes like rectangles and circles on it using OpenCV.
5. Load an image and apply Gaussian blur, crop at specific region using OpenCV, NumPy array slicing.
6. Load an image and add text to it using OpenCV, repeat this for creating a blank image and add text on that as well.
7. Load a grayscale image and apply binary thresholding to it and the rotate at 60^0 using OpenCV.
8. Load two images and blend them using OpenCV, and the convert it to grayscale and apply histogram equalization to enhance contrast.
9. Create binary images and perform bitwise AND, OR, XOR, and NOT operations using OpenCV and NumPy.
10. Load an image and convert its pixel values to a Pandas DataFrame, then analyze basic statistics, also apply a mask to it using bitwise AND operation.

Note: Use Subplot function where more than one image is required to be displayed/shown.