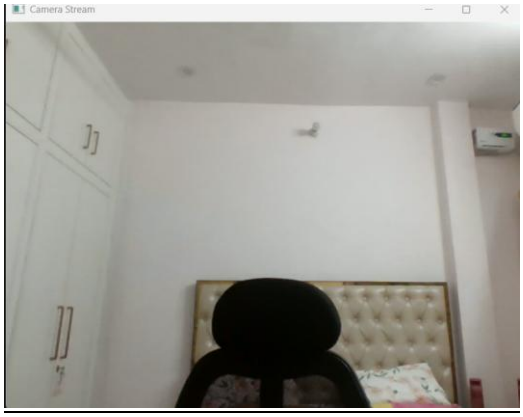
COMPUTER VISION

CODE 1:

Concept used: Capturing live video using OpenCV.

OUTPUT: Live stream



OBSERVATIONS:

1 The program turns on a camera

2 It keeps taking pictures from the camera continuously

3 Each picture is shown on your screen in a window

4 You can stop the program by pressing a key like q

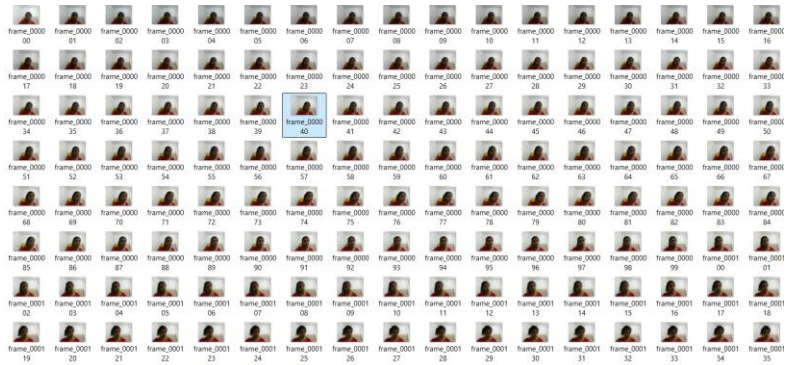
5 After stopping it closes the camera and the window properly

---

CODE 2

Concept used: Saving frames from webcam stream.

OUTPUT: Live camera feed + saved frames



OBSERVATIONS:

1 The program turns on a camera
2 It captures frames continuously from the camera
3 Each frame is shown on screen
4 Each frame is saved in a folder named frames
5 You can stop the program by pressing a key like q
6 After stopping it releases the camera and closes windows

CODE 3

C**oncept used:** Loading and displaying an image using Pillow and OpenCV.

OUTPUT: Display a single image

OBSERVATIONS:
1 The program loads an image from disk
2 It checks if the image exists
3 Shows the image in a window
4 Waits until a key is pressed to close
5 Closes the window after key press

CODE 4

Concept used: Flipping an image using OpenCV in Python.

OUTPUT: Original and flipped images

OBSERVATIONS:

1 The program loads an image

2 It creates vertical, horizontal, and both-direction flips

3 Shows original and flipped images

4 Waits for a key press to close

5 Closes all windows after key press

CODE 5

Concept used: Resizing an image using OpenCV in Python.

OUTPUT: Original and resized images

OBSERVATIONS:

1 Loads an image from disk

2 Checks if image loaded correctly

3 Resizes the image to 300x300 pixels

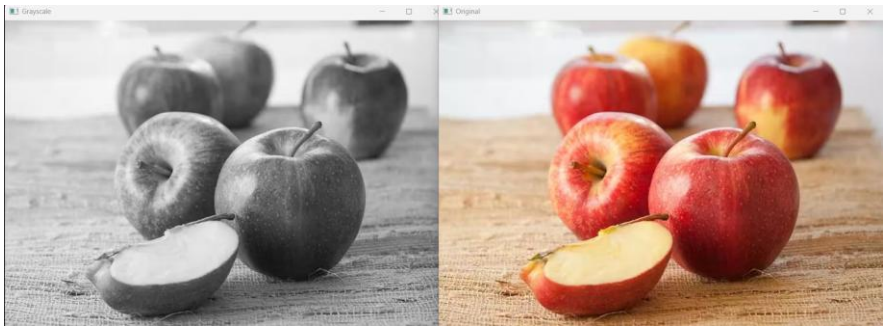
4 Shows original and resized images

5 Saves the resized image to disk
6 Closes windows after key press

CODE 6

Concept used: Converting an image to grayscale using OpenCV.

OUTPUT: grayscale images

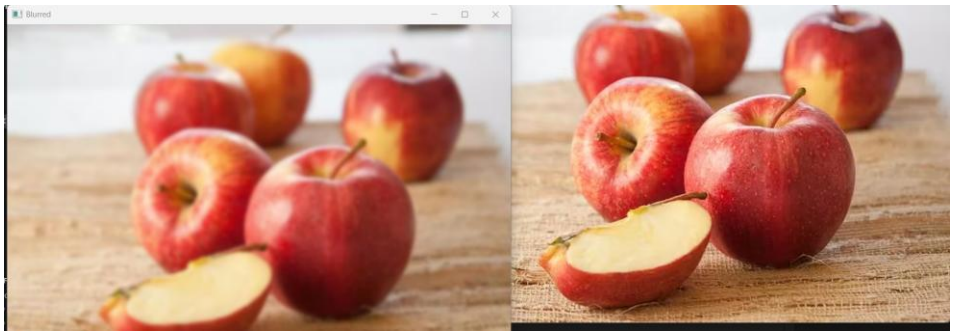


OBSERVATIONS:
1 Loads an image from disk
2 Converts the image to grayscale
3 Shows original and grayscale images
4 Optionally saves the grayscale image
5 Waits for a key press then closes all windows

CODE 7

Concept used: Applying Gaussian blur using OpenCV in Python.

OUTPUT: Original and blurred images



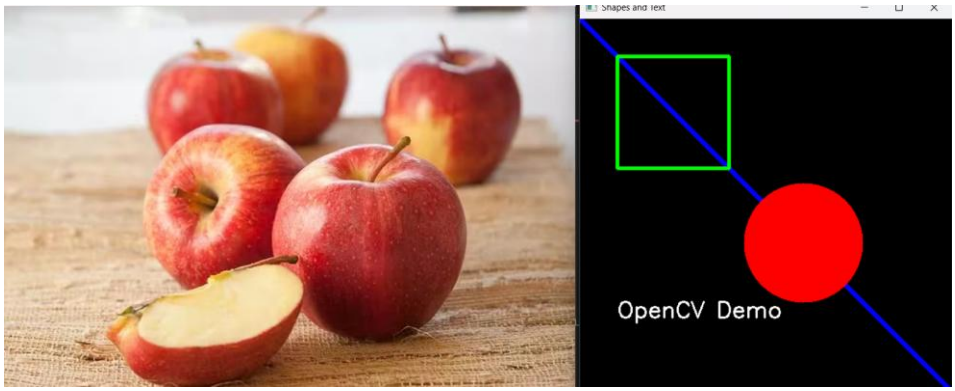
OBSERVATIONS:
1 Loads an image
2 Applies Gaussian blur with a 15x15 kernel
3 Shows original and blurred images

4 Optionally saves the blurred image
5 Waits for a key press then closes all windows

CODE 8

Concept used: Drawing shapes and text using OpenCV in Python.

OUTPUT: Image with shapes and text

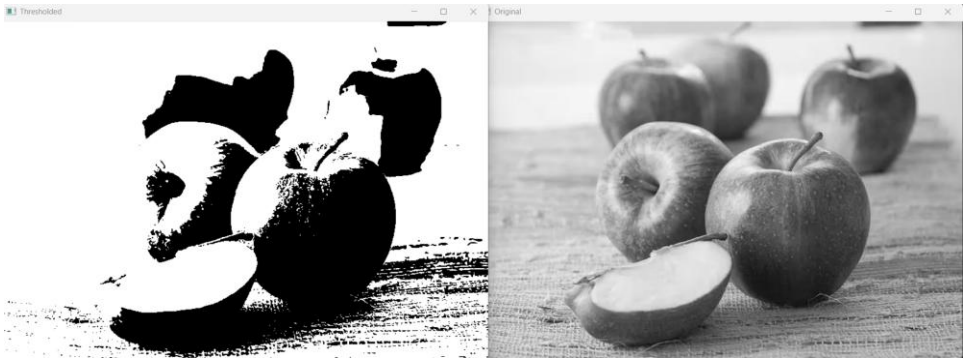


OBSERVATIONS:
1 Creates a black blank image
2 Draws a line, rectangle, and circle
3 Adds text on the image
4 Displays the image
5 Closes window after key press

CODE 9

Concept used: Performing binary thresholding using OpenCV.

OUTPUT: Original and threshold images



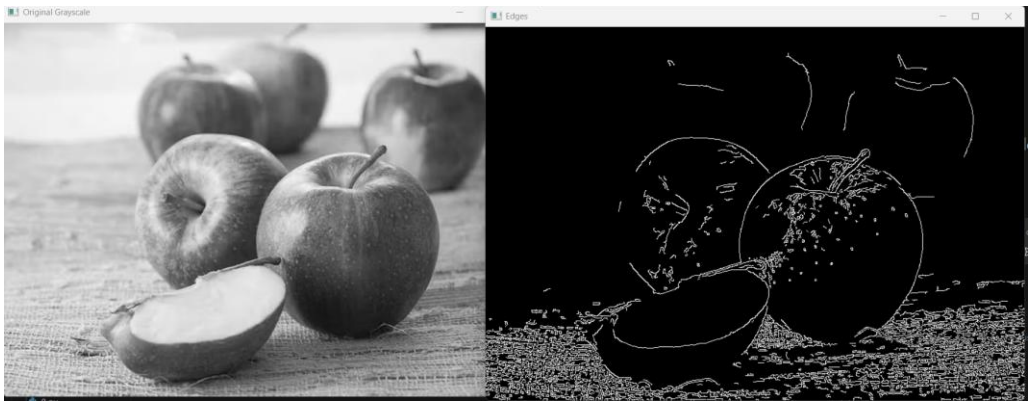
OBSERVATIONS:
1 Loads a grayscale image

2 Applies binary thresholding to convert image into black and white
3 Shows original and thresholded images
4 Waits for a key press then closes all windows

CODE 10

Concept used: Detecting edges using Canny operator in OpenCV.

OUTPUT: Edge-detected image

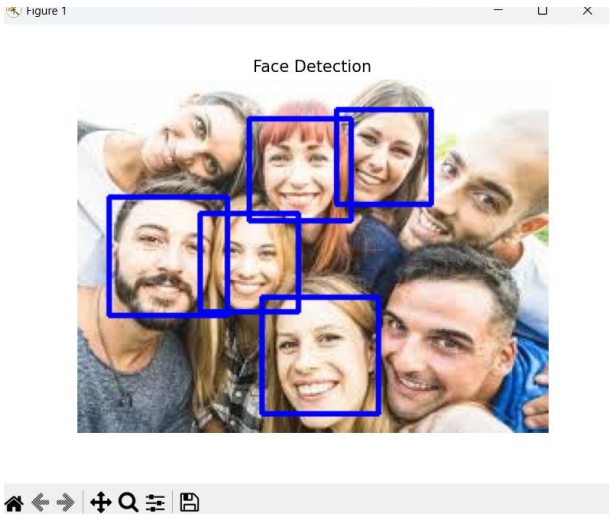


OBSERVATIONS:
1 Loads a grayscale image
2 Detects edges using Canny edge detector
3 Shows edges in a window
4 Waits for a key press then closes all windows

CODE 11

Concept used: Face detection using OpenCV.

OUTPUT: Image with faces detected



OBSERVATIONS:

1 Loads an image and Haar cascade classifier for face detection

2 Converts image to grayscale

3 Detects faces in the image

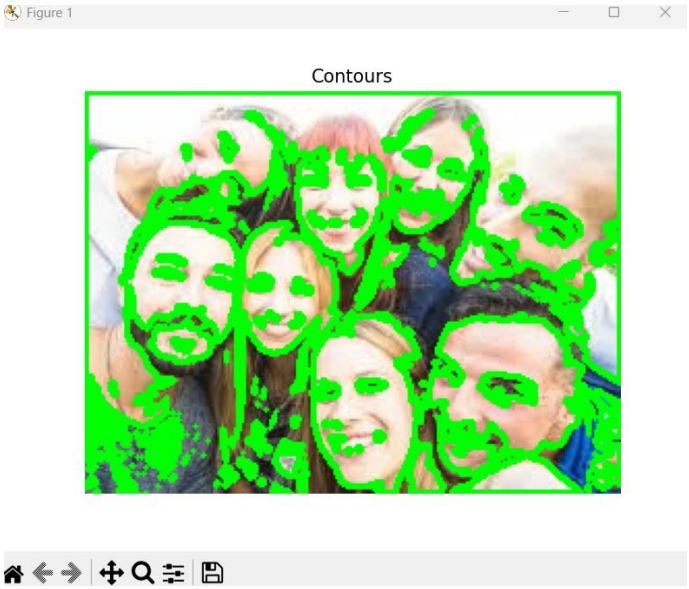
4 Draws rectangles around detected faces

5 Shows the result and closes window after key press

CODE 12

Concept used: Detecting and drawing contours using OpenCV.

OUTPUT: Image with contours drawn



OBSERVATIONS:

1 Loads an image and converts it to grayscale

2 Applies thresholding to get binary image

3 Finds contours in the image

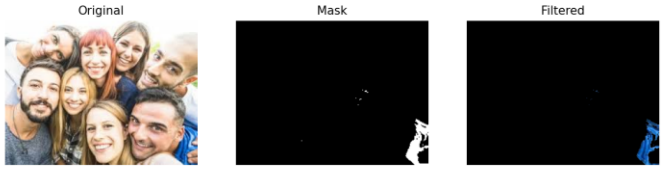
4 Draws contours on original image

5 Shows the result and closes window after key press

CODE 13

Concept used: Image dilation using OpenCV in Python.

OUTPUT: Color detection and masking



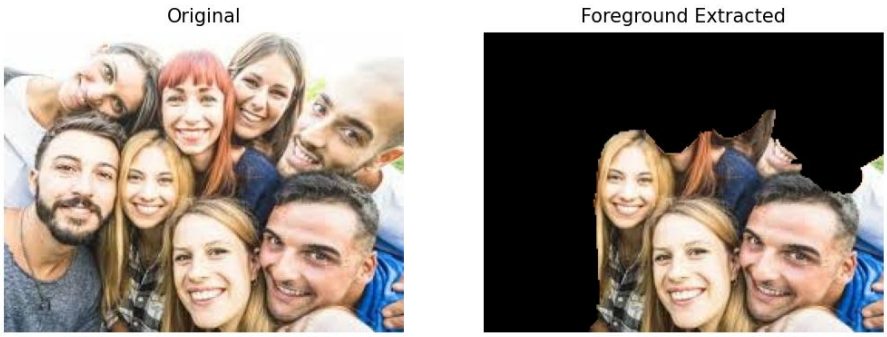
OBSERVATIONS:

1 Loads an image and converts it to HSV color space
2 Defines blue color range
3 Creates mask for blue color
4 Applies mask to original image to filter blue objects
5 Shows original, mask, and filtered images
6 Waits for key press then closes all windows

CODE 14

Concept used: Foreground extraction from image.

OUTPUT:



OBSERVATIONS:

1 Loads an image and creates mask for GrabCut
2 Initializes background and foreground models
3 Defines rectangular ROI for GrabCut
4 Applies GrabCut algorithm to extract foreground
5 Shows original and foreground images
6 Waits for key press then closes all windows

CODE 15

Concept used: Live blue object tracking.

OUTPUT: Live blue object tracking

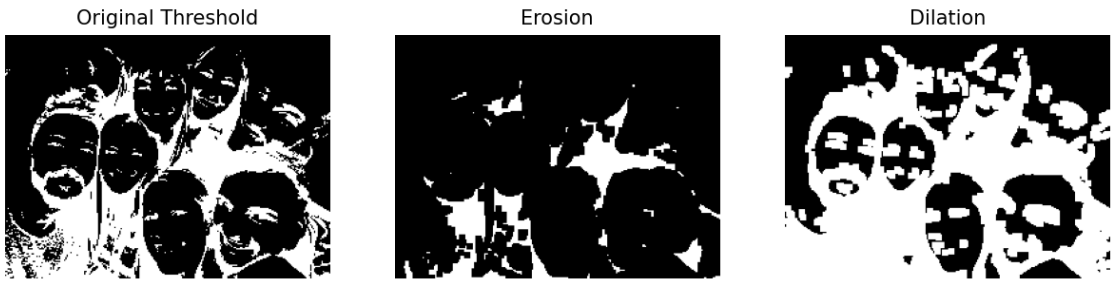
OBSERVATIONS:

1 Turns on the camera
2 Captures frames continuously
3 Converts frames to HSV color space
4 Detects and masks blue objects
5 Shows original frame, mask, and tracked output
6 Stops when key q is pressed
7 Releases camera and closes all windows

CODE 16

Concept used: Morphological operations on text image

OUTPUT:



OBSERVATIONS:

1 Loads a grayscale text image

2 Applies binary inverse thresholding
3 Performs erosion to shrink white regions
4 Performs dilation to enlarge white regions
5 Shows original, erosion, and dilation results
6 Waits for key press then closes all windows