**Aim:**

1. To Find the number of coins in the given image

**Software/ Packages Used:**

1. Pycharm IDE

2. Libraries used:

• NumPy

• opencv-python

**Introduction:**

Coin detection using OpenCV involves leveraging the library's capabilities to analyze images and extract relevant features that distinguish different coins. The process typically encompasses tasks such as edge detection, contour analysis, and template matching. With the aid of OpenCV, developers can create robust and efficient coin detection algorithms that can be integrated into diverse applications, enhancing the automation and accuracy of coin-related processes.

**Theory:**

To understand coin detection using OpenCV, it's essential to grasp the underlying theory and concepts involved.

Image Preparation:

Convert the image to grayscale to simplify processing.

Reduce noise in the image through techniques like Gaussian blurring.

Detecting Edges:

Use the Canny edge detector to find edges in the image.

Finding Contours:

Identify contours (object boundaries) in the image.

Circle Detection:

Use the Hough Circle Transform to find circular objects, like coins.

Coin Recognition:

Classify coins based on their size and diameter.

Optionally, improve accuracy using template matching.

Thresholding:

Convert the image to binary format through thresholding.

Clean-up:

Remove false positives and refine detected coin regions.

OpenCV Implementation:

Use OpenCV functions like cv2.cvtColor(), cv2.Canny(), and cv2.HoughCircles() for implementation.

**PROGRAM:**

**# Import necessary libraries**

import cv2

import numpy as np

import matplotlib.pyplot as plt

**# Read the image from file**

image = cv2.imread("C:\\C3.jpg")

**# Convert the image to grayscale**

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

plt.imshow(gray, cmap='gray')

**# Apply Gaussian blur to the grayscale image**

blur = cv2.GaussianBlur(gray, (11, 11), 0)

plt.imshow(blur, cmap='gray')

**# Use Canny edge detector to find edges in the blurred image**

canny = cv2.Canny(blur, 30, 150, 3)

plt.imshow(canny, cmap='gray')

**# Dilate the edges to connect nearby edges and close gaps**

dilated = cv2.dilate(canny, (1, 1), iterations=2)

plt.imshow(dilated, cmap='gray')

**# Find contours in the dilated image**

(cnt, hierarchy) = cv2.findContours(dilated.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

**# Convert image to RGB for visualization**

rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

**# Draw contours on the RGB image**

cv2.drawContours(rgb, cnt, -1, (0, 255, 0), 2)

plt.imshow(rgb)

**# Print the number of detected coins**

print('Coins in the image: ', len(cnt))

**# Display the images**

plt.show()

**# Wait for a key press and close all OpenCV windows**

cv2.waitKey(0)

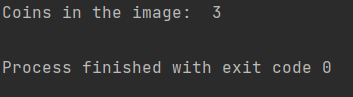
cv2.destroyAllWindows()

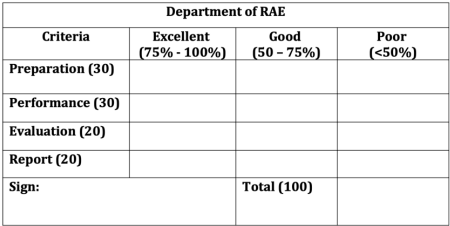
**INPUT:**



**OUTPUT:**



****



**RESULT:**

Thus we have detected the number of coins present successfully in the image using the above code.