Bharati Vidyapeeth (Deemed to be University)

Department of Engineering and Technology

Kharghar, Navi Mumbai

**Department of Artificial Intelligence and Machine Learning**



**EXPERIMENT 8**

|  |  |
| --- | --- |
| **Name: Shivam Bhosle** | **Roll No: 02** |
| **Subject: COMPUTER VISION** | **Class/Batch: B1** |
| **Date of Performance:** | **Date of Submission:** |

|  |
| --- |
| **AIM** |

To detect circles in an image using the Hough Transform method with OpenCV and Python.

|  |
| --- |
| **Theory/Procedure/Algorithm** |

The Hough Transform is a technique used in image analysis, computer vision, and digital image processing to detect simple shapes, such as lines and circles, in images. The Circular Hough Transform (CHT) is specifically used for detecting circles in images.

The key principle is that a circle can be represented in parametric form as: (x−a)2+(y−b)2=r2(x - a)^2 + (y - b)^2 = r^2(x−a)2+(y−b)2=r2 Where:

* (a,b)(a, b)(a,b) are the coordinates of the circle's center,
* rrr is the radius of the circle.

For each edge point in the image, a set of potential circles (for different radii) is drawn, and a voting mechanism is employed in a 3D accumulator space (center coordinates and radius). Peaks in this space correspond to potential circles in the image.

OpenCV provides the HoughCircles() function for detecting circles using the Hough Transform.

**Procedure:**

1. **Install Libraries**: Install the required libraries, especially OpenCV.
   1. pip install opencv-python
2. **Load Image**: Load the input image using OpenCV.
3. **Preprocess the Image**:
   1. Convert the image to grayscale for easier processing.
   2. Apply a Gaussian blur to reduce noise.
4. **Apply Hough Transform**: Use the HoughCircles() function to detect circles in the image.
5. **Display the Result**: Draw the detected circles on the image and display the result.

**Algorithm:**

1. Start
2. Import the required libraries (OpenCV, NumPy).
3. Load the input image.
4. Convert the image to grayscale.
5. Apply GaussianBlur to reduce noise.
6. Use the HoughCircles function to detect circles:
   * Provide parameters like the image, method, dp (inverse ratio of accumulator resolution), minimum and maximum radius, etc.
7. Draw the detected circles on the original image.
8. Display the result.
9. End

|  |
| --- |
| **IMPLEMENTATION** |

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load an image (replace with the path to your image)

image = cv2.imread(r"C:\\Users\\Shivam 007\\Downloads\\tanjiro.webp", cv2.IMREAD\_COLOR)

# Convert the image to grayscale

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

# Apply GaussianBlur to reduce noise and improve detection accuracy

blurred\_image = cv2.GaussianBlur(gray\_image, (9, 9), 2)

# Detect circles in the image using HoughCircles

circles = cv2.HoughCircles(blurred\_image, cv2.HOUGH\_GRADIENT, dp=1.2, minDist=30,

param1=50, param2=30, minRadius=15, maxRadius=50)

# If some circles are detected, process them

if circles is not None:

circles = np.round(circles[0, :]).astype("int") # Round the coordinates to integers

# Draw each circle on the original image

for (x, y, r) in circles:

# Draw the outer circle

cv2.circle(image, (x, y), r, (0, 255, 0), 4) # Green circle

# Draw the center of the circle

cv2.circle(image, (x, y), 2, (0, 0, 255), 3) # Red dot at the center

# Display the result using OpenCV's imshow (or matplotlib if needed)

cv2.imshow("Detected Circles", image)

cv2.waitKey(0) # Press any key to close the window

cv2.destroyAllWindows()

# Optionally, use matplotlib to display the image inline if you're in Jupyter or prefer this method

# Convert the BGR image to RGB format for displaying in matplotlib

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

plt.imshow(image\_rgb)

plt.axis("off")

plt.show()



|  |
| --- |
| **Conclusion** |

The experiment successfully demonstrated how to detect circles in an image using the Hough Transform method with OpenCV. The method is highly effective for detecting circular objects in images but requires appropriate parameter tuning for different image conditions (lighting, noise, etc.).

|  |
| --- |
| **Assessment** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Timely Submission**  **(7)** | **Presentation**  **(06)** | **Understanding**  **(12)** | **Total**  **(25)** | **Sign** |
|  |  |  |  |  |