```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

#defining imshow function to get image
def imshow(title = "Image", image = None, size = 10):
    w, h = image.shape[0], image.shape[1]
    aspect_ratio = w/h
    plt.figure(figsize = (size*aspect_ratio, size))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.show()


def detecting_coins(image_path, radius_range):
    #converting to grayscale to reduce computational complexity
    image = cv2.imread(image_path, 0)
    #gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Applying GaussianBlur to reduce noise and improve edge
detection
    blurred = cv2.GaussianBlur(image, (5, 5), 0)

    # Applying Canny edge detection to detect edges
    edges = cv2.Canny(blurred, 100, 200)

    # here the Hough accumulator
    rows, cols = edges.shape
    min_radius, max_radius = radius_range
    radius_count = max_radius - min_radius
    accumulator = np.zeros((rows, cols, radius_count),
dtype=np.uint16)

    # Pre-computing sine and cosine values for efficiency
    #calculating theta cos-theta and sine-theta to get the centre of
circles
    theta = np.deg2rad(np.arange(0, 360))
    cos_theta = np.cos(theta)
    sin_theta = np.sin(theta)

    # Populating the accumulator
    # using np.argwhere to getting indices of array elements
    edge_pixels = np.argwhere(edges > 0)
    for x, y in edge_pixels:
        #r is actual radius and r_idx is radius index required for
indexing into accumulator array
        for r_idx, r in enumerate(range(min_radius, max_radius)):
            x_vals = (x - r * cos_theta).astype(int) # x-cordinates
of centre of circles for a given edge pixel
```

```
            y_vals = (y - r * sin_theta).astype(int) # y-cordinates
of centre of circles for a given edge pixel
            valid = (0 <= x_vals) & (x_vals < rows) & (0 <= y_vals)
& (y_vals        < cols)
            a_vals = a_vals[valid]
            b_vals = b_vals[valid]
            accumulator[a_vals, b_vals, r_idx] += 1

    # Finding circles based on accumulator peaks
    threshold = 0.5 * np.max(accumulator)
    circles = np.argwhere(accumulator > threshold)

    # Creating a binary mask
    binary_mask = np.zeros((rows, cols), dtype=np.uint8)
    for a, b, r_idx in circles:
        r = min_radius + r_idx
        cv2.circle(binary_mask, (b, a), r, (255), thickness=-1)

    return binary_mask

if __name__ == "__main__":
    #path of the image that I cropped from pdf
    image_path = "picture/image1.png"
    # radius just consider to avoid unnecessary computation

    radius_range = (10, 100)

    binary_mask = detecting_coins(image_path, radius_range)

    imshow("Binary Mask", binary_mask)
```