|  |  |  |
| --- | --- | --- |
|  | C:\Users\Hussi\OneDrive\Desktop\logo.jpeg |  |

# **Count coins In an image**

**Prepared By:**

**Name: Mahmoud Bahaa Eldeen Abdallah**

**ID: 210733**

**Name: Andrew Hany Nadi**

**ID: 212179**

**Introduction :**

This project focuses on developing a robust and efficient algorithm to count coins in an image using MATLAB, addressing practical needs in industries such as banking, retail, and vending systems. The algorithm leverages MATLAB's image processing toolbox to preprocess the image, isolate the coins, and accurately count them. The process begins with converting the image to grayscale, which simplifies the image by reducing it to a single intensity channel. Adaptive thresholding is then applied to convert the grayscale image to a binary image, effectively handling varying lighting conditions. Next, the image is complemented to prepare it for morphological operations, ensuring the coins are represented as solid white objects. Hole filling is performed to solidify the coin regions, followed by the removal of small objects to eliminate noise and artifacts, leaving only significant objects. Finally, connected component analysis is used to identify and accurately count the coins. Each step in this process is meticulously designed to enhance the accuracy of coin detection and counting.

This report details the methodology and purpose of each step, demonstrating the effectiveness of digital image processing techniques in solving practical object counting problems. Through this project, we highlight the potential for further applications and improvements in the field of digital image processing..

**Explanation of the method/filter/algorithm;**

**1-Image Reading and Display:**

* **Image Reading:** The ‘**imread’** function reads the image file and loads it into the workspace. This step is essential to start processing the image.
* **Image Display:** The ‘**imshow’** function displays the image to visualize the original data. This helps in understanding the content and quality of the image before processing.

### **2-Grayscale Conversion:**

* **Simplify Image Processing:** Converting the image to grayscale reduces it to a single channel (intensity), making it easier to process compared to a three-channel (RGB) image. This step is crucial as most image processing techniques operate more effectively on grayscale images.

### **3-Adaptive Thresholding**

* **Binarization:** Converts the grayscale image to a binary image, where the coins are represented as white (foreground) and the background as black. Adaptive thresholding adjusts the threshold value for different regions of the image, making it robust to varying lighting conditions. This step separates the coins from the background, which is essential for further analysis.

### **4-Image Complementation:**

* **Invert Binary Image:** The ‘**imcomplement’** function inverts the binary image so that the coins become white (foreground) and the background becomes black. This inversion is necessary because subsequent morphological operations (like hole filling and object removal) typically work with white objects on a black background.

### **5. Hole Filling:**

* **Create Solid Objects:** The ‘**imfill’** function fills any holes inside the white regions (coins). This step ensures that each coin is represented as a solid white object, which is important for accurately counting the coins.

**6-Filter the image:**

* **Remove Noise and Artifacts:** The ‘**bwareaopen’** function removes small white objects that have fewer than 100 pixels. This step eliminates small noise and artifacts that are not coins, ensuring that only significant objects (coins) remain in the image.

### **7. Coin Counting:**

* **Identify and Count Coins:** The ‘**bwconncomp’** function identifies connected components in the binary image. Each connected component corresponds to a separate coin. The **NumObjects** property of the **coins** structure provides the total number of connected components, which is the number of coins. This final step quantifies the number of coins present in the image.

**Code**

%Image read

img = imread (‘ P3 - eight - count the number of coins.tif ’);

figure,

imshow(img);

% convert it into Binary vertion Image

BW = imbinarize(img,0.5); % 0.5 is atest threshold value

figure,

imshow(BW);

% complement the image

BW1 = imcomplement(BW);

figure,

imshow(BW1);

% Fill the holes to make a solid objects

BW2 = imfill(BW1,'holes');

figure,

imshow(BW2);

% Filter the image

BW3 = bwareaopen(BW2,10);

% Now find the number of coins

coins = bwconncomp(BW2);

NoOfCoins = coins.NumObjects;

disp('The Total Number of Coins are :');

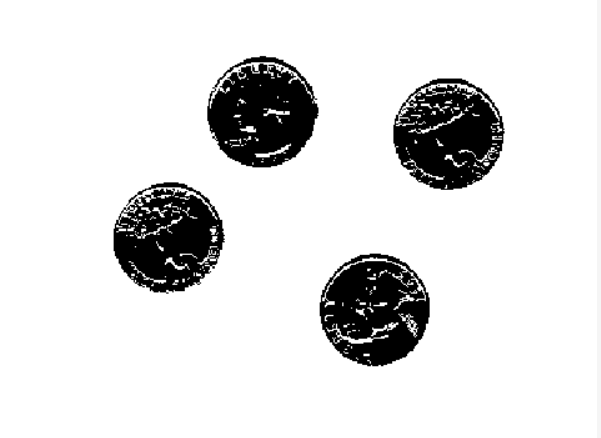
disp(NoOfCoins);

**screenshots for every phase**

**1-Original image**

****

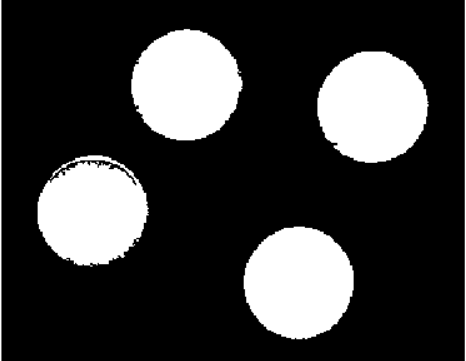
**2-convert it into Binary vertion Image**

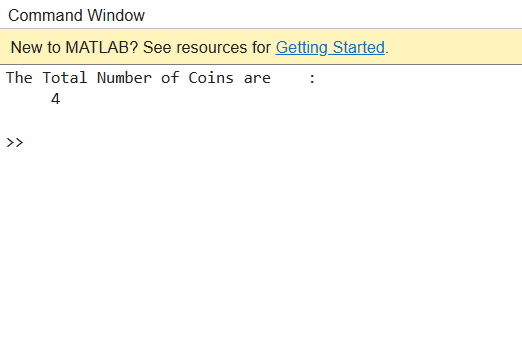
****

**3- complement the image**

****

**4- Fill the holes to make a solid objects**

****

**Output number of coins**