<u>Assignment no:7</u>

1. **Aim:** To implement image segmentation using global thresholding.
2. **Software Tool Used:** MATLAB
3. **Theory:**

In an image segmentation using thresholding techniques, **global thresholding** and **Otsu's thresholding** are commonly used methods to separate foreground from background. Here's an overview of the theory behind each method and how they apply to different types of images:

1. **Global Thresholding:** Global thresholding is a straightforward segmentation technique where a single threshold value T, is applied across the entire image to distinguish foreground pixels from background pixels. This approach works well when there is a clear intensity difference between foreground and background. The basic principle is:

Pixel Segmentation Rule:
- If f (x, y) >T, the pixel is classified as foreground (white).
- If f (x, y) ≤T, the pixel is classified as background (black).

The choice of T is crucial, and it can be selected based on histogram analysis. However, global thresholding is less effective for images with varying illumination or noise, as it assumes uniform lighting across the image.

2. **Otsu's Thresholding:** Otsu's thresholding is a global thresholding method that automatically determines an optimal threshold T by maximizing the variance between foreground and background classes. This method assumes a bimodal histogram, meaning there are two prominent intensity peaks representing foreground and background. Otsu's method works by:
   - Compute the Histogram:

$$p(i) = \frac{\text{number of pixels with intensity } i}{\text{total number of pixels}}.$$

   - Calculate Cumulative Sums and Means:

$$\text{Cumulative Sum } \omega(t) = \sum_{i=0}^{t} p(i)$$

$$\text{Cumulative Mean } \mu(t) = \sum_{i=0}^{t} i \cdot p(i)$$

- Compute Global Mean Intensity:

$$\mu = \sum_{i=0}^{255} i \cdot p(i).$$

- Calculate Between-Class Variance:

$$\sigma_B^2(t) = \frac{[\mu \cdot \omega(t) - \mu(t)]^2}{\omega(t) \cdot [1 - \omega(t)]}$$

  This equation maximizes the separation between the foreground and background classes.
- Select Optimal Threshold: Find the threshold t that maximizes $\sigma_B^2(t)$. This value of t is the optimal threshold t^* that best separates the foreground from the background.
- Apply Threshold to Create Binary Image:

- Use the optimal threshold t* to convert the grayscale image into a binary image:
- Set pixels with intensity greater than t* to 1 (foreground).
- Set pixels with intensity less than or equal to t* to 0 (background).

3. **Result:**

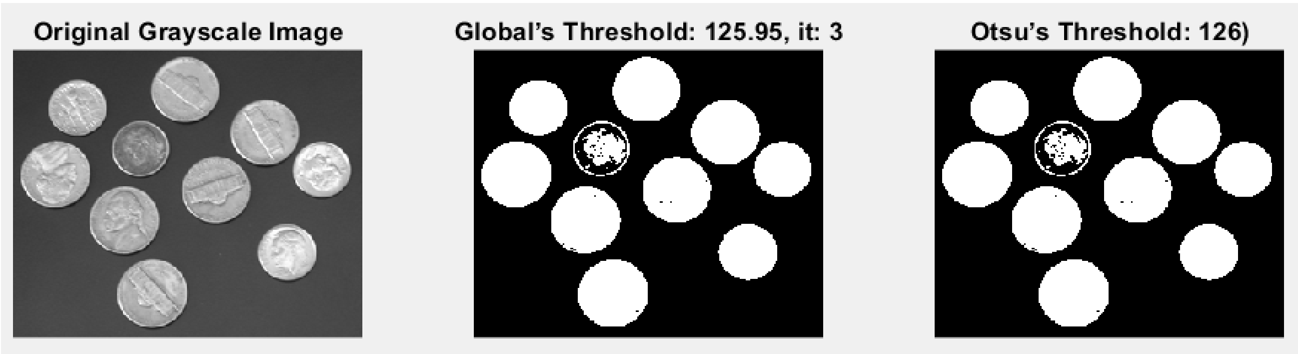

Fig (1): thresholding on the normal image

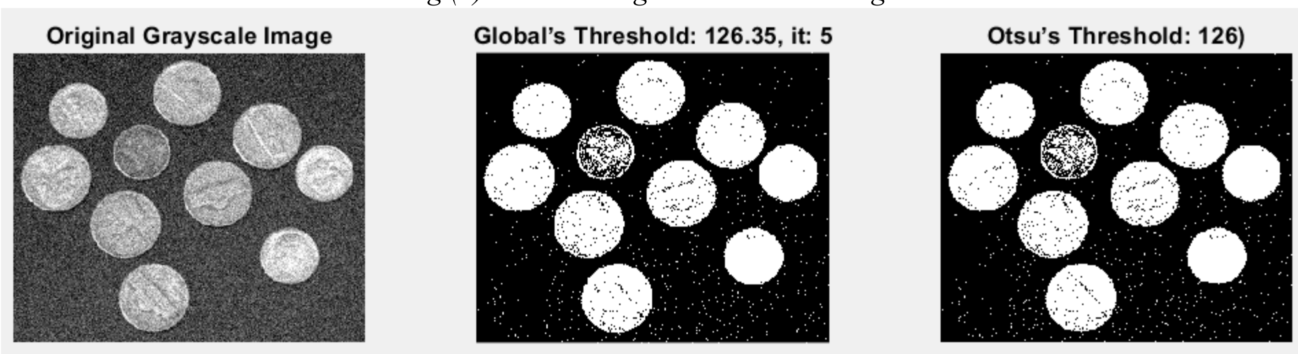

Fig (2): thresholding on the noised image variance 0.01

Fig (3): thresholding on the de-noised image

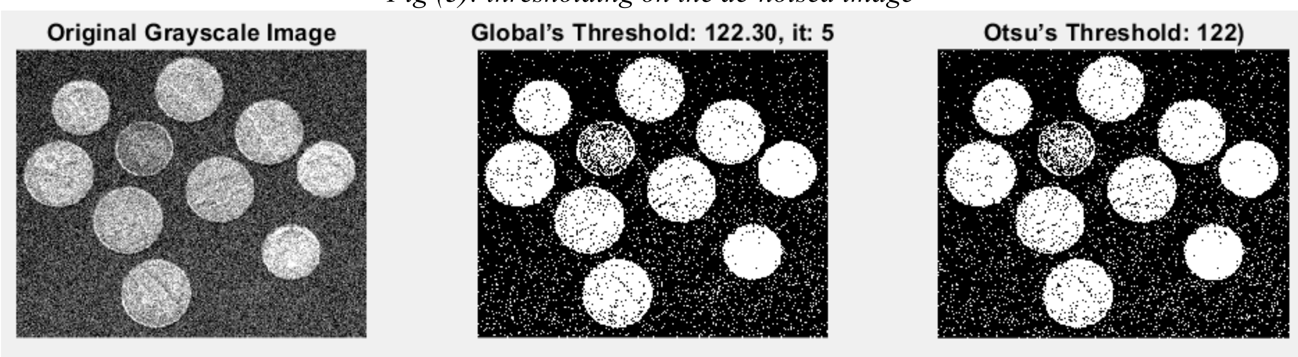

Fig (4): thresholding on the noised image variance 0.02

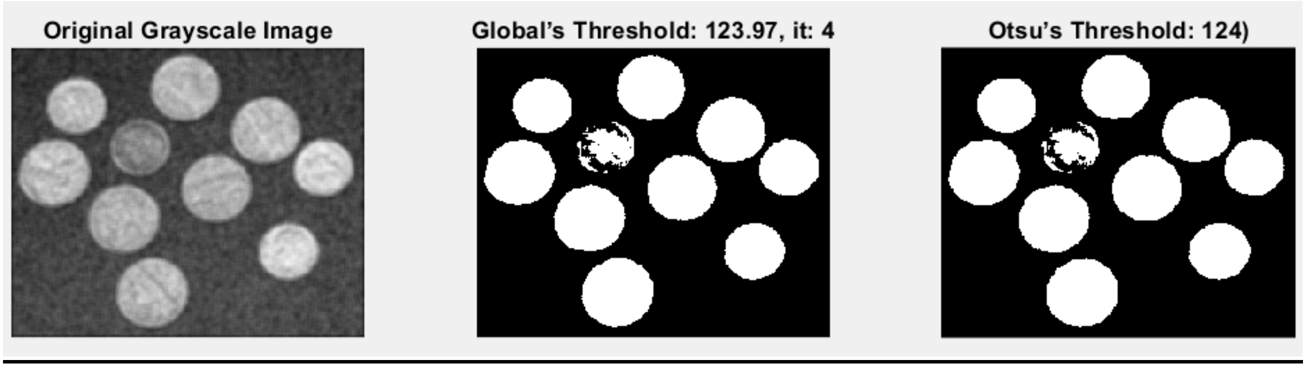

Fig (5): thresholding on the de-noised image

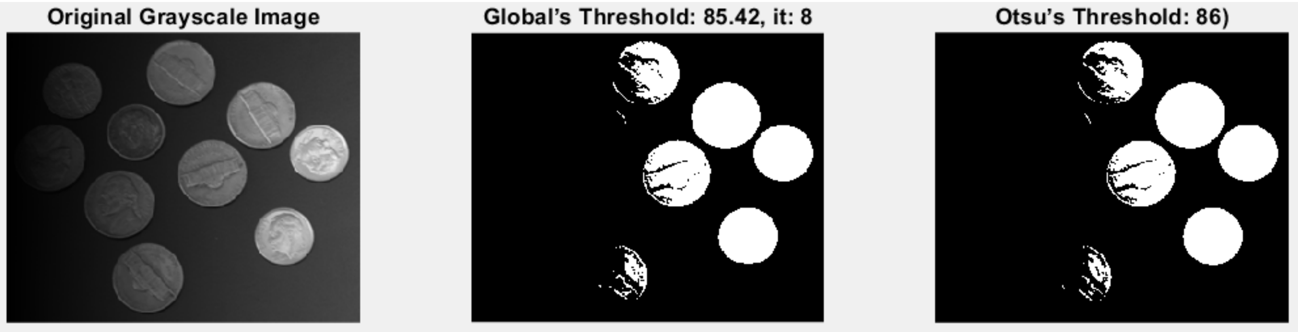

Fig (6): thresholding on the non-uniformed illuminated image

4. Discussion:

Appling **global thresholding** and **Otsu's thresholding** on different types of images, observe various outcomes due to the inherent characteristics of each image type and the limitations of each thresholding method. Here's a detailed discussion on how each type of image influences the performance of these thresholding techniques:

1. Normal Image

- **Global Thresholding**:
  - For a standard, well-lit image with good contrast between foreground and background, global thresholding can be effective.
  - A manually chosen threshold can successfully segment the object from the background if the object intensity is clearly distinguishable from the background.
  - If the contrast isn't distinct enough, choosing a single threshold value may not capture all details accurately.
- **Otsu's Thresholding**:
  - Otsu's method performs well with a normal image, especially if the histogram has a bimodal distribution (two clear peaks).
  - It calculates an optimal threshold by maximizing the variance between object and background classes, making it more reliable than manually selecting a threshold.
  - produces a clean binary image with well-separated foreground and background regions.

- **Noised Image (Variance 0.01 and 0.02):**

Variance 0.01: Introduces minor noise, creating a light grainy effect. The histogram remains bimodal but with some intensity overlap.

Variance 0.02: Higher noise leads to significant intensity fluctuations, resulting in a wider spread of pixel values. The histogram becomes less distinct, complicating threshold selection.

Thresholding Performance(0.01 variance):

- Global Thresholding might struggle due to noise, resulting in a less distinct separation between coins and background. Noise tends to blur the edges, so more misclassified pixels may appear.
- Otsu's Thresholding will also experience some degradation in performance due to noise, as the added noise increases intra-class variance, making it harder to select an optimal threshold. However, Otsu's method might still perform better than the global threshold approach in separating the two regions.

Thresholding Performance (0.02 variance):

- **Global Thresholding** will likely perform poorly here due to the higher noise level. The threshold calculation will be influenced by the widespread noise, leading to a noisy binary output.
- **Otsu's Thresholding** will also be impacted by the high noise level. While it might still give a relatively reasonable segmentation, the overall quality of separation between foreground and background is likely to decrease.

3. Denoised Image:

Denoising Effects:

- The arithmetic mean filter effectively reduces noise, especially in the image with variance 0.01, smoothing out minor fluctuations without significant detail loss.
- For the higher-variance image (0.02), a larger kernel (e.g., 5x5) further smooths the image but may blur some edges, compromising fine details.

Thresholding Performance (0.01 variance)

- **Global Thresholding** may improve compared to the previous figure as filtering will smooth out some of the noise, making the threshold more effective.
- **Otsu's Thresholding** should also see an improvement, as the reduced noise allows for better segmentation. However, the smoothing effect might slightly blur the edges of the objects (coins).

Thresholding Performance (0.02 variance)

- Global **Thresholding** will benefit from the filtering as it reduces the noise. However, as with the previous filtering, some edges might appear blurred.
- **Otsu's Thresholding** will also see improvement due to the filtering, providing a more distinct separation than without filtering.

4. Non-Uniform Illuminated Image

Non-uniform illumination introduces brightness gradients across the image, which significantly challenges both global and Otsu's thresholding methods.

- **Global Thresholding**:

- In images with varying brightness, a single threshold value is insufficient to handle regions with different lighting levels.
  - For example, brighter areas may be correctly segmented, but darker areas may not meet the threshold requirement, leading to incomplete or inaccurate object segmentation.
  - **Observation**: The output may show parts of the object merged with the background or fragmented areas due to inconsistent intensity across the image.
- **Otsu's Thresholding**:
  - Otsu's method also struggles with non-uniform illumination because it relies on a bimodal intensity distribution to identify the optimal threshold.
  - The brightness variations disrupt the histogram shape, making it difficult for Otsu's algorithm to select a threshold that segments the image accurately.
  - **Observation**: Otsu's thresholding may produce better results than global thresholding but still falls short in accurately segmenting objects with non-uniform illumination. Some areas of the object may be segmented correctly, while others remain indistinguishable from the background.

4. Conclusion:

Global and Otsu's thresholding methods provide effective results on uniformly illuminated, noise-free images. However, their performance degrades significantly on noisy and non-uniformly illuminated images. Pre-filtering improves segmentation accuracy for noisy images, especially when using Otsu's thresholding. For non-uniform illumination, adaptive methods are preferable to achieve accurate segmentation across varying brightness levels. This study demonstrates the importance of choosing appropriate thresholding methods and pre-processing techniques depending on the image characteristics to achieve optimal segmentation.

5. <u>CODE:</u>

```
image = imread("coins.png");
globalThresholdSegmentation(image);% figure(1)
noised_img = imnoise(image, "gaussian", 0, 0.01);
globalThresholdSegmentation(noised_img);% figure(2)
kernal = ones(5, 5) / 25;
filteredImage = imfilter(noised_img, kernal, "same");
globalThresholdSegmentation(filteredImage);% figure(3)
noised_img = imnoise(image, "gaussian", 0, 0.02);
globalThresholdSegmentation(noised_img);% figure(4)
filteredImage = imfilter(noised_img, kernal, "same");
globalThresholdSegmentation(filteredImage);% figure(5)
[rows, cols] = size(image);
ramp = repmat(linspace(0, 1, cols), rows, 1);
output_image = uint8(double(image) .* ramp);
globalThresholdSegmentation(output_image);% figure(6)

function globalThresholdSegmentation(image)
  image = double(image);
  threshold = mean(image(:));
  difference = 1;
  it = 0 ;
  while difference > 0.5
```

```
        lowerGroup = image(image <= threshold);
        upperGroup = image(image > threshold);
        meanLower = mean(lowerGroup(:));
        meanUpper = mean(upperGroup(:));
        newThreshold = (meanLower + meanUpper) / 2;
        difference = abs(newThreshold - threshold);
        threshold = newThreshold;
        it = it +1 ;
    end
    fprintf("%d \n",it);
    binaryImageGlobal = image > threshold;
    pixelCounts = imhist(uint8(image)); % Histogram of the image
    totalPixels = sum(pixelCounts); % Total number of pixels
    cumulativeSum = cumsum(pixelCounts); % Cumulative sum
    cumulativeMean = cumsum((0:255)' .* pixelCounts); % Cumulative mean
    globalMean = cumulativeMean(end) / totalPixels; % Global mean
    betweenClassVariance = zeros(256, 1);
    for t = 1:256
        wB = cumulativeSum(t);
        wF = totalPixels - wB;
        if wB == 0 || wF == 0
            continue;
        end
        mB = cumulativeMean(t) / wB;
        mF = (cumulativeMean(end) - cumulativeMean(t)) / wF;
        betweenClassVariance(t) = wB * wF * (mB - mF)^2;
    end
    [~, optimalThreshold] = max(betweenClassVariance);
    optimalThreshold = optimalThreshold - 1; % Otsu's optimal threshold
    binaryImageOtsu = image > optimalThreshold;
    figure;
    subplot(1, 3, 1);
    imshow(uint8(image));
    title('Original Grayscale Image');
    subplot(1, 3, 2);
    imshow(binaryImageGlobal);
    title(sprintf('Global’s Threshold: %0.2f, it: %d', threshold, it));
    subplot(1, 3, 3);
    imshow(binaryImageOtsu);
    title(['Otsu’s Threshold: ', num2str(optimalThreshold), ')']);
end
```