C:\opencv\build\x64\vc16\lib
opencv_world4110d.lib
C:\opencv\build\include
D:/FresherXavisTech/Image/

https://commons.wikimedia.org/w/index.php?curid=67144384
https://en.wikipedia.org/wiki/Otsu%27s_method

Slide 5:

Below is an example of applying thresholding.

We have the original image here. When we apply thresholding, the result consists of black regions and white regions. However, it's important to note that the quality of the output image depends entirely on the threshold value **T**. If we choose a good value for **T**, the resulting image will have high quality. On the other hand, When the chosen threshold is too low, some information may be lost and conversely when the threshold is too high, the result appears here.

The best value of **T** for thresholding is typically located between the two peaks of the histogram, as this is the optimal point to separate foreground and background pixels. At this midpoint, the confusion between the two classes is minimized, allowing for the most accurate object-background separation.

Unfortunately, the histogram of an image does not always have clearly defined peaks. Its shape can be affected by several factors. especially **noise** and **lighting conditions**.

Slide 6:

Let’s examine how noise and lighting affect thresholding performance through an example. The first column shows a Noiseless image. Below it is the histogram of that image. As we can see, the histogram has only two line, so thresholding is not really necessary in this case.

The second column shows the image corrupted by Gaussian noise with a standard deviation of 10 intensity levels. From the corresponding histogram, we can still easily identify a valley between the two peaks, which can serve as a suitable threshold value.

However, when the standard deviation increases to 50, the histogram no longer exhibits two clear peaks. This result makes it much more difficult to determine a reasonable threshold value.

Slide 7:

These are the thresholding results of the two images above. We can see that the quality of the result below is very good, but the one above is not.

Every black pixel in the white region and every white pixel in the black region is a thresholding error, so the segmentation in the less successful case was highly inaccurate.

Noise affects thresholding performance, so before applying thresholding, it is advisable to use noise reduction techniques such as mean filtering, Gaussian filtering, and so on.

Slide 8:

The deep valley between peaks was corrupted to the point where separation of the modes without additional processing is no longer possible.

Slide 16:

This image shows the results of the Iterative Selection Thresholding method, which produced many unwanted white regions in areas that are not part of the main objects. In contrast, the result from Otsu's Thresholding method is significantly cleaner; the circles are detected more clearly and sharply, with very little white noise in the background areas.

Otsu's Thresholding method (right image) performs markedly better than the Iterative Selection Thresholding method (left image) for thresholding this image. The result from Otsu's method contains less noise, the objects are more distinctly separated, and the overall image appears cleaner.

Slide 17:

The **Triangle Thresholding** method works by analyzing the histogram of a grayscale image. First, it identifies the peak of the histogram—the value with the highest frequency. Then, it locates one end of the histogram (either the leftmost or rightmost non-zero value). A straight line is drawn from the peak to this endpoint. For each point between the peak and the end, the perpendicular distance from the histogram value to the line is calculated. The threshold is chosen as the point with the maximum distance. This creates a triangle shape in the histogram, and the optimal threshold lies at the point farthest from the triangle's base.