**CV Practical No.: 4**

**Aim: Converting rgb image to binary image using thresholding method**

**New Concept:**

**i. cv2.IMREAD\_GRAYSCALE:** This is a flag used when reading an image with OpenCV (cv2). It specifies that the image should be loaded in grayscale (single-channel) mode, rather than the default color mode (BGR). When cv2.IMREAD\_GRAYSCALE is passed as an argument to the cv2.imread() function, the image is converted into a 2D array, where each pixel represents the intensity of light (from black to white), rather than having three channels (BGR).

**ii. cv2.threshold:** This function is used for image binarization in OpenCV. It converts an image into a binary image based on a specified threshold value. The cv2.threshold function applies a threshold to an image and converts pixel values to either a high value (maxval) or a low value (usually 0), depending on whether they exceed a given threshold value (thresh). This is useful for segmenting objects from the background.

**iii. cv2.THRESH\_BINARY:** This is a thresholding type used with cv2.threshold function. It applies a binary threshold to the image, where pixel values greater than the threshold are set to the maximum value, and those less than the threshold are set to 0. When using cv2.THRESH\_BINARY, the pixel intensity is either set to the maximum value (for example, 255) if it exceeds the threshold or to 0 if it doesn't. This creates a stark black-and-white image, useful for binary image analysis.

**iv. image.shape:** The shape attribute of an image represents the dimensions of the image in the form of a tuple (height, width, channels). The image.shape provides the dimensions of the image:

* height is the number of rows (the number of pixels along the vertical axis),
* width is the number of columns (the number of pixels along the horizontal axis),
* channels refers to the number of color channels (3 for RGB/BGR, 1 for grayscale).

**v. threshold\_value:** A value used in thresholding to define the cut-off point. Pixel values above the threshold are assigned one value (e.g., 255), and pixel values below the threshold are assigned another value (e.g., 0). The threshold\_value is the numerical value that is used to distinguish between foreground and background pixels during thresholding. For example, in binary thresholding, any pixel with a value greater than threshold\_value will be converted to 255 (white), and those less than the threshold will be converted to 0 (black).

**Theory:**

**i. RGB to Binary:**

* **RGB to Binary** is the process of converting an image from the RGB (Red, Green, Blue) color space to a binary (black and white) format. This conversion is crucial for many computer vision tasks, such as object detection, segmentation, and feature extraction.
* **RGB Color Model**: In the RGB model, images are represented by three color channels: Red, Green, and Blue. Each pixel has a combination of these three channels, and each channel can have intensity values ranging from 0 to 255. This allows for millions of different colors to be represented.
* **Binary Image**: A binary image is an image that consists of only two colors (usually black and white), represented by pixel values of 0 (black) and 255 (white). In a binary image, pixel values are either on (white) or off (black), with no intermediate shades.
* **Applications:**
* **Object detection**: Binarization is useful for detecting objects against a contrasting background.
* **Image segmentation**: Dividing an image into regions (foreground and background) based on pixel intensities.
* **Edge detection**: Helps to highlight features by emphasizing high-intensity regions.

**ii. Threshold method:**

* The Threshold Method is a fundamental image segmentation technique used to convert an image into a binary form by applying a specific threshold value. The method classifies pixel values based on their intensity (or grayscale value) into two distinct categories: foreground (object) and background.
* **Steps Involved:**
* **Gray Level Intensity**: An image typically has pixel values that represent different shades of gray (for grayscale images) or color intensities (for RGB images). In the thresholding process, each pixel intensity is compared to a predefined threshold value.
* **Thresholding**: The core idea of the threshold method is:
* If the pixel value is greater than or equal to the threshold value, it is assigned the maximum value (often 255 for white in binary images).
* If the pixel value is less than the threshold value, it is assigned the minimum value (usually 0 for black in binary images).
* This effectively segments the image into two regions:
* Foreground (objects of interest, often white),
* Background (everything else, often black).
* The threshold value can be manually selected or determined using algorithms (e.g., Otsu's method) that compute the optimal threshold based on the image's histogram.
* **Applications:**
* **Image Binarization**: Essential for creating binary images that are easier to analyze and manipulate.
* **Object Detection and Recognition**: By segmenting the foreground from the background, thresholding helps identify objects within an image.
* **Edge Detection**: A form of thresholding is often applied after edge detection algorithms to enhance or simplify the results.

**Program:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load the grayscale image

image = cv2.imread('cartoon.jpeg', cv2.IMREAD\_GRAYSCALE)

total=np.sum(image)

w,h=image.shape

avg=int(total/(w\*h))

# Apply thresholding

threshold\_value = avg # You can adjust this value

\_, thresholded\_image = cv2.threshold(image, threshold\_value, 255, cv2.THRESH\_BINARY)

# Display the thresholded image

#cv2.imshow('Thresholded Image', thresholded\_image)

#Image1

plt.figure(figsize = (10,5))

plt.subplot(1,2,2)

#plt.imshow(image)

plt.imshow(image, cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image2

plt.subplot(1,2,1)

#plt.imshow(thresholded\_image)

plt.imshow(thresholded\_image, cmap = 'gray')

plt.title('Thresholded Image')

plt.axis('off')

plt.show()

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output:**

