DIP Assignment-1 Report

1. Histogram Computation

The implemented function `histogram` takes the path (images/coins.png) of a grayscale image as input and computes its histogram. Converting an image to grayscale simplifies histogram analysis by reducing data complexity and focusing on intensity distribution. This approach is also computationally efficient as Grayscale images consume less memory and are faster to process because they contain only one channel of data rather than three (R, G, B). The conversion using a weighted-sum approach using `convert('L')` method in the PIL library is as follows:

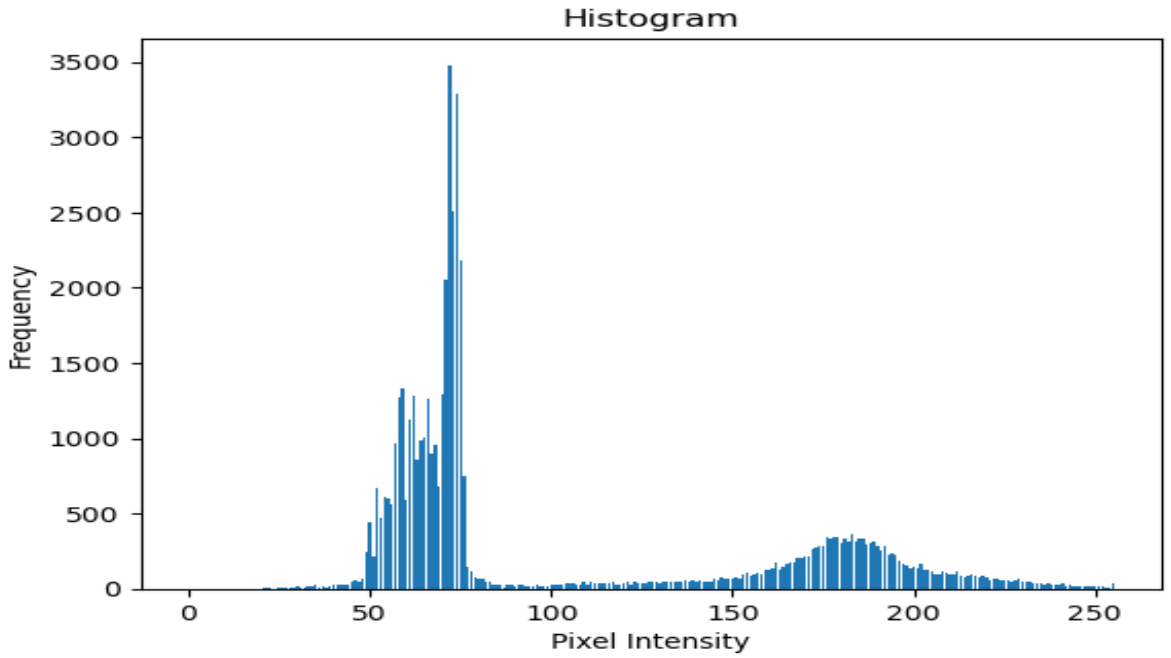
$$L = 0.2989*R + 0.5870*G + 0.1140*B$$

Where, L is the luminance (grayscale intensity), and R, G, and B are the red, green, and blue color components, respectively.

The histogram shows the frequency of pixels at each intensity level ranging from 0 to 255. The function returns this histogram as a NumPy array of size 256.

Inference from Observations:

- The plotted histogram successfully visualizes the distribution of pixel intensities across the image.
- The histogram demonstrates a bimodal distribution, indicating the presence of two predominant pixel intensity groups in the image. This bimodal nature, especially with well-separated clusters, suggests that the image has two primary regions of distinct intensities, which could correspond to foreground and background or two contrasting regions.
- The calculated average intensity based on the histogram matches exactly with the actual average intensity of the image, validating the correctness of the histogram computation.
- Average intensity from histogram: 103.30500158906722
  Actual average intensity: 103.30500158906722
- The match between the calculated and actual average intensities confirms that the histogram is a reliable tool for statistical analysis of an image's grayscale levels.

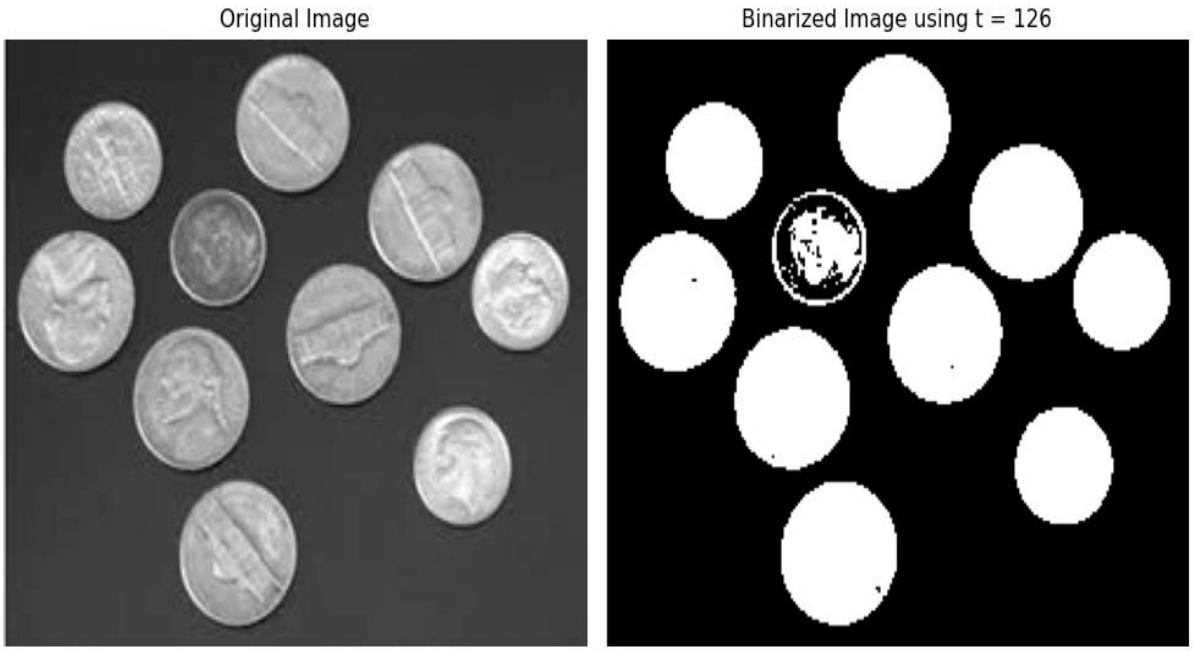


2. Otsu’s Binarization

Otsu's Binarization is implemented by finding the optimal threshold in two ways: minimizing within-class variance and maximizing between-class variance. It then verifies that the optimal thresholds from both methods are equivalent and also measures the time taken by each method.

Inference from Observations:

- Both methods for determining the optimal threshold yielded the same value of 126, confirming their equivalence.
- The time taken to minimize within-class variance (0.0750 seconds) is slightly less than the time taken to maximize between-class variance (0.1140 seconds). This is because Minimizing within-class variance is slightly faster as it directly computes the variance for the two classes (below and above the threshold), whereas maximizing between-class variance requires an additional computation to find the total variance of the image and then subtract the within-class variance.



3. Depth based Extraction

The Implementation consists of two main functions: `binarize` and `imageSuperImpose`.

The `binarize` function takes in the depth image and a threshold value. It iterates over each pixel, converting it to binary based on the threshold, thereby producing a binary depth map.

The `imageSuperImpose` function uses this binary depth map to decide where to overlay text from a given text image onto a background image. If a pixel in the binary depth map is 1, it overlays the corresponding pixel from the text image; otherwise, it keeps the original background image's pixel.

For determining the optimal threshold, Otsu's method is employed, which effectively minimizes within-class variance or maximizes between-class variance, thus aiding in more accurate text extraction and overlay.

Inference from Observations:

- The optimal threshold for binarization using Otsu's method was numerically determined to be 144.0. This value effectively segmented the text from the background in the depth image.
- The superimposed image generated using this optimal threshold displayed precise text extraction and overlay. The numerical value of the threshold (144.0) can be considered highly effective as it produced the desired output.
- Note that for some threshold values, the superimposition may not work as expected. A non-optimal threshold might lead to incomplete or erroneous extraction of text from the depth map and consequently, an imperfectly segmented text is superimposed on the background.
  For example, if the threshold is too low, it might include parts of the background in the depth map, leading to a 'noisy' overlay. If the threshold is too high, it might miss out some parts of the text, causing incomplete text overlay on the background.

4. Connected Components

- The implemented code uses a two-pass connected component algorithm to label and then filter out the regions based on a minimum pixel count to count the total characters in the image.
- **Algorithm**:
    - The *connected component algorithm* processes a binary image to identify, and label connected regions where adjacent pixels share the same value (usually '1' for foreground).
    - In the *first pass*, each pixel is visited in a row-by-row order. For each foreground pixel, a label is assigned based on its 8 neighboring pixels' labels.
    - *Equivalence relationships* are established between different labels that belong to the same connected component. These relationships are stored in an equivalence dictionary.
    - In the *second pass*, the algorithm revisits each pixel to assign it the smallest label in its equivalence set, effectively uniting various parts of the same connected region under a single label.
    - Finally, a *filtering step* removes components with a pixel count below certain thresholds to remove punctuations.

Inference from Observations:

- The optimal threshold value for binarization was found to be 143.0 using Otsu's method, which is crucial for the accuracy of the connected components.
- A pixel count threshold of 60 was used to filter out smaller components, presumed to be punctuation. When using a different threshold, the count of characters deviated from the correct number of 64, highlighting the importance of this parameter in ensuring accurate character count.

5. MSER (optional)

Why might Otsu's method not work well here:

Otsu's method determines a global threshold for the entire image. This makes it less effective for images with varying lighting conditions or where different characters have significantly different grayscale values. MSER, on the other hand, operates in the local regions, making it more adaptive to such variations. Otsu's may result in loss of some characters or inclusion of noise, whereas MSER's adaptability can better distinguish between actual characters and noise or varying light conditions.