Digital Image Processing
Assignment 1

August 30, 2025

Pritam Kumar
24021
MTech AI

---

1. **Histogram Computation:**

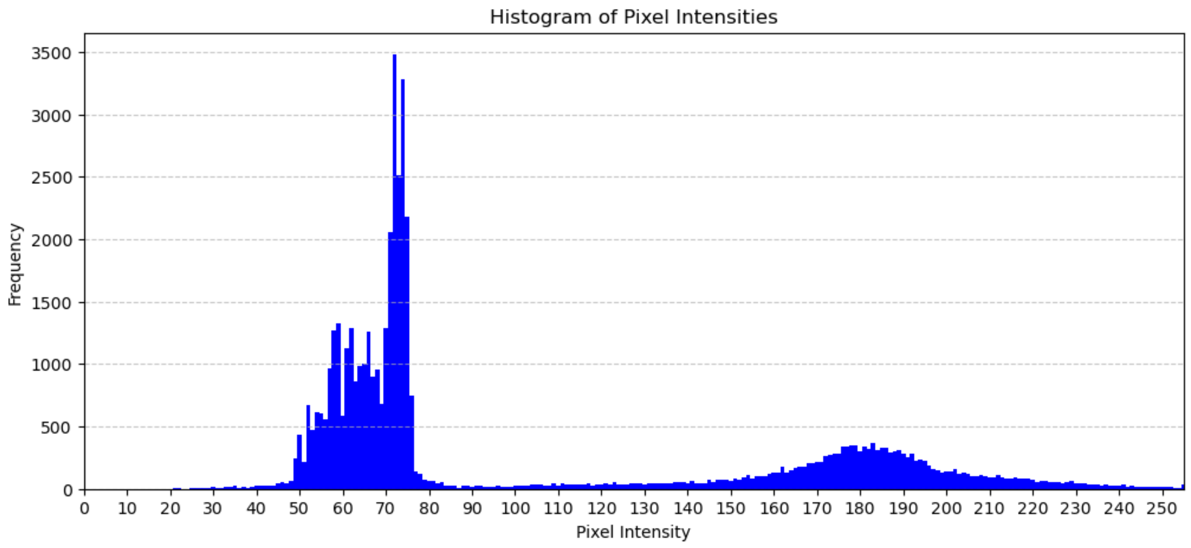


Figure 1: Histogram of a binary image coin

Observations

- A large cluster between ∼50-80 (dark grayish intensities). The highest spike is near intensity 70.
- Another smaller cluster between ∼170–200 (lighter gray/near white region).
- Very few pixels at the extremes (near 0 or 255). Therefore, the image doesn't have large areas of pure black or pure white.
- The image is grayscale with two dominant intensity groups.
  1. A darker background or foreground region (60–80).
  2. A lighter foreground or background region (170–200).

Average Intensity: Using the histogram, we can calculate the average intensity by summing the products of frequency and intensity.

$$\text{Avg Intensity} = \frac{1}{MN}\sum_{i=1}^{K} x_i f_i$$

where x_i is the intensity value and f_i is the frequency of the pixel.

- Average pixel intensity using histogram: 103.31
- Average pixel intensity using NumPy: 103.31

We can verify that the average intensity from both methods is the same.

2. **Otsu's Binarization**

   (a) Minimize the within-class variance $\sigma_w^2(t)$.

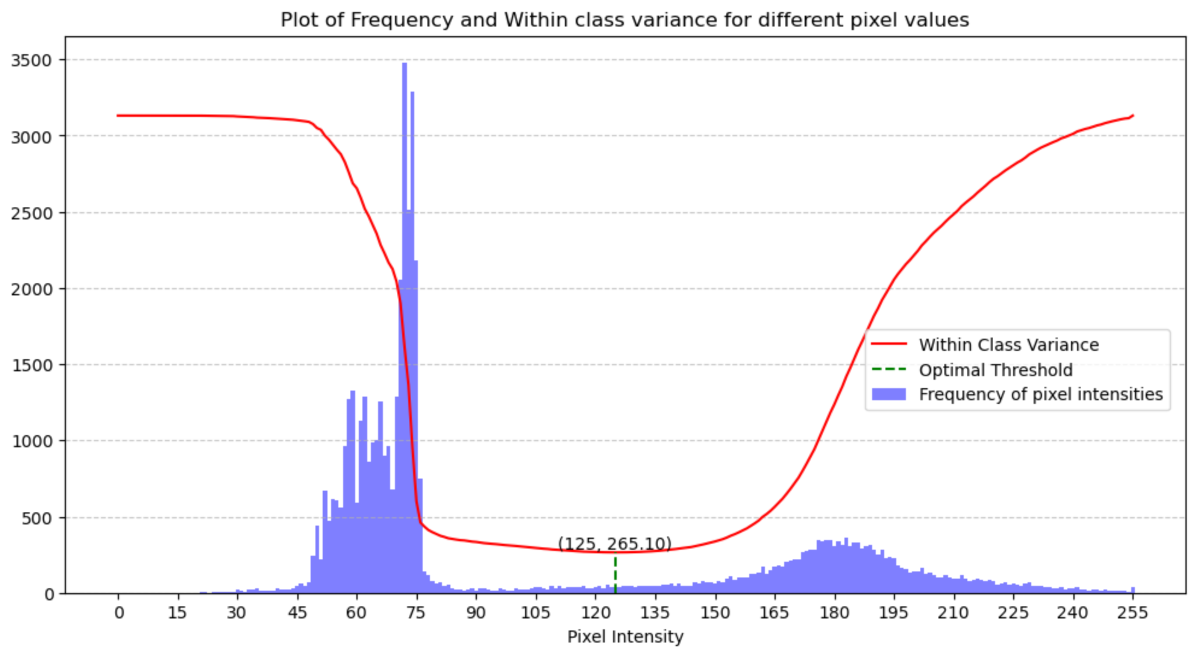
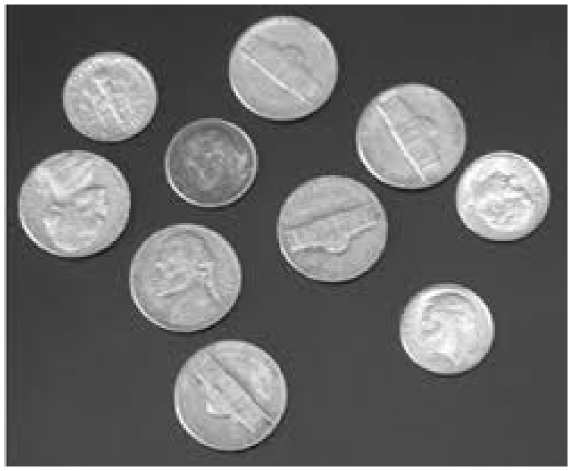
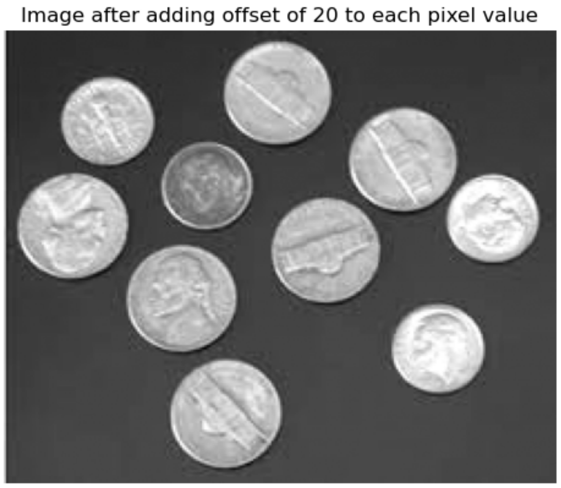


Figure 2: Frequency distribution (vertical blue lines). The solid red curve shows the within-class variance, and the dashed vertical line indicates the threshold intensity corresponding to the minimum variance.

*The optimal threshold intensity is **125**, with a corresponding minimum within-class variance of **265.10**.*

(b) **Maximize the between-class variance on an offset of 20 on the coin image**


(a) Original coin image



(b) Coin image with intensity offset of 20

Figure 3: Comparison of two coin images: (a) original image and (b) image with intensity offset.

The optimal threshold is ***145****, and the maximum between-class variance is* ***2852.96***

1. Using within-class variance minimization on the original coin image, the optimal threshold was 125, while applying between-class variance maximization on the offset version of the same image gave a threshold of 145. Please refer to Figure 4.
2. The difference arises because applying an intensity offset shifts the histogram to the right, moving the separation point between the background and foreground peaks to a higher gray level.
3. After adding a +20 intensity offset, the Otsu threshold shifts from 125 $\rightarrow$ 145. This is just the addition of 20 to the original image's optimal threshold.

3. **Adaptive Binarization:**
Otsu's thresholding method can automatically determine the optimal value of T, assuming a bimodal distribution of pixel intensities in our input image.
Issue with one global threshold: In this approach, only one value of T will be used to test all the pixels to segment the image into background and foreground. But, lighting conditions, shadows, etc, vary in images, and it may not be appropriate to use a single global threshold. The T may be effective in certain regions but may fail in others. That's why we use *Adaptive Binarization.*

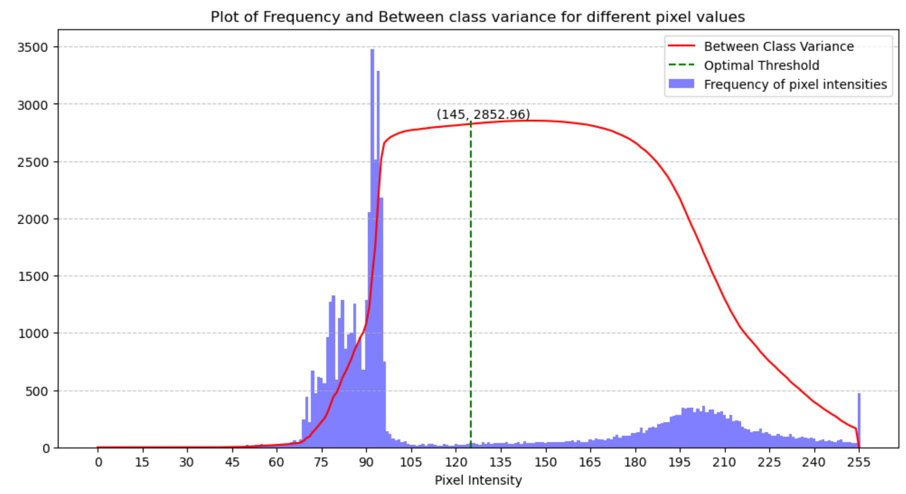


Figure 4: Between-class variance on the coin image with intensity offset of 20

We use Adaptive Binarization considering different block sizes. Block size impacts the performance of the binarization. We shall see the performances of each block.

The total number of blocks checked to find the local threshold is given by

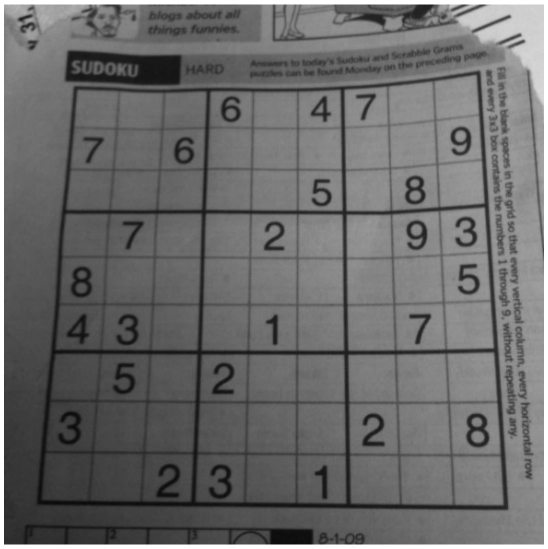
$$\text{Total Blocks } = \left\lceil \frac{m}{N\,(1-\text{overlap})} \right\rceil \cdot \left\lceil \frac{n}{N\,(1-\text{overlap})} \right\rceil \tag{1}$$

where m and n are the number of rows and columns in the image, N is the square block size, and the overlap is 0.2.

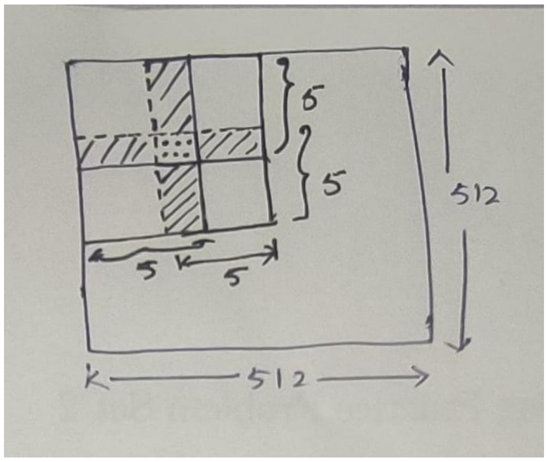
For the given image `sudoku.png`, $m = n = 512$.

Substituting $N = 5$,

$$\text{Total Blocks} = \left\lceil \tfrac{512}{5\times(1-0.2)} \right\rceil^2 = 16384$$



(a) Original Sudoku image



(b) Block movement visualization. The dotted shaded area represents pixels covered in four blocks, while the solid-line regions are covered in two blocks.

Figure 5: Original Sudoku image and corresponding block movement visualization.

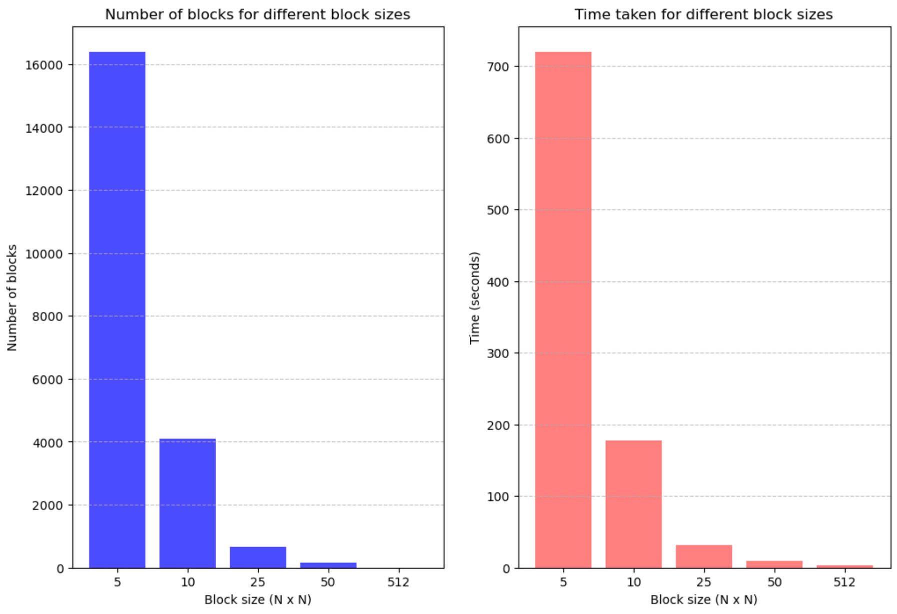
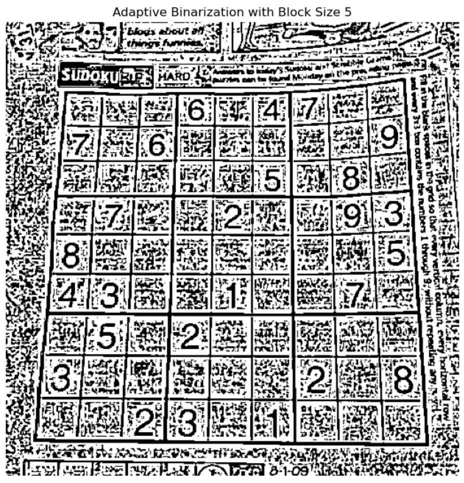
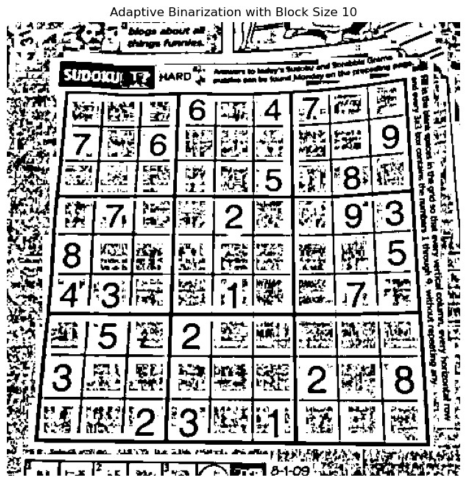


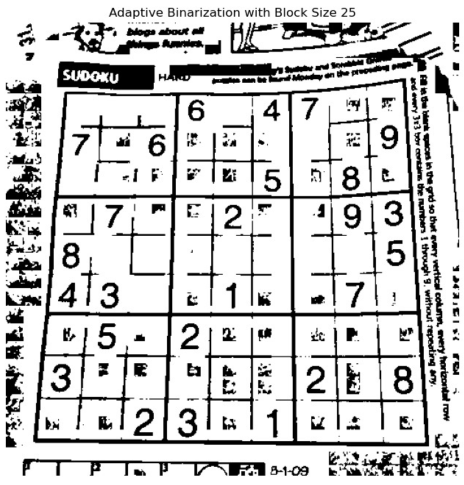
Figure 6: Left bar plot shows the number of blocks operated for each block size, and right plot shows the total time taken to complete the binarization



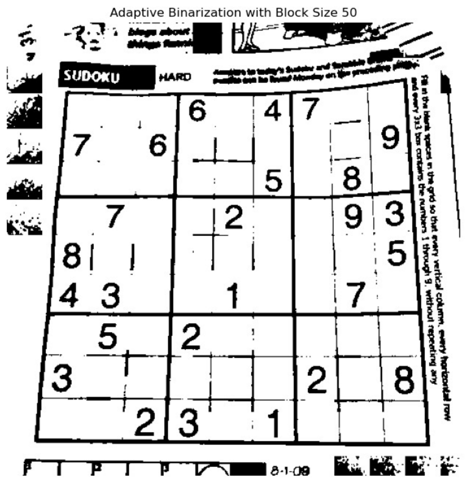
(a) 5 × 5 Sudoku image



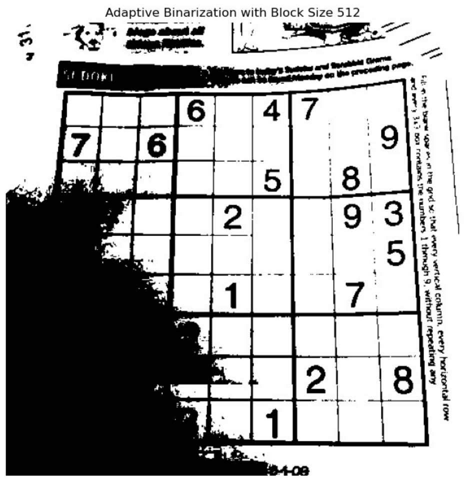
(b) 10 × 10 Sudoku image



(c) 25 × 25 Sudoku image



(d) 50 × 50 Sudoku image



(e) Full Sudoku image binarization

Figure 7: Comparison of binarized Sudoku images at different grid sizes.

Observations on Block Sizes

a) 5×5 **blocks:** Very fine detail preserved; digits and grid lines are clear even under uneven lighting, but background noise is also emphasized.

b) 10×10 **blocks:** Still captures detail well, with slightly less noise than 5×5; a good balance between readability and robustness.

c) 25×25 **blocks:** Digits remain visible, but some thin strokes start to fade in darker regions; background looks cleaner.

d) 50×50 **blocks:** Smoother background, but many small digit details are lost; works only if illumination is fairly uniform.

e) 512×512 **(full image binarization):** Ignores local variations; large dark patches obscure digits, and uneven lighting causes significant loss of information.

4. **Connected Components:**
The 8-neighbours of a pixel (x, y) are defined as:

$$N_8(x,y) = \{(x-1,y),(x-1,y-1),(x,y-1),(x+1,y-1),(x+1,y),(x+1,y+1),(x,y+1),(x-1,y+1)\}.$$

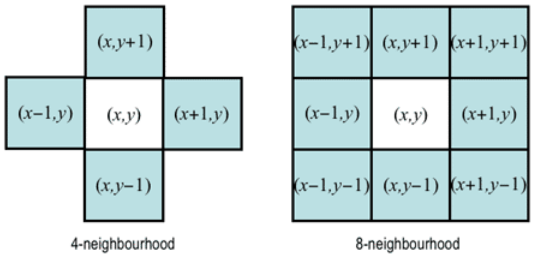


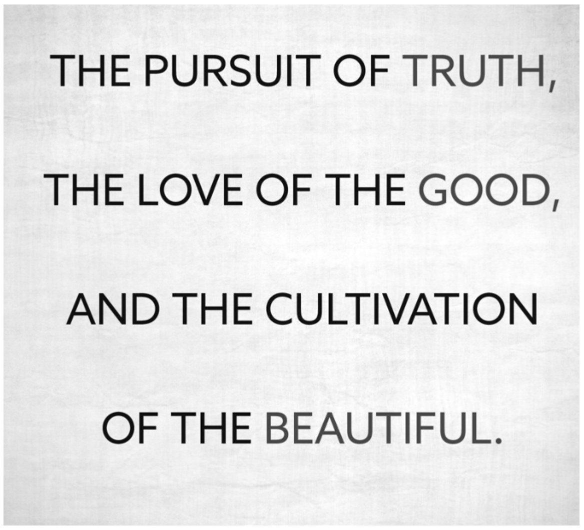
Figure 8: Illustration of 8-connected neighbours in an image grid. The left one is for 4-adjacency, and the right one is for 8-adjacency.

Algorithm 1 Find

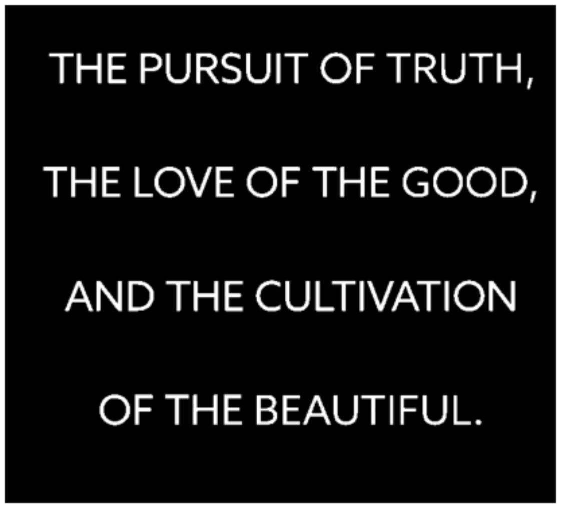
```
function FIND($x$)
    while $parent[x] \neq x$ do
        $parent[x] \leftarrow parent[parent[x]]$        ▷ path compression
        $x \leftarrow parent[x]$
    end while
    return $x$
end function
```

Algorithm 2 Union

```
1: function UNION($x, y$)
2:     $r_x \leftarrow$ FIND($x$),   $r_y \leftarrow$ FIND($y$)
3:     if $r_x \neq r_y$ then
4:         $parent[r_y] \leftarrow r_x$
5:     end if
6: end function
```



(a) Original image of quote.png



(b) Quote image after Otsu's binarization. The white background was made black and black into white.

Figure 9: Comparison of original and binarized quote images.

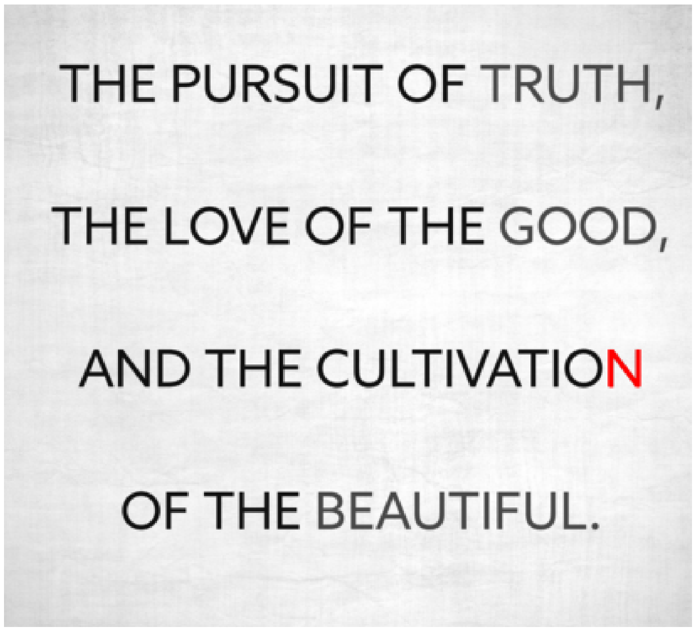


Figure 10: The modified image where the largest component is highlighted in red over the original image.

- The largest character was identified as **N**, with a total of **145 connected components**.