Digital Image Processing

Atharva Bhawalkar

The LNM Institute of Information Technology, Jaipur

30 April 2024

1 Preliminary Steps

1.1 Reading Source Image

imread is a MATLAB function used for reading images from various file formats. For color images, it returns a 3D array, where each dimension represents a color channel. *RGBImage* is the variable name used to store the RGB image data read from the file.

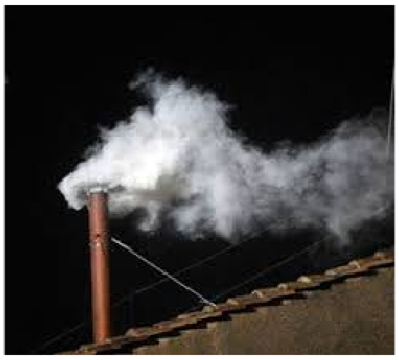

Figure 1: Source Image

1.2 Determining Dimensions

size is a MATLAB function which returns the dimensions of the input array. When applied to an image array *RGBImage*, it returns a vector containing the size of each dimension.

2 Colored to Grayscale

2.1 What is Grayscale

A grayscale image is an image where each pixel is represented by a single intensity value, typically ranging from 0 (black) to 255 (white) in an 8-bit image.

2.2 Why Grayscale

Grayscale images are favored in image processing for their simplicity, requiring less memory compared to color images. They offer computational efficiency as algorithms operate on a single intensity channel.

2.3 Lightness Method

To convert a RGB image into a Grayscale image, I chose to implement the lightness method. The lightness method, also known as luminance-preserving grayscale conversion, is a technique using which the grayscale intensity of each pixel is computed as the average of the maximum and minimum RGB values of the corresponding pixel in a color image.

$$\textit{grayValue} = \frac{\max(R, G, B) + \min(R, G, B)}{2} \tag{1}$$

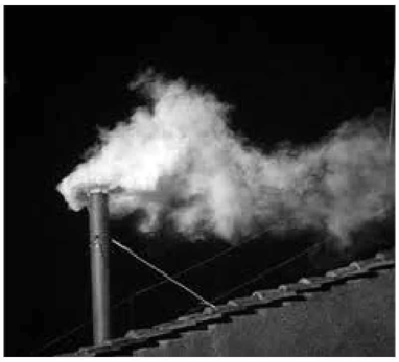

Figure 2: Gray-scale Image

3 Adding Noise

Generally, noise is added to the degraded image to simulate the imperfections or disturbances that occur during the image acquisition or transmission process.

3.1 Salt & Pepper Noise

The salt & pepper noise model describes the phenomenon where random bright and dark pixels occur sporadically in digital images. *Salt* refers to randomly

occurring bright pixels with high intensity values. *Pepper* refers to randomly occurring dark pixels with low intensity values.
In mathematical terms, the salt and pepper noise model can be represented as follows:

$$noisyImage = \begin{cases} 0, & noise < P_a \\ 255, & noise > (1 - P_b) \end{cases} \quad (2)$$

where,
P_a is the probability of occurrence for the salt noise
P_b is the probability of occurrence for the pepper noise.

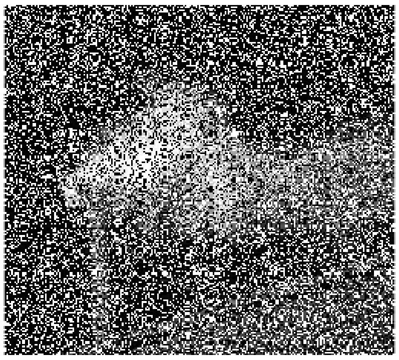
Figure 3: Noisy Image

4 Adaptive Median Filtering

The median filter is particularly effective at removing this type of noise because it replaces each pixel's value with the median value of its local neighborhood. Since the median is less affected by extreme values than the mean, it can effectively eliminate the influence of the noisy pixels.

4.1 Why Adaptive

The adaptive median filter adjusts the size of the neighborhood window based on the characteristics of the noise. This allows it to preserve image details while removing salt-and-pepper noise.

4.2 Levels of Operation

Level A
$A_1 = Z_{med} - Z_{min}$
$A_2 = Z_{med} - Z_{max}$
If $A_1 > 0$ and $A_2 < 0$, proceed to level A
otherwise, increase the *windowSize* by 2
If the *windowSize* is $\leq$ to S_{max}, repeat Level A
otherwise, output Z_{xy}
Level B
$B_1 = Z_{xy} - Z_{min}$
$B_2 = Z_{xy} - Z_{max}$
If $B_1 > 0$ and $B_2 < 0$, output Z_{xy}
otherwise, output Z_{med}
where,
Z_{min} : minimum gray value in S_{xy}
Z_{max} : maximum gray value in S_{xy}
Z_{med} : median of gray levels in S_{xy}
Z_{xy} : gray value of the image at (x,y)
S_{max} : maximum allowed size of S_{xy}

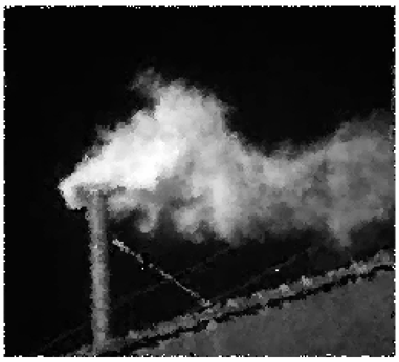
Figure 4: Filtered Image

5 Otsu's Thresholding

Otsu's thresholding, also known as Otsu's method or Otsu's algorithm, is a popular technique for automatically segmenting an image into two classes (e.g., foreground and background) based on the intensity values of its pixels.

5.1 Method

We begin by computing the histogram of the input image to analyze the distribution of pixel intensities, followed by the determination of the weighted sum of pixel values.
Through an iterative process, the algorithm explores various threshold values, calculating the between-class variance for each. This involves computing weights for background and foreground pixels, as well as their respective means.
The optimal threshold is updated whenever a higher between-class variance is encountered.

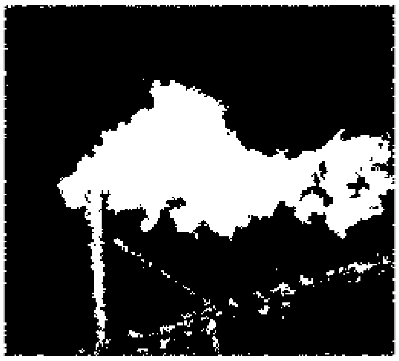
Figure 5: Binary Image

6 Morphological Analysis: Opening

Morphological Opening is used to remove small objects or noise from a binary image.
The opening operation (*erosion* followed by *dilation*) effectively removes small objects or noise in the image while preserving the larger structures.

6.1 Method

Erosion: This step involves shrinking or eroding the boundaries of foreground (white) regions in the binary image. It is done by moving a structuring element (often a small kernel or window) over the image and replacing each pixel with the minimum pixel value within the neighborhood defined by the structuring element.
Dilation: Following erosion, dilation is applied. Dilation expands or fattens the boundaries of foreground regions. It involves moving the structuring element over the eroded image and replacing each pixel with the maximum pixel value within the neighborhood defined by the structuring element.

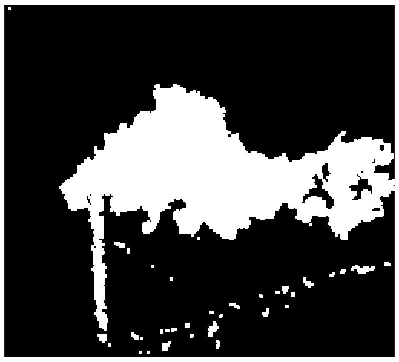
Figure 6: Opened Image

7 Morphological Analysis: Connected Components

Dilation, a fundamental morphological operation, expands or thickens the boundaries of objects in a binary image by replacing each pixel with the maximum pixel value within the neighborhood defined by a structuring element.

7.1 Method

We group together pixels in an image that belong to the same object, forming connected components.
connected components are identified through an iterative process based on dilation. The algorithm starts with the input binary image and repeatedly applies dilation until no further changes occur in the image. Each iteration of dilation merges adjacent foreground pixels that are connected, gradually growing the connected components.

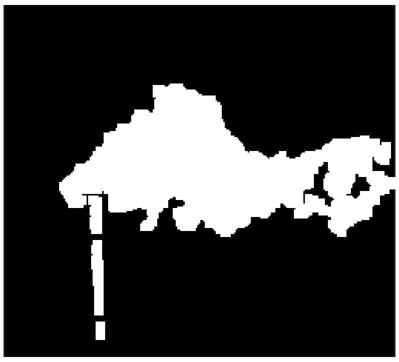
Figure 7: Connected Image

8 Results

Area of Source Image: 60088 pixels
Area of Largest Connected Component: $\approx$ 11000 pixels