Tree counting in Different Scenarios, using satellite remote sensing

(CE671 Term Paper Report)

Tenzing Pema Thungon (231030063)

Amisha Dhiman (231030006)

Civil Engineering, Geoinformatics, IIT Kanpur, 208016, UP, India

tenzing23@iitk.ac.in

amishad23@iitk.ac.in

Abstract

Accurate and efficient tree counting is essential for various ecological, environmental, and land management applications. Traditional methods of tree counting, such as manual surveys, can be labor-intensive and time-consuming. Remote sensing technologies, on the other hand, have emerged as significant instruments for optimizing the process of tree counting, with advantages such as rapid data collection, cost-effectiveness, and large-scale coverage. This report explores the application of satellite remote sensing, specifically leveraging Google Earth imagery, for the task of tree counting. With the increasing availability of high-resolution satellite imagery, this study demonstrates the feasibility and efficiency of using such resources to estimate tree populations. By utilizing image processing and computer vision techniques, the report outlines the methodology and algorithms employed in tree counting using remote sensing data, including image processing, and data fusion techniques, also emphasizing the advantages of remote sensing in providing a cost-effective and wide-scale solution for environmental monitoring. Image processing techniques were used in this paper for autonomous segmentation of the satellite image and extraction of trees from the segmented image, for both the assessment of tree density in forests and the monitoring of individual trees in urban environments. The results of this research provide valuable insights into the potential of satellite remote sensing as a tool for tree counting and its significance in various environmental and conservation applications.

Keywords- Image Processing, Remote Sensing, Google Earth.

1 Introduction

Forests and trees play a crucial role in maintaining the planet's ecological balance, contributing to carbon storage, the preservation of biodiversity, and the overall health of ecosystems. To effectively manage and conserve these vital resources, it is imperative to have accurate information about tree populations. Satellite remote sensing has swiftly emerged as a potent tool for tree counting over vast areas, revolutionizing our capacity to collect essential data for environmental monitoring and management. The use of this technology, combined with the availability of resources such as Google Earth photos, has widened the scope of our environmental study efforts. Google Images is a convenient and accessible source of satellite imagery. It provides access to a vast library of images from a variety of satellites, including Landsat, Sentinel-2, and PlanetScope. These images have a spatial resolution of up to 3 meters, which is sufficient for tree counting in most cases, makes GEE a valuable tool for tree counting using satellite remote sensing.

There are number of different approaches that can be used to count trees using satellite remote sensing. One common approach is to use image segmentation to identify and extract individual tree crowns. Once the tree crowns have been identified, they can be counted using a variety of methods, such as connected component labeling or edge detection methods.

Another approach to tree counting is to use machine learning techniques to classify pixels in the satellite image as trees or non-trees. This approach is particularly useful for counting trees in dense forests or urban areas.

We utilized Google imagery and MATLAB in this study to segment tree crowns in images from two different areas: one where trees are separated from one other and one where trees are very near to each other, making it difficult to distinguish tree borders. The purpose of this research is to compare the effectiveness of various tree counting methods in these two scenarios.

2 Methodology

In this study, we use two satellite images from Google Earth to evaluate our tree counting algorithm. The first image is from IITK campus near OAT and has a ground resolution of 380 meters. The second image is from the Lindale, United States, and has a ground resolution of 80 meters. The two images were selected to represent a variety of tree densities and landscapes. The IITK image shows a dense forest, while the Lindale image shows a more open landscape with scattered trees.

In this study, MATLAB served as the pivotal tool for conducting the image processing tasks. MATLAB's extensive suite of image processing functions and capabilities played a fundamental role in enhancing the quality and utility of the remote sensing data. The significance of MATLAB in remote sensing lies in its versatility, efficiency, and robustness. It empowers researchers to manipulate, analyze, and visualize remote sensing imagery, facilitating the extraction of valuable insights and information from satellite and aerial data. Its ability to handle large datasets, apply advanced image processing algorithms, and streamline workflows makes it an indispensable resource for remote sensing studies, ensuring the accurate and reliable analysis of Earth's ever-evolving landscapes.

2.1 Study area 1

The image utilized consist of three bands i.e., RGB a.k.a TrueColor. The resolution of the image is 1485*829 pixels, taken on 11th May 2022. The image is not very good and consist of fewer details. For further processing various image enhancement techniques are used.

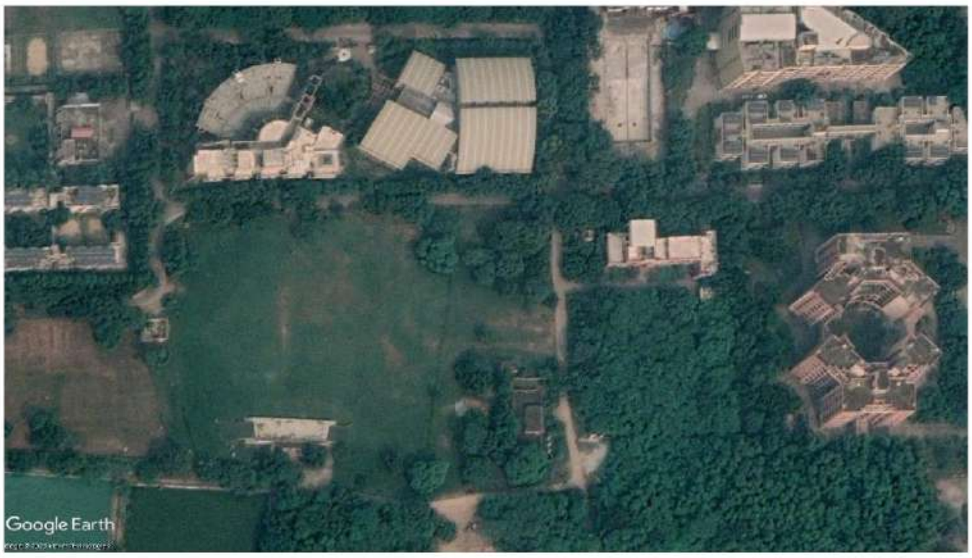


Figure 1: Google Earth view of IITK, area near OAT

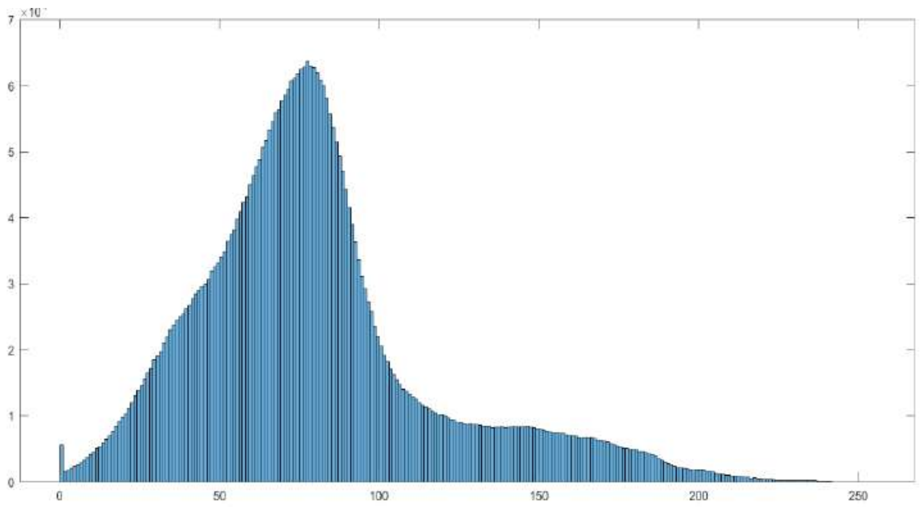
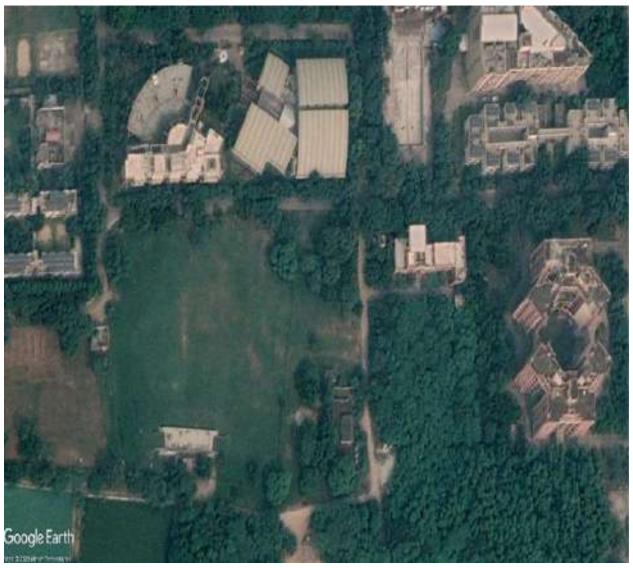


Figure 2: Histogram of the original image

2.1.1 Pre-Processing: image enhancement techniques

A. Local-brighten

The imlocalbrighten() function in MATLAB performs a local brightness adjustment on an image. First it calculates the average brightness of the surrounding pixels for each pixel in the image. The brightness of each pixel is then adjusted by a factor that is proportional to the difference between the pixel's brightness and the average brightness of the surrounding pixels. This means that the brightness of each pixel in the image is adjusted independently, based on the brightness of the surrounding pixels.



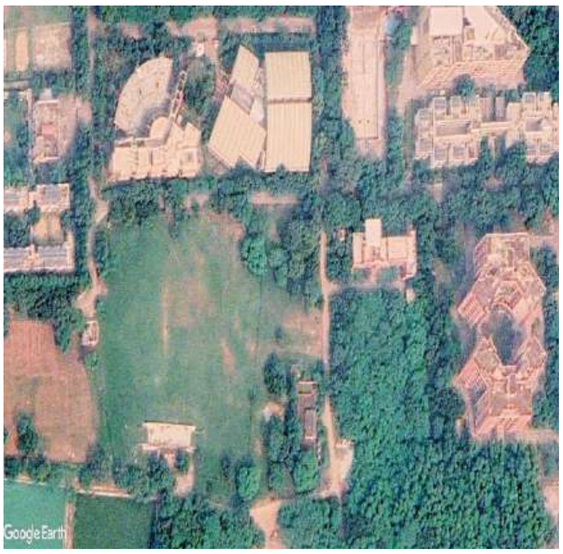
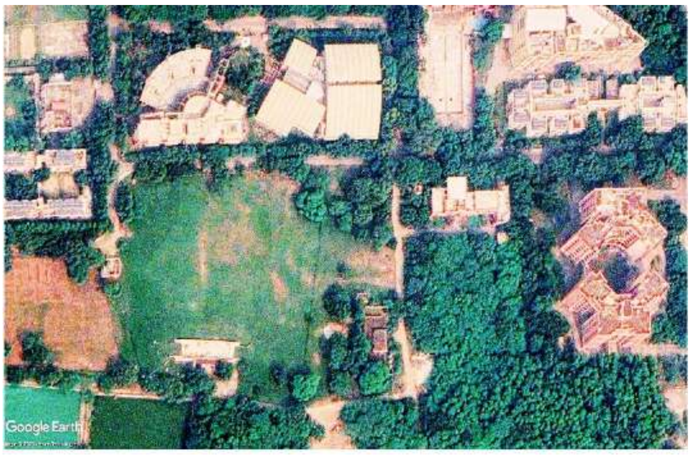


Figure 3: Original image on the (left) and enhanced image on the (right).

B. Histogram equalization

The histeq() function in MATLAB performs histogram equalization on an image. This means that the brightness of the image is adjusted so that the pixels are distributed evenly across the brightness range. This can also be useful for improving the contrast of an image, especially in images that have a wide range of brightness values.
The histeq() function first calculates the histogram of the image. The histogram is then transformed so that the pixels are distributed evenly across the brightness range. The transformed histogram is then used to map the brightness values of the pixels in the original image to new brightness values.

The combined use of "localbrighten" and "histeq" can make an image appear brighter and have better contrast, which often leads to an improved visual appearance.



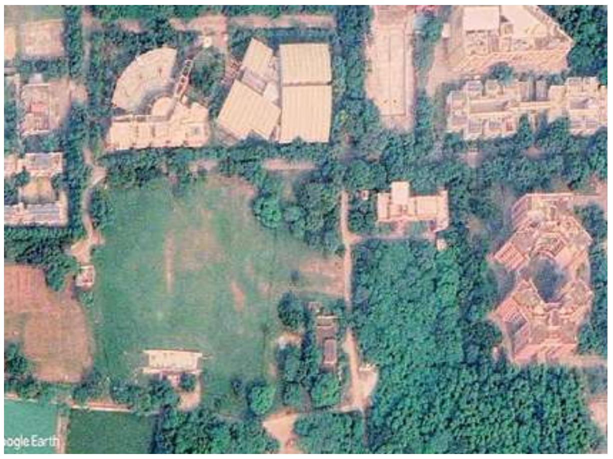


Figure 4: image after histeq (left) and image after localbrighten (right).

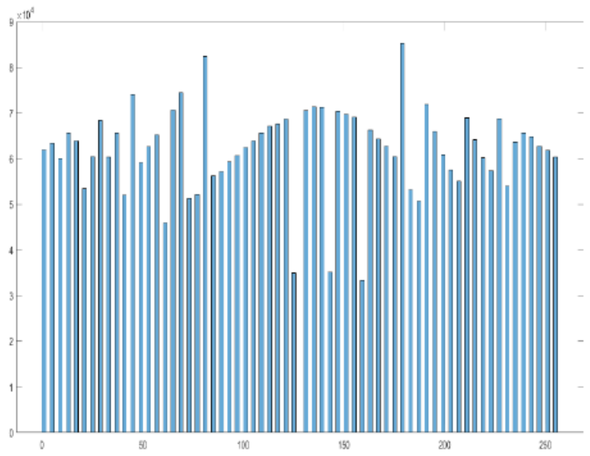
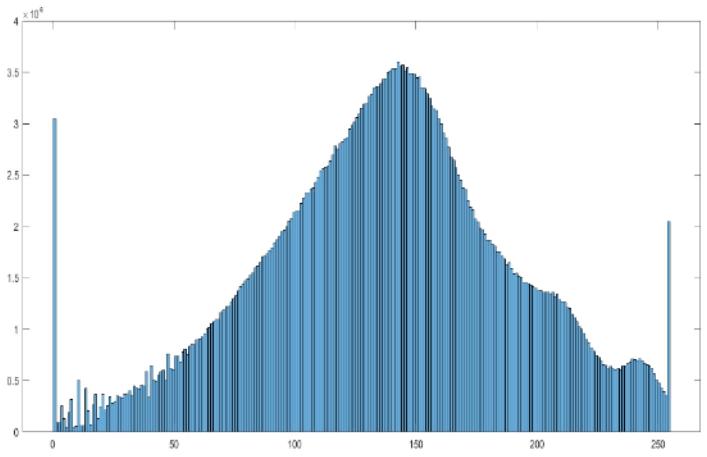



Figure 5: histogram of histeq image (left) and histogram of localbrighten image (right).

2.1.2 RGB to Gray conversion

Converting an image from RGB to grayscale (also known as "grayscale conversion") is a common step in image processing and analysis. Histogram equalization is applied individually to each color channel in an RGB image. Converting to grayscale simplifies this process, as there is only one channel to equalize. The gray image helps to normalize the brightness of the image, making it easier to identify the trees.

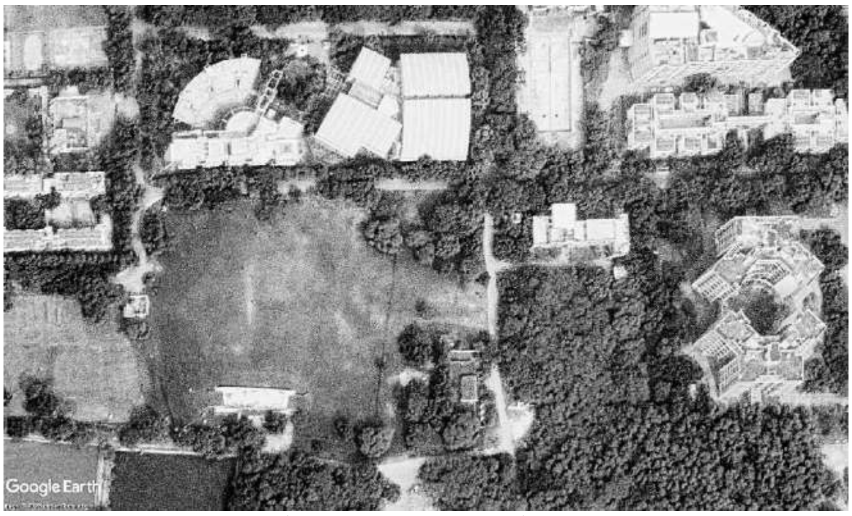


Figure 6: RGB to Gray image

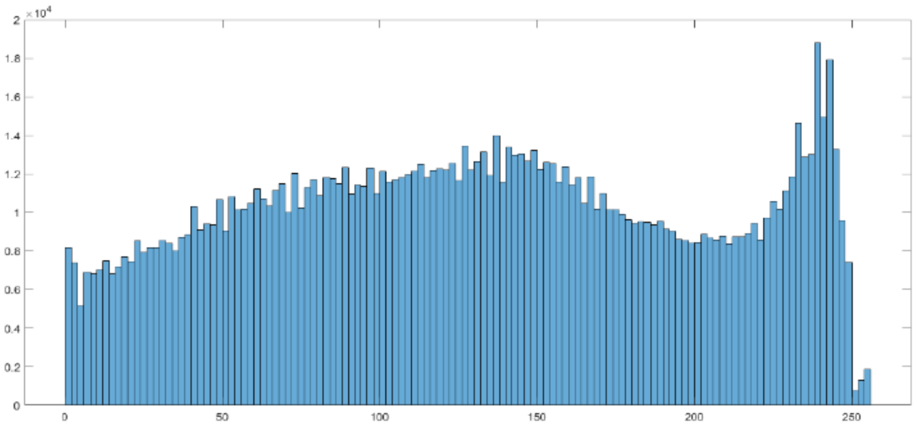


Figure 7: histogram of Gray image

After histogram equalization, RGB and gray image histograms are evenly distributed, which is a good sign. It indicates that the histogram equalization algorithm increased the image's contrast without adding any unwanted effects.

2.1.3 Gray to Binary conversion ()

The gray image is binarized using imbinarize() to convert it to a black and white image aka binary images. They are commonly observed as a logical array, with 0 representing a black pixel and 1 representing a white pixel. This is done by thresholding the image, meaning that all pixels with a brightness value above the threshold are set to white, and all pixels with a brightness value below the threshold are set to black. The goal is to choose a threshold value that separates the foreground objects (trees) from the background (buildings). Hence, we can see that buildings are segmented as white pixels and trees as black pixels.

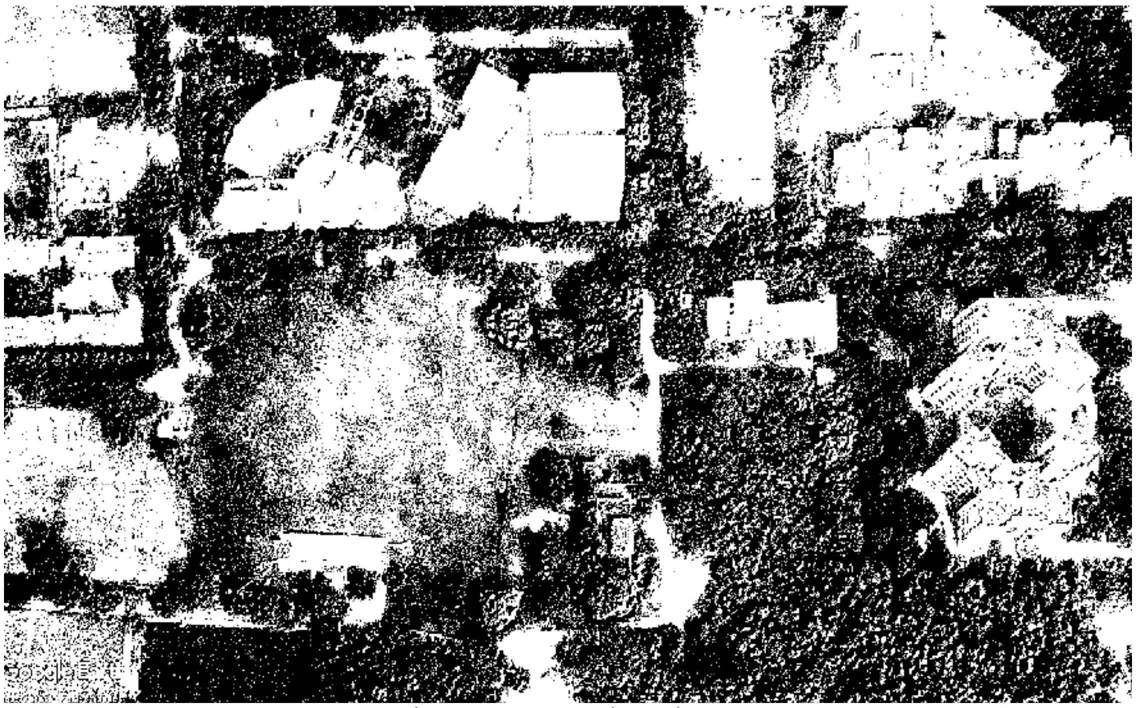


Figure 8: Gray to Binary image

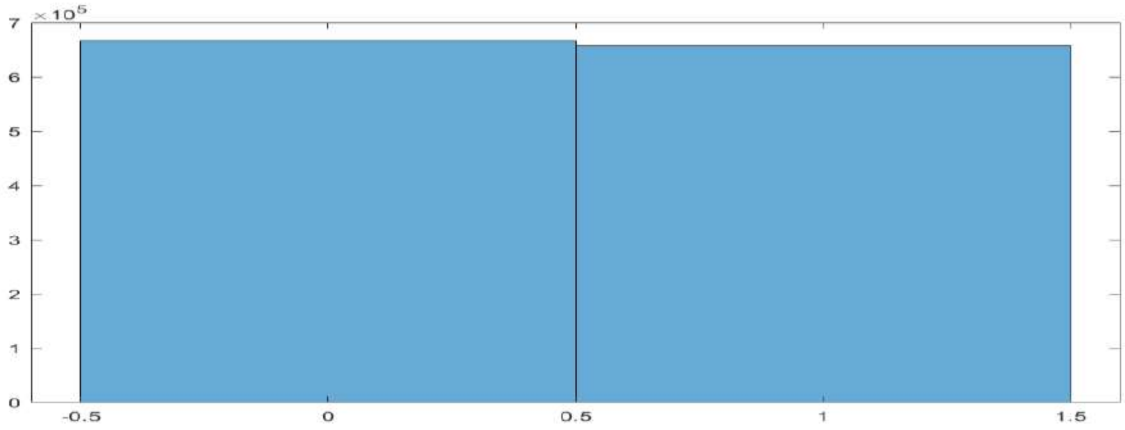


Figure 9: Histogram of binary image

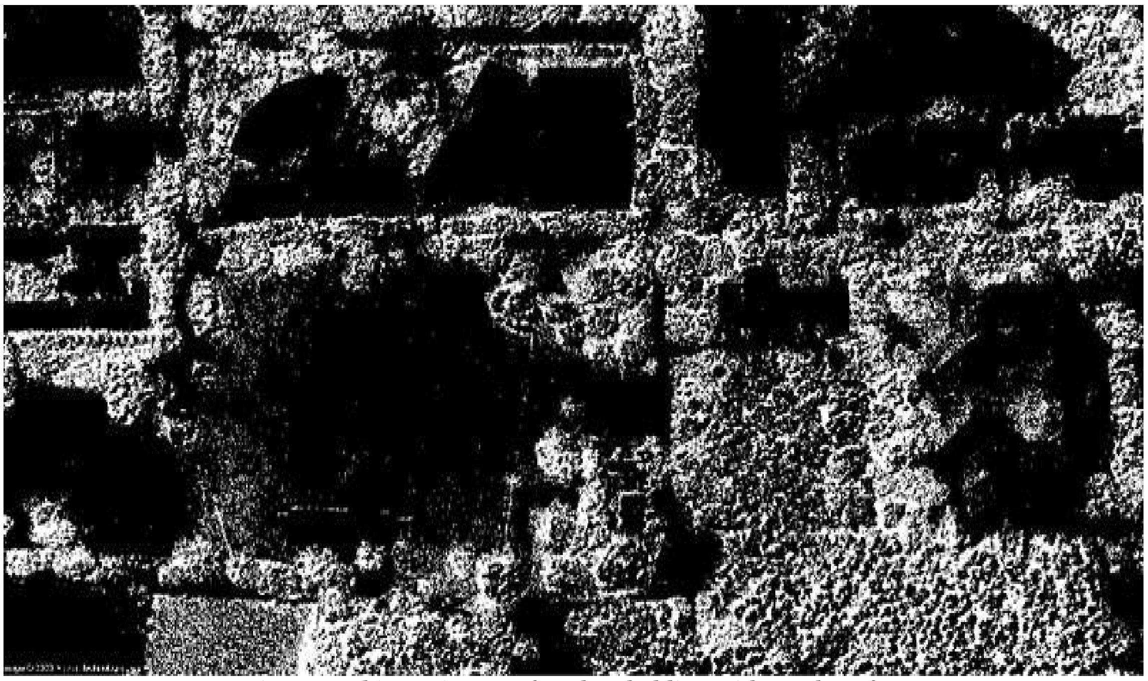


Figure 10: binary image after thresholding with a value of 65

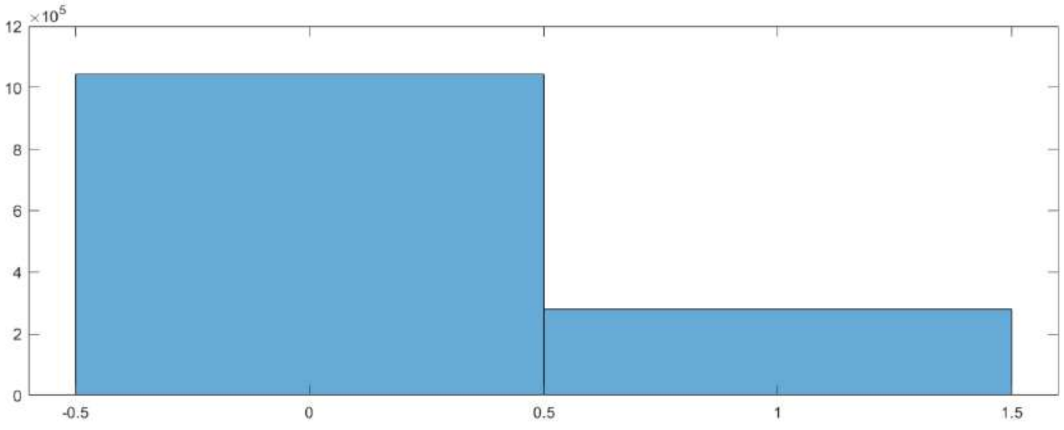


Figure 11:

- The fact that the binary image histogram is unequally distributed after thresholding with a value of 65 indicates that the thresholding operation has successfully separated the trees from the buildings.
- The image still has a lot of noise after histogram equalization and thresholding. Histogram equalization is only able to improve the contrast of an image, not remove noise. Thresholding can also introduce noise into the image.

2.1.4 Improving Segmentation: Cleaning Binary Masks (Morphological Operations)

The image in Fig 10 is noisy, so we use MATLAB's bwareaopen() function for image segmentation. The bwareaopen() function in MATLAB is used to remove small objects from a binary image. It works by identifying connected components in the image and removing any connected components that have fewer than a specified number of pixels. In this case, we are using a value of 40 for the bwareaopen() function. This means that any connected components with fewer than 40 pixels will be removed from the image.

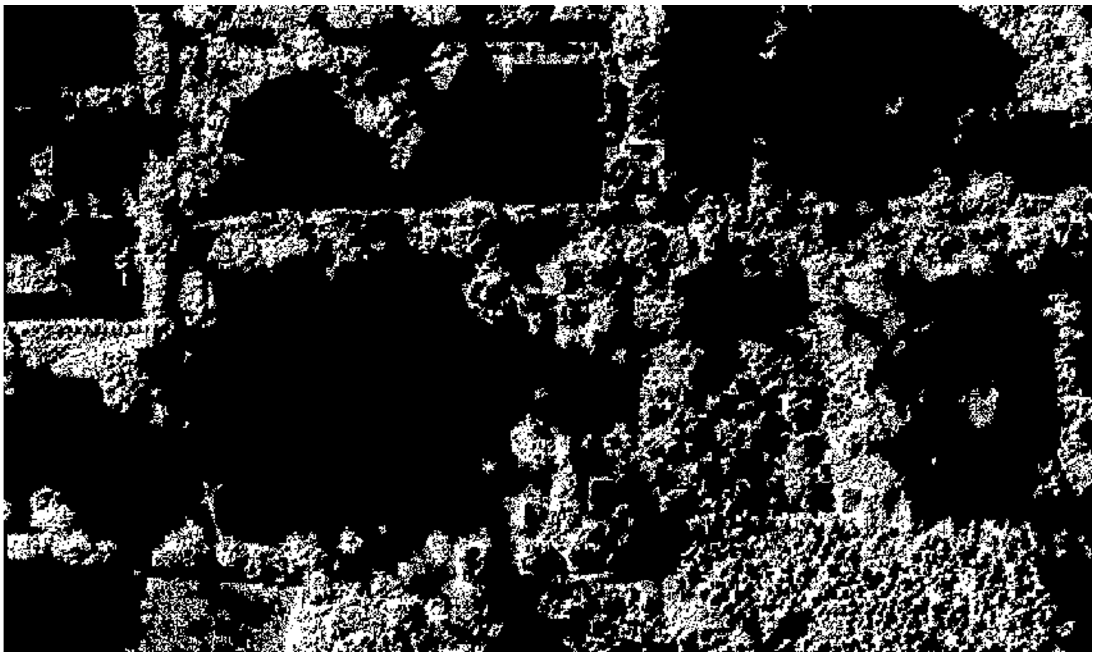


Figure 12: Masked image after applying bwareaopen

- Unfilled masked images preserve the boundaries of individual trees. This is important for accurate tree counting, especially in images where the trees are close to each other.
- Unfilled masked images are less likely to merge neighboring trees together. This is because the filling process can be fooled by noise or other artifacts in the image, which can lead to neighboring trees being merged together.

2.1.5 Tree counting using connected components

We use the bwconncomp() function to calculate the number of trees in the image. By identifying and labeling connected components in the image, we can count the number of connected components to determine the number of trees in the image. A connected component of a mask is a group of neighboring foreground (white) pixels.
In 8-connected pixel tree counting, each pixel in the image is considered to be connected to its 8 nearest neighbors. This means that a pixel can be connected to up to 8 other pixels.

4-connected pixel tree counting is generally better for counting trees that are very close to each other, however, it is important to note that 4-connected pixel tree counting may not be accurate for all images. For example, if the trees in the image are very large and have many branches, 4-connected pixel tree counting may underestimate the number of trees in the image. Number of trees found to be 550.

2.2 Study Area 2

Colour Spaces

The image utilized consist of three bands i.e., RGB a.k.a truecolour. The resolution of the image is 6447x4920 pixels, 9th Oct 2021. This image is significantly larger than the image 1. It has a higher pixel count both in width and height, which means it contains more data and detail. The larger image (6447x4920 pixels) captures finer details because it has more pixels to represent the scene. It provides a higher level of detail as we can see in fig 13.

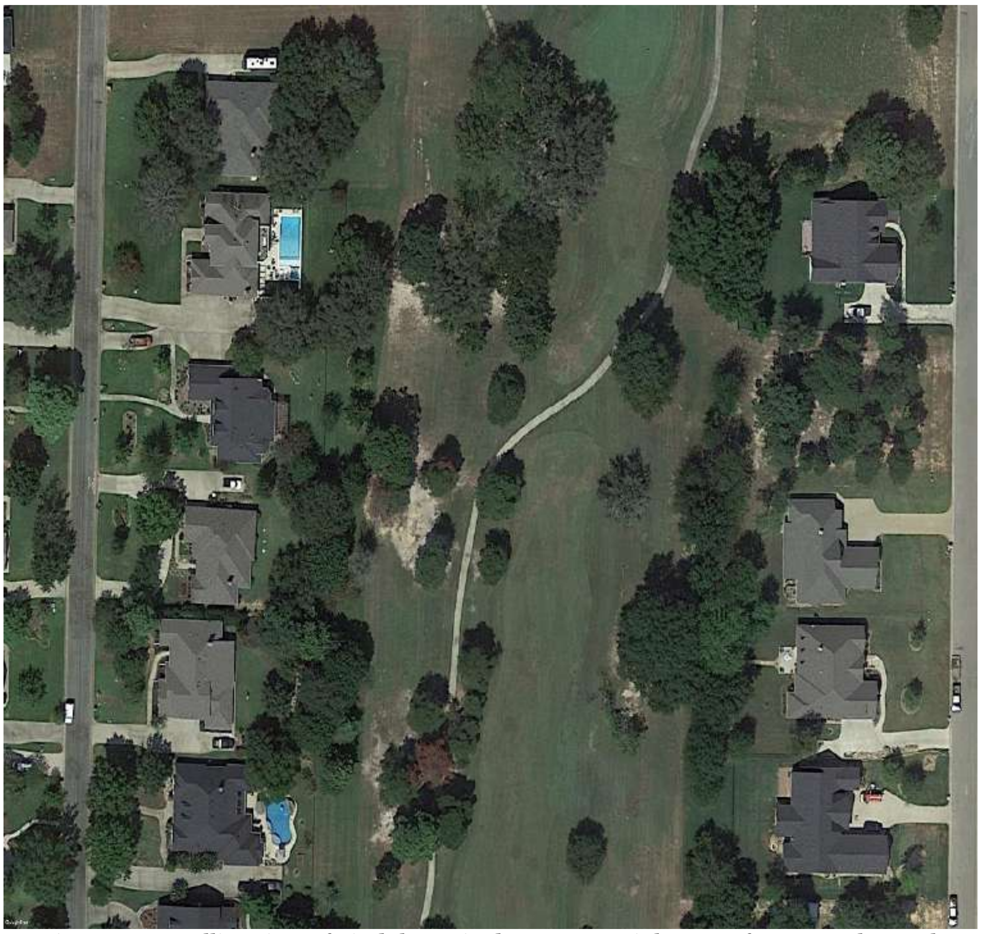


Figure 13: Satellite View of Lindale, United States - Aerial Image from Google Earth

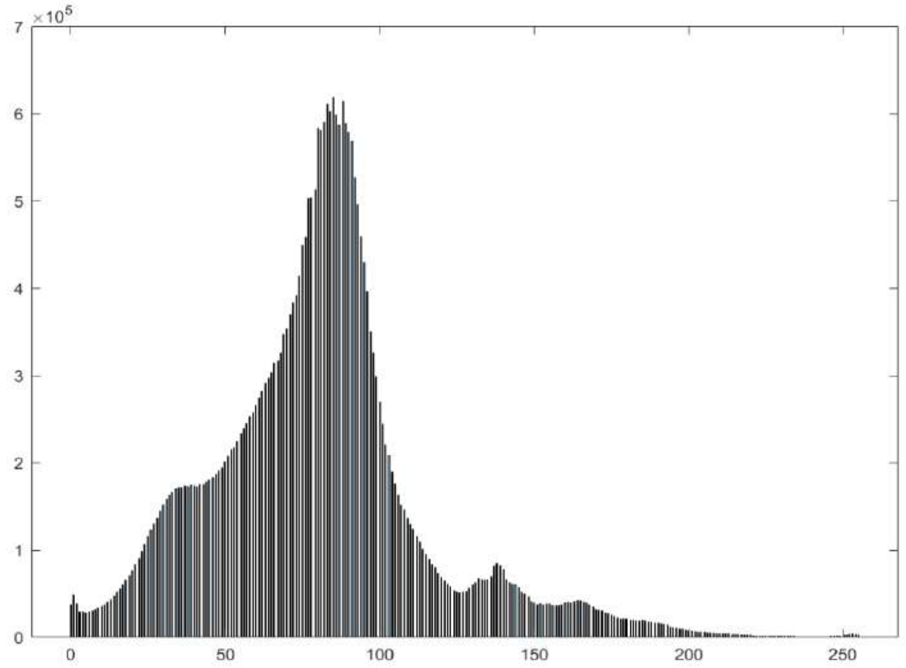


Figure 14: Histogram of the Image 2

2.2.1 Pre-Processing: image enhancement techniques

Histogram equalization

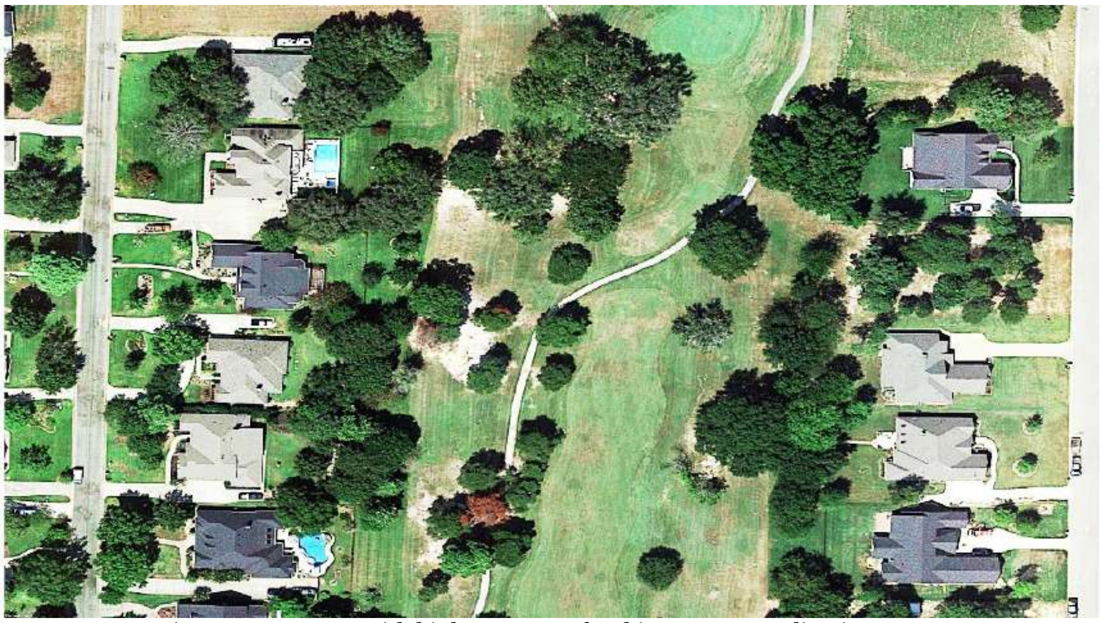


Figure 15: Image2 with high contrast after histogram equalization

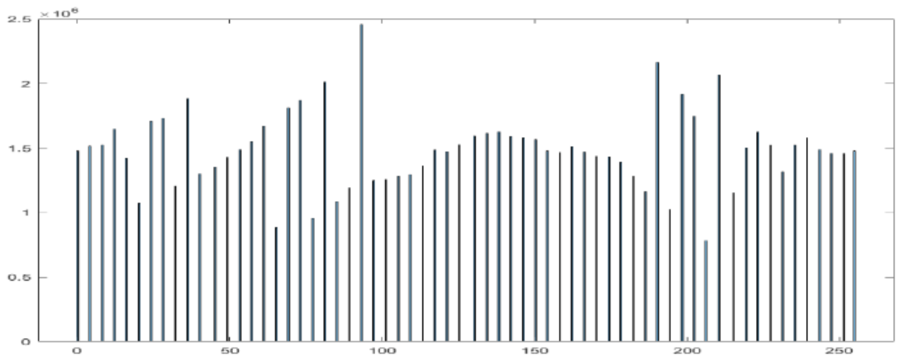


Figure 16: Enhanced Image2 Histogram after Histogram Equalization

2.2.2 RGB to Gray conversion

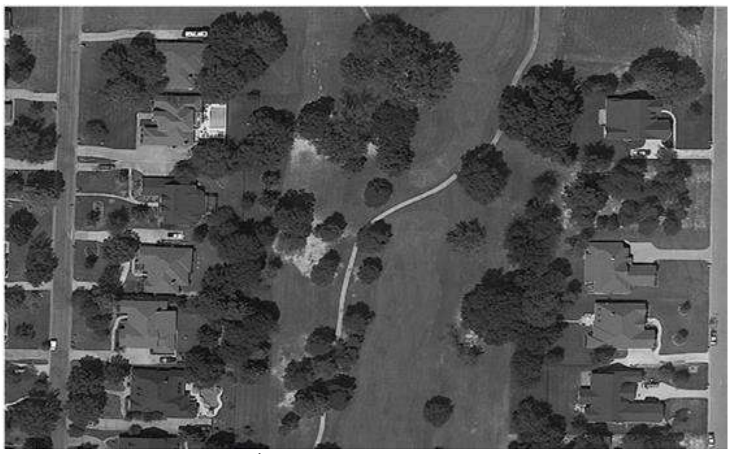


Figure 17: Gray Image 2

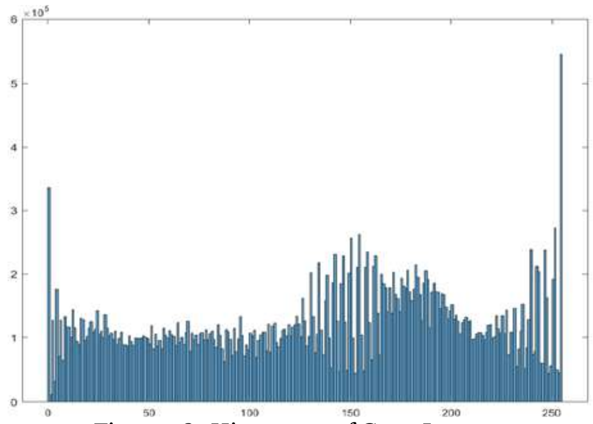


Figure 18: Histogram of Gray Image2

2.2.3 Gray to Binarize (Thresholding)

Binarization converts the image to a black-and-white image, which is much simpler to process than a grayscale image. This makes it easier to detect the edges of the trees in the image.
Binarization process converts all pixel values below a certain threshold to black and all pixel values above the threshold to white. This can result in a histogram with two peaks, one corresponding to the black pixels and the other corresponding to the white pixels.

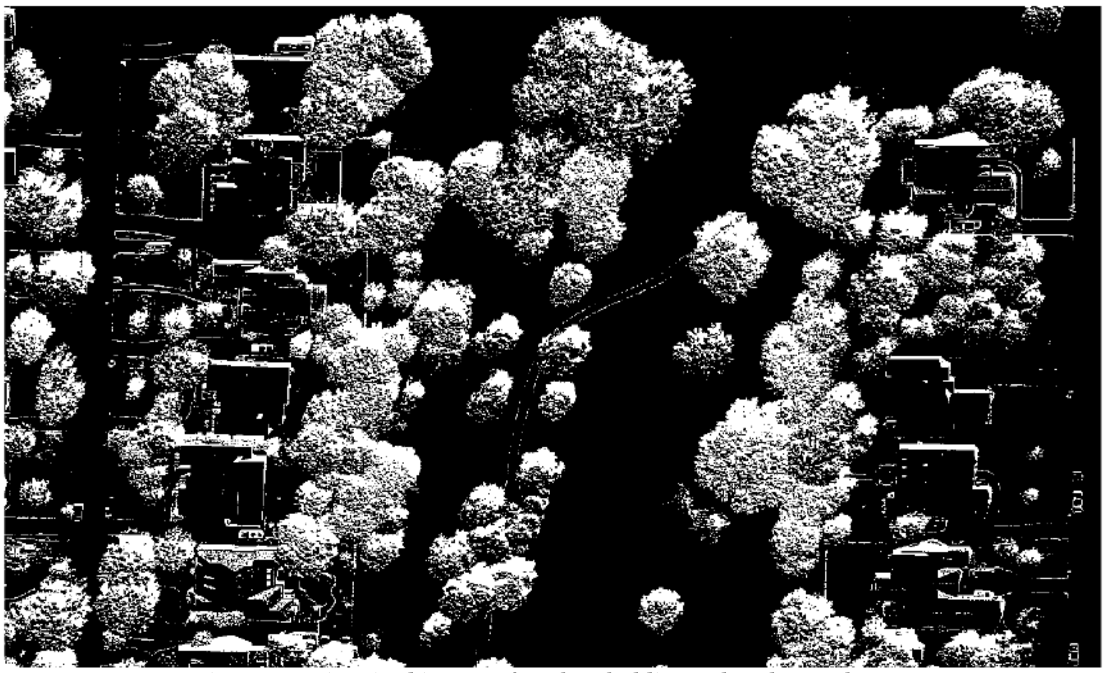


Figure 19: Binarized image after thresholding values lesser than 65

Thresholding is a technique used to convert a grayscale image to a binary image. A binary image is an image that consists of only two pixel values: black and white.

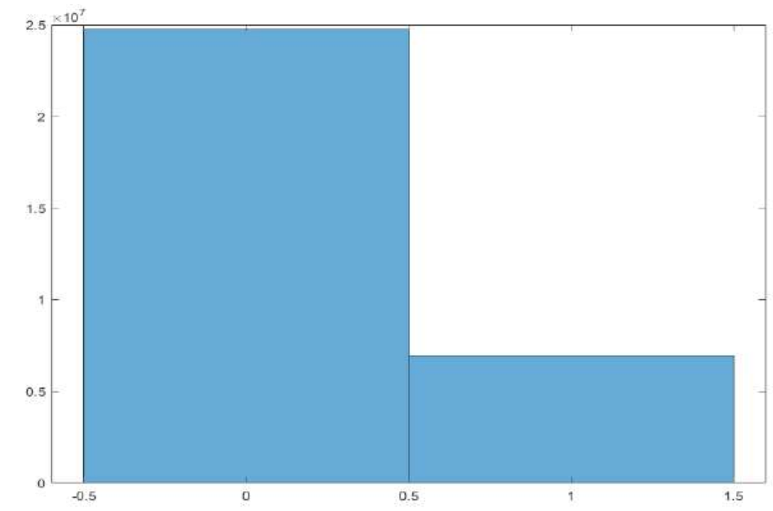


Figure 20: Histogram of binarized image

2.2.4 Edge detection (post-processing)

we used Sobel edge detection to detect the edges of trees in satellite images. Sobel edge detection is a gradient-based edge detector that calculates the gradient of the image at each pixel. The gradient of an image is a measure of how quickly the intensity of the image changes at a particular pixel. the intensity of the image also changes slightly inside the trees. This is because the trees are not perfectly uniform. The Sobel edge detector can also detect these small changes in intensity and mark them as edges. This is why you see small dots inside the trees after applying Sobel edge detection to a binarized image.

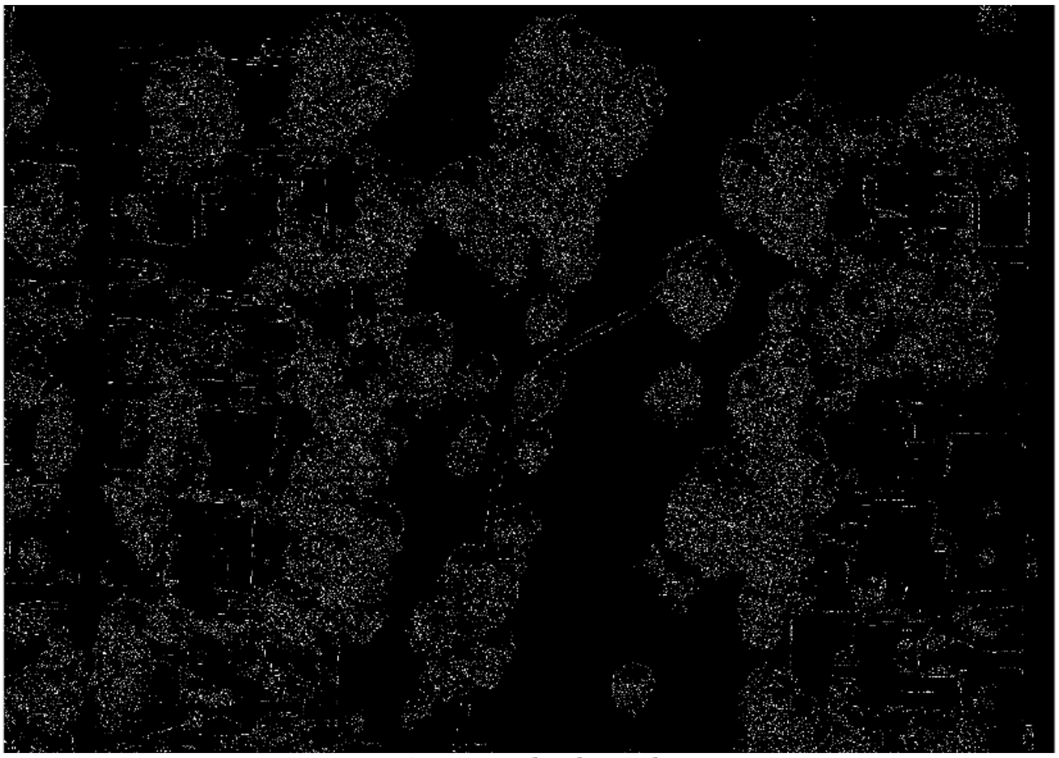


Figure 21: Edge detected

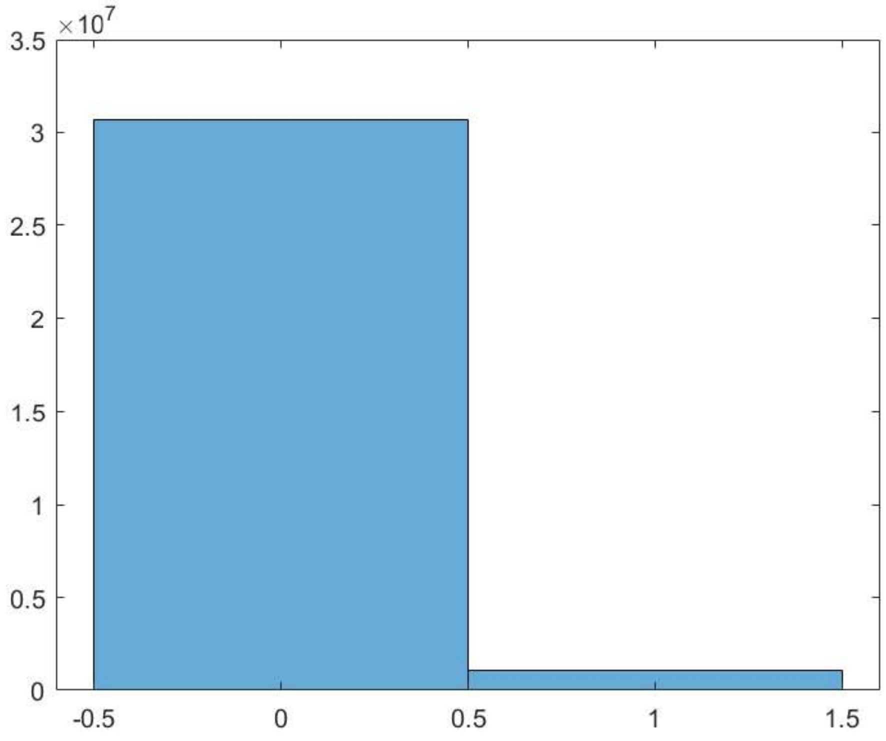


Figure 22: Histogram of edge detected

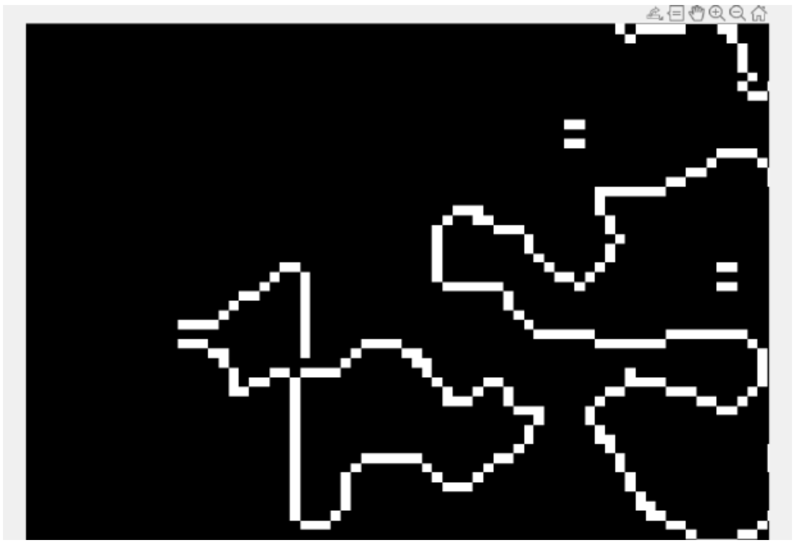


Figure 23: unconnected edges zoomed in

2.2.5 Edge closing (Morphological Operations)

To remove these small dots, we use the imclose function with a disk structuring element of radius 4. The closing operation is a combination of dilation and erosion. It works by first dilating the image to fill in any small gaps in the foreground. Then, it erodes the image to remove any small protrusions of the background. This will close any gaps in the edges of the trees and remove small objects.
The imclose function closes any gaps in the edges of the trees. This is important because gaps in the edges can lead to errors in the tree detection algorithm. For example, if a tree has a gap in its edge, the tree detection algorithm may mistake the gap for another tree. By closing the gaps in the edges, we can reduce the likelihood of these errors.
The imclose function also removes small objects from the image. This is important because small objects inside the trees can be confused with trees by the tree detection algorithm. For example, if there is a small branch or leaf inside a tree, the tree detection algorithm may mistake it for a tree. By removing small objects from the image, we can reduce the likelihood of these errors.
The strel('disk',4) function creates a disk-shaped structuring element with a radius of 4 pixels. The size and shape of the structuring element will affect the outcome of the morphological operation. This structuring element is used by the imclose function to close the gaps in the edges and to remove small objects. The radius of 4 pixels is a good choice for this application because it is large enough to close most gaps in the edges and to remove most small objects, but it is not so large that it merges neighboring trees together. Overall, the imclose function helps to connect and smooth the objects, improving their shape and structure.

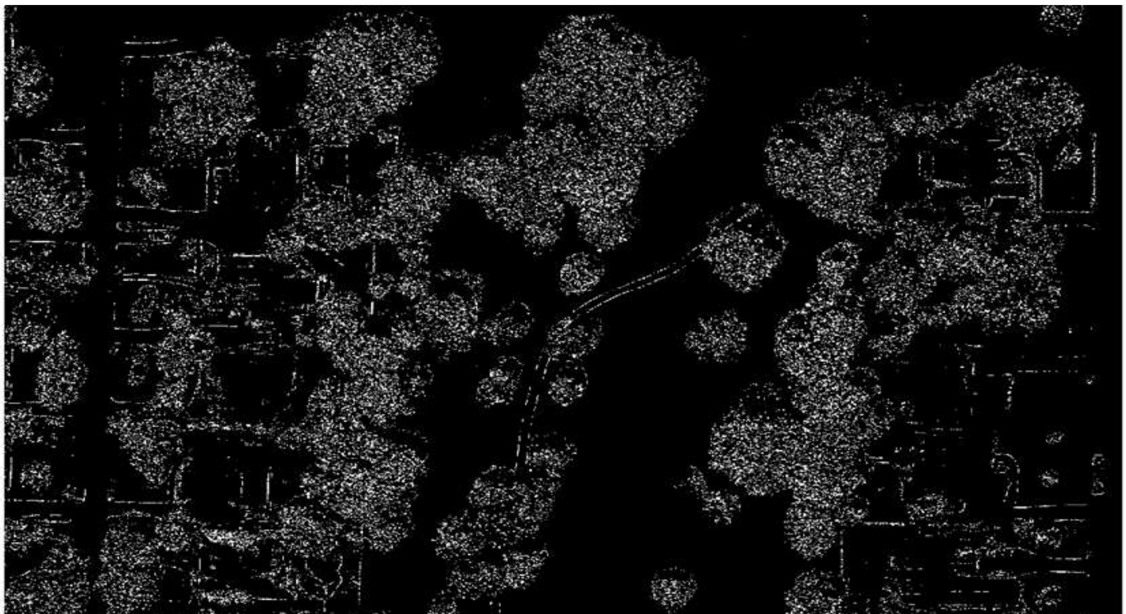


Figure 24: Closed edges of a binary image

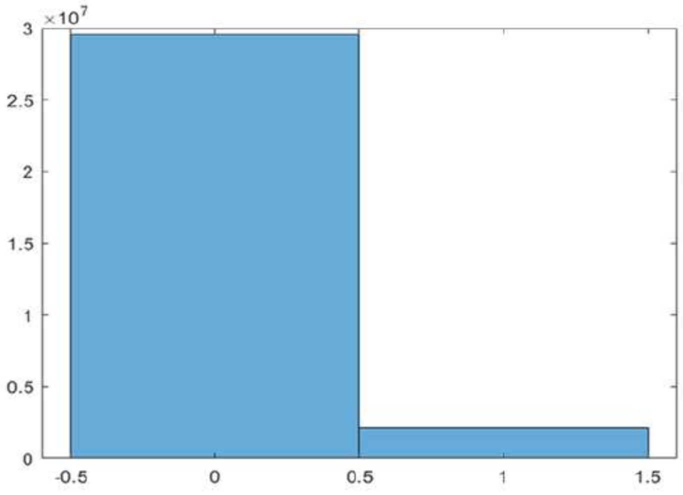


Figure 25: Histogram of closed edge binarized image

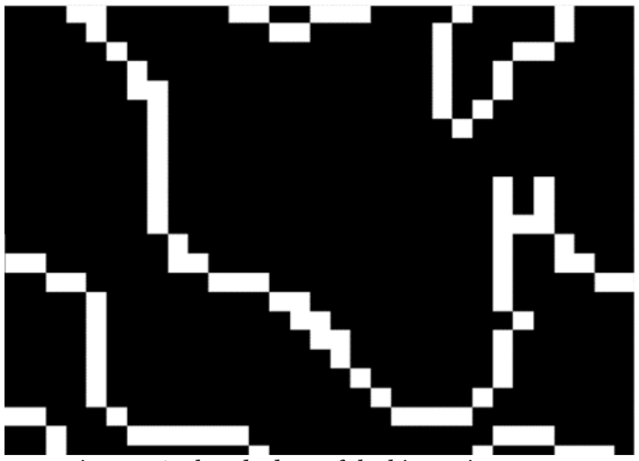


Figure 26: closed edges of the binary image

2.2.8 Improving Segmentations: Cleaning binary Masks

After closing the objects, we still have small, undesired objects or noise present in the binary image. We use the bwareaopen function and bixtor in the binary image and the edge detected image with a threshold value of 2000 to remove small objects and noise from the image.

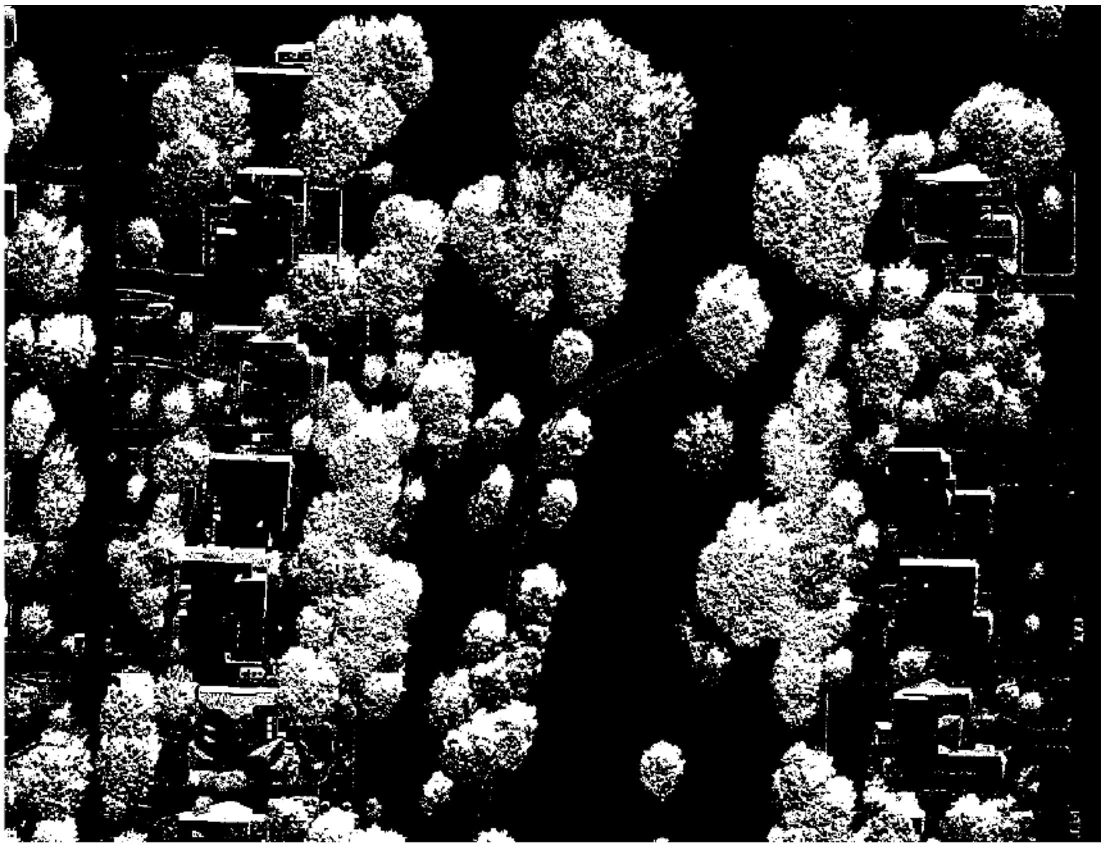


Figure 27: Segmented image using bixtor

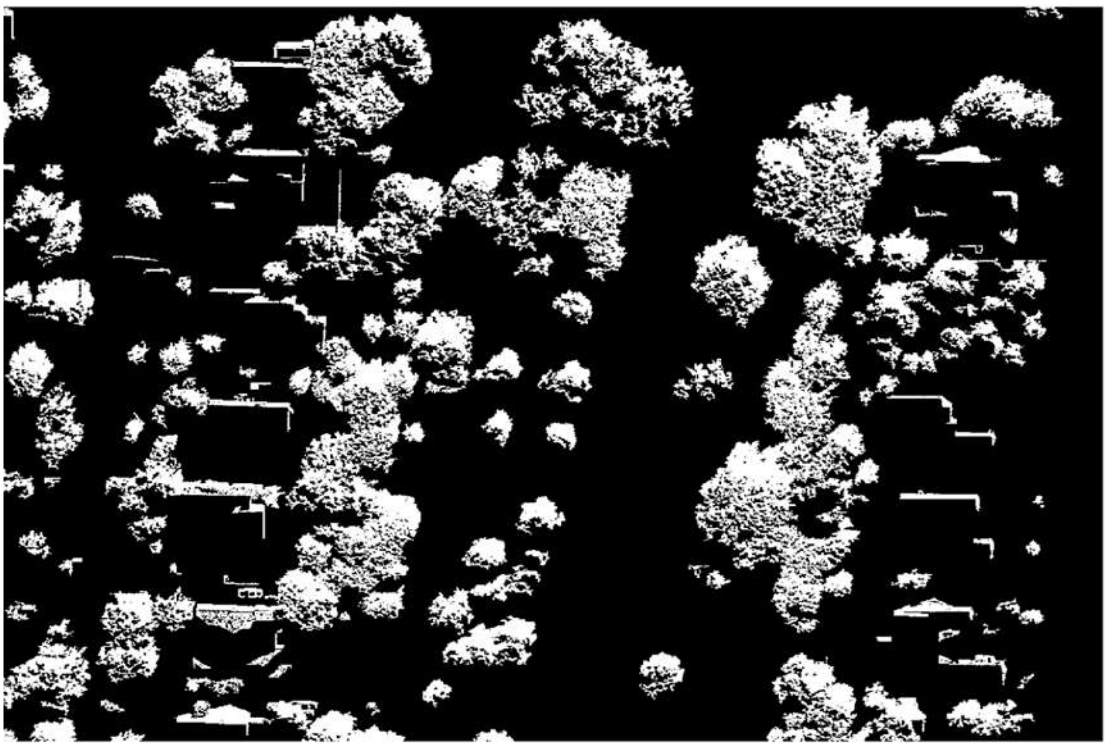


Figure 28: Segmented image using bwareaopen

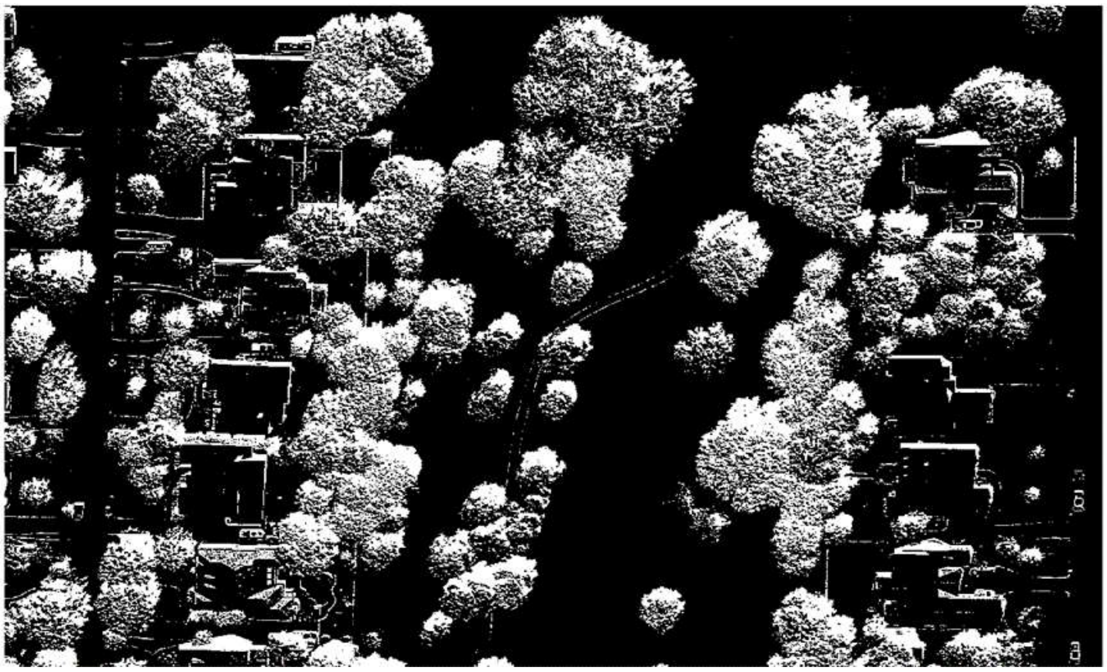


Figure 29: Initial binary image

- The bixtor function performs a bitwise XOR operation on two binary images. This operation flips the bits of the first image wherever the corresponding bits of the second image are set.
- The bwareaopen function removes small objects from a binary image. It does this by comparing the area of each object in the image to a specified threshold value. If the area of an object is smaller than the threshold value (2000), then the object is removed from the image.
- The threshold value of 2000 is used to determine which objects should be removed from the binary image by the bwareaopen function. If the area of an object is smaller than 2000 pixels, then the object is removed from the image.

This makes the trees more visible and reduces the amount of noise in the image, which makes it easier to count the trees. The bwareaopen function removes small objects from a binary image. It does this by comparing the area of each object in the image to a specified threshold value. If the area of an object is smaller than the threshold value, then the object is removed from the image. The bixtor operation between a binary image and a mask that isolates the region of interest that is the forested area in this case, it helped in emphasizing the tree objects within that region.

2.2.9 Finding and analyzing Objects

Upon obtaining a well-segmented image of the trees, the tree counting process is executed using the bwconncomp function. This function efficiently identifies and labels connected components, enabling the precise enumeration of individual tree canopies.

3 Algorithm:

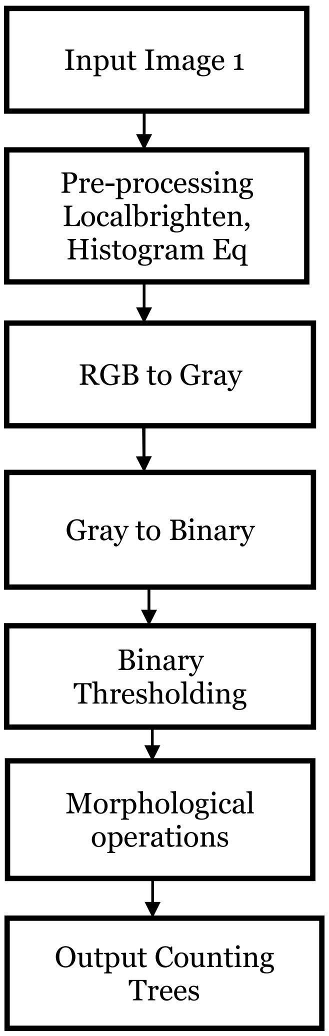


Figure 30: Block Diagram 1

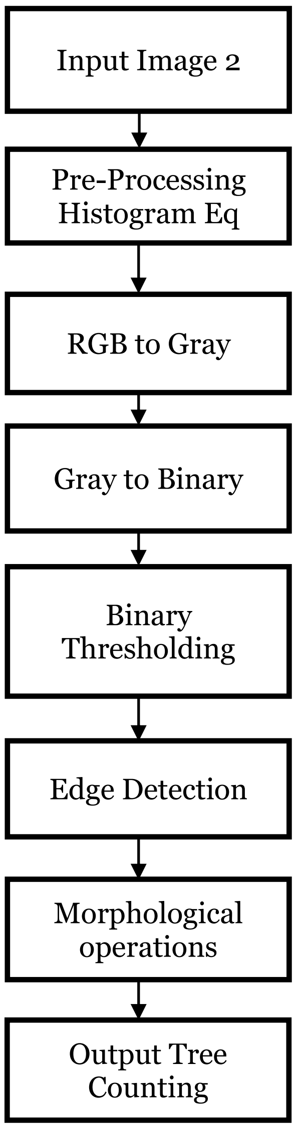


Figure 32: Block Diagram 2

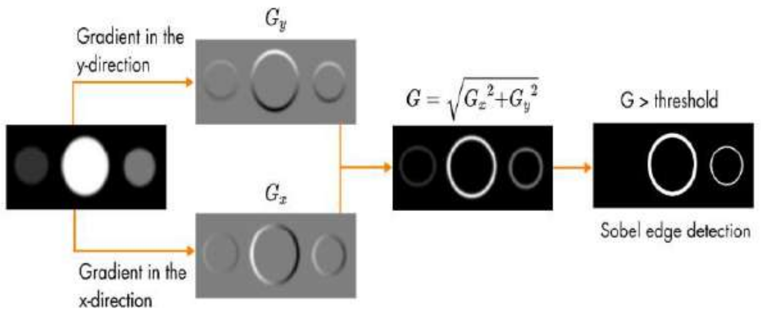
Problem Identification: Taking satellite images of various regions with trees, one heavily populated and the others sparsely populated, with complex background.

Preprocessing: Such as Localbrighten and Histogram Equalization may be necessary to modify the contrast to enhance the visual quality of an image or optimize the effectiveness of other image processing algorithms. Histogram equalization is a method used to enhance the contrast of an image by redistributing the pixel intensity values in such a way that the full dynamic range of intensity levels is better utilized. This algorithm is applied to the histogram of the image to stretch or equalize the pixel intensity distribution.

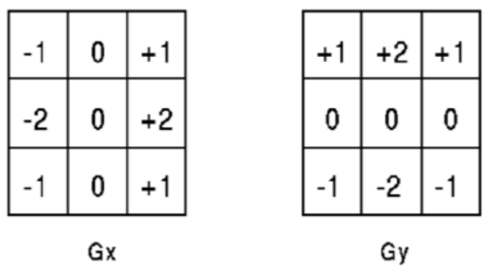
Binary Thresholding: Segment the image by thresholding the gray image.

Morphological operations: Improving Segmentations, by Cleaning Binary Masks.
Morphological operations are filter operations in image processing that are based on the shape and structure of objects in an image. These operations involve replacing pixels with either the minimum or maximum values from their neighborhood or a combination of these values. The four main operations are erosion, dilation, opening, and closing. Bwareaopen, is an opening operation, achieved by first eroding and then dilating an image, eliminates small foreground regions enclosed by background while preserving the size of larger objects.

Edge Detection: The Sobel edge detector uses a pair of 3 x 3 convolution masks, one estimating gradient in the x-direction and the other estimating gradient in y–direction.



Gradient in the y-direction
G_y
$G = \sqrt{G_x^2 + G_y^2}$
G > threshold
G_x
Gradient in the x-direction
Sobel edge detection



-1	0	+1
-2	0	+2
-1	0	+1

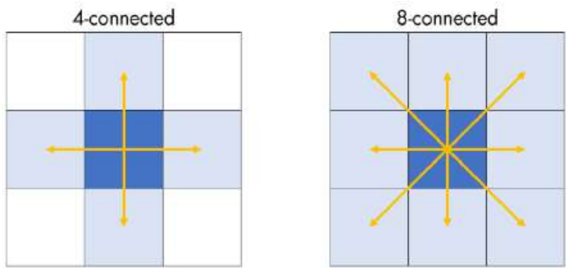
Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 31: Sobel convolution kernels

Finding and Analysing Objects: bwconncomp function was used for counting the number of trees. bwconncomp uses a connected component labelling algorithm to identify and label all the connected components in a binary image. To label a connected component, bwconncomp first finds an unlabelled pixel in the image. It then uses the flood-fill algorithm to fill in all the adjacent pixels that have the same value as the starting pixel. Once all the adjacent pixels have been filled in, bwconncomp assigns a unique label to the connected component. bwconncomp repeats this process until all the pixels in the image have been labelled. With 8-connectivity, when the algorithm labels connected components, it is more likely to group adjacent pixels as part of the same tree, even if they are diagonally adjacent. This helps in reducing the risk of overcounting trees, especially when trees are close to each other. 4-connectivity could result in counting multiple trees as separate entities when they are, in fact, part of the same tree canopy.



4 Results

	Sample 1	Sample 2
Actual Trees	650	200
Detected Trees	550	198
Composition	RGB	RGB
Percentage Accuracy	84.6%	99%

5 Conclusion

The aim of this study was to gather precise information about individual trees in both a densely populated environment and a sparsely populated one, where the backdrop presented a complex and intricate landscape. Filling in the holes in a masked image of trees can reduce the number of trees counted, especially if the trees are very close to each other. This is because the filling process can merge neighboring trees together, creating a single tree object instead of two or more tree objects. Tree counting through satellite remote sensing offers a cost-effective and efficient method for monitoring large forested areas. However, accuracy and spatial resolution limitations should be considered. Tree counting in different scenarios using satellite remote sensing can be achieved with varying degrees of accuracy depending on the spatial resolution of the imagery and the complexity of the scene. In general, higher spatial resolution imagery will result in easier image detection and more accurate tree counts. However, even with high-resolution imagery, some scenes may be too complex to accurately count trees using traditional methods. In these cases, different morphological methods can be used to improve the accuracy of tree counts. Another approach to tree counting is to use machine learning techniques to classify pixels in the satellite image as trees or non-trees. This approach is particularly useful for counting trees in dense forests or urban areas.
Overall, the study contributes to our understanding of the challenges and opportunities in tree counting through satellite remote sensing. It reinforces the significance of this methodology in environmental monitoring and management while acknowledging that a combination of techniques is often required to achieve the most precise results. Further research and refinement of these methods will continue to advance our capacity to monitor and conserve tree populations effectively.

References

R. K. (2018). Tree Crown Detection, Delineation and Counting in UAV Remote Sensed Images: A Neural Network Based Spectral–Spatial Method. *Journal of the Indian Society of Remote Sensing*.

Ankit Arya, S. N. (2017). Carbon Sequestration Analysis of dominant tree species using Geo-informatics Technology in Gujarat State (INDIA). *International Journal of Environment and Geoinformatics (IJEGEO).*

Aparna P, H. M. (2018). CNN Based Technique for Automatic Tree Counting . *International Conference on Design Innovations for 3Cs Compute Communicate Control.*

Giriraj Amarnath, S. B. (2017). Evaluating MODIS-vegetation continuous field products to assess treecover change and forest fragmentation in India – A multi-scale satellite. *The Egyptian Journal of Remote Sensing and Space Sciences*.

VibhaL, P. V. (2009). RobustTechniqueforSegmentationandCounting. *IEEEInternationalAdvanceComputingConference(IACC2009)*.

Xinyu Dong, Z. Z. (2020). Extraction of Information about Individual Trees from High-Spatial-Resolution UAV-Acquired Images of an Orchard. *Remote Sens*.

MATLAB