A Versatile Image Pre-processing Application for Geographic Information Systems.

Spencer Lobo
Dept. of Geomatics Engineering
University of Calgary
Calgary, Canada
spencer.lobo@ucalgary.ca
30228989

Anan Ghosh
Dept. of Geomatics Engineering
University of Calgary
Calgary, Canada
anan.ghosh@ucalgary.ca
30220833

Debistuti Kundu
Dept. of Geomatics Engineering
University of Calgary
Calgary, Canada
debistuti.kundu@ucalgary.ca
30227696

Himadri Das
Dept. of Geomatics Engineering
University of Calgary
Calgary, Canada
himadri.das@ucalgary.ca
30214525

Ariful Islam Anik
Dept. of Geomatics Engineering
University of Calgary
Calgary, Canada
arifulislam.anik@ucalgary.ca
30230898

***Abstract*—The emergence of machine learning, computer vision and geographic information systems (GIS) has made image pre-processing an essential stage in image enhancement and manipulation. The need for a robust image processing tool is required to perform versatile image pre-processing applications that can perform a variety of techniques on images. For Instance, geographic information systems (GIS) relies on a well-designed image processing system that can extract crucial information from maps, aerial photographs or spatial (satellite) images. This is done to reduce the noise and improve the quality of the image. This program would include image processing tools, such as resizing, rotating, and flipping on an image. Enhancement techniques such as image sharpening and blurring. Using filters to reduce the noise in an image and technical image processing techniques such as Edge Detection, Erosion, and Dilation. The design of the image processing application would be user-friendly, allowing users to upload images and save them to their desired location. Additionally, it enables the user to apply multiple image-processing techniques to generate the desired output the user requires.**

***Index Terms*—image processing, pre-processing, Geographic Information Systems (GIS), Data interpretation, Image manipulation**

I. Introduction

Our Project "A versatile image pre-processing application designed for Geographic Information Systems (GIS)" focuses on image processing and pre-processing tools in the field of geographic information systems (GIS). It explores the various tools that can be applied for image manipulation and enhancement of the visual data.

This project uses multiple image manipulation and enhancement techniques such as image enhancements such as image sharpening, and filters such as Gaussian Blurr and Prewitt Filters, using these tools for feature extraction and text extractions. While focusing on using these techniques in the GIS domain, attempting to retrieve features, and data from map images the same time developing a robust tool with multiple image processing tools that can be used by the user to get the desired image results at the same time offering the user a range of functions for altering pictures, such as scaling, rotating, and flipping. With capabilities like picture sharpening, blurring, and noise reduction through a filter at the same time seamlessly integrating technical image processing techniques such as Edge Detection, Erosion, and Dilation. All this is achieved while prioritizing user-friendliness, making it simple for the user to upload their images by utilizing Google Drive and selecting the image processing tools that they prefer to apply

This research applies image pre-processing in the field of Geographic Information Systems by presenting a versatile and forward-thinking solution where technology meets simplicity.

II. Literature Review

Maps offer vital information about the Earth's surface and are fundamental tools in geoscience and remote sensing. In recent years, Geographic Information Systems (GIS) have emerged as transformative tools at the nexus of technology and spatial analysis, revolutionizing the perception, interpretation, and interaction of geographical information. Through the lens of GIS images, this technology has been instrumental in capturing, managing, and analyzing spatial data. The transition from conventional paper maps to digital formats has increased the demand for sophisticated methods in automatic text recognition and feature extraction.[1] The goal of this integrated methodology is to build an integrated system that can extract textual information in addition to spatial features, resulting in a more accurate and comprehensive digital representation of geographic features. The current state of automatic feature extraction and text recognition from scanned topographic maps is evident that continuous advancements in this area are crucial for enhancing the efficiency and accuracy of geographic data processing. To improve text recognition

accuracy on topographic maps, the integration of spatial relationships, linguistic analysis, and contextual information has been investigated. The digital version of a map from GIS images serves as a bridge between raw spatial data and actionable intelligence, from monitoring environmental changes and land use patterns to urbanization. The study "An improved contrast enhancing approach for color-to-grayscale mappings" [2] employed a method of adaptive thresholding for grayscale images in terms of computational efficiency. As a part of basic image processing, applying Gaussian Blur on GIS images not only minimizes artifacts but also outperforms conventional techniques to adapt to varying noise levels.[3] To extract the inside information of the GIS image the edge detection algorithm showcases efficiency, a critical aspect of the requirements of computational power. A paper titled "Color-Based Image Segmentation Using Adaptive Thresholding ", analyzes the superior performance under different conditions for binary thresholding.[4] The advancement of morphological image processing outperforms GIS images to evaluate dilation techniques and emphasizes improved feature connectivity.[5] A novel approach in the field of histogram analysis with equalization based on statistical modeling illustrates the efficacy of preserving image information and overall image quality. [6]. A research paper titled Image Gradients "A Comparative Analysis of Gradient Boosting Algorithms" introduced a robust methodology to compute image processing with high accuracy by adapting various image characteristics.[7] To improve image quality, the algorithm from [8] goes beyond the traditional deblurring methods and addresses the challenges of image degradation. The dynamic landscape of GIS image processing unfolds as a symphony of innovation to understand spatial data. As a cutting-edge contribution, it becomes the synergy between GIS and image processing is not merely a scientific pursuit but an art- a canvas upon which the strokes the spatial insights converge.

III. Methodology

The Project Consists of Multiple functions that define the different image processing tools implemented into this application. Each function is designed to contribute towards image manipulation and the enhancement of the visual data. using these image processing tools. This application consists of 9 parts, these are:

1) Importing Data.
2) Basic Image Processing.
3) Morphological Operations.
4) Image Restoration.
5) Advanced Image Processing.
6) Feature Extraction.
7) Text Extraction.
8) User Input System.

A. Importing Data.

The Process begins by importing the data that is the image to be processed this is done by first mounting the Google Drive to the Google Colab Notebook, ensuring that the user can upload the image of their choice onto the Google Drive and copying the path (of image) and pasting the path in the Google Colab Notebook. This selects the image and imports it to Google Colab Notebook, where the image can be enhanced and manipulated using various processing tools ranging from basic image processing to advanced processing such as feature and text extraction. The original image is then displayed to the user.

B. Basic Image Processing.

The Basic Image Processing consists of the following tools:

1) Grayscale Conversion: Using the Pillow (PIL) library, a grayscale conversion is applied to the original image. This results in a grayscale image and an array corresponding to the grayscale image. This is an important process in image processing as this step can aid in finding the edges of an image, applying erosion, dilation, and advanced image processing algorithms this also aids with the text extraction process.

2) Gaussian Blur: The Gaussian Blur is a filter that can applied to a grayscale image to reduce the noise present in the image. The blurred image is then displayed with the resulting array for the processed image.

3) Image Sharpening: The image sharpening is a process that is used by combining Gaussian blur with alpha blending. The value of alpha or the sharpening value is based on the input value by the user, and the sharpening is then applied to the image. The image is then displayed.

4) Image Rotation: The Image Rotation is a user-specified process using the rotation angle (degree) specified by the user, and this is then applied to the image using a 2D rotation matrix the processed image is then displayed.

5) Image Resizing: Image Resizing is a user-specified process that uses the scaling factor defined by the user, and this is then applied to the original image, and the processed image is then displayed.

6) Canny Edge Detection: The Canny Edge Detection is an edge detection algorithm that is applied to the grayscale image. The Grayscale image is retrieved by passing the image through the "GrayScale" function block and the Canny edge detection algorithm is then applied to this image using the lower and upper thresholds based on the user's input, the edges are detected and displayed.

7) Binary Thresholding: The binary thresholding is a segmentation algorithm that is applied to the grayscale image. The Canny Edge function is called in this algorithm and the binary thresholding is applied based on the user-specified threshold values. The resulting binary image is displayed.

C. Morphological Operations.

The Morphological Operations consists of Erosion and Dilation

1) Erosion: Erosion is an image processing algorithm that shrinks the image pixels, It removes the pixels on object boundaries. The erosion is applied on the Edge detected image and the processed image is displayed.

2) Dilation: Dilation is an image processing algorithm that expands the image pixels, It adds pixels on object boundaries. The dilation is applied on the Edge detected image and the processed image is displayed.

D. Image Restoration.

Image Restoration consists of Histogram Analysis and Histogram Equalization

1) Histogram Analysis: The histogram is a graph or plot which is plotted is a graph that displays the frequency of pixels of a grayscale image, where the pixel values range from 0 to 255. The function utilizes the CV2 method cv2.calcHist to display the histogram plot.

2) Histogram Equalization: The Histogram Equalization is a restoration process that uses the cv2.equalizeHist function from the CV2 library to adjust the contrast using the histogram of the image this, intensifies the intensities, making the lower contrast gain a higher contrast, this is achieved by spreading the most frequent intensity value. The equalized image is then displayed.

E. Advanced Image Processing.

The Advanced Image Processing technique used in this paper are Double Thresholding and Prewitt Filters

1) Double Thresholding: Double Thresholding is a technique used in image processing used in edge detection and is used to highlight the edges in an image. This uses two threshold values on an image for better results. The threshold values are taken as inputs from the user and applied to the image. The image is then displayed.

2) Prewitt Filter: The Prewitt Filter is an image-processing algorithm used for edge detection. These filters highlight the regions of the intensity changes in an image. This is done by using a 3x3 convolution kernel, which detects and changes the intensities in the direction of the type of Prewitt filter applied. The Horizontal filter emphasizes edges that run horizontally, while the vertical filter emphasizes edges that run vertically, and the +45 and -45 filters emphasize edges that run diagonally. The Prewitt filter is applied to the image based on user input for the type of filter, and the filtered image is displayed.

F. Feature Extraction.

The Feature extraction block utilizes the canny edge detection along with the contour identification, by drawing contours on the original image, the features are highlighted and the results are displayed.

G. Text Extraction.

The text extraction is performed by using the Tesseract OCR engine, In order to extract the text, the grayscale image is used, Using the grayscale image and the Tesseract OCR engine the text is extracted and displayed.

H. User Input Operation.

This code block takes input from the user in order to determine the type of operation to be applied to the image. When a user enters a number, the corresponding image processing algorithm is applied, and the result is displayed. If the user displays an incorrect input, an error is displayed stating Invalid operation. If the user wishes to exit the code block, the user can do so by typing quit into the input box, which ends the application.

IV. Results

Gray scaling of image – Function 1 is used for the gray scaling of the uploaded image. Where user gets the output as a set of array data and a Grayscale of the original image [Fig1]

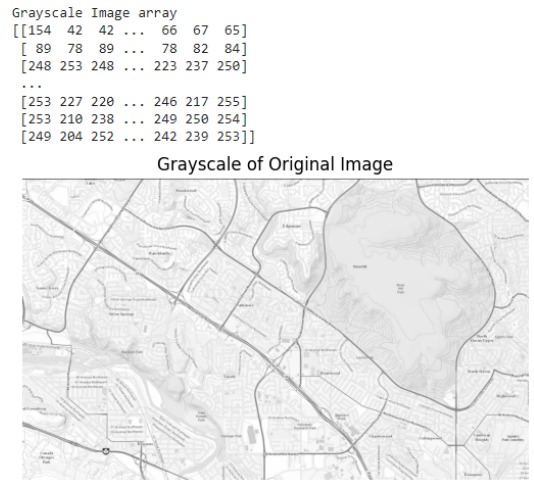


Fig. 1. Gray scaling of image

Gaussian Blur –Here an user can reduce the grain of a noisy image to get a well polished image with the function. [Fig 2]

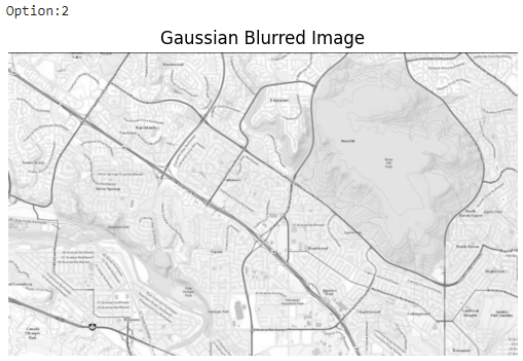


Fig. 2. Gaussian Blur

Image Rotation – In option 3 the user must need to put a desired angle first in case of rotating the image in a particular

Fig. 3. Image Rotation

direction. In [Fig 3] the input was a rotation of 90 degree angle.

Image Sharpening – While using the function for the sharpening of the uploaded image, at first the system turns the colored original image into a grayscale version then after the inclusion of alpha sharpening value user gets a sharpened image. [Fig 4]

Fig. 4. Image Sharpening

Image Resizing – Using this function requires scaling level to provide the user a resized image with desired aspect ratio. Here is a resized image of the original file with the aspect ratio of 16:10.

Fig. 5. Image Resizing

Edge Detection – The canny edge function works in multiple stage process. At first using the Gaussian convolution to smoothen out the original image and then a simple 2D first derivative operator is applied to the smoothened image to highlight regions of the image with high first spatial derivatives.

Binary Threshold – In this function it takes the threshold value from the user and compares each pixel to that particular

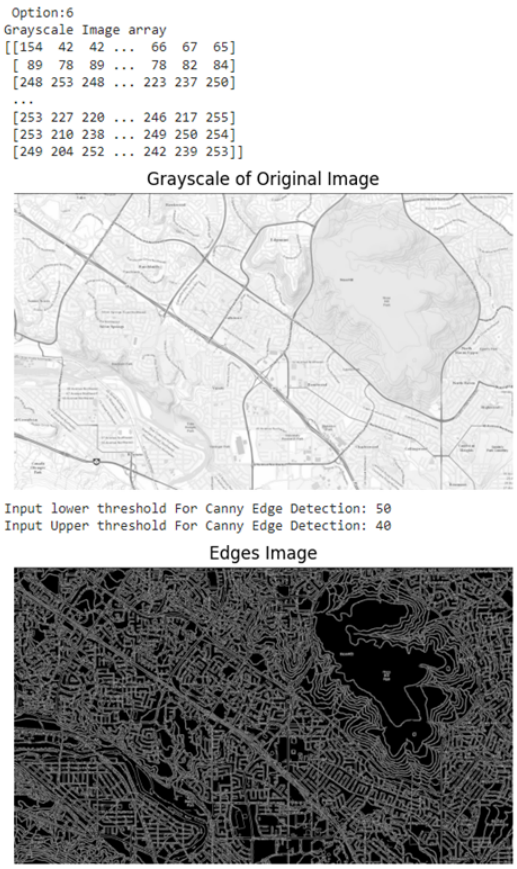


Fig. 6. Edge Detection

threshold value. If the pixel value is less than or equal to the assigned threshold, then the pixel value is set to zero or to the maximum value. [Fig 7 and 8]

Erosion – This function removes pixel on the object boundaries. Erosion shrink-ens the image pixels i.e. it is used for shrinking of element A by using element B. Erosion removes pixels on object boundaries.: The value of the output pixel is the minimum value of all the pixels in the neighborhood. A pixel is set to 0 if any of the neighboring pixels have the value 0.

Dilation – While running the Dilation function, it adds pixels to the boundaries of objects in the uploaded image. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image. [Fig10]

Histogram - The histogram plots the number of pixels in the image (vertical axis) with a particular brightness or tonal value (horizontal axis). Algorithms in the digital editor allow the user to visually adjust the brightness value of each pixel and to dynamically display the results as adjustments are made.

Double Threshold - The double thresholding function is to use the output from the higher threshold to select all the touching regions in the output from the lower threshold. [Fig12]

Prewitt Filter- The Prewitt filter is similar to the Sobel in

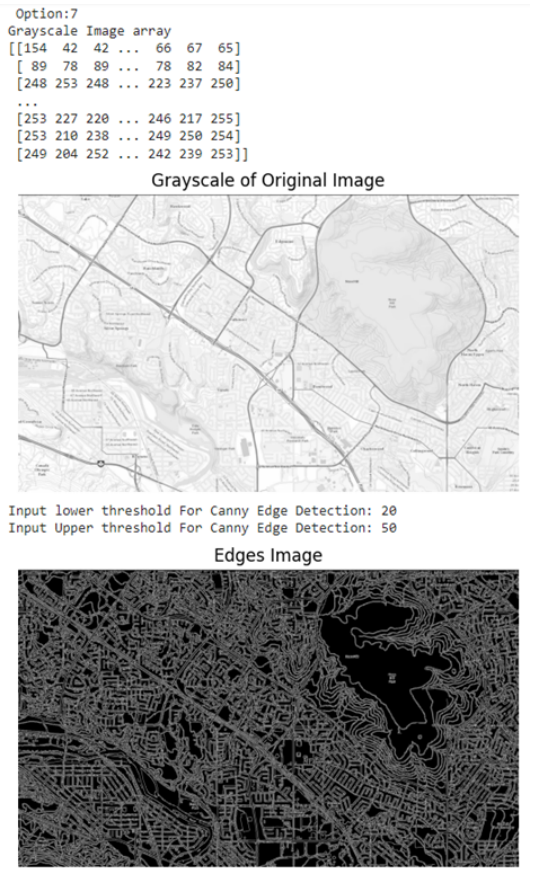


Fig. 7. Binary Threshold

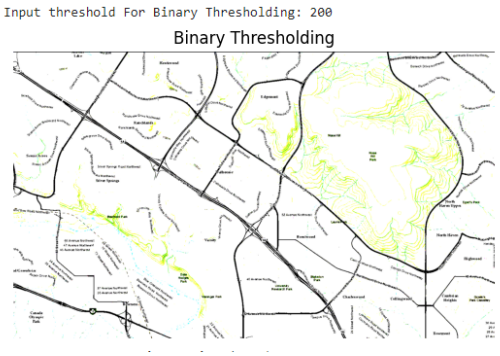


Fig. 8. Gray scaling of image

that it uses two 3 x 3 kernels. One for changes in the horizontal direction, and one for changes in the vertical direction. The two kernels are convolved with the original image to calculate the approximations of the derivatives. Users can use various types of filters and get different types of array data. Fig [13,14,15,16]

Text Extraction - Text extraction can be achieved by applying text detection that identifies image parts containing text, text localization finds the exact position of the text, text segmentation separates the text from its background and binarization process converts the colored images into binary.

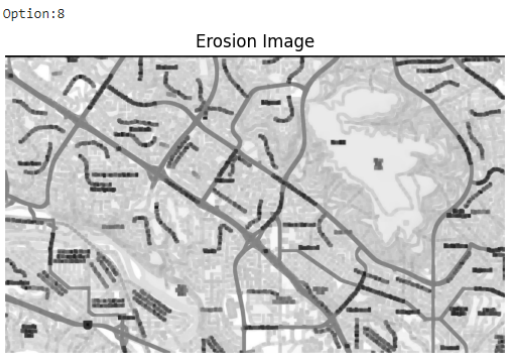


Fig. 9. Erosion

Fig. 10. Dilation

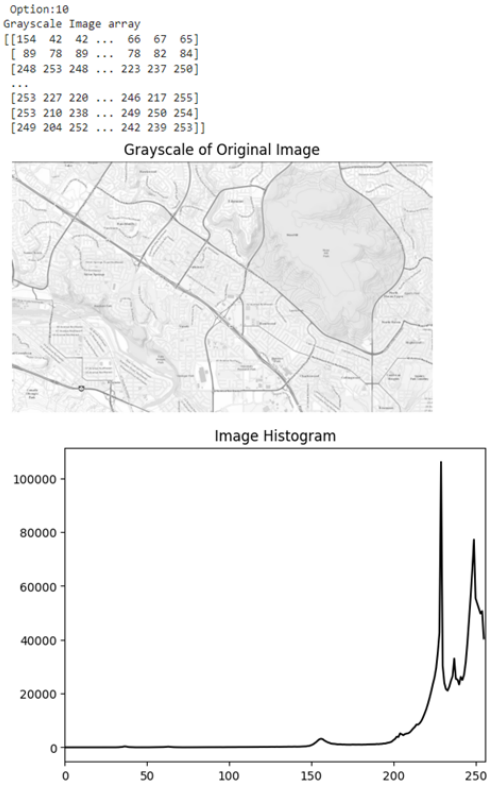


Fig. 11. Histogram

```
Option:11
Enter Lower Threshold: 10
Enter Higher Threshold: 200
```

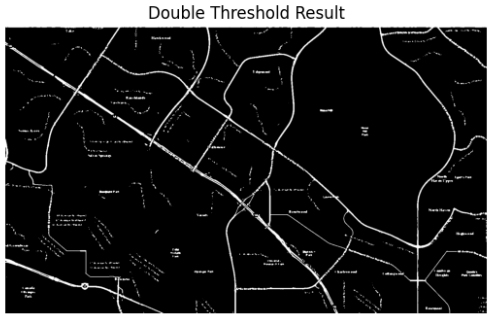


Fig. 12. Gray scaling of image

```
Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 1
Prewitt Mask
[[-1 -1 -1]
 [ 0  0  0]
 [ 1  1  1]]
After applying Prewitt Mask
[[ 94 238 139 ... 211 210 203]
 [113   1 152 ...  21   0 228]
 [ 12  23  16 ...  51  77  93]
 ...
 [ 30  12   8 ... 149 123 156]
 [ 31 251 246 ... 235 240 238]
 [ 14 252 247 ...   9  19  13]]
```

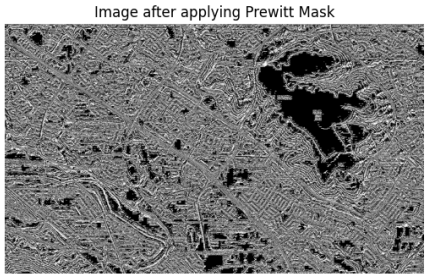


Fig. 13. Gray scaling of image

```
Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 2
Prewitt Mask
[[-1  0  1]
 [-1  0  1]
 [-1  0  1]]
After applying Prewitt Mask
[[235 224 240 ... 247 252   2]
 [118 112 252 ... 241 224 243]
 [ 12  11 252 ...   9 233 244]
 ...
 [ 73  81 193 ...  84 204 149]
 [114  45 143 ...  17 231 200]
 [133   9 119 ... 254 229 224]]
```

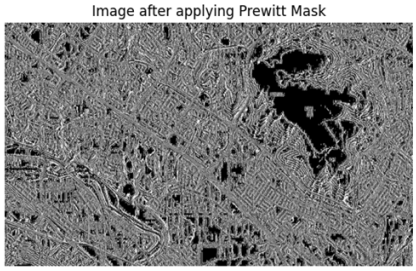


Fig. 14. Gray scaling of image

```
Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 3
Prewitt Mask
[[-1 -1  0]
 [-1  0  1]
 [ 0  1  1]]
After applying Prewitt Mask
[[253 141 163 ... 221 224 222]
 [ 74 207  97 ... 180 152 142]
 [183 188 188 ... 204 198 221]
 ...
 [ 63  81 206 ... 234 122 127]
 [ 96  39 158 ... 249 222 218]
 [ 98   4 152 ...  10 252 240]]
```

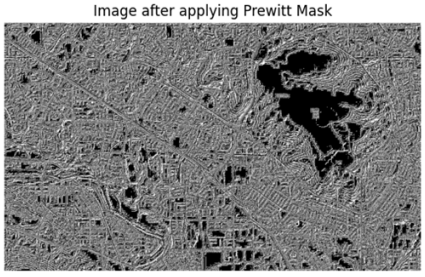


Fig. 15. Gray scaling of image

```
Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 4
Prewitt Mask
[[ 0  1  1]
 [-1  0  1]
 [-1 -1  0]]
After applying Prewitt Mask
[[ 94 195  71 ...  22  29  38]
 [ 55 161 151 ...  60  66  99]
 [ 80  79  70 ...  51   8  10]
 ...
 [ 36  33 219 ... 131  73 240]
 [ 61  21 200 ...  10   4 234]
 [ 80   2 175 ... 250 222 226]]
```

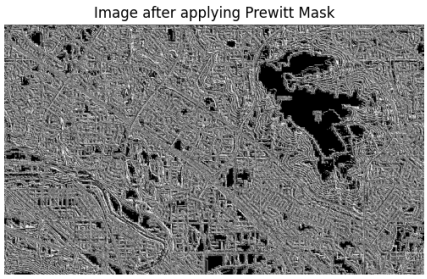


Fig. 16. Gray scaling of image

Feature Extraction - Feature Extraction uses an object-based approach to classify imagery, in the main image an object (also called segment) is a group of pixels with similar spectral, spatial, and/or texture attributes. Traditional classification methods are pixel-based, meaning that spectral information in each pixel is used to classify imagery.

Image Restoration - Image restoration is performed by reversing the process that blurred the image and such is performed by imaging a point source and use the point source image, which is called the Point Spread Function (PSF) to restore the image information lost to the blurring process.

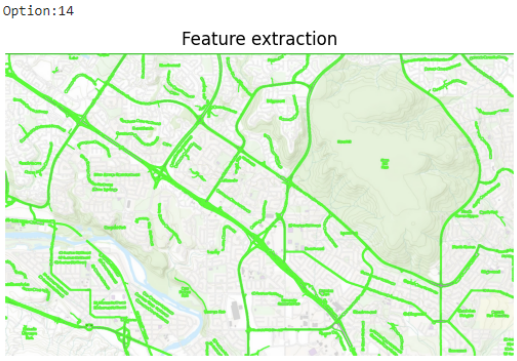


Fig. 17. Feature Extraction

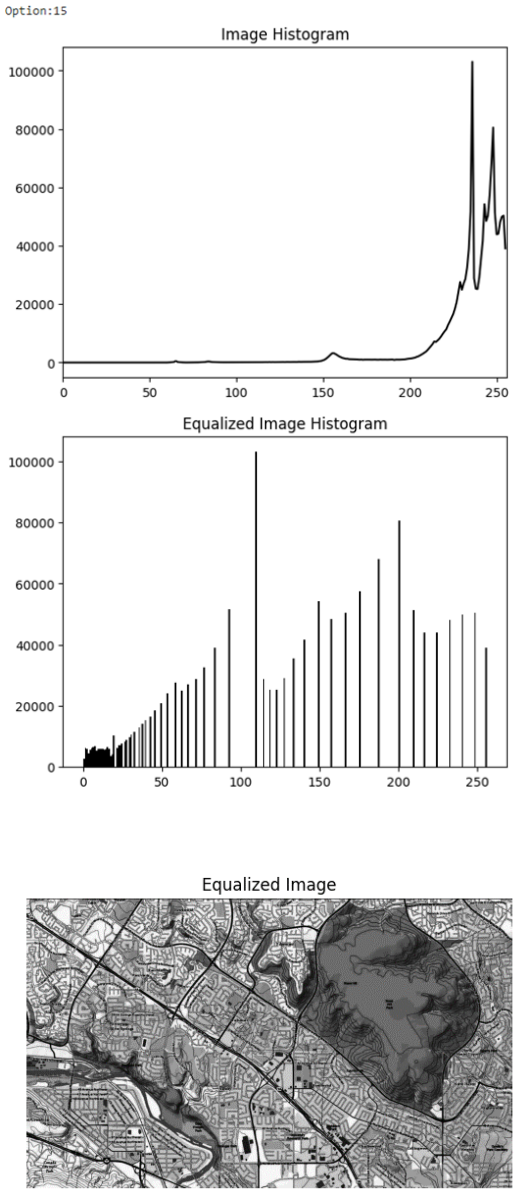


Fig. 18. Image Restoration

V. Discussion.

In recent years, Image processing has had a significant impact to revolutionize GIS images to extract the insights and interact with spatial data. With the vast expansion of image processing, it has opened new frontiers from remotely sensed data. This project is a pioneer because of its smooth integration to complex geospatial image processing and analysis requirements. With careful planning and the application of Python code, this project aims to produce a stable, easily navigable program that may serve as a multi-purpose image processing tool. With an eye toward being as user-friendly as possible, a variety of functions, including scaling, rotating, and flipping, are provided. Additionally, the range of offerings is extended by the inclusion of technical image processing techniques such as Edge Detection, Erosion, Dilation, Feature, and Text Extraction which are accessible through allowing the user to upload their choice of spatial image.

While using the functions to grayscale, the function converts the main image into a monochrome representation [Fig1] by providing each pixel a single intensity value. The main work behind the function is it calculates the weighted average of the three RGB channels to account human eye's sensitivity to each color.

The picture that a user will use is obviously 2 dimensional and Gaussian blur uses two mathematical functions (one for x-axis and one for the y-axis) to create a third function which is known as convolution. This third function creates a normal distribution of the pixel values, smoothing out some of the randomness.

If the user feels like the image is not in proper orientation, the Image Rotation function helps the user to get desire output. Image rotation is a common image processing routine with applications in matching, alignment, and other image-based algorithms. The input to an image rotation routine is an image, the rotation angle , and a point about which rotation is done.

The sharpening function takes the main image and subtracts a fraction of neighboring pixels from each pixel. This is a valuable creative tool for emphasizing texture. In this process it increases the contrast between adjacent pixels to enhance the details in an image. Sharpening an image in OpenCV can be done using a simple convolution operation. A convolution operation involves taking a small region of the image and multiplying each pixel value by a corresponding weight in a filter matrix. The filter matrix is used to modify the pixel values in the image, and the result of the convolution operation is a new image with enhanced details. To get the proper aspect ratio, the user provides the desired value for the axis and the function provides the output.

Edge detection uses linear filtering with a Gaussian kernel to smooth noise and then computes the edge strength and direction for each pixel in the smoothed image. The Canny filter is a multi-stage edge detector. It uses a filter based on the derivative of a Gaussian to compute the intensity of the gradients. Simple thresholding, which is basically binary thresholding, is a technique that sets a threshold value and

compares each pixel to that threshold value. If the pixel value is less than or equal to the assigned threshold, then the pixel value is set to zero or to the maximum value. In binary thresholding, if the value of the pixel is less than the threshold, it will be given a 0 value, i.e., black. If it is greater than the threshold, it will be assigned 255, i.e., white.

If the user uses the image for erosion, it shrinks the boundaries of an objective in the image. This is done by convolving the image with a structuring element, which determines the size and shape of the erosion. The output of the erosion operation is a new image where the pixels in the original image are eroded or shrunk. For dilation, the image is replaced by the kernel anchor at the center. This causes the white regions in the image to grow bigger in size and the image size increases as well. The dilate function returns an image with increased size of white shade in the given image. The histogram is considered as a graph or plot, which gives the user an overall idea about the intensity distribution of an image. It is a plot with pixel values (ranging from 0 to 255, not always) in X-axis and corresponding number of pixels in the image on Y-axis. It is just another way of understanding the image.

The double thresholding function is to use the output from the higher threshold to select all the touching regions in the output from the lower threshold. Morphological reconstruction can do exactly that. In the section of Prewitt filter it is seen that it can detect two types of edges Vertical and Horizontal. By choosing the filter type the user can get different sets of data to work with. Edges are calculated by using the difference between corresponding pixel intensities of an image.

If the user wants to extract text from an image, they can use Text Recognition function where the process involves ranking the characters that are to be extracted. It primarily involves two algorithms i.e. Matrix Matching and Feature Extraction. In matrix matching, the image is compared to an already existing glyph of characters. This matching is done based on pixel-by-pixel comparison. Text extraction can be achieved by applying text detection that identifies image parts containing text, text localization finds the exact position of the text, text segmentation separates the text from its background and binarization process converts the colored images into binary.

In image restoration it performs the process in a reversed way, where it blurs the image and such is performed by imaging a point source and use the point source image, which is called the Point Spread Function (PSF) to restore the image information lost to the blurring process. Feature extraction is a great tool to reduce the amount of redundant data from the uploaded image. In [Fig 14] the function took all the main roads and edges and to reduce the machine effort didn't detect the more complicated small roads.

This is achieved while keeping the application's user friendliness at the forefront by providing the image processing tools from the vast array of options present in the application.

VI. Conclusion.

In Conclusion, this project creates a robust, user-friendly application using which the user can select multiple image processing application tools that can be used by the user in the field of GIS to get the desired image results at the same time this offers the user a range of functions for such as scaling, rotating, and flipping, while also incorporating features such as picture sharpening, blurring, and noise reduction through the use of filters at the same time integrating technical image processing techniques such as Edge Detection, Erosion, Dilation, Feature, and Text Extraction. This is achieved while keeping the application's user friendliness at the forefront, allowing the user to easily upload their choice of image using Google Drive and selecting the image processing tools from the vast array of options present in the application.

References

[1] M. N. Abedin et al., "2011 Index IEEE Transactions on Geoscience and Remote Sensing Vol. 49," IEEE Transactions on Geoscience and Remote Sensing, vol. 49, no. 12, pp. 5167, 2011.

[2] G. R. Kuhn, M. M. Oliveira, and L. A. F. Fernandes, "An improved contrast enhancing approach for color-to-grayscale mappings," The Visual Computer, vol. 24, pp. 505-514, 2008.

[3] A. Jalobeanu, L. Blanc-Féraud, and J. Zerubia, "An adaptive Gaussian model for satellite image deblurring," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 613-621, 2004.

[4] P. Sharma and P. Abrol, "Color based image segmentation using adaptive thresholding," Int. J. Sci. Tech. Adv, vol. 2, no. 3, pp. 151-156, 2016.

[5] K. A. M. Said and A. B. Jambek, "Analysis of Image Processing Using Morphological Erosion and Dilation," Journal of Physics: Conference Series, vol. 2071, no. 1, IOP Publishing, 2021.

[6] O. Patel, Y. P. S. Maravi, and S. Sharma, "A comparative study of histogram equalization based image enhancement techniques for brightness preservation and contrast enhancement," arXiv preprint arXiv:1311.4033, 2013.

[7] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," Artificial Intelligence Review, vol. 54, pp. 1937-1967, 2021.

[8] B. Rasti et al., "Image restoration for remote sensing: Overview and toolbox," IEEE Geoscience and Remote Sensing Magazine, vol. 10, no. 2, pp. 201-230, 2021.

Appendix

A Versatile Image Pre-processing Application for Geographic Information Systems
Python Code

Group Members:

Name: Spencer Lobo **UCID:** 30228989

Name: Anan Ghosh **UCID:** 30220833

Name: Debistuti Kundu **UCID:** 30227696

Name: Himadri Das **UCID:** 30214525

Name: Ariful Islam Anik **UCID:** 30230898

0.1 Prerequisite

Running the below code will mount the google drive to the collab notebook. This will ensure the that the Images can be selected and saved to the desired Google Drive Location.

[1]:
```
from google.colab import drive      # Mount Google Drive to Colab
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

[2]:
```
!pip install Pillow                 # Install Pillow library
```

```
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages
(9.4.0)
```

[3]:
```
!sudo apt install tesseract-ocr     # Install tesseract-ocr library
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  tesseract-ocr-eng tesseract-ocr-osd
The following NEW packages will be installed:
  tesseract-ocr tesseract-ocr-eng tesseract-ocr-osd
0 upgraded, 3 newly installed, 0 to remove and 15 not upgraded.
Need to get 4,816 kB of archives.
After this operation, 15.6 MB of additional disk space will be used.
```

```
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-ocr-eng
all 1:4.00~git30-7274cfa-1.1 [1,591 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-ocr-osd
all 1:4.00~git30-7274cfa-1.1 [2,990 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-ocr amd64
4.1.1-2.1build1 [236 kB]
Fetched 4,816 kB in 0s (11.0 MB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78,
<> line 3.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package tesseract-ocr-eng.
(Reading database ... 120899 files and directories currently installed.)
Preparing to unpack .../tesseract-ocr-eng_1%3a4.00~git30-7274cfa-1.1_all.deb ...
Unpacking tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr-osd.
Preparing to unpack .../tesseract-ocr-osd_1%3a4.00~git30-7274cfa-1.1_all.deb ...
Unpacking tesseract-ocr-osd (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr.
Preparing to unpack .../tesseract-ocr_4.1.1-2.1build1_amd64.deb ...
Unpacking tesseract-ocr (4.1.1-2.1build1) ...
Setting up tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Setting up tesseract-ocr-osd (1:4.00~git30-7274cfa-1.1) ...
Setting up tesseract-ocr (4.1.1-2.1build1) ...
Processing triggers for man-db (2.10.2-1) ...
```

[4]:
```
!pip install pytesseract            # Install pytesseract library
```

```
Collecting pytesseract
  Downloading pytesseract-0.3.10-py3-none-any.whl (14 kB)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from pytesseract) (23.2)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.10/dist-
packages (from pytesseract) (9.4.0)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.10
```

1 Importing Library

[5]:
```
from google.colab import drive                          # Connect Google Drive
from matplotlib import pyplot as plt                    # Used to Plot Graphs
import numpy as np                                      # While working on image array
from PIL import Image                                   # Used to read as well as work␣
 ↪on an image
from google.colab.patches import cv2_imshow             # Importing CV2 library in␣
 ↪google collab it also patches any errors caused by cv2 while operating on␣
 ↪google collab
import cv2                                              # Importing CV2 library in␣
 ↪google collab
import time                                             # Set delays (counter)
import pytesseract                                      # Used for text extraction
from scipy import misc,ndimage                          # Importing Scipy library also␣
 ↪while applying sharpening of the image
```

2 Main Code Block

2.1 Importing Image from Google Drive

[7]:
```
drive.mount('/content/drive')                                   # Mounting Google Drive
image_path = input(str("Enter the path to the image: "))        # Input the image path␣
 ↪of the Google Drive
image = Image.open(image_path)                                  # Read the image using␣
 ↪PIL
image = np.array(image)                                         # Convert the PIL␣
 ↪image to a numpy array
img = Image.open(image_path)                                    # Create a copy of the␣
 ↪image with img
image = img
org_image = image
image = np.array(img)                                           # Create an array of␣
 ↪the image
plt.imshow(img)                                                 # Display the Image
plt.title("Original Colour Image")                              # Add Title
plt.axis('off')                                                 # Remove the axis
plt.show()                                                      # Display
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
Enter the path to the image: /content/drive/MyDrive/Colab Notebooks/Nose
Hill.png
```

2.2 Basic Image Processing

2.2.1 GrayScale of Image

[8]:
```
def GrayScale(image):                                  # Function to Convert image to␣
↪grayscale
 image = img.convert('L')                              # Convert code to 'L' grayscale
 global ar                                             # Make variable ar global
 ar = np.array(image)                                  # Assign the grayscale image array␣
↪to variable ar
 print("Grayscale Image array")                        # Displaying text
 print(ar)                                             # Display the array
 plt.title("Grayscale of Original Image")              # Display the title
 plt.imshow(image)                                     # Display the image
 plt.gray()                                            # Display only the grayscale
 plt.axis('off')                                       # Removing the axis
 plt.show()                                            # Display the processed image

 return image, ar                                      # Return Image and Array
```

2.2.2 Gaussian Blur

[9]:
```
def GaussianBlur(image, ar):                           # Function to Apply Gaussian Blur␣
↪on the image
 image = cv2.GaussianBlur(ar, (5, 5), 0)               # Apply Gaussian Blur with a 5x5␣
↪kernel
```

```
  plt.imshow(image, cmap='gray')              # Display Image with cmap='gray'
  plt.title('Gaussian Blurred Image')         # Add Title to Image
  plt.axis('off')                             # Turn Off the axis
  plt.show()                                  # Display Image

  return image                                # Return Image
```

2.2.3 Image Sharpening

[10]:
```
def Sharpen_Image(image):                     # Function to Sharpen Image
  image = misc.face(gray=True).astype(float)
  blur = ndimage.gaussian_filter(img, 5)      # Applying Gaussian filter
  plt.imshow(blur)                            # Plot Image
  plt.axis('off')                             # Turn Off the axis
  plt.show()                                  # Display the Image
  blur_G = ndimage.gaussian_filter(blur, 1)
  alpha = int(input('Enter value of alpha (sharpening value): ')) # User Input␣
↪for the alpha (sharpening value)
  sharp = blur+alpha*(blur-blur_G)            # Blur + the Alpha multiplied␣
↪with the blur_G
  plt.imshow(sharp)                           # Plot Image
  plt.axis('off')                             # Turn Off the axis
  plt.show()                                  # Display Image
```

2.2.4 Image Rotation

[11]:
```
def RotateImage(image):                       # Function to Rotate Image
  r = int(input('Enter the rotation degree: ')) # User Input the rotation degree
  rows, cols, _ = image.shape                                    # Extracts␣
↪the height and width of original image
  rotation_matrix = cv2.getRotationMatrix2D((cols/2, rows/2), r, 1)  # 2D␣
↪rotation of 45 degrees
  image = cv2.warpAffine(image, rotation_matrix, (cols, rows))       # Apply the␣
↪45 degree rotation
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)# Convert BGR format to RGB
  plt.imshow(image)                           # Plot Image
  plt.title('(Rotated Image')                 # Add Title
  plt.axis('off')                             # Removing the axis
  plt.gray()                                  # Display only the grayscale
  plt.show()                                  # Display Image
  return image                                # Return Image
```

2.2.5 Image Resizing

[12]:
```
def ResizedImage(image):                                # Function to Resize the image
  x = int(input('Enter Scaling for x-axis: '))          # User Input the X - axis␣
↪scaling factor
  y = int(input('Enter Scaling for y-axis: '))          # User Input the Y - axis␣
↪scaling factor
  image = cv2.resize(image, (0, 0), fx=x, fy=y)         # Resize using cv2.resize() the␣
↪image to half (0.5) both height and width
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert BGR format to RGB
  plt.imshow(image)                                     # Plot Image
  plt.title('Resized Image')                            # Add Title
  plt.axis('off')                                       # Turn off the axis label and␣
↪title
  plt.gray()                                            # Display only the grayscale
  plt.show()                                            # Display Image
  return image                                          # Return Image
```

2.2.6 Canny Edge Detection

[13]:
```
def CannyEdge(image):                                   # Function to detect edges using␣
↪canny edge detection
  GrayScale(image)                                      # Run the GrayScale(image)␣
↪function block
  x = int(input('Input lower threshold For Canny Edge Detection: '))    # User␣
↪Input lower threshold
  y = int(input('Input Upper threshold For Canny Edge Detection: '))    # User␣
↪Input Upper threshold
  images = cv2.Canny(image, x, y)                       # Appling Canny Edge detection␣
↪with lower threshold and upper threshold
  plt.imshow(images, cmap='gray')                       # Plot Image
  plt.title('Edges Image')                              # Add Title
  plt.axis('off')                                       # Removing the axis
  plt.show()                                            # Display Image
  return image                                          # Return Image
```

2.2.7 Binary Thresholding

[14]:
```
def BinaryThreshold(image):                             # Function to apply Binary␣
↪Thresholding on an image
  CannyEdge(image)                                      # Run the CannyEdge(image)␣
↪function block
  x = int(input('Input threshold For Binary Thresholding: '))  # User Input␣
↪threshold value For Binary Thresholding
  _, image = cv2.threshold(image, x, 255, cv2.THRESH_BINARY)    # Applying binary␣
↪thresholding on the grayscale image with threshold values
```

```
  plt.imshow(image, cmap='gray')                # Plot Image
  plt.title('Binary Thresholding')              # Add Title
  plt.axis('off')                               # Removing the axis
  plt.gray()                                    # Display only the grayscale
  plt.show()                                    # Display Image
  return image                                  # Return Image
```

2.3 Morphological Operations

2.3.1 Erosion

[15]:
```
def Erosion(image,ar):                          # Function to apply Ersion on image
  kernel = np.ones((5, 5), np.uint8)
  image = cv2.erode(ar, kernel, iterations=2)  # Apply Erosion on the image using␣
↪cv2
  plt.imshow(image, cmap='gray')                # Plot image as grayscale
  plt.title('Erosion Image')                    # Add Title
  plt.axis('off')                               # Removing the axis
  plt.gray()                                    # Display only the grayscale
  plt.show()                                    # Display Image
  return image                                  # Return Image
```

2.3.2 Dilation

[16]:
```
def Dilation(image, ar):                        # Function to apply Dilation on␣
↪image
  kernel = np.ones((5, 5), np.uint8)
  image = cv2.dilate(ar, kernel, iterations=2)# Apply Dilation on the image␣
↪using cv2
  plt.imshow(image, cmap='gray')                # Display image as gray
  plt.title('Dilation Image')                   # Add Title
  plt.axis('off')                               # Removing the axis
  plt.gray()                                    # Display only the grayscale
  plt.show()                                    # Display Image
  return image                                  # Return Image
```

2.4 Image Restoration

2.4.1 Histogram Analysis

[17]:
```
def Hist(image):                                # Function to get Histogram␣
↪Analysis
  GrayScale(image)                              # Run the GrayScale(image)␣
↪function block
  original_hist = cv2.calcHist([image], [0], None, [256], [0, 256])
  plt.plot(original_hist, color='black')        # plotting an histogram
  plt.title('Image Histogram')                  # Adds Title
  plt.xlim([0, 256])                            # x-axis limit
```

```
  plt.show()                                    # Display Image
```

2.4.2 Histogram Equalization

[18]:
```
def Image_restoration(image):                   # Function to apply Image␣
↪Restoration using histogram equalization
  image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)  # Read the image as␣
↪grayscale using cv2
  original_hist = cv2.calcHist([image], [0], None, [256], [0, 256])
  plt.plot(original_hist, color='black')        # Plot the original histogram image
  plt.title('Image Histogram')                  # Adds Title
  plt.xlim([0, 256])                            # x-axis limit
  plt.show()                                    # Display Image

  equalized_image = cv2.equalizeHist(image)     # Equalize the histogram

  plt.hist(equalized_image.ravel(), 256, [0, 256], color='black')
  plt.title('Equalized Image Histogram')        # Adds Title
  plt.show()                                    # Plot the image

  plt.imshow(equalized_image, cmap='gray')      # Display the equalized image as␣
↪grayscale
  plt.title('Equalized Image')                  # Adds Title
  plt.axis('off')                               # Removing the axis
  plt.show()                                    # Return Image
```

2.5 Advanceed Image Processing

2.5.1 Double Threshold

[19]:
```
def double_threshold(image): # Function for applying Double Threshold
  gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)      # Convert the image␣
↪to grayscale
  blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)   # Apply Gaussian␣
↪Blur to reduce noise
  low_threshold = int(input("Enter Lower Threshold: "))     # User Input lower␣
↪threshold value
  high_threshold = int(input("Enter Higher Threshold: "))   # User Input upper␣
↪threshold value
  _, low_threshold_mask = cv2.threshold(blurred_image, low_threshold, 255, cv2.
↪THRESH_BINARY)         # low Treshold mask
  _, high_threshold_mask = cv2.threshold(blurred_image, high_threshold, 255, cv2.
↪THRESH_BINARY_INV)   # high Treshold mask

  double_thresholded_image = cv2.bitwise_and(low_threshold_mask,␣
↪high_threshold_mask)   # Combine the two threshold masks
  plt.imshow(image, cmap='gray')                # Display image as grayscale
```

```
    plt.title('Double Threshold Result')          # Adds Title
    plt.gray()                                    # Display only the grayscale
    plt.axis('off')                               # Removing the axis
    plt.show()                                    # Display Image

    return image                                  # Return Image
```

2.5.2 Prewitt Filter

[20]:
```
def Prewitt(image, ar):                           # Function for applying Prewitt␣
↪filters
    ask = int(input("Enter the Type of Prewitt Filter: \n 1: Prewitt Horizontal, 2:
↪ Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: ")) # User input
    if ask == 1:
        pv = [[-1, -1, -1], [0, 0, 0], [1, 1, 1]]  # Prewitt Horizontal filter
    elif ask == 2:
        pv = [[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]]  # Prewitt Vertical filter
    elif ask == 3:
        pv = [[-1, -1, 0], [-1, 0, 1], [0, 1, 1]]  # Prewitt +45 filter
    elif ask == 4:
        pv = [[0, 1, 1], [-1, 0, 1], [-1, -1, 0]]  # Prewitt -45 filter
    else:
        print("Invalid operation")                 # Print Invalid Statement
        return None                                # Return none
    pv = np.array(pv)                              # Convert the prewitt filter to␣
↪array
    print("Prewitt Mask")                          # Print statement
    print(pv)                                      # Print the Prewitt filter
    ar_pv = ndimage.convolve(ar, pv)               # Convolve array with Prewitt␣
↪filter
    print("After applying Prewitt Mask")           # Print statement
    print(ar_pv)                                   # Print array
    image = Image.fromarray(ar_pv)                 # Image from array
    plt.imshow(image)                              # Displays the image
    plt.title("Image after applying Prewitt Mask") # Adds Title
    plt.axis('off')                                # Removing the axis
    plt.gray()                                     # Display only the grayscale
    plt.show()                                     # Display Image

    return image                                   # Return Image
```

2.6 Features and Text Extraction

2.6.1 Feature Extraction

[21]:
```
def feature_ext(image):
 ↪# Function for feature extraction
  gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
 ↪# Convert the image to grayscale
  blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)
 ↪# Apply Gaussian Blur to reduce noise and improve feature extraction
  edges = cv2.Canny(blurred_image, 50, 150)
 ↪# Use Canny edge detection to find edges in the image
  contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.
 ↪CHAIN_APPROX_SIMPLE)   # Find contours in the image
  gis_map_with_contours = image.copy()
 ↪# Draw the contours on the original image
  cv2.drawContours(gis_map_with_contours, contours, -1, (0, 255, 0), 2)
  plt.imshow(gis_map_with_contours)
 ↪# Plot Image
  plt.title('Feature extraction')
 ↪# Add Title
  plt.axis('off')
 ↪# Removing the axis
  plt.gray()
 ↪# Display only the grayscale
  plt.show()
 ↪# Display the Image
```

2.6.2 Text Extraction

[22]:
```
def text_ext(image):
 ↪# Function for text extraction
  pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'
 ↪# Configure pytesseract to use the tesseract-ocr executable
  gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
 ↪# Convert the image to grayscale
  extracted_text = pytesseract.image_to_string(Image.fromarray(gray_image))
 ↪# Perform OCR using pytesseract
  print("Extracted Text:\n", extracted_text)
 ↪# Print the extracted text
```

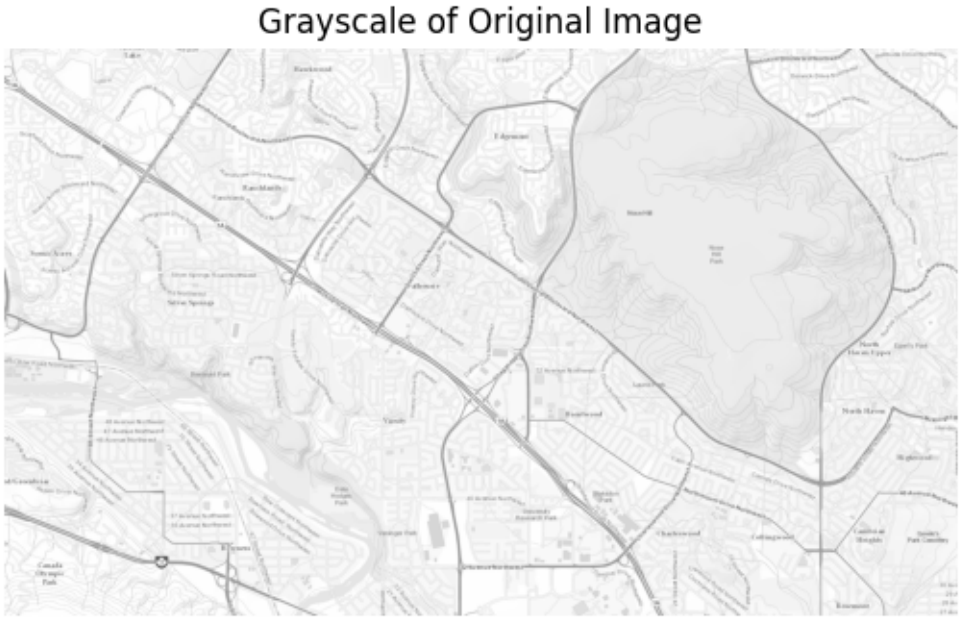
2.7 Executing Image Processing Tasks:

[23]:
```
import time
while True:
  time.sleep(1)       # Set a deplay for 1 sec
```

```
  operation = input( 'Enter Image Processing Option (1-15) \n     "1 : Grayscale␣
↪Image", "2 : Gaussian Blur on Image", "3 : Rotate Image", "4: Sharpen Image",␣
↪"5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold", "8 : Erosion", \n ␣
↪   "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt␣
↪Filter","13: Text Extraction","14: Feature Extraction", "15: Image␣
↪Restoration" \n Type "quit" to quit \n Option:')

  if operation == "1":
    GrayScale(image)
  elif operation == "2":
    GaussianBlur(image, ar)
  elif operation == "3":
    RotateImage(image)
  elif operation == "4":
    Sharpen_Image(image)
  elif operation == "5":
    ResizedImage(image)
  elif operation == "6":
    CannyEdge(image)
  elif operation == "7":
    BinaryThreshold(image)
  elif operation == "8":
    Erosion(image, ar)
  elif operation == "9":
    Dilation(image, ar)
  elif operation == "10":
    Hist(image)
  elif operation == "11":
    double_threshold(image)
  elif operation == "12":
    Prewitt(image, ar)
  elif operation == "13":
    text_ext(image)
  elif operation == "14":
    feature_ext(image)
  elif operation == "15":
    Image_restoration(image)
  elif operation == "quit":
    break
  else:
    print("Invalid operation")
    continue
# Add a delay of 10 second before the next iteration
time.sleep(1)
```

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
```

```
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:1
Grayscale Image array
[[153  44  44 ...  65  65  65]
 [ 65  65  65 ...  65  65  65]
 [253 255 246 ... 230 255 254]
 ...
 [253 229 214 ... 254 252 255]
 [254 203 240 ... 252 242 255]
 [247 200 254 ... 239 243 255]]
```



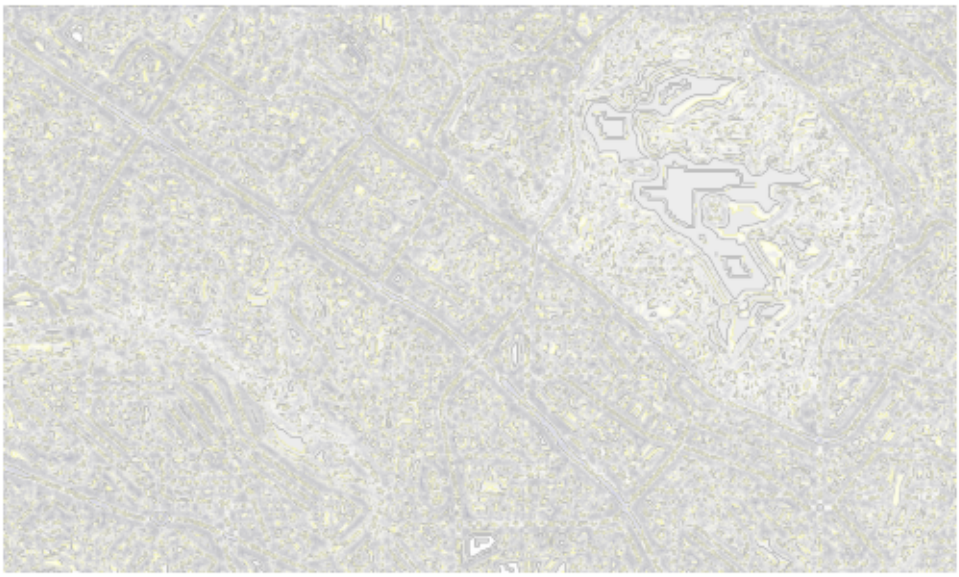

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:2
```


```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:3
Enter the rotation degree: 45
```

(Rotated Image

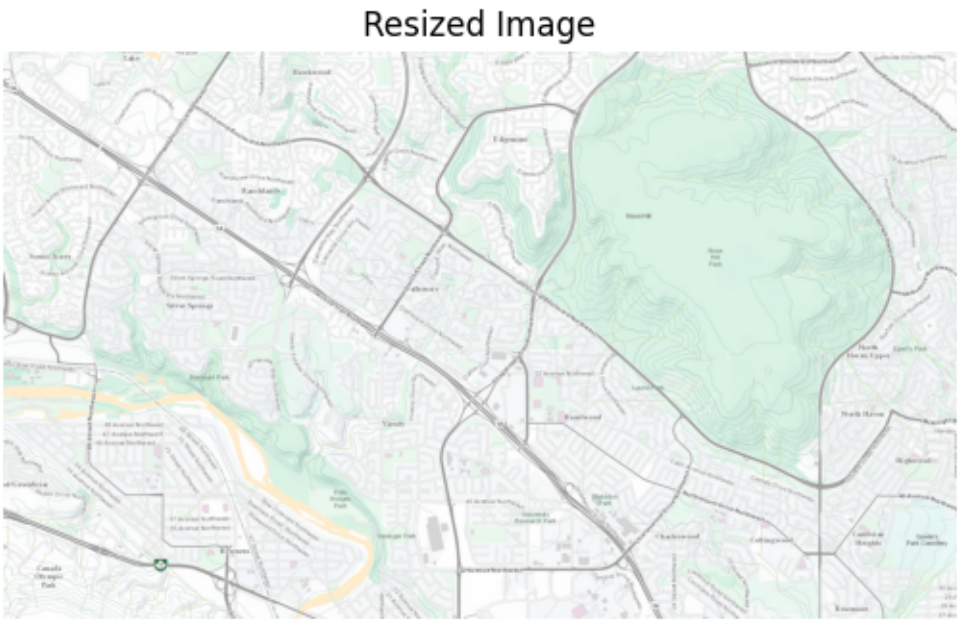
```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:4

<ipython-input-10-09c2c640a354>:2: DeprecationWarning: scipy.misc.face has been
deprecated in SciPy v1.10.0; and will be completely removed in SciPy v1.12.0.
Dataset methods have moved into the scipy.datasets module. Use
scipy.datasets.face instead.
  image = misc.face(gray=True).astype(float)
```

```
Enter value of alpha (sharpening value): 40
```



```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
```

```
 Type "quit" to quit
 Option:5
Enter Scaling for x-axis: 1
Enter Scaling for y-axis: 1
```




```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:6
Grayscale Image array
[[153  44  44 ...  65  65  65]
 [ 65  65  65 ...  65  65  65]
 [253 255 246 ... 230 255 254]
 ...
 [253 229 214 ... 254 252 255]
 [254 203 240 ... 252 242 255]
 [247 200 254 ... 239 243 255]]
```

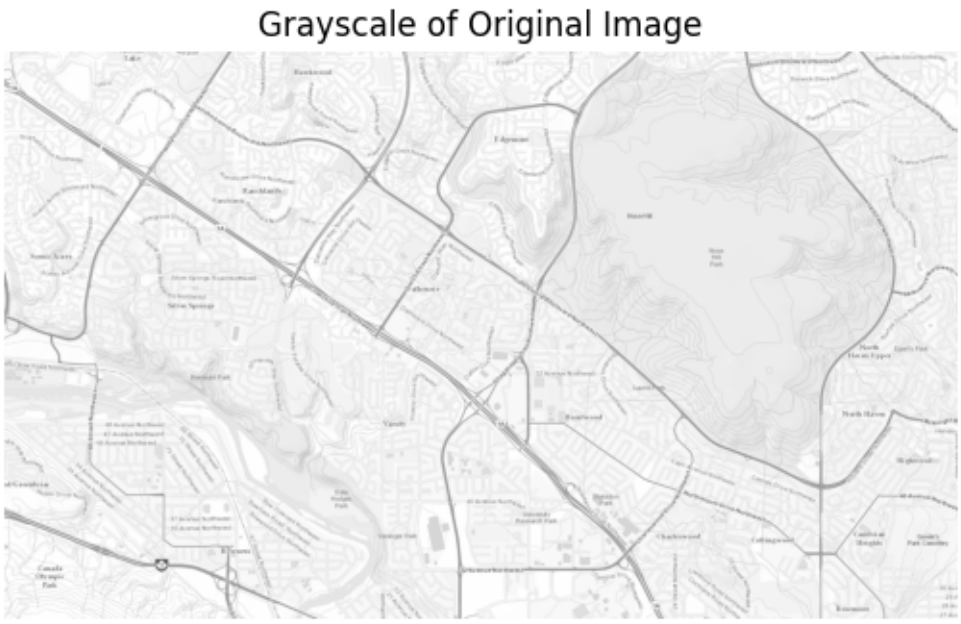
Grayscale of Original Image

```
Input lower threshold For Canny Edge Detection: 100
Input Upper threshold For Canny Edge Detection: 300
```

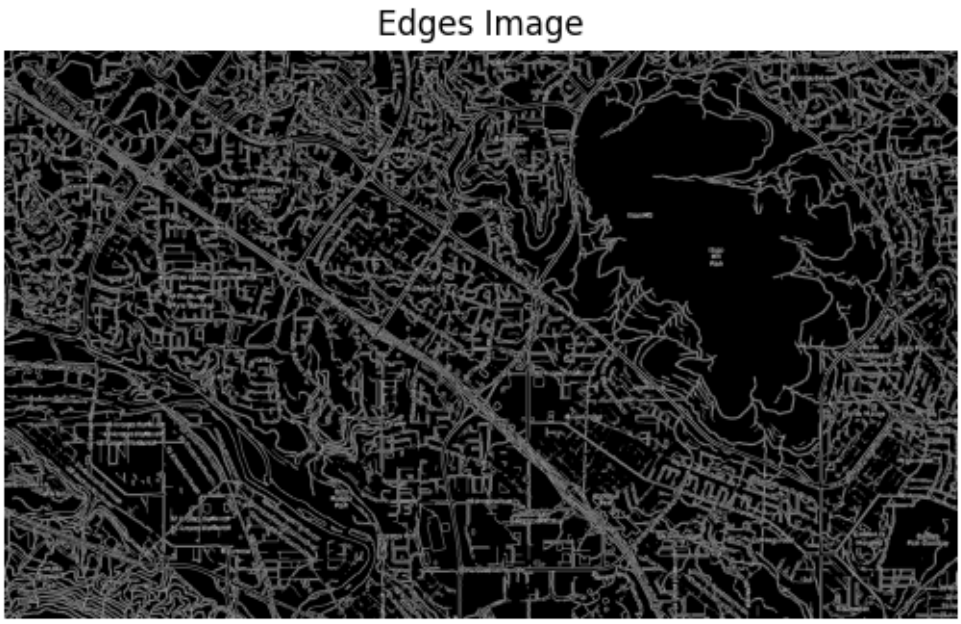
Edges Image

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
```

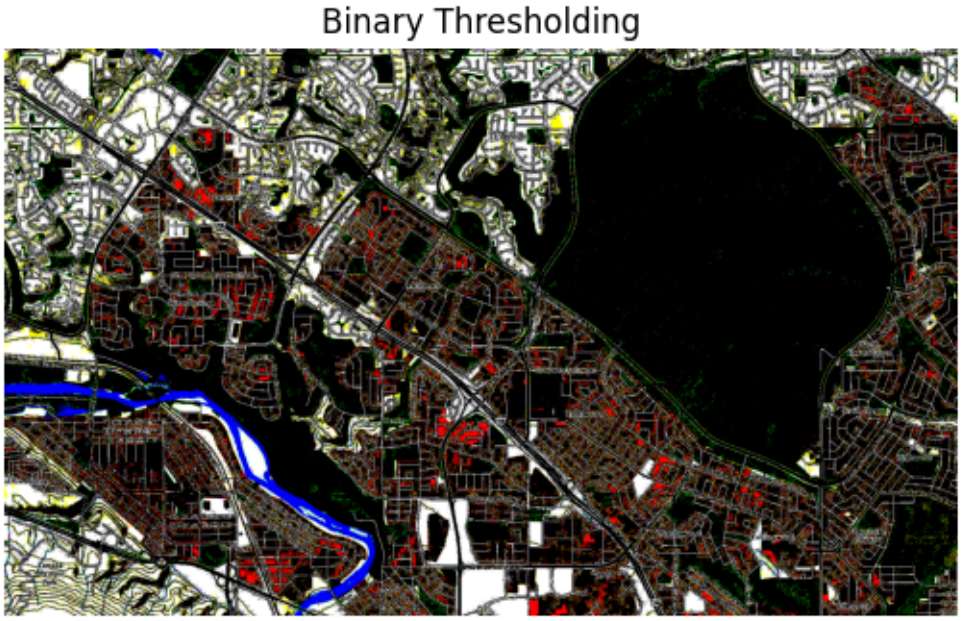
```
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:7
Grayscale Image array
[[153  44  44 ...  65  65  65]
 [ 65  65  65 ...  65  65  65]
 [253 255 246 ... 230 255 254]
 ...
 [253 229 214 ... 254 252 255]
 [254 203 240 ... 252 242 255]
 [247 200 254 ... 239 243 255]]
```




```
Input lower threshold For Canny Edge Detection: 100
Input Upper threshold For Canny Edge Detection: 200
```

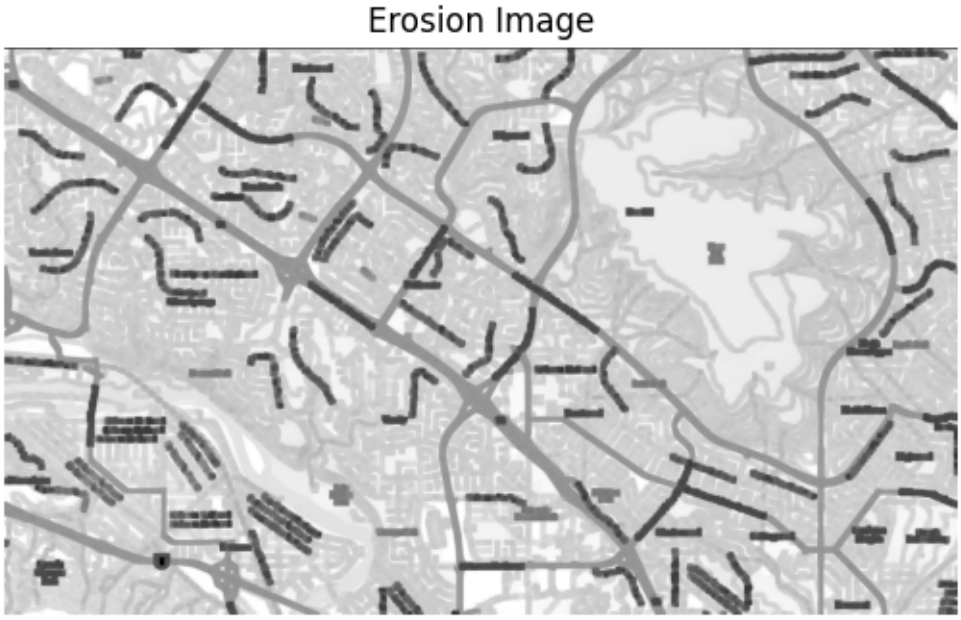



```
Input threshold For Binary Thresholding: 250
```



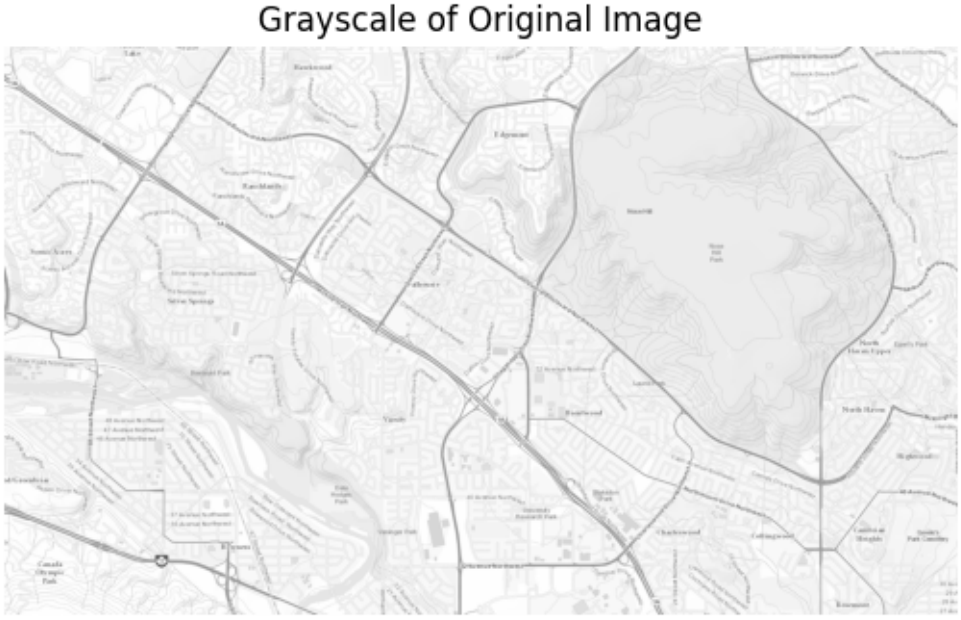

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
```

```
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:8
```




```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:9
```


```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:10
Grayscale Image array
[[153  44  44 ...  65  65  65]
 [ 65  65  65 ...  65  65  65]
 [253 255 246 ... 230 255 254]
 ...
 [253 229 214 ... 254 252 255]
 [254 203 240 ... 252 242 255]
 [247 200 254 ... 239 243 255]]
```


Grayscale of Original Image

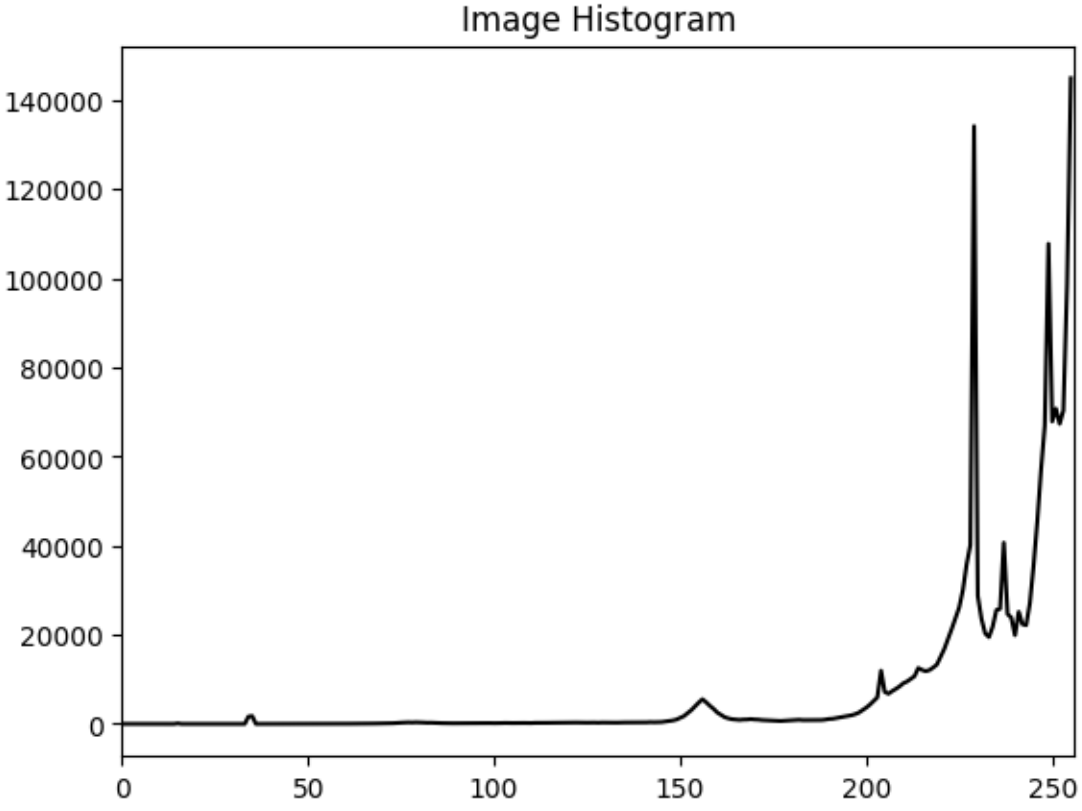
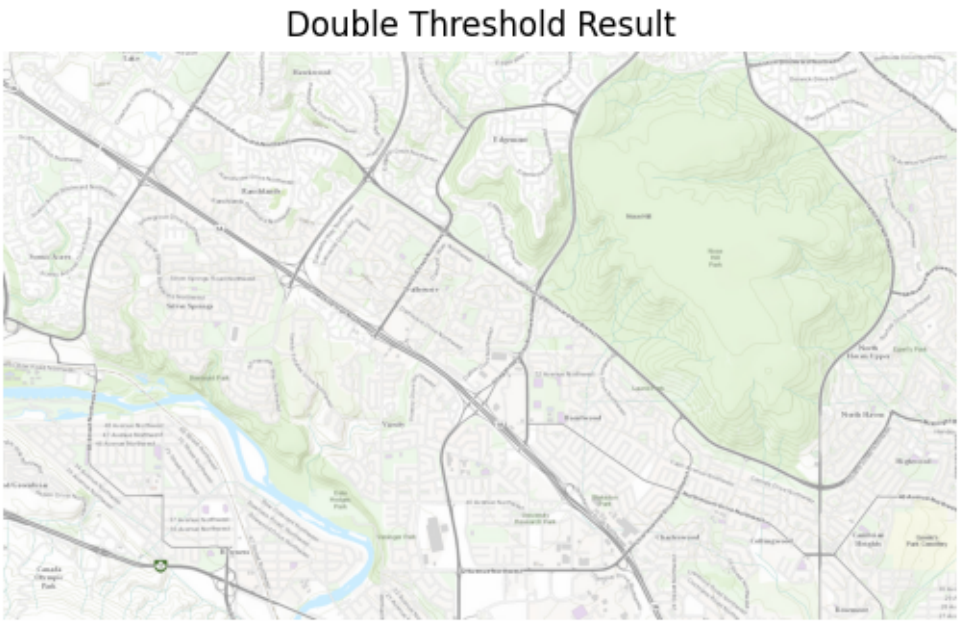


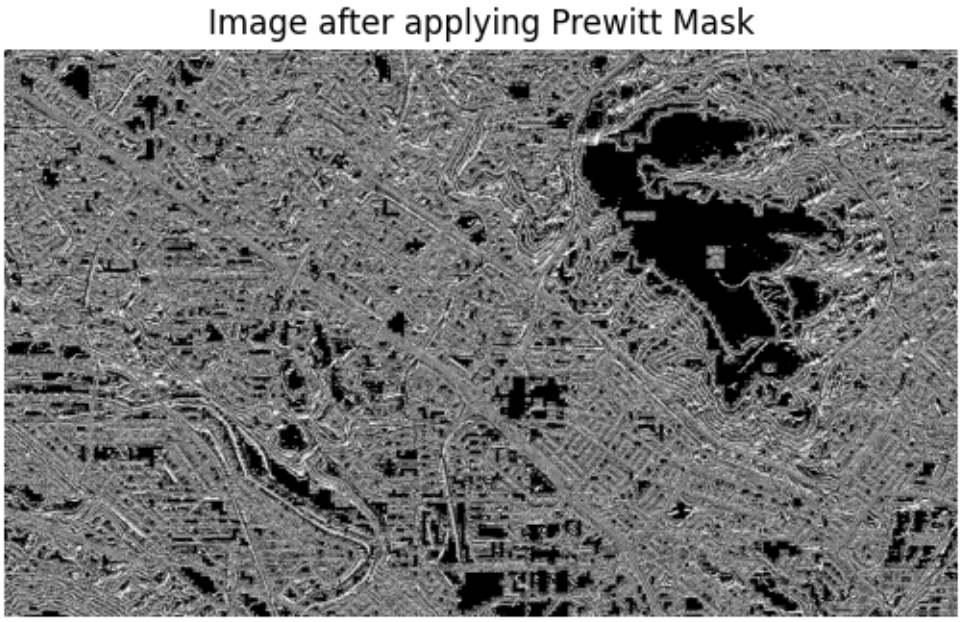
Image Histogram
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:11
Enter Lower Threshold: 30
Enter Higher Threshold: 50
```



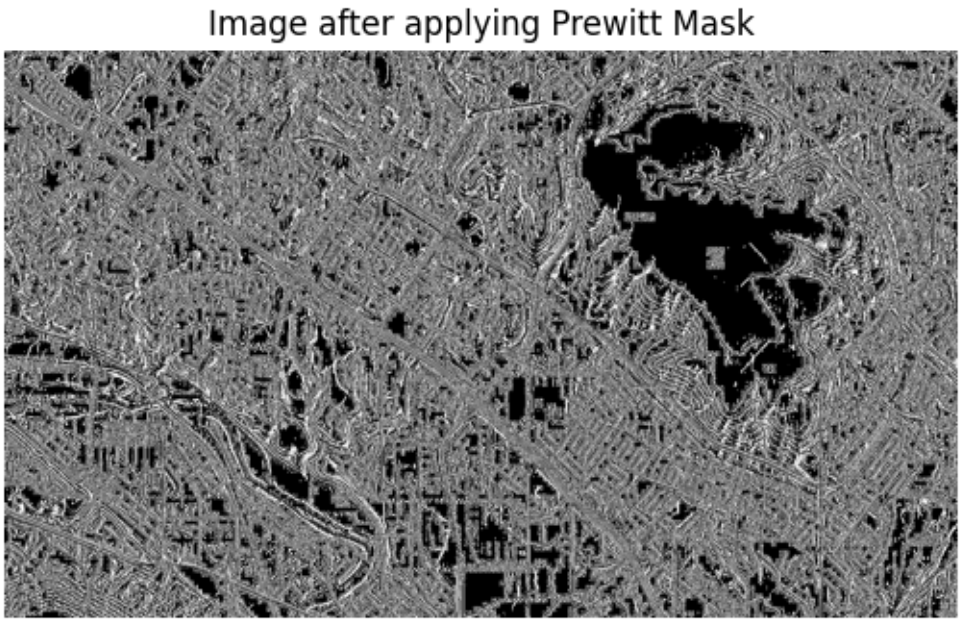

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 1
Prewitt Mask
[[-1 -1 -1]
 [ 0  0  0]
 [ 1  1  1]]
After applying Prewitt Mask
[[155  46 193 ...   0   0   0]
```

```
 [101 255 148 ...  16 224 200]
 [208 208 213 ...  34  52  40]
 ...
 [ 44   3   3 ... 151 133 153]
 [ 41 251 242 ...  23  24   9]
 [ 17 252 245 ...   0  12 255]]
```

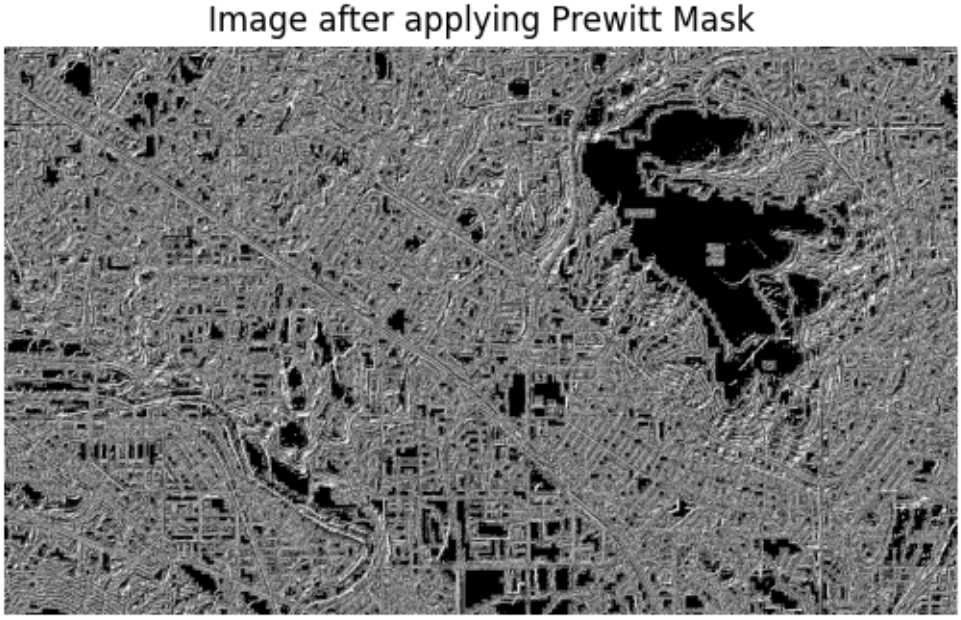



```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 2
Prewitt Mask
[[-1  0  1]
 [-1  0  1]
 [-1  0  1]]
After applying Prewitt Mask
[[218 218   0 ...   0   0   0]
 [107 116   4 ... 207 232   1]
 [  8   7 255 ... 254 220 228]
 ...
 [ 73 108 185 ...  98 229 124]
```

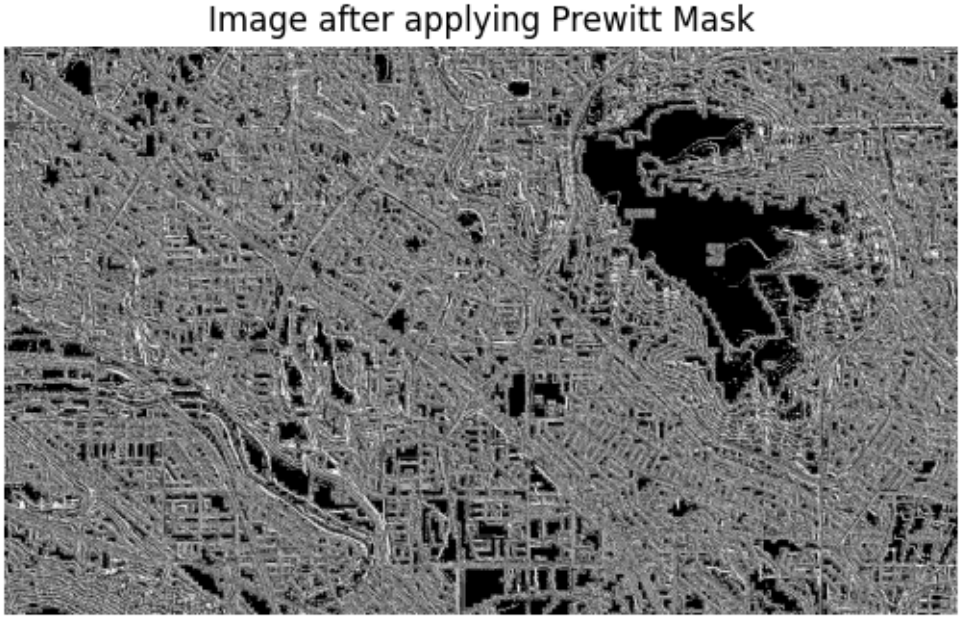
```
 [122  46 129 ... 244 236 228]
 [145   0  97 ... 254 221 219]]
```




```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 3
Prewitt Mask
[[-1 -1  0]
 [-1  0  1]
 [ 0  1  1]]
After applying Prewitt Mask
[[ 29 176 214 ...   0   0   0]
 [ 54 208 103 ... 157 133 134]
 [140 149 141 ... 170 183 179]
 ...
 [ 69 100 189 ... 239 129 137]
 [110  42 140 ...   9   5 240]
 [108 252 137 ...   7 236 231]]
```




```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:12
Enter the Type of Prewitt Filter:
 1: Prewitt Horizontal, 2: Prewitt Vertical, 3: Prewitt +45, 4: Prewitt -45.: 4
Prewitt Mask
[[ 0  1  1]
 [-1  0  1]
 [-1 -1  0]]
After applying Prewitt Mask
[[ 42 151  42 ...   0   0   0]
 [ 53 164 157 ...  50  99 123]
 [122 121 118 ...  35  13  50]
 ...
 [ 28  47 230 ... 108  99 240]
 [ 63  18 194 ... 227 228 231]
 [ 84 253 162 ... 250 225 232]]
```




```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:13
Extracted Text:
```

```
eave: 2
e z rmuda Drive North
eX 3 ere ce Orval eewer
= Hawkwood eat
2 Berwick Drive Now "Sem,
2 2
```

Sead y
& S
Ae, z
‘‘2, 5
i vey, ‘=
Bin. ‘‘ng =
rest 3,
*ranortnwett

Northwves

a ri
= ik cine OPV Nortnyy
quulevard Nori,

sf 2
Re

Ranchlands,
Ranchlands 5.
wa

syeraiove Dri,
3S

Silver Springs Road Northwest

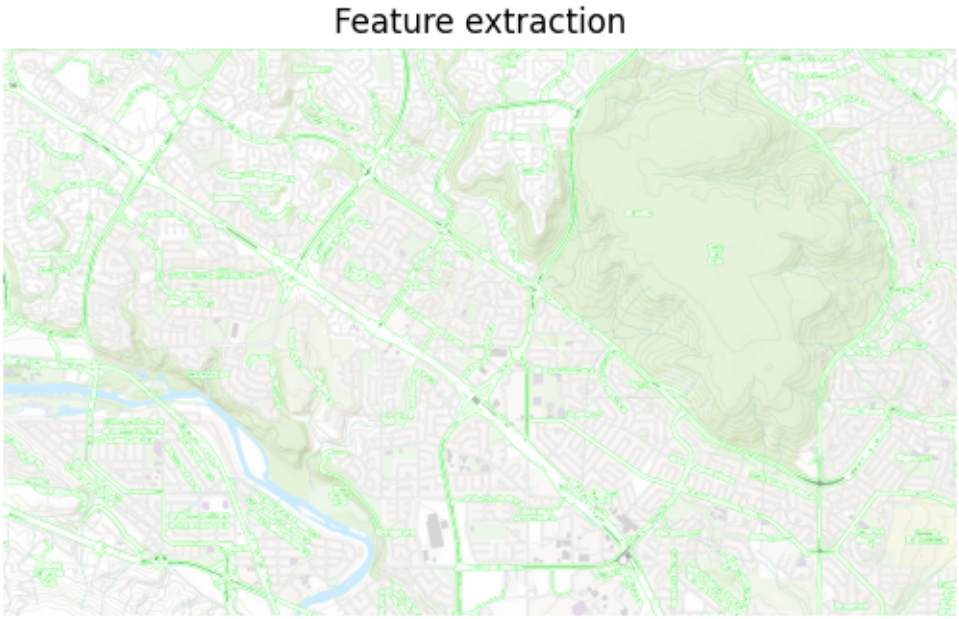
o

```
rd Northwest
Silver Springs

Eget's Park
'Haven Upper \
```

```
£
z
4p avenue Northwest MegavsB
§ 47 Avenue Nonthwest <5. Hendy
ay, $-46-Avenue Northwest 5
S B4eAvenueNorest yO
2 Yo \ BN
% 4
ort Uy % \ 2g")
se eH, <4, \
\diGreenbri KS 4 \ DN
»d/Greenbriar e, , .
```

```
Aspen nve %,
one %lakston Lassonclle
40 Avenue Nor @ eo
by Des, he.
%, Univers
a7 Avenue Northvest-\ ciliary
36 avenue Northwest \
Veranger Park cuieen's
Heights Park Cemetery
z 3
z %
Boy Ye
Ft ep %,
```

```
INR BS, Ne 30K
6% % 3. A, 2 292
: ~ fe * Vhs Rosemont, 28:AV
BAK 74 %, 2TAN

Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:14
```




```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:15
```

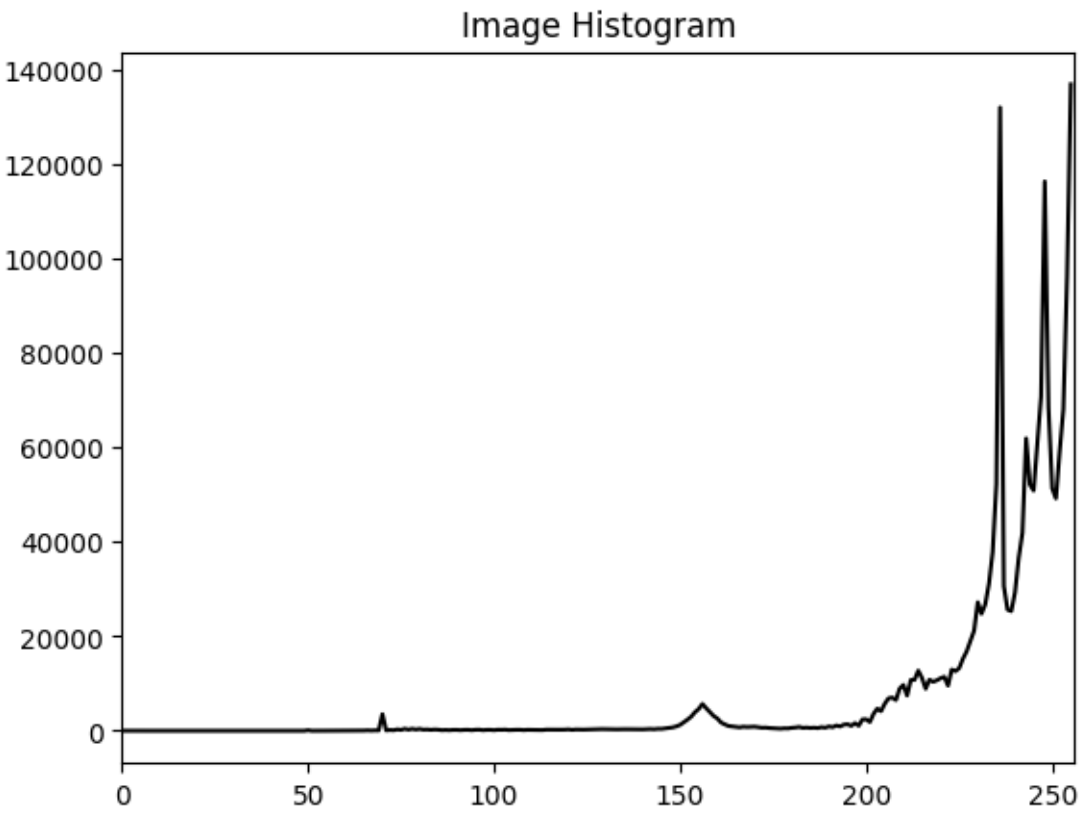
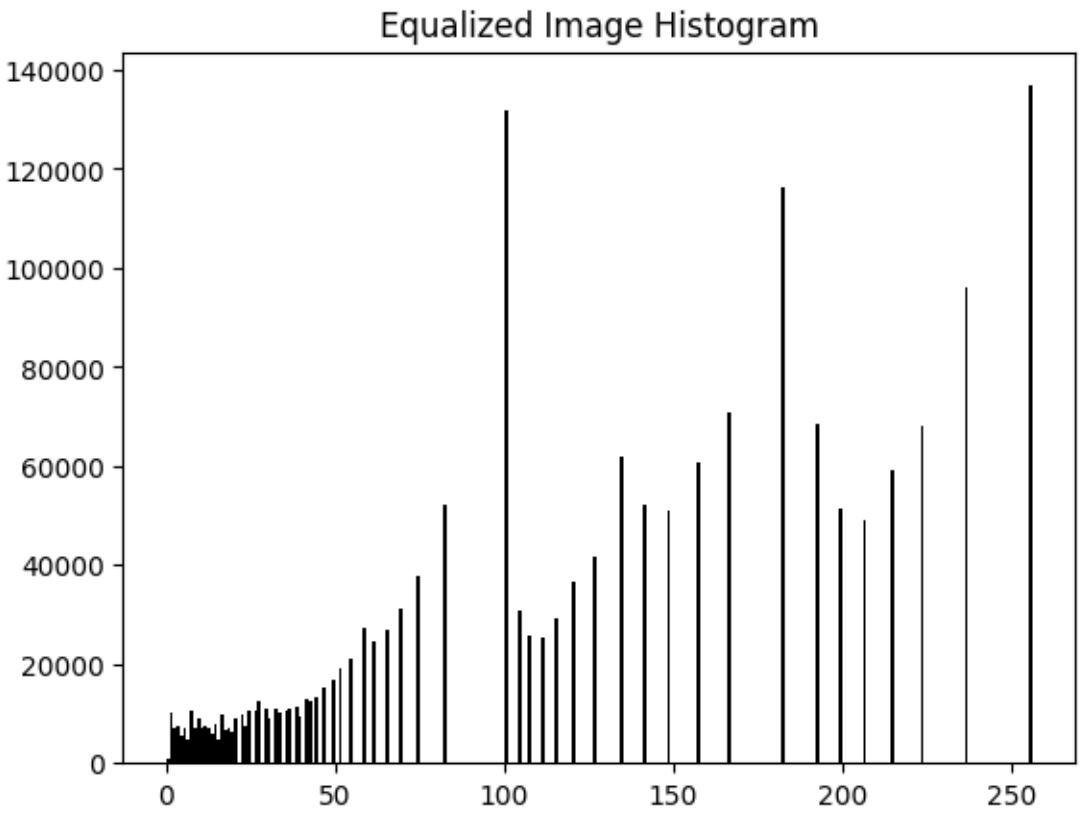
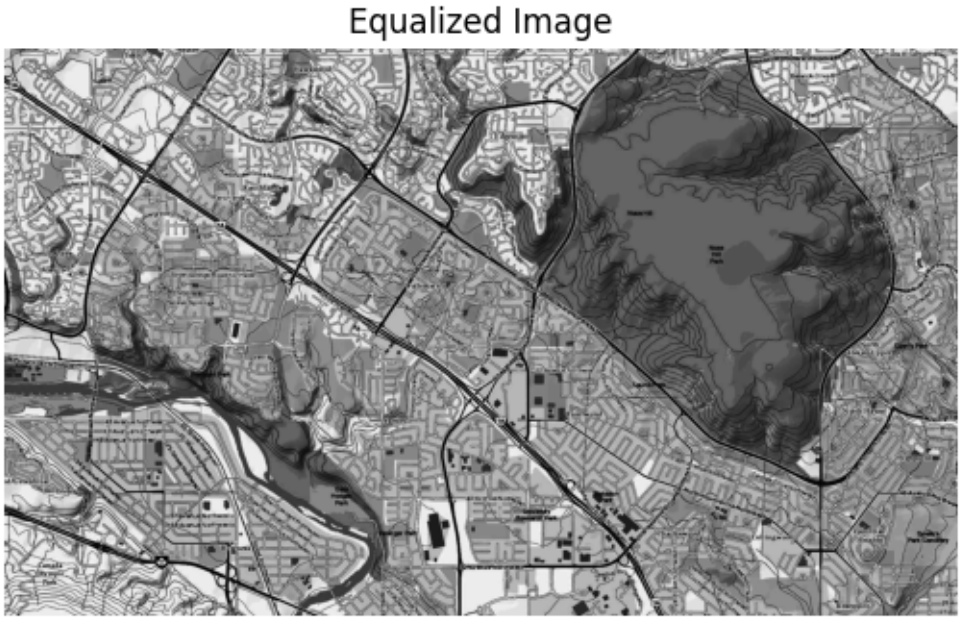
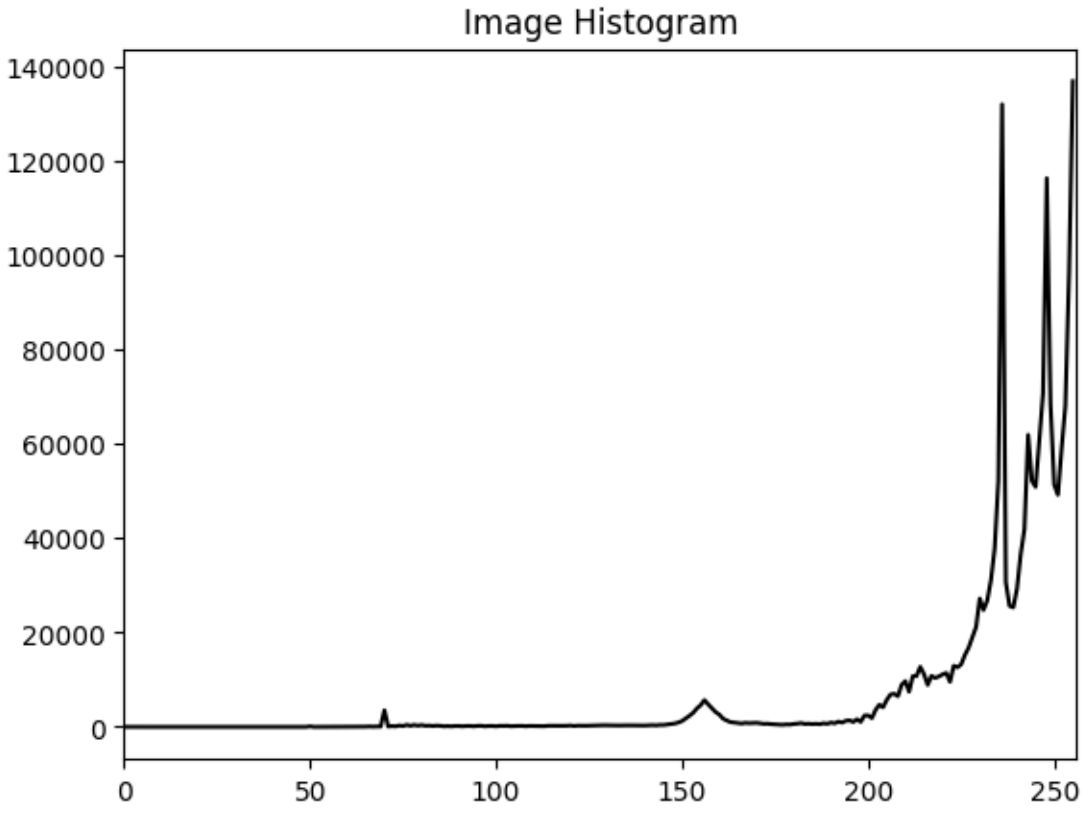


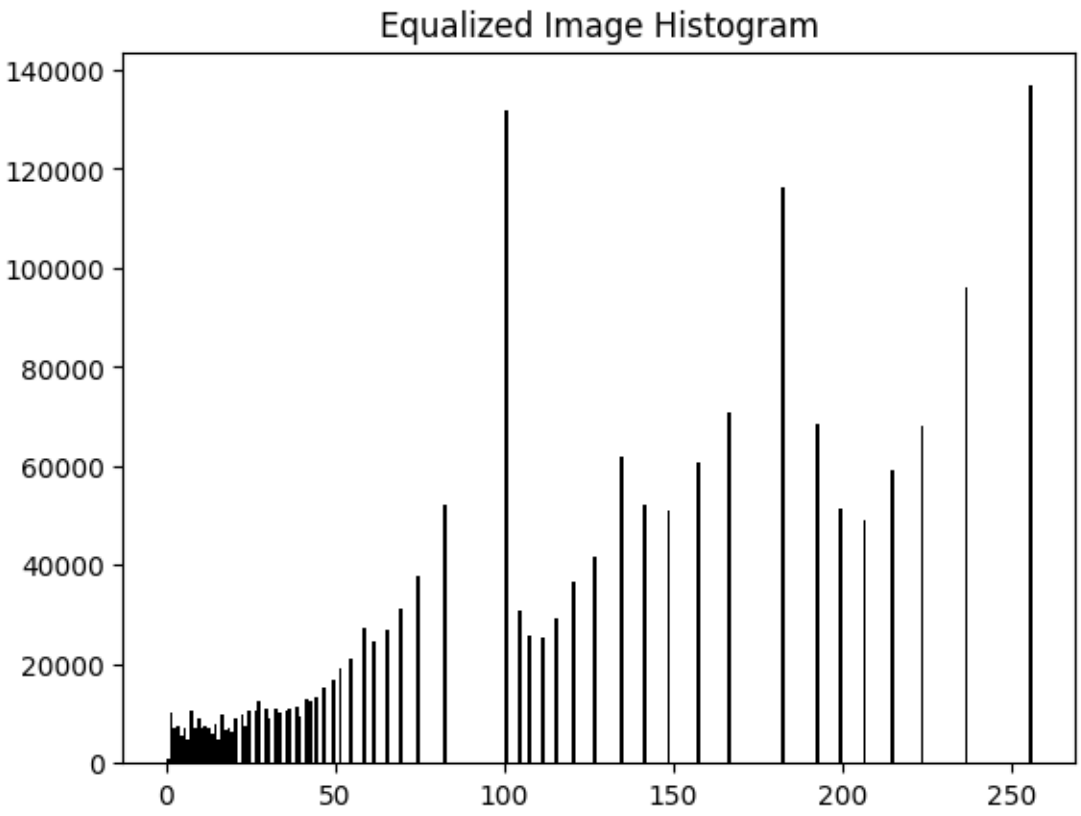
Image Histogram
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250

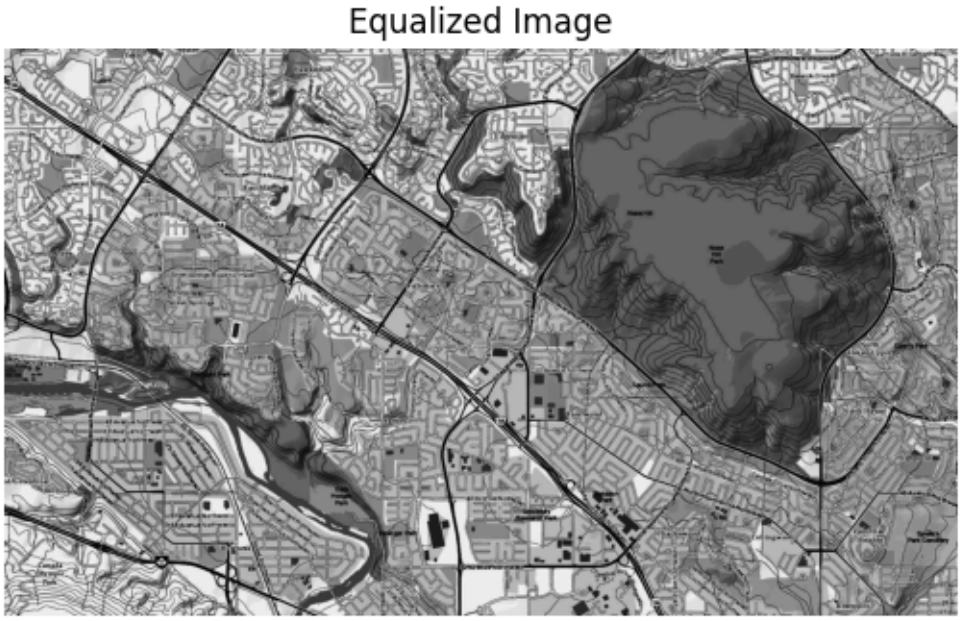

Equalized Image Histogram
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250


Equalized Image

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:15
```




Equalized Image Histogram
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250


Equalized Image

```
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:16
Invalid operation
Enter Image Processing Option (1-15)
     "1 : Grayscale Image", "2 : Gaussian Blur on Image", "3 : Rotate Image",
"4: Sharpen Image", "5 : Resized Image","6 : CannyEdge", "7 : BinaryThreshold",
"8 : Erosion",
     "9: Dilation", "10: Histogram", "11: Double Thresholding", "12: Prewitt
Filter","13: Text Extraction","14: Feature Extraction", "15: Image Restoration"
 Type "quit" to quit
 Option:quit
```

2.8 The End