SPL-1 Project Report

An Image Processing Tool

Submitted by :
MD. Sabbir Hosen
BSSE Roll No : 1333
BSSE Session : 2020-21

Supervised by :
Dr. Emon Kumar Dey
Designation : Associate Professor.
Institute Of Information Technology

Institute Of Information Technology
University Of Dhaka
21 - 05 - 2023

Table Of Contents

1.0 Introduction:

The project focuses on the development and implementation of a comprehensive image enhancement and segmentation tool using C++.
The objective of the project is to build a solid foundation in image processing by incorporating various techniques and algorithms, and to compare their performance for different applications. The project covers a wide range of functions, including image enhancement, segmentation, edge detection, and the calculation of the Normalized Difference Vegetation Index (NDVI) for remote sensing.

The project begins with image enhancement techniques such as brightening, negation, and smoothing. These techniques are employed to improve the visual quality of the images and provide a better foundation for subsequent analysis. Brightening enhances the overall brightness of grayscale images, while negation improves contrast by inverting intensity values. Smoothing techniques are utilized to reduce noise and enhance image clarity.

The segmentation phase focuses on isolating regions of interest within the images.

- Manual thresholding.
- Otsu's thresholding.

are implemented to separate foreground objects from the background. Additionally, Histogram equalization is employed to enhance the contrast of the images, making it easier to identify and extract important features.

Edge detection plays a crucial role in identifying object boundaries and sharp transitions in pixel intensities. The project explores various edge detection algorithms, including :

- Laplacian convolution.
- Sobel edge detection.
- Prewitt edge detection.
- Canny edge detection.

These algorithms enable precise localization of edges, which is essential for object recognition and analysis.
Furthermore, the project implements a research paper,
“Improved Edge Detection Algorithm for Brain Tumor Segmentation ”

that focuses on the Sobel operator for enhanced edge detection and thresholding techniques in the context of brain tumor segmentation. By incorporating additional thresholding methods and evaluating their performance, the project offers a comprehensive analysis of the Sobel operator's effectiveness in accurately delineating tumor boundaries.

Additionally, the project includes the calculation of the NDVI from a Landsat 4 dataset, showcasing the versatility of image processing techniques. NDVI is a widely used index in remote sensing applications to assess vegetation health and density. Its inclusion in this project highlights the ability to apply image processing techniques to diverse domains.

And then , a comprehensive metric was developed to compare the performance of various techniques in terms of noise reduction and edge detection capabilities.

By implementing these functions using C++ without external libraries, the project demonstrates a strong understanding of fundamental image processing algorithms and their practical implementation.
Overall, this project serves as a valuable resource for those seeking to gain hands-on experience with image enhancement and segmentation techniques. The comprehensive analysis, along with the inclusion of additional functions and the research paper implementation, contributes to the understanding and application of image processing methods in various domains.

1.1 Background :

1.1.1 BMP image :

A BMP (Bitmap) image is a commonly used raster graphics file format for storing digital images.BMP images are uncompressed and store pixel data in a straightforward manner. They can be either monochrome (black and white) or color images.

The structure of a BMP file consists of a file header, an optional bitmap information header, and the pixel data. The file header contains information about the file format and size, while the bitmap information header specifies details about the image, such as its width, height, color depth, and compression method (if any).

The pixel data in a BMP image is stored row by row, starting from the bottom-left corner of the image and moving horizontally to the right. Each pixel is represented by a certain number of bits depending on the color depth of the image. For example, in a 24-bit BMP image, each pixel is represented by 3 bytes (8 bits for each color channel: red, green, and blue), resulting in a total of 24 bits per pixel. This allows for a wide range of colors and more realistic representations.

1.1.2 PGM image :

A PGM (Portable Gray Map) image is a simple and widely used file format for grayscale images. PGM images are often used in computer vision, image processing, and scientific applications.

PGM images can have different variations, including PGM P2 (ASCII) and PGM P5 (binary). In both variations, the image data consists of a header section followed by the pixel intensity values.

The header section of a PGM image contains information about the file format, image size, and maximum pixel intensity value. In the P2 (ASCII) format, the header is text-based and includes information such as the "P2" identifier, width and height of the image, and the maximum intensity value. In the P5 (binary) format, the header is stored in binary form, providing the same information in a more compact way.

1.1.3 Threshold Based Segmentation :

The basic idea behind threshold-based segmentation is to select a threshold value that separates pixels into two categories: those that meet a certain criterion (above or below the threshold) and those that do not. The criterion is typically based on the intensity values of the pixels.
The threshold value can be determined in different ways :

- **Manual Thresholding:** The threshold value is selected manually by the user, based on their knowledge or visual inspection of the image. This method provides direct control over the segmentation process but may be subjective and time-consuming.
- **Global Thresholding:** A single threshold value is computed to separate the image into foreground and background regions. Common methods for global thresholding include Otsu's thresholding, which maximizes the separability between the foreground and background classes based on their intensity distributions.
- **Adaptive Thresholding:** Instead of using a single threshold value for the entire image, adaptive thresholding techniques adapt the threshold value locally, considering the local characteristics of each pixel's neighborhood. This approach is useful when illumination or intensity variations exist within the image.

Once the threshold value is determined, the image is segmented by assigning pixels above the threshold to the foreground or object class, and pixels below the threshold to the background or non-object class. The result is a binary image where the object of interest is represented by white pixels and the background by black pixels.

1.1.4 Edge Detection :

Edge detection is a fundamental image processing technique used to identify and locate boundaries or edges within an image. Edges represent significant changes in pixel intensity values and often correspond to object boundaries or transitions between different regions in an image.

There are several common edge detection algorithms, each with its own characteristics and strengths. Some of the popular edge detection techniques include:

Gradient-based Methods:

- Sobel operator: Computes the gradient magnitude of the image by convolving it with a set of predefined gradient kernels in the horizontal and vertical directions.
- Prewitt operator: Similar to the Sobel operator, it estimates the gradient magnitude by applying convolution with specific kernels.

Laplacian of Gaussian (LoG):

The LoG operator involves convolving the image with a Gaussian kernel to smooth it, followed by applying the Laplacian operator to detect edges based on the second derivative of the image.

Canny edge detector:

The Canny edge detector is a multi-stage algorithm that combines various techniques to achieve robust edge detection. It involves smoothing the image with a Gaussian filter, calculating gradient magnitudes and orientations, performing non-maximum suppression to thin the edges, and applying hysteresis thresholding to finalize the detected edges.

The output of an edge detection algorithm is often represented as a binary image, where the edge pixels are marked as white (or 1) and the non-edge pixels as black (or 0). However, some algorithms may produce a gradient magnitude image, highlighting the strength of the edges across the image.

1.1.5 NDVI :

NDVI (Normalized Difference Vegetation Index) is a commonly used vegetation index in remote sensing and satellite imagery analysis. It provides a quantitative measure of vegetation health and density by analyzing the reflectance properties of plants in the visible and near-infrared (NIR) spectral bands.

NDVI is calculated using the following formula:

NDVI = (NIR - Red) / (NIR + Red)

In the formula, "NIR" refers to the reflectance value in the near-infrared band, and "Red" refers to the reflectance value in the red band of the electromagnetic spectrum. These reflectance values are typically obtained from remote sensing data, such as multispectral satellite images.

interpreting NDVI values can provide valuable information about vegetation characteristics. Here are some common interpretations:

1. Negative Values: Typically associated with non-vegetated areas, such as water bodies, barren land, or built-up areas.
2. Near Zero Values: Correspond to areas with very low vegetation cover, such as deserts or sparsely vegetated regions.
3. Low Positive Values (0.1 to 0.3): Represent areas with moderate vegetation cover, such as grasslands or agricultural fields.
4. High Positive Values (0.3 to 0.8): Indicate dense and healthy vegetation, such as forests or areas with abundant plant growth.

NDVI is widely used in applications related to agriculture, forestry, ecology, and environmental monitoring. It helps assess plant health, monitor crop growth, detect vegetation stress, identify land cover changes, and analyze spatial patterns of vegetation distribution.

1.2 Challenges :

Implementing a project on image processing and segmentation in C++ without external libraries can present several challenges. Here are some specific points highlighting the potential difficulties:

1. Image Loading and Processing: Loading and manipulating image data without external libraries can be challenging. Image formats such as BMP and PGM have specific file structures and compression methods that need to be handled manually. Implementing functions like brightening, negation, smoothing, and histogram equalization will require careful parsing and manipulation of pixel data.
2. Edge Detection Algorithms: Implementing edge detection algorithms, such as Sobel, Prewitt, and Canny, involves convolutions and complex mathematical operations. Without external libraries, you'll need to manually perform these computations and handle data structures, such as convolution kernels and gradient calculations, efficiently.
3. Thresholding Techniques: Implementing user input thresholding and Otsu's thresholding requires analyzing the image histogram, computing threshold values, and segmenting the image accordingly. Handling these operations manually without the assistance of external libraries may require significant effort and attention to detail.

4. NDVI Calculation: Calculating NDVI involves extracting specific spectral bands, performing arithmetic operations, and generating the final index values. You need to know what is a landsat dataset, how to generate image using them. Implementing these calculations from scratch in C++ without external libraries may require knowledge of image bands, radiometric calibration, and appropriate mathematical transformations.
5. Metric Development: Designing and implementing a metric to compare edge detection techniques based on noise reduction adds an additional layer of complexity. Ensuring accuracy, fairness, and meaningful comparisons while considering the limitations of the project's constraints can be demanding.

2. Project Overview :

2.1 Image Read and Write :

To read and write an image we must know the header structure of that particular image. In our case we used two types of image PGM and BMP these are the header structures of these two images :

Header structure for PGM image.

The header of a PGM image file consists of:

1. First line containing signature of the image file either p2 or p5.
2. Second line is the comment line
3. Third line provides the information about height and the width of the image.
4. Fourth line specifies maximum intensity level of the image.

Data follows the header information and is written in pixel values (in text or binary format). Data is in raster order, which indicates that all data for the first row of the image is written first, then for the second row, and so on. The origin of the coordinate system for a PGM image is located on the top left corner.

2. Header structure for BMP image :

(a) File Header (14 Bytes).
(b) Information Header (40 Bytes).
(c) Color Table (4 * number of colour bytes).
(d) Pixel Data (variable bytes).

The functions used for read and write images are :
For PGM :

1. bool readImage(PGMImage* pgm , const char* filename)

2. void constructImage(PGMImage* pgm, char const* filename)

For BMP :

1. bool readBmpImage(BMPImage* bmp , const char* filename)

2. void constructBmpImage(BMPImage* bmp, char const* filename)

2.2 Image enhancement :

2.2.1 Brightening : Normally brighten the image by increasing the pixel value one by one.

2.2.2Smoothing :

- Box Filter:

  A simple smoothing technique where each pixel is replaced with the average of its neighboring pixels within a defined window or kernel. The box filter gives equal weight to all pixels in the neighborhood.
- Gaussian Filter:

  The Gaussian filter applies a weighted average to each pixel, giving more importance to nearby pixels and less to those farther away. The weights are determined by a Gaussian distribution, which results in a smoother and more natural blurring effect.

2.2.3. Histogram equalization :

Histogram equalization is a technique used in image processing to enhance the contrast and improve the overall visual appearance of an image. It redistributes the pixel intensities in such a way that the resulting image has a more balanced and stretched histogram.
Algorithm :
The process of histogram equalization involves the following steps:

1. Computing the Histogram: The first step is to calculate the histogram of the input image. The histogram represents the frequency distribution of pixel intensities, showing the number of pixels at each intensity level.
2. Cumulative Distribution Function (CDF): The next step is to compute the cumulative distribution function of the histogram. The CDF represents the accumulated probability of each intensity level in the image.
3. Histogram Equalization Transformation: Using the CDF, a transformation function is created that maps the original pixel intensities to new values. This transformation aims to spread out the intensity levels across the entire dynamic range.
4. Applying the Transformation: The transformation function is applied to each pixel in the image, mapping its original intensity to a new value based on the equalization process.
5. Generating the Equalized Image: The resulting transformed values form the equalized image, which has an enhanced contrast compared to the original image.

$$pdf(x) = p(r_k) = \frac{total\ pixels\ with\ intensity\ r_k}{total\ pixels\ in\ image\ x}.$$

From this we calculated the CDF (cumulative distribution function) as follows :

$$cdf(x) = \sum_{k=0}^{m-1} p(r_k)$$

Mathematically:

$$p(s_k) = \sum_{k=0}^{m-1} p(r_k)$$

Image Segmentation :

2.3 Thresholding :

2.3.1 Manual Thresholding :

Takes an integer as user input between (0 - 255) and sets the threshold value Th = input.

$$E(x,y) = \begin{cases} 255 & g(x,y) \geq Th, \\ 0 & otherwise \end{cases}$$

2.3.2 Otsu's Method :

Otsu's thresholding is a widely used technique for automatic thresholding. It determines an optimal threshold value to separate foreground and background regions based on their histogram of pixel intensities.

Algorithm:

Step 1: Compute histogram for a 2D image.

Step 2: Calculate foreground and background variances (measure of spread) for a single threshold.

(i) Calculate weight of background pixels and foreground pixels.

(ii) Calculate mean of background pixels and foreground pixels. iii)Calculate variance of background pixels and foreground pixels.

Step 3: Calculate "within class variance"

Mathematically :

1) Calculate weight for background and foreground pixels

$$W \text{ (Weight)} = \frac{w_1+w_2+\ldots w_n}{\text{Total number of pixels}}$$

1) Calculate mean

Histogram value =h

$$\mu \text{ (Mean)} = \frac{(h_1*w_1) + (h_2*w_2) + \ldots (h_n*w_n) + (h_n*w_n)}{\text{Sum of weights (W)}}$$

2) Calculate variance

$$V = \frac{((h_1-\mu)^2*w_1) + ((h_2-\mu)^2*w_2)\ldots. ((h_n-\mu)^2*w_n) + ((h_n-\mu)^2*w_n)}{\text{Sum of weights (W)}}$$

3) Calculate within class variance

I:e sum of two variances multiplied by their associated weights.

Within class variance = $W_b*V_b+W_f*V_f$

2.4 Edge detection :

Gradient based detection :

2.4.1 Sobel operator :

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

2.4.2 Prewitt Operator :

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

2.4.3 Canny Edge detection :

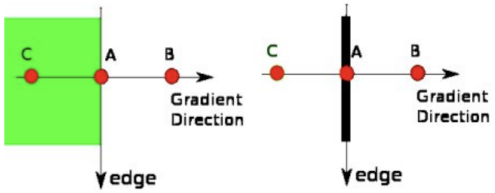
Algorithm :

1. Noise reduction : we use the Gausian filter, which is convolved with the image and removes the noise, preventing the desired edges in output images
2. Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction G(x) and vertical direction G(y). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

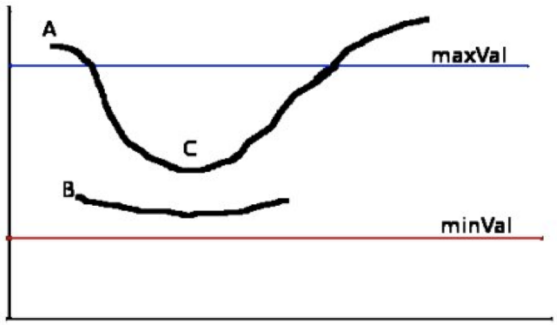
3. After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:



image

Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

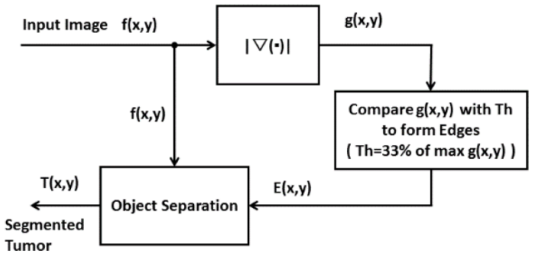
4. **Hysteresis Thresholding :** This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:



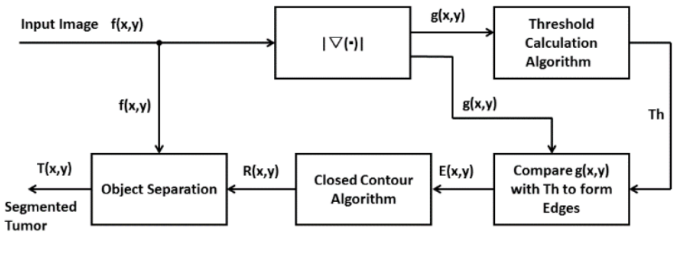
2.5 Application of image segmentation :

Asra Aslam , Ekram Khan , M.M. Sufyan Beg.
Improved Edge Detection Algorithm for Brain Tumor Segmentation
Procedia Computer Science, 58 (2015), pp. 430-437
It combines edge detection and region growing techniques to provide an enhanced algorithm for brain tumor segmentation.
These are the procedures that it follows :

1. Finding gradient image using Sobel Operator
2. Calculate image dependent threshold iteratively
3. Apply Closed-Contour Algorithm
4. Object segmentation based on pixel intensity within closed contour.



(a) Conventional Algorithm



(b) Proposed Algorithm

2.5.1 Iterative thresholding :

Algorithm :

1. Let the initial threshold be Th0 which is equal to the average intensity of gradient image g(x, y), as defined.

$$Th^0 = \frac{\sum_{j=1}^{h}\sum_{i=1}^{w} g(x,y)}{h \times w}$$

2. Set iteration index l = 0, separate g(x,y) into two classes, where the lower class consists of those pixels of g(x,y) which have gradient values less than Thl, and the upper class contains rest of the pixels.

3. Compute the average gradient values mL and mH of lower and upper classes respectively.

4. Set iteration l = l+1 and update threshold value as:

$$Th^l = \frac{m_L + m_H}{2}$$

5. Repeat steps 2 to 4 until is satisfied, where and take Thl as final threshold and denote it by Th.

2.5.2 Closed Contour Algorithm :

Closed-Contour Main

1. for $i = 1$ to h
2. for j=1 to w
3. if $E(i,j)$=0
4. { for $k = 1$ to r
5. if($E(i,j) \in$ region R_k)
6. { goto step 1 to scan next pixel
7. }
8. $r = r + 1$ //increment the region index by 1
9. $E(i, j) \rightarrow Rr$ //insert element in region R_r+1
10. call Search(i,j)
11. }

Closed-Contour Search(i_0, j_0)

1. for $n = 1$ to 8 neighbours of pixel (E(i_0, j_0))
2. { for $k = 1$ to r
3. { if(E(i_n, j_n) $\in$ region R_k)
4. { goto step 1 to scan next neighbour
5. }
6. }
7. for x = −2 to 2
8. for y = −2 to 2
9. if(E($i_n + x, j_n + y$) = 255) goto step 1
10. E(i_n, j_n) → R_r //insert element in region R_r
11. call search(i_n, j_n)
12. }

2.5.3 Object separation :

1. for $t = 1$ to r where $R_t \in R(x, y)$
2. { count number of pixels $(i, j) \in R_t$ as n_t
3. set sum $S_t = 0$.
4. for all pixels $(i, j) \in R_t$
5. $S_t = S_t + f(x, y)$
6. $\mu_t = S_t / n_t$
7. }
8. find R_k such that $\mu_k = \max\{\mu_1, \mu_2, \ldots, \mu_r\}$
9. set all pixels $T(i,j) \leftarrow 0$
10. for all pixels $(i, j) \in R_k$
11. $T(i, j) = f(i, j)$

2.6 Edge detection comparison metrics :

We implemented two widely used metrics to compare between Sobel, Prewitt and Canny edge detection technique.

2.6.1 Mean Square Error :

The mean square off the errors is the difference between the original image and the edge detected image. The contrast happens due to randomness .The MSE incorporates degradation function and statistical characteristics of noise in the edge detected image. The higher MSE indicates a greater difference between the original and processed image.

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M,N}$$

2.6.2 Peak Signal To Noise Ratio :

According to a paper by : D. Poobathy , Dr. R Manicka Chezian
Edge Detection Operators : Peak Signal To Noise Based Ratio
MECS (2014), 10, pp (55-61)

Peak signal-to-noise ratio, is ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. The PSNR usually expressed in terms of the decibel (dB) scale. PSNR is a rough estimation to human perception of reconstruction quality. Although a higher PSNR generally indicates that the reconstruction is of higher quality in image compression. But in some cases like edge detection PSNR should lesser to achieve proper results. The PSNR calculated based on the MSE by,

$$PSNR = 10\,log_{10}\left(\frac{R^2}{MSE}\right)$$

2.6.3 Comparison Results :

(a) Blackbuck :




Fig : a | Fig : b | Fig c | Fig : d

Type	Sobel	Prewitt	Canny
MSE :	5611.25	5775.4	7809.41
PSNR	2.93855	2.93229	2.86677

(b) GreenLand :

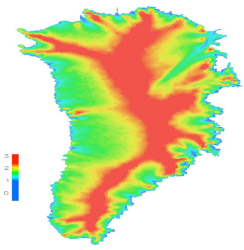
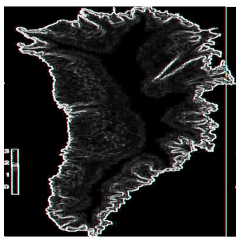
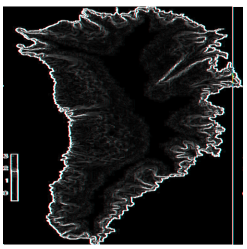





Fig : a | Fig : b | Fig c | Fig : d

Type	Sobel	Prewitt	Canny
MSE :	35717	36616.8	40058.6
PSNR	2.53664	2.53124	2.51173

3. Results :

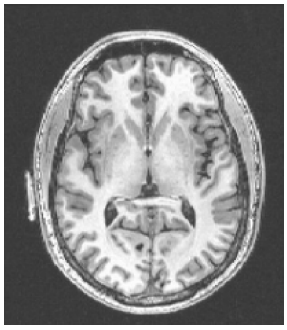
3.1 Image Enhancement :

3.1.1 Brightening :

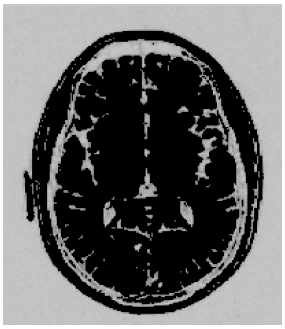
Sample Image

Brighten Image

3.1.2 . Negation :



Sample Image



Negative Image

3.1.3 Histogram Equalisation :

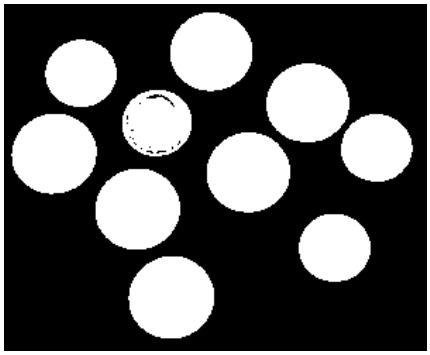
Sample Image

Histogram Equalised Image

3.2 Image Segmentation :

3.2.1 Otsu's Thresholding :

Sample Image



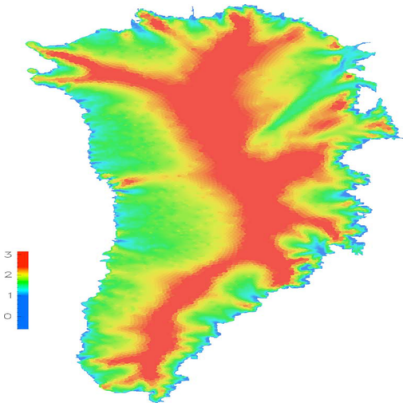
Threshold Image

3.2.1 Sobel Edge Detection :

Sample Image

Sobel Output Image

3.2.2 Prewitt Edge Detection :



Sample Image

Prewitt Output Image

3.2.3 Canny Edge Detection :

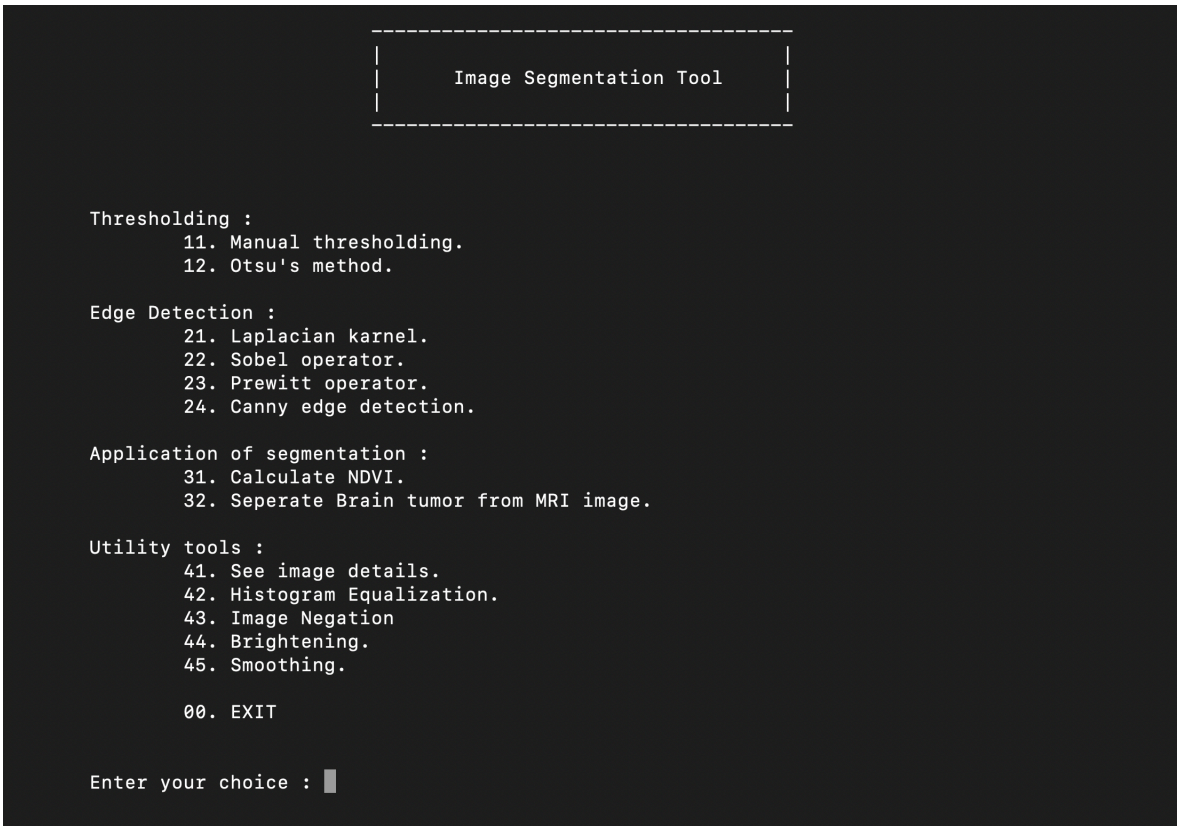
Sample Image

Canny Output Image

4. User manual :

Using of this tool is very much easy. You just need to type an integer number based on your choice. Then there will be a message **“Enter file name : ”** , you should enter any .bmp or .pgm file suggested by the tool. If your operation is successful this will show you a confirmation message and give you the output file name. If there is an error it will show you the error message.

Step 1:



Step 2 :

```
Enter your choice : 24

 Enter file name (.bmp) : sample.bmp
Operation successful!
Output file name : CannyOut.bmp
```

The following image shows the sample image and the output image :

(a) sample.bmp

(b) CannyOut.bmp

5. Conclusion :

In conclusion, this project implemented various image enhancement and segmentation techniques in C++ without external libraries. The goal was to build a basic image enhancement tool and compare different functions. Techniques such as brightening, negation, smoothing, histogram equalization, and thresholding were implemented. Edge detection methods like Sobel, Prewitt, and Canny were also used. A research paper on brain tumor segmentation was implemented using the Sobel operator. NDVI calculation and a metric comparing Sobel and Canny edge detection for noise reduction were performed. The project successfully achieved its objectives of implementing a range of image enhancement and segmentation techniques, comparing their performance, and applying them to specific domains such as brain tumor segmentation and vegetation analysis. The project demonstrated the capabilities of C++ for image processing tasks and provided valuable insights into the strengths and limitations of different techniques. Future work could involve further optimization of algorithms for performance improvement and exploring additional image processing techniques for enhanced analysis and segmentation tasks.

6. References :

1. https://users.wpi.edu/~cfurlong/me-593n/pgmimage.html#PGMTop
   13/02/2023
2. PCSK Gaddam, P Sunkara - 2016 - diva-portal.org
   Advanced Image Processing Using Histogram Equalization and Android Application Implementation.
   https://www.diva-portal.org/smash/get/diva2:1063170/FULLTEXT01.pdf
3. http://www.ue.eti.pg.gda.pl/fpgalab/zadania.spartan3/zad_vga_struktura__pliku_bmp_en.html
   09/04/2023
4. http://www.ripublication.com/ijaerdoi/2015/ijaerv10n9_20.pdf
   11/04/2023
5. https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html
   07/05/2023
6. Asra Aslam , Ekram Khan , M.M. Sufyan Beg.
   Improved Edge Detection Algorithm for Brain Tumor Segmentation
   Procedia Computer Science, 58 (2015), pp. 430-437
   https://www.sciencedirect.com/science/article/pii/S1877050915021687
   14/05/2023
7. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm#:~:text=The%20Gaussian%20smoothing%20operator%20is,(%60bell%2Dshaped%27)%20hump.
   14/05/2023
8. https://up42.com/blog/5-things-to-know-about-ndvi
   17/05/2023
9. Landsat 8 dataset https://earthexplorer.usgs.gov/
   18/05/2023
10. D. Poobathy , Dr. R Manicka Chezian
    Edge Detection Operators : Peak Signal To Noise Based Ratio
    MECS (2014), 10, pp (55-61)
    https://www.semanticscholar.org/paper/Edge-Detection-Operators:-Peak-Signal-to-Noise-Poobathy-Chezian/ebada25e647fd0501f5b2386b3f17b210cecb4af
    21/05/2023