POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies

Politecnico
di Torino

Master's Degree Thesis

Analysis of Topological Distribution of Skin Lesions

Supervisors

Prof. Marco Piras

Prof. Guido Pagana

Candidate

Nikoo Arjang

JULY 2024

ABSTRACT

This thesis presents a comprehensive study on the segmentation of dermoscopic images for the purpose of accurately identifying and delineating lesion areas. The developed pipeline integrates multiple stages of image preprocessing and segmentation techniques to ensure high accuracy and reliability in lesion detection. Given the critical role of lesion shape in diagnosing skin diseases, the objective is to assist clinicians in accurately extracting lesions from surrounding skin, thereby enhancing subsequent diagnosis and treatment. Additionally, by quantifying the geometric area of the lesions, it becomes feasible to monitor their progression over time.

A dataset of 200 high-resolution dermoscopic images, each accompanied by ground truth annotations, is utilized. The images are resized from their original dimensions to 512x512 pixels to standardize the dataset. This resizing ensures consistent input dimensions and facilitates more efficient processing. Subsequently, grayscale conversion is applied to decrease computational load and prioritize key features. Afterwards, some pre-processing techniques including histogram equalization, advanced noise removal were implemented to make features more distinguishable and eliminate artifacts such as hair from the images, respectively. These techniques include a calibrated thresholding approach for binary masking. Gaussian smoothing, and median filtering are then applied which collectively ensure that the images are clean and of high quality for segmentation.

Quantitative analysis is performed to measure the lesion areas in square millimeters, based on pixel spacing calculated from the field of view at 20x magnification. This step provides meaningful clinical measurements of lesion extents. Finally, three segmentation methods are evaluated: Binarization, Canny edge detection, and Sobel edge detection. For assessment of the accuracy of each method, the performance evaluation has been carried out using Dice coefficient, incorporating comprehensive statistical analysis such as mean, median, and variability metrics. The binarization method is emerged as the most effective indicating superior accuracy and consistency.

ACKNOWLEDGEMENT

To my parents, whose steadfast support and belief in me have been a constant source of motivation, I am profoundly grateful. This thesis is a testament to your enduring love and encouragement.

This master's thesis represents the culmination of my academic journey at Politecnico di Torino, pursuing an MSc in ICT for Smart Societies. This endeavor has been both challenging and rewarding, and I am deeply grateful to those who have supported me along the way. I would like to extend my heartfelt thanks to my supervisors, Prof. Marco Piras and Prof. Guido Pagana. Their guidance, encouragement, and insightful feedback have been crucial to the completion of this thesis. I am also thankful to the LINKS Foundation for their support and belief in my work, which provided essential resources for this research. A special thanks goes to my friends Morteza Mollaei and Aslan Nouri, whose assistance and unwavering support were invaluable throughout this process.

Thank you to everyone who contributed to this achievement.

Nikoo Arjang

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Skin cancer is one of the most common cancers globally, with melanoma being the most dangerous form due to its high potential for metastasis. Early detection and accurate diagnosis are critical for improving survival rates. Dermoscopy, also known as dermatoscopy, is a non-invasive imaging technique that enhances the visualization of pigmented skin lesions, aiding in the differentiation between benign and malignant lesions. The detailed dermoscopic images provided by this technique allow clinicians to identify various structures and patterns not visible to the naked eye, facilitating early and accurate diagnosis.

Dermoscopic images are specialized magnified images of the skin taken using a dermatoscope[1]. This handheld device combines a magnifying lens with a light source to illuminate and visualize the skin in detail. Dermoscopic images reveal various skin lesions, each with distinct characteristics. Common nevi (moles) are typically benign and not associated with a high risk of skin cancer. However, atypical nevi (dysplastic nevi) and melanoma are frequently studied due to their significant potential for developing into skin cancer. A common nevus, or mole, is a benign skin lesion resulting from the proliferation of melanocytes. These moles are typically small, uniformly colored, and have well-defined, smooth borders. They are prevalent in the general population and generally harmless. However, changes in their appearance can sometimes indicate malignancy (Fig. 1-1)[1].

Fig. 1-1 Common nevus dermoscopic image [1]

Atypical Nevus (Dysplastic Nevus) are unusual moles that can resemble melanoma but are generally benign. They are larger than common nevi and display a heterogeneous color pattern. Their presence can be associated with an increased risk of melanoma. Dermoscopically, atypical nevi exhibit an asymmetrical shape, irregular and poorly defined borders, and non-uniform pigmentation. Additionally, they may present different structures such as dots, globules, and streaks (Fig. 1-2)[1].

Fig. 1-2 Atypical nevus dermoscopic image [1]

Melanoma is a malignant tumor of melanocytes and represents the most dangerous form of skin cancer. Early detection and treatment are critical for improving survival rates. Melanomas often exhibit asymmetry, irregular or poorly defined borders, and a diverse color palette. They vary greatly in size and shape and often show changes over time. Dermoscopic features of melanoma include uneven color distribution, and the presence of atypical networks, streaks, blue-white veils, and regression structures (Fig. 1-3)[1].

Fig. 1-3 Melanoma dermoscopic image [1]

Dermoscopy enhances the examination of pigmented skin lesions, enabling the identification of various features that are not visible to the naked eye. This technique is particularly valuable in the early detection and diagnosis of skin cancers, including

melanoma. The use of dermoscopic images in research and clinical practice has led to the development of various diagnostic algorithms and criteria, such as the ABCD rule (Asymmetry, Border, Color, Diameter) and the 7-point checklist, which help standardize the assessment of dermoscopic features and guide clinical decision-making [2].

In recent years, advancements in image processing techniques have revolutionized the diagnosis and management of skin diseases. By harnessing the power of digital imaging technologies and computational algorithms, healthcare professionals can now analyze skin lesions with unprecedented accuracy and efficiency. These tools enable clinicians to differentiate between benign and malignant lesions, track disease progression, and guide treatment decisions with greater confidence. Image processing has emerged as a cornerstone technology in healthcare, revolutionizing diagnostic and therapeutic approaches across various medical disciplines. The integration of image processing techniques into healthcare applications has transformed the landscape of medical diagnostics, enabling non-invasive visualization and quantification of anatomical structures, physiological processes, and pathological abnormalities. From radiology and pathology to dermatology and ophthalmology, image processing plays a pivotal role in augmenting the diagnostic capabilities of clinicians and improving patient outcomes. [3]

In this study, several pre-processing techniques are utilized on dermoscopic images and various segmentation methods are applied to detect skin lesions. The shape of these lesions is crucial for diagnosing skin diseases, and our goal is to support clinicians in precisely isolating lesions from the surrounding skin to improve diagnosis and treatment accuracy. Furthermore, by measuring the geometric area of the lesions, it becomes possible to track their progression over time, providing valuable insights for ongoing patient monitoring and care. The methodology in this study involves several key steps of image pre-processing to ensure the quality and consistency of the data, and the application of various segmentation techniques including Binarization, CANNY and SOBEL edge detection. The performance of each segmentation method is evaluated using the Dice coefficient, which measures the similarity between the segmented results and the ground truth. The average of this Dice coefficient, standard deviation, and other statistical measures for each method are calculated to assess their accuracy and robustness. The Binarization method has been emerged as the most effective, achieving the highest average Dice coefficient, demonstrating superior accuracy and consistency.

CHAPTER 2
LITERATURE REVIEW

Computer-assisted diagnosis (CAD) systems and image processing have been extensively researched over the past two decades to assist skin cancer specialists in improving lesion detection. The concept was initially proposed around 1985. By utilizing image processing techniques, skin lesion images are analyzed to identify specific features indicative of malignancy, such as asymmetrical shape, irregular border, color variation, and diameter. []Photography, a common tool in dermatological practice, has become more accessible with the development of low-cost imaging systems, enabling clinicians to obtain high-resolution medical images of areas of interest. Screening pigmented skin lesions (PSLs) aids clinicians and dermatologists in better visualization, documentation, and tracking of lesion evolution over time. Clinical photography and dermoscopy are two widely used lesion screening methods, each with its own advantages and differences. Clinical photography captures images from the skin's surface, akin to what a clinician observes, while dermoscopy magnifies and visualizes lesion patterns and substructures not visible to the naked eye. Dermoscopy, when used by trained practitioners, has been shown to enhance lesion diagnosis accuracy. Computer-aided diagnostic systems offer quantitative and objective evaluation of PSLs, serving as a supportive tool for physicians by providing automated or semi-automated diagnoses. Figure 2.1 depicts the essential components of lesion CAD tools, which are image processing algorithms aimed at diagnosing lesion types.

Fig. 2-1- The components of lesion detection system.

The human skin consists of three primary layers: the epidermis, dermis, and hypodermis (also known as subcutaneous tissue). These interconnected layers play crucial roles in maintaining skin health (see Figure 2-2). The dermis, comprised of collagen and elastic fibers, is divided into two layers: the papillary dermis (upper thin

layer) housing the epidermis and dermis, and the reticular dermis (lower thick layer) containing blood and lymph vessels, nerve endings, sweat glands, and hair follicles. It plays essential roles in providing energy and nutrients to the epidermis, as well as regulating temperature, aiding in healing, and facilitating the sense of touch.

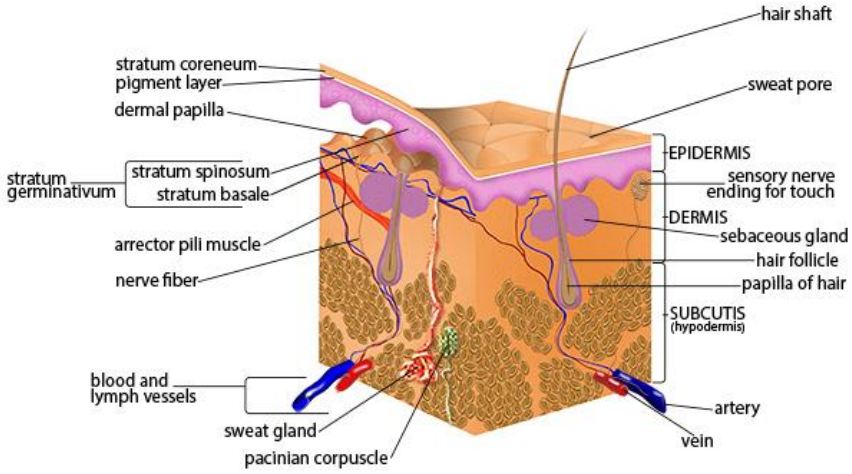


Fig. 2-2- The skin's anatomy comprising of different layers [5].

The epidermis serves as the outermost protective layer of the skin, comprised of four layers: the basal layer (Stratum Basale), stratum spinosum, stratum granulosum, and stratum corneum. Within the epidermis, four types of cells exist: keratinocytes (making up 95% of cells), melanocytes, Langerhans' cells, and Merkel cells. Melanocytes produce melanin, which can transfer to nearby keratinocytes. This process can intensify through tanning reactions from sun exposure or UV radiation, potentially increasing the risk of malignant transformation. Basal cell carcinoma and squamous cell carcinoma, arising from non-pigmented basal and squamous keratinocytes, are the most prevalent skin cancer types [6].

The analysis of dermoscopic images has seen significant advancements through the integration of various computational methods. Early studies concentrated on extracting specific features such as color, shape, and texture from these images. Rubegni et al. (2002) demonstrated the effectiveness of pattern analysis in differentiating between benign and malignant lesions, with techniques like the ABCD (Asymmetry, Border, Color, and Diameter) rule being developed to quantify these features. Machine learning has played a pivotal role in enhancing image analysis, with classifiers like support vector machines (SVM) and k-nearest neighbors (k-NN) being employed to distinguish between various lesion types. More recently, convolutional neural networks (CNNs) have shown remarkable performance in image classification tasks. Esteva et al. (2017) illustrated that deep learning models could achieve dermatologist-level accuracy in identifying skin cancer [7]. Topological Data Analysis (TDA) has emerged as a powerful tool for examining the topological

features of data. This method considers the shape and connectivity of data points, which is particularly useful for analyzing skin lesion distributions. Researchers such as Perea and Harer (2015) have applied TDA to study the spatial patterns of lesions, identifying topological invariants that correlate with malignancy. Spatial and morphological analyses have also been a focus, with recent studies highlighting the importance of considering lesion distribution across the patient's entire skin surface [8]. Marghoob et al. (2013) emphasized techniques like Voronoi diagrams and Delaunay triangulations to model the spatial arrangement of lesions. These methods help in identifying clusters and patterns that may indicate a higher risk of melanoma [9].

Furthermore, integrating dermoscopic image analysis with clinical data, such as patient history and genetic information, has shown promise in improving diagnostic accuracy. This holistic approach allows for a more comprehensive assessment by considering both visual and non-visual factors, ultimately enhancing the reliability of diagnoses. To sum up, it's vital to identify the most robust and efficient features to extract lesion from skin images. Thus, in this study, we examine and discuss different methods for lesion detection using dermoscopic images.

CHAPTER 3
MATERIALS AND METHODS

In this study, a comprehensive pre-processing pipeline is initially designed to enhance dermoscopic images, facilitating accurate label production through MATLAB. The primary objective is to improve the lesion diagnosis process by enabling the automatic segmentation of lesion areas. The pre-processing pipeline incorporates several steps, including center cropping, resizing, gray-scale transformation, histogram equalization, noise removal, Gaussian smoothing, and median filtering. During the pre-processing, image dimensions are standardized, computational load is reduced, contrast is enhanced, noise is removed, and image quality is refined. These steps prepare the dermoscopic images for segmentation. Following pre-processing, the segmentation phase is carried out utilizing three methods: Binarization, Sobel, and Canny edge detection techniques, each contributing to the delineation of lesion boundaries. The entire pipeline is illustrated in Figure 3-1. Subsequently, the performance of the pre-processing and segmentation steps is rigorously assessed using the Dice coefficient. This metric is crucial for validating the precision of lesion localization, ensuring that the pre-processing pipeline produces high-quality, reliable results. The subsequent sections of this study provide detailed descriptions of the methodologies, tools, and algorithms employed.

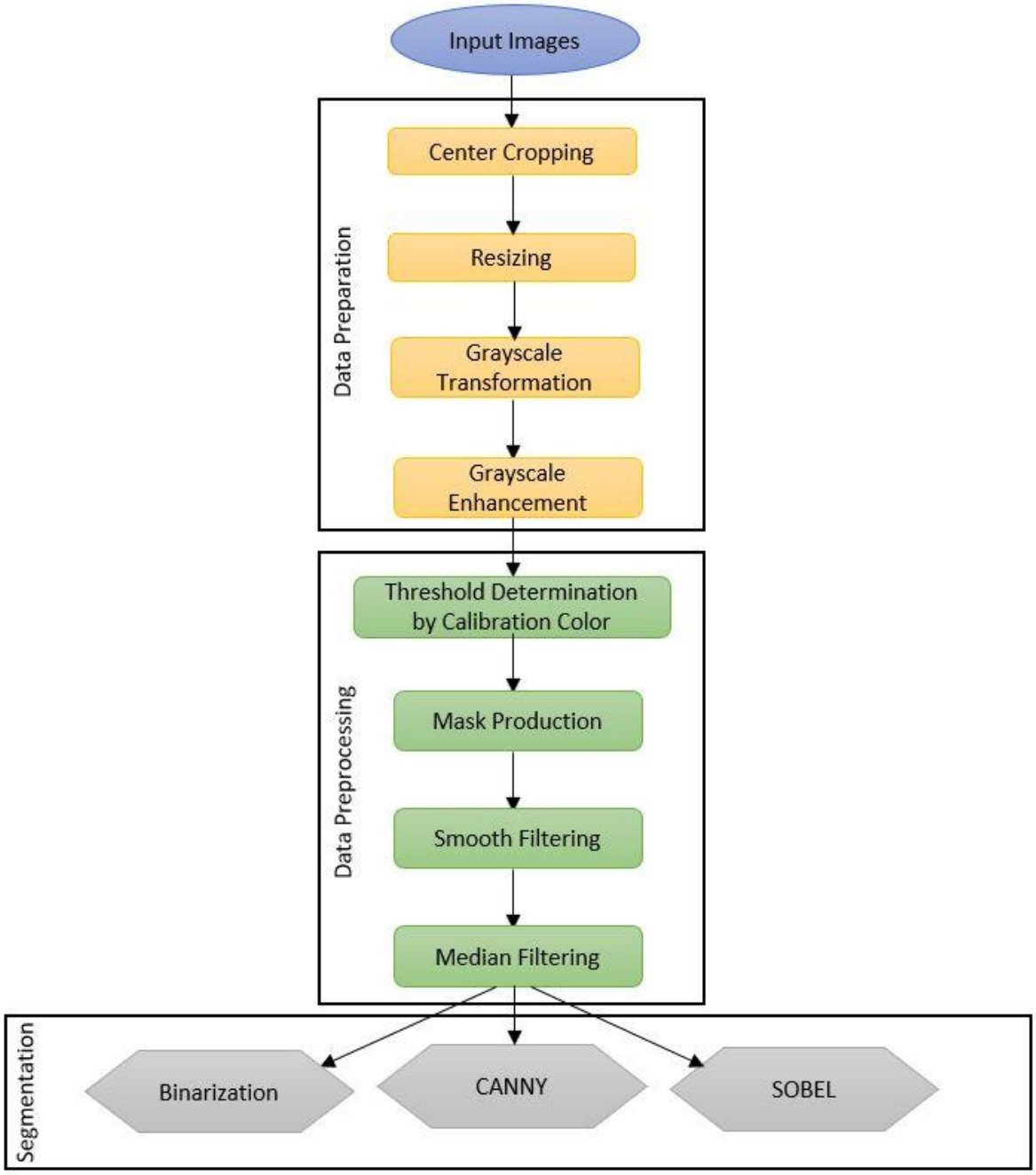


Fig. 3-1- Dermoscopic images pre-processing segmentation pipeline

3.1 DATASET

In this work, we deal with dermoscopic images for various types of skin anomalies. The PH_2 dataset is utilized which is publicly available for research and educational purposes [1]. This dataset comprises dermoscopic images acquired at the Dermatology Service of Hospital Pedro Hispano in Matosinhos, Portugal. These

images were taken under standardized conditions using the Tuebinger Mole Analyzer system. Each image is accompanied by detailed medical annotations, including lesion segmentation, clinical and histological diagnoses, and the evaluation of various dermoscopic criteria such as colors, pigment network, dots/globules, streaks, regression areas, and blue-whitish veil [1]. In dermatology, imaging techniques play a crucial role in diagnosing and monitoring skin conditions. Further information about the images of the dataset is given in the Table. 3-1:

Aspect	***Details***
Image Magnification	*20x*
Image Type	*8-bit RGB color images*
Resolution	*768x560 pixels*
Categories	*80 common nevi, 80 atypical nevi, 40 melanomas*

Table. 3-1- Dermoscopic image dataset information [1]

Intended for an in-depth examination of skin lesions, dermoscopic imaging enables healthcare specialists to visualize structures beneath the skin's surface. This kind of imaging is used in dermatology aiding in diagnosing conditions such as melanoma and other skin anomalies. Images are captured with a dermatoscope, utilizing polarized or non-polarized light and often immersion fluids to reduce surface reflections, revealing fine details like pigmentation patterns and vascular structures invisible to the naked eye. With high magnification and resolution, these images focus on specific lesions or areas of interest, providing a limited but detailed field of view. The high diagnostic value of dermoscopic imaging lies in its ability to visualize structures such as pigment networks and vascular patterns, enhancing specificity in differentiating between benign and malignant lesions. The controlled lighting, often with polarized light, minimizes reflections and enhances the visibility, color contrast, and clarity of subsurface structures, making it a crucial tool for diagnosing dermatological conditions. [2]

3.2 PRE-PROCESSING

Pre-processing encompasses operations performed on images at the most fundamental level of abstraction, wherein both input and output manifest as intensity images. Such images typically mirror the original data captured by any device, wherein an intensity image is commonly represented by one or more matrices of brightness values. It is important to note that pre-processing does not augment the information content of the image; rather, it often diminishes information, as quantified by entropy. Thus, from an information-theoretic perspective, the optimal pre-processing strategy would entail none at all; indeed, the most effective approach to circumventing the need for elaborate pre-processing is to prioritize high-quality image acquisition.

Nevertheless, pre-processing remains invaluable across a spectrum of scenarios owing to its capacity to attenuate information extraneous to the particular image processing or analysis task at hand. Consequently, the principal objective of pre-processing is to refine the image data by mitigating undesirable distortions or amplifying pertinent image features requisite for subsequent processing. Additionally, geometric transformations of images, such as rotation, scaling, and translation, are categorized as pre-processing methods herein, given their analogical utility in this context [10].

The inherent redundancy of information within the majority of images provides sufficient opportunity for image pre-processing techniques to analyze data and discern image characteristics in a statistical context. These identified characteristics serve a dual purpose: either to mitigate inadvertent degradation, such as noise, or to augment the image's quality. Notably, in real images, adjacent pixels corresponding to objects typically exhibit analogous or identical brightness values [10]. In the following, we delve into different preprocessing steps that are considered in this work.

3.2.1 Image Resizing

Image resizing is a crucial preprocessing step in numerous computer vision and many other applications. Resizing images ensures uniform input sizes, standardizing all images to the same dimensions. This uniformity is crucial for efficient processing and input into the networks, maintaining consistent image dimensions during each stage. Smaller, resized images demand fewer computational resources and memory, speeding up training and inference processes. This also helps manage large image

data sizes, making them more manageable in terms of memory and storage. Proper resizing techniques preserve the original aspect ratio, preventing unintended distortions that could affect model performance. Some methods also allow for controlled distortion, useful for data augmentation by generating varied versions of the same image. Correct resizing improves model accuracy by highlighting relevant image features and minimizing irrelevant details. Normalization through resizing ensures the model learns from standardized data, enhancing its ability to generalize across different datasets[11].

There are different techniques to perform image resizing such as adjusting size in which mainly eliminates portions of the image to achieve specific dimensions that are utilized in this study. To ensure preserving the informative part of each image, 'imresize' function using the 'bilinear' method is utilized to resize the images to 512x512. It would be cautious to say that 'imresize' changes the resolution, as the definition of resolution can vary. In image processing, resolution often refers to the ability to distinguish between two neighboring fine objects. While 'imresize' alters the number of pixels and pixel density, it does not change the pixel size itself. This means it can maintain the ability to separate two objects that are distinguishable in the original image. To assess the quality of the resized image in terms of spatial resolution, the Structural Similarity Index Measure (SSIM) function can be used. This involves recovering the resized image back to its original dimensions and comparing it with the original image. In this case, the spatial resolution preservation can be evaluated. A value closer to one indicates a higher similarity to the original image, reflecting better preservation of spatial resolution.

3.2.2 Gray-Scale Transformation

Gray-scale transformation is a technique used to simplify image processing by converting images to grayscale, thus reducing computational load and focusing on essential features. In an RGB image, each pixel comprises red, green, and blue components. The grayscale conversion process involves combining these components into a single intensity value. This is typically done using a weighted sum of the red, green, and blue values, reflecting the human eye's sensitivity to different colors. For example, the formula:

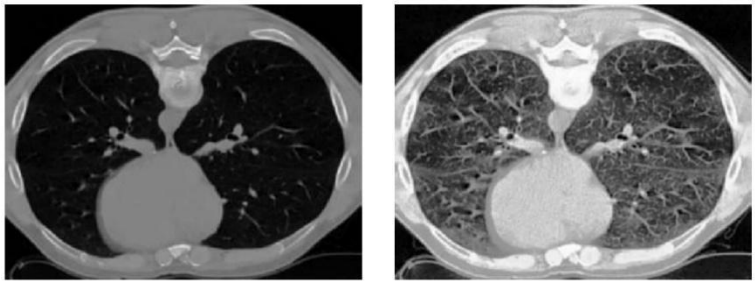
$$Gray = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B,$$

is commonly used, where $R, G\ and\ B$ are the intensities of the red, green, and blue channels, respectively. This ensures the grayscale image accurately represents the perceived brightness of the original RGB image.

3.2.3 Histogram Equalization

To further enhance the quality of the grayscale images, histogram equalization is applied. This technique redistributes the brightness values to span the entire range of possible values, thereby enhancing the contrast of the image. Histogram equalization is particularly useful for making features more distinguishable, which is crucial in medical imaging where clear visibility of details is necessary for accurate diagnosis. By adjusting the input histogram $H(p)$ of grayscale intensity values to create a more uniform output histogram $G(q)$, we improve the contrast by balancing dark and light areas.

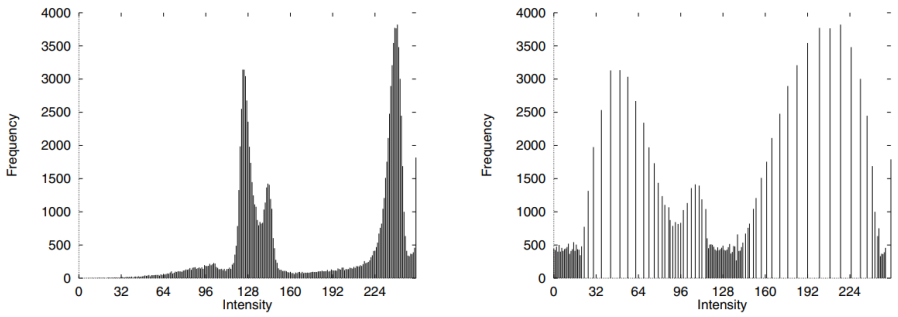
Histogram equalization is performed by first calculating the histogram of the original grayscale image to determine the frequency of each intensity value. Then, the cumulative distribution function (CDF) is computed from this histogram. Each original intensity value p is mapped to a new intensity value q using the CDF, ensuring that the new values are spread more evenly across the entire range of possible intensities. This is achieved by normalizing the CDF and scaling it to the desired range of intensity values. The result is an image with enhanced contrast, where details are more visible due to the more uniform distribution of brightness levels. [12]



(a) (b)

Fig. 3-2- Histogram equalization of a lung image, a) original image, b) equalized image [13].

The histograms of both images depicted in Figure 3-2 are presented in Figure 3-3, which assesses the alterations resulting from the application of histogram equalization to the image.



(a) (b)

Fig. 3-3- Histogram equalization diagram, a) original image, b) equalized image [13].

3.2.4 Noise Removal Using Masking

During image preprocessing, it is frequently essential to eliminate different forms of noise, such as hair and other unwanted elements in the images, which may disrupt the precise evaluation of skin lesions. The rationale behind this can be divided into three main reasons. First, it enhances the clarity of images by eliminating obstructions that may obscure important details, thereby reducing the risk of incorrect diagnoses or misinterpretations. Second, the elimination of noise improves image quality, which in turn enhances the performance of models used for tasks such as segmentation and classification. Lastly, it ensures consistency by providing uniform images that are free from artifacts, thereby maintaining the integrity of diagnostic procedures.

One effective method for achieving this objective involves employing masking techniques. This section outlines the strategy of employing masking to eliminate hair from images. The masking methodology involves several steps, including converting the image to a binary format, creating a mask, and then applying this mask to remove noise from the original image.

3.2.5 Image Smoothing

Image smoothing leverages the redundancy inherent in image data to mitigate noise, typically achieved through some variation of brightness value averaging within a designated neighborhood O. However, smoothing introduces the challenge of blurring sharp edges. Hence, we shall focus on edge-preserving smoothing methods herein, where the average is computed solely from points within the neighborhood exhibiting similar properties to the point undergoing processing. Local image smoothing can effectively eradicate impulse noise or degradations manifesting as thin

stripes. Nonetheless, it proves ineffective in scenarios where degradations manifest as large blobs or thick stripes. In such cases, image restoration techniques may be employed to address these issues [14].

Let us assume that the noise value ν at each pixel is an independent random variable characterized by a zero mean and a standard deviation of σ. If we capture the same static scene under identical conditions n times, from each captured image, a specific pixel value g_i, where i = 1, . . . , n, is selected. An estimate of the correct value can be derived by averaging these selected values, accounting for their corresponding noise values $\nu_1, \ldots, \nu_n$ [13]:

$$\frac{g_1 + \ldots + g_n}{n} + \frac{\nu_1 + \ldots + \nu_n}{n}$$

The subsequent term in this expression delineates the noise component, which once more constitutes a random value characterized by a mean of zero and a standard deviation. Consequently, if n images capturing the same scene are at our disposal, smoothing can be executed without inducing image blur through [13]:

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j)$$

This line of reasoning aligns with a well-established statistical principle: when a random sample is drawn from a population, and the corresponding sample mean is computed, it yields a distribution of sample mean values when this process is repeated multiple times. The distribution of sample means offers several beneficial properties. it has a mean equal to the population mean and a variance of $\frac{\sigma}{\sqrt{n}}$,which is smaller than that of the original population. If the original data follows a normal distribution, the distribution of sample means will also be normal. Importantly, the central limit theorem indicates that the distribution of sample means tends to be normal regardless of the original distribution. Practically, this theorem reduces the need to explicitly generate the distribution of sample means. In statistical analysis, approximately 30 samples are generally considered the minimum number of observations necessary.

Typically, only one noise-corrupted image is available, and averaging is subsequently conducted within a local neighborhood. This approach yields satisfactory results when the noise exhibits a smaller size than the smallest objects of interest in the image. However, a notable drawback is the blurring of edges.

Averaging can be viewed as a specific instance of discrete convolution. For instance, a neighborhood size of 3 × 3, the convolution mask h is as follows [13]:

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

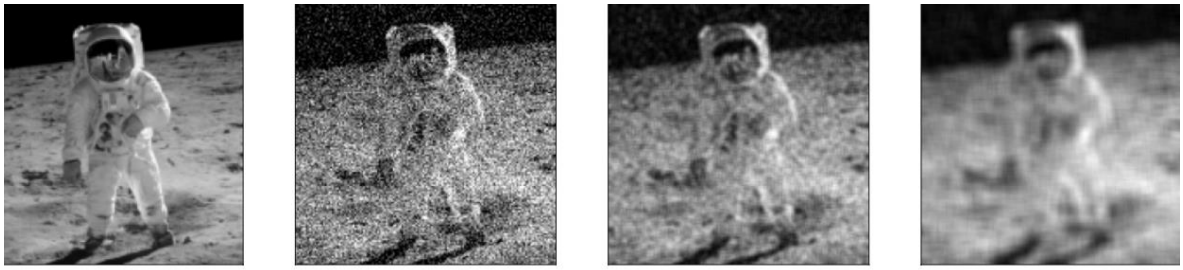
The importance of the pixel situated in the center of the convolution mask h or its 4-neighbors is occasionally heightened, as it offers a closer approximation to the characteristics of noise following a Gaussian probability distribution [13].

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Two widely employed smoothing filters feature coefficients that gradually diminish, approaching near-zero values at the edges of the window. This strategy effectively mitigates spurious oscillations in the frequency spectrum. The Gaussian and Butterworth filters are prominent examples. In the case of the Gaussian filter, larger convolution masks for averaging are generated based on the Gaussian distribution formula, with the mask coefficients normalized to yield a unit sum. Conversely, the Butterworth filter operates within the realm of local pre-processing in the frequency domain [15-16].

An illustrative example serves to elucidate the efficacy of noise suppression techniques. To highlight the discrete nature of the process, low-resolution images measuring 256 × 256 pixels were deliberately selected. Figure 3-4 (a) portrays the original image of Prague Castle. In contrast, Figure 3-4 (b) displays the same image overlaid with additive noise following a Gaussian distribution. Following the application of averaging with a 3 × 3 convolution mask, depicted in Figure 3-4 (c), noise is noticeably reduced, albeit at the expense of slight image blurring. Subsequently, Figure 3-5 (d) showcases the outcome of averaging with a larger mask size (7 × 7), resulting in significantly more pronounced blurring [17].

While filters of this nature may incur significant computational overhead, the computational burden is substantially alleviated in the crucial instance of separable filters. In the context of two-dimensional (2D) separability, the convolution kernel can be decomposed into a product of two one-dimensional vectors. Theory offers insights into identifying which convolution masks exhibit separability.



(a) (b) (c) (d)

Fig. 3-4- Image smoothing, a) original image, b) Gaussian noise, c) 3×3 averaging and d) 7×7 averaging [18]

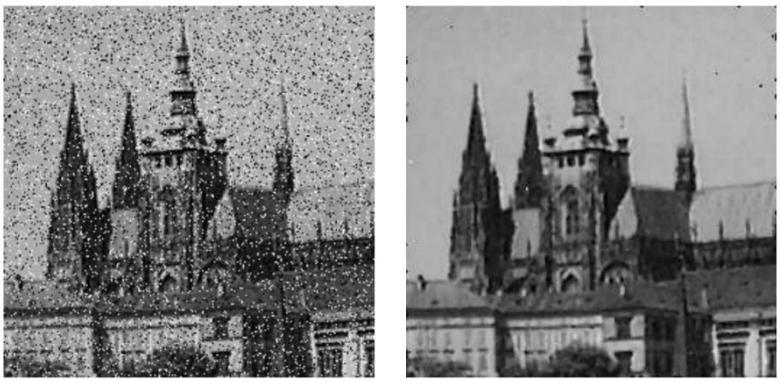
3.2.6 Median Filtering

In probability theory, the median delineates the boundary between the higher and lower halves of a probability distribution. For a random variable x, the median M represents the value at which the probability of x being less than M equals 0.5. Calculating the median of a finite list of real numbers involves arranging the list in ascending order and selecting the middle value. Lists are often crafted with an odd number of elements to ensure the uniqueness of the median.

Median filtering represents a non-linear smoothing technique aimed at mitigating edge blurring, wherein the central concept involves substituting the current pixel in the image with the median value derived from the brightness values within its local neighborhood. This method proves effective in eliminating impulse noise, as the median value within the neighborhood remains unaffected by individual noise spikes. Moreover, due to its minimal impact on edge sharpness, median filtering allows for iterative application. However, the computational complexity associated with sorting pixels within a potentially large rectangular window at each pixel position can be prohibitive. An alternative, more efficient approach involves recognizing that as the window shifts across a row by one column, the only alteration to its contents entails discarding the leftmost column and introducing a new right column [19].

The process of median filtering is depicted in Figure 3-5. An inherent limitation of median filtering within a rectangular neighborhood lies in its tendency to distort fine lines and sharp corners. This drawback can be circumvented by employing alternative neighborhood shapes. For instance, to preserve horizontal or vertical lines, an appropriate neighborhood configuration can be adopted. Median smoothing represents a specialized manifestation of broader rank filtering methodologies, which involve organizing pixels within a neighborhood into a sequential arrangement. Pre-processing outcomes are then derived through statistical analysis of this sequence, with the median serving as one potential measure. Additionally, variants such as the

maximum or minimum values within the sequence offer generalized extensions of dilation and erosion operations in images featuring a wider range of brightness values. This approach is known as order statistics (OS) filtering, wherein values within the neighborhood are once again ordered into a sequence, and a new value is determined as a linear combination of these ordered values [20].



(a) (b)

Fig. 3-5- Median filtering, a) impulse noise corrupted image, b) 3×3 filtering [13].

3-3 BINARIZATION

After completing the preprocessing steps, the images were binarized using MATLAB's ‘imbinarize’ function. Binarization is a critical process that converts grayscale images into binary images, consisting of only black and white pixels. This step simplifies the image data and highlights the regions of interest, which is essential for accurate lesion detection and analysis. Binarization decreases the complexity of the images by converting the grayscale intensity values into a binary format, aiding in easier and more efficient analysis. Region of Interest emphasizes the boundaries and shapes of lesions, making it easier to distinguish and analyze these regions. Additionally, binary images are crucial for subsequent processes like segmentation and edge detection, which rely on clear delineation of object boundaries.

The threshold determination is done using calibration color which involves several key steps. Initially, specific calibration colors representing the range of intensities present in the images are selected based on their ability to distinguish between the lesion and the surrounding skin. Subsequently, using these selected calibration colors, the threshold value for binarization is calculated through a predetermined linear equation, which involves analyzing the intensity values of the calibration colors to

determine an effective threshold that separates the lesion from the background. Finally, this calculated threshold is applied to the grayscale images using the 'imbinarize' function in MATLAB, which converts the grayscale images into binary images by setting pixels above the threshold to white (indicating the lesion) and pixels below the threshold to black (indicating the background).

3.4 LESION GEOMETRICAL AREA CALCULATION

In order to quantitatively analyze the lesions in the processed dermoscopic images, a series of calculations are performed to determine the portion and size of the lesions in millimeters which can be useful in comparison of lesion size over time. This comparison has not been performed in this study due to lack of related data. This step is crucial for assessing the extent of the lesions and for making meaningful medical interpretations.

Initially, Pixel Spacing Calculation that converts the pixel dimensions of the images to physical dimensions (millimeters), The pixel spacing is calculated based on the field of view (FOV) dimensions at 20x magnification. Assuming the FOV dimensions were 7 mm x 7 mm, the pixel spacing in both the x and y directions is determined by dividing the FOV dimensions by the original image dimensions (768x560 pixels). The pixel spacing for the original image dimensions (768x560 pixels) is calculated to determine the physical size each pixel represents in millimeters.

Next, the pixel spacing for the resized images (512x512 pixels) is adjusted to maintain accurate physical measurements. The adjusted pixel spacing for the resized images is calculated, ensuring that the physical dimensions are correctly mapped to the new pixel dimensions. The binary image is analyzed to calculate the number of white pixels, representing the lesion area, and the number of black pixels, representing the non-lesion area. The proportions of white and black pixels are calculated to understand the relative size of the lesion compared to the entire image area in which the proportion of white pixels in the binary image provides the percentage of the image occupied by the lesion and the proportion of black pixels represent the percentage of the image that does not contain the lesion. To convert the pixel area of the lesion to a physical area in square millimeters, the number of white pixels (lesion

area in pixels) is multiplied by the pixel spacing values in the x and y directions. The calculated lesion area in pixels is converted to square millimeters, providing a meaningful measurement of the lesion size in physical units. The code snippet provided in the annex helps to clarify the process.

3.5 SEGMENTATION

To perform more research and compare different techniques, another method named segmentation is utilized to detect the edges. Segmentation in biomedical image processing is a crucial step that involves partitioning an image into meaningful regions, which typically correspond to different anatomical structures, tissues, or areas of interest within the body. This process facilitates the detailed analysis of complex medical images, such as those obtained from MRI, CT, ultrasound, or microscopy, enabling more accurate diagnosis, treatment planning, and research [21].

The primary goal of segmentation is to isolate and delineate specific features within an image, such as organs, tumors, blood vessels, or other critical structures, allowing for quantitative analysis and better visualization. Segmentation techniques can be broadly categorized into several types based on their underlying methodologies. Thresholding is one of the techniques that involves setting a specific intensity value (threshold) to separate different regions. Pixels with intensity values above the threshold are classified as one region, while those below are classified as another. Thresholding is simple and fast but may be inadequate for complex images with overlapping intensity ranges [22-23]. Edge Detection is another method that involves techniques such as the Sobel, Canny, and Prewitt operators identifying the boundaries between different regions based on the changes in intensity. These methods are effective in highlighting structures but can be sensitive to noise [24-25]. Moreover, Region-Based Segmentation method groups pixels into regions based on predefined criteria such as intensity homogeneity [26]. Furthermore, Clustering algorithms like K-means and Gaussian Mixture Models (GMM) segment the image by grouping pixels with similar characteristics. These methods are effective for distinguishing between multiple regions but may require prior knowledge about the number of clusters [27]. In addition, recent advances in machine learning, particularly deep learning, have revolutionized image segmentation. Convolutional Neural Networks (CNNs) and fully convolutional networks (FCNs) can learn complex features from annotated training data, providing highly accurate and automated segmentation. Examples include U-Net and Mask R-CNN, which are widely used in biomedical

applications for their robustness and precision [28]. Each of these methods has its strengths and limitations, and the choice of technique often depends on the specific characteristics of the biomedical images and the segmentation task at hand. Combining multiple methods or using hybrid approaches can also enhance segmentation accuracy and robustness. In this study, Edge Detection is utilized as selected method to perform the segmentation process.

3.5.1 Edge Detection

Edge detection algorithms comprise a critical set of localized image pre-processing techniques employed to identify alterations in the intensity function, with edges representing pixels where this function, typically brightness, undergoes abrupt changes. Neurological and psychophysical investigations indicate that regions within the image characterized by abrupt changes in function values play a pivotal role in image perception. Notably, edges demonstrate a certain degree of invariance to variations in illumination and viewpoint. Focusing solely on edge elements with pronounced magnitude, termed edges, often yields adequate information for comprehending the image. This process offers a notable advantage in significantly reducing the volume of image data. However, such reduction does not compromise the interpretative understanding of image content in numerous instances. Edge detection facilitates a pertinent abstraction of the image data; for example, line drawings exemplify such abstraction.

The physical phenomena inherent in the image formation process is explored to give rise to sudden alterations in image values, as illustrated in Figure 3-6. Differential calculus serves as a framework for elucidating changes in continuous functions, whereby an image function, contingent upon two variables denoting coordinates in the image plane, necessitates operators articulated through partial derivatives to characterize edges. A modification in the image function is delineated by a gradient vector, which denotes the direction of maximum ascension within the image function.

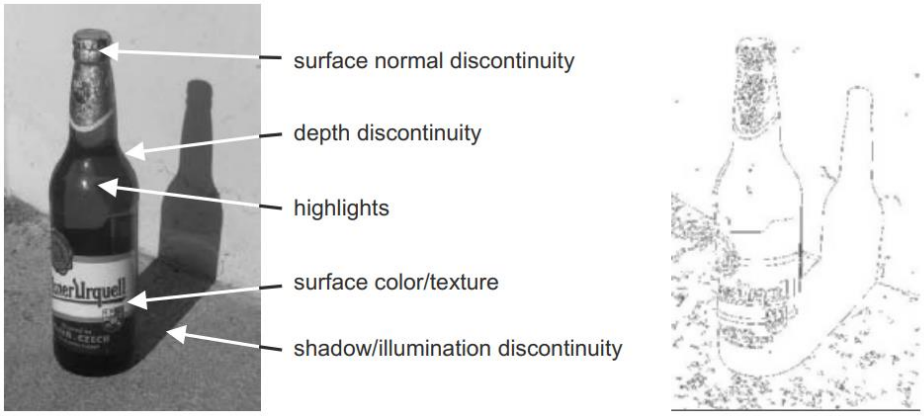


Fig. 3-6- Edge detection procedure [13].

An edge represents an attribute associated with an individual pixel, derived from the behavior of the image function within the vicinity of that pixel. It is characterized as a vector variable encompassing two components: magnitude and direction. The magnitude of the edge corresponds to the magnitude of the gradient, while the edge direction, denoted as φ, is orthogonal to the gradient direction, ψ, by a rotation of -90 degrees. The gradient direction signifies the direction of maximal change in the function, such as the transition from black (f(i, j) = 0) to white (f(i, j) = 255). This concept is elucidated in Figure 3-7, wherein closed lines delineate regions of uniform brightness. The orientation of 0 degrees points towards the east [18].

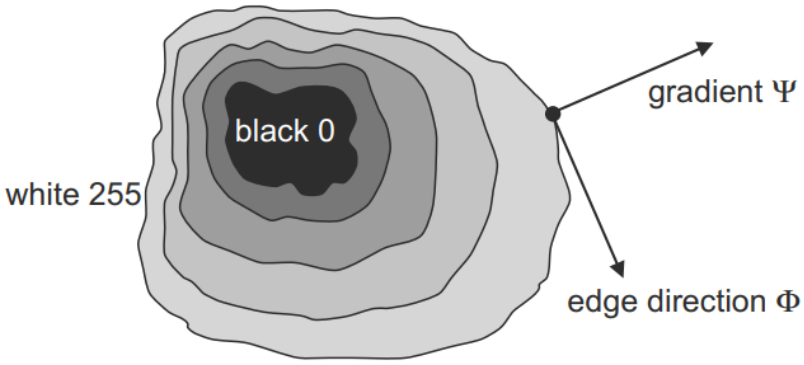


Fig. 3-7- Gradient and edge direction definition [13].

Edges serve as prominent tools in image analysis for delineating boundaries between regions. When a region exhibits homogeneous brightness, its boundary coincides with pixels where variations occur in the image function. In an ideal scenario devoid of noise, these boundary pixels typically demonstrate high edge magnitudes. Notably, the boundary and its constituent segments (edges) exhibit perpendicularity with respect to the gradient direction. Figure 3-8 provides illustrations of various standard edge profiles. Edge detection algorithms are commonly tailored to detect specific types of edge profiles.

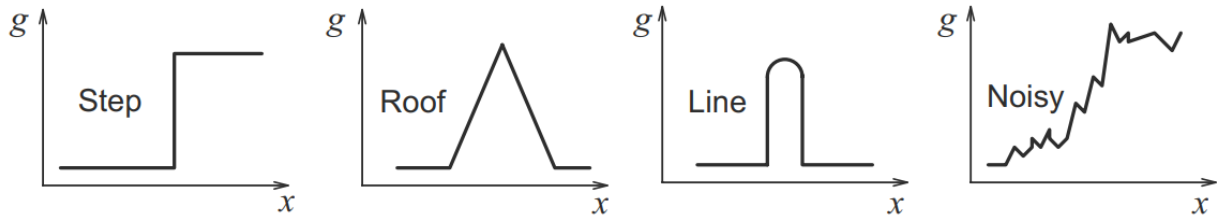


Fig. 3-8- Edge profiles [13].

The continuous functions of gradient magnitude and gradient direction are computed as follows [13]:

$$\left|\operatorname{grad} g(x,y)\right| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2},$$

$$\psi = \arg\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right),$$

The term arg(x, y) represents the angle (measured in radians) from the x-axis to the point (x, y). In certain scenarios, there is a focus solely on edge magnitudes, disregarding their orientations. In such cases, a linear differential operator known as the Laplacian is employed. The Laplacian possesses uniform properties across all directions and, as a consequence, remains unchanged under rotation. Its formulation is expressed as [13]:

$$\nabla^2 g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2}$$

The aim of image sharpening is to enhance the steepness of edges, with the resultant sharpened image intended for human observation. The sharpened output image, denoted as f, is derived from the input image g through the following process [13]:

$$f(i,j) = g(i,j) - C\,S(i,j)$$

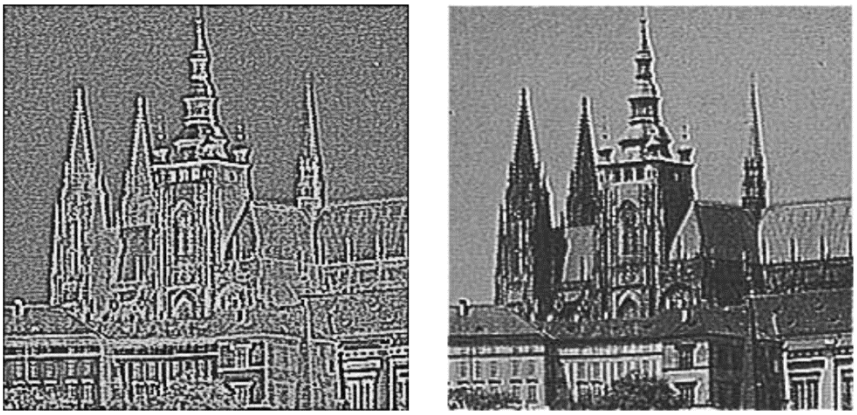
In the given equation, C represents a positive coefficient that signifies the degree of sharpening, while S(i, j) denotes a measure of the sharpness of the image function, computed utilizing a gradient operator. The Laplacian operator is frequently employed for this purpose, as depicted in Figure 3-9, which provides an illustrative example of image sharpening utilizing the Laplacian. Additionally, image sharpening can be interpreted within the frequency domain. As known, the Fourier transform outcome comprises a blend of harmonic functions [14]. The derivative of the sinusoidal function sin(nx) yields ncos(nx); thus, higher frequencies correspond to larger magnitudes of their derivatives. To execute image sharpening, a signal

proportional to an unsharp image, such as one significantly blurred by a smoothing operator, is subtracted from the original image. Given the discrete nature of digital images, approximation via differences is imperative. The first-order differences of the image g in the vertical direction (with i fixed) and horizontal direction (with j fixed) are articulated as follows [13]:

$$\Delta_i\, g(i,j) = g(i,j) - g(i-n,j)\ ,$$
$$\Delta_j\, g(i,j) = g(i,j) - g(i,j-n)\ ,$$

Here, n denotes a diminutive integer, commonly designated as 1. The selection of n should be such that it remains sufficiently small to offer a precise approximation to the derivative, yet adequately large to disregard negligible alterations in the image function. Symmetric formulations for these differences are sought [15]:

$$\Delta_i\, g(i,j) = g(i+n,j) - g(i-n,j)\ ,$$
$$\Delta_j\, g(i,j) = g(i,j+n) - g(i,j-n)\ ,$$



(a) (b)

Fig. 3-9- Laplace operator, a) laplace edged image, b) sharpened image using laplace operator [13].

Gradient operators, serving as a metric for the sharpness of edges, are categorized into three main groups. First, the operators that estimate derivatives of the image function through differences. Some of these operators exhibit rotational invariance (e.g., Laplacian), thereby necessitating the utilization of a single convolution mask. Conversely, others, approximating first derivatives, employ multiple masks. The orientation is determined based on the optimal alignment with various basic patterns. Second, the operators predicated on the zero-crossings of the second derivative of the image function (e.g., Marr-Hildreth or Canny edge detectors). Third, the operators that endeavor to align an image function with a parametric model of edges.

Edge detection represents a pivotal stage enabling sophisticated image analysis and continues to be a subject of ongoing investigation. Various methodologies showcased in contemporary literature encompass fuzzy logic, neural networks, and wavelets. Selecting the optimal edge detection strategy may pose challenges due to the diverse array of available approaches. Consequently, two famous methods SOBEL and CANNY are selected for this purpose.

3.5.1.1 SOBEL Operator

The Sobel operator is frequently employed as a straightforward method to detect the horizontal and vertical orientation of edges, primarily utilizing masks h1 and h3 [30].

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \dots$$

Upon obtaining the responses y and x from masks h1 and h3 respectively, the determination of edge strength (magnitude) can be formulated as follows [30] in which direction is arctan(y/x).

$$\sqrt{x^2 + y^2} \qquad \text{or} \qquad |x| + |y|$$

3.5.1.2 CANNY Edge Detection

Canny introduced a method for edge detection that is deemed optimal for identifying step edges affected by white noise. The optimality of this detector is predicated on three criteria [31]: First, the detection criterion underscores the necessity of capturing significant edges accurately while minimizing spurious responses. Second, the localization criterion emphasizes the importance of minimizing the discrepancy between the actual position of the edge and its detected location. Third, the single response criterion aims to mitigate multiple responses to a solitary edge. This aspect is partially encompassed by the first criterion, as in instances where two responses occur for a single edge, one must be regarded as erroneous. The third criterion addresses the challenge posed by noisy edges and counters the effects of non-smooth edge operators.

Canny approach is underpinned by several key concepts [32]. Initially, the edge detector was formulated for a one-dimensional signal and the first two optimality criteria were addressed. A solution was attained through closed-form expressions derived via the calculus of variations. Incorporating the third criterion pertaining to multiple responses necessitates numerical optimization to identify the optimal solution. The resultant filter can be accurately approximated, with an error margin of less than 20%, using the first derivative of a Gaussian smoothing filter with a specific standard deviation. This approach is favored due to the availability of efficient implementation methods. Subsequently, the detector is extended to operate in two dimensions. A step edge is characterized by its position, orientation, and potentially its magnitude. It can be demonstrated that convolving an image with a symmetric two-dimensional Gaussian filter and subsequently differentiating in the direction of the gradient (perpendicular to the edge direction) yields a straightforward and efficient directional operator.

Let G denote a two-dimensional Gaussian function, and consider the scenario where there is a desire to perform convolution of the image with an operator G_n, which represents the first derivative of G in a specified direction denoted as n [32]:

$$G_n = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \nabla G$$

The objective is to ensure that n is orthogonal to the edge; however, this direction is not predetermined. Instead, it is inferred robustly based on the smoothed gradient direction. Given an image f, the estimation of the normal to the edge, denoted as n, is computed as follows [32]:

$$\mathbf{n} = \frac{\nabla (G * f)}{|\nabla (G * f)|}$$

Subsequently, the edge position is identified as the local maximum attained by convolving the image f with the operator G_n in the direction of n [32]:

$$\frac{\partial}{\partial \mathbf{n}} G_n * f = 0$$

This equation elucidates the process of identifying local maxima in the direction orthogonal to the edge, a procedure commonly known as non-maximal suppression.

Given the associativity of convolution and differentiation, the initial step involves convolving an image f with a symmetric Gaussian function G, followed by the computation of the directional second derivative using an estimation of the direction

n obtained. The determination of the edge's strength, represented by the magnitude of the gradient of the image intensity function f, is then evaluated as [32]:

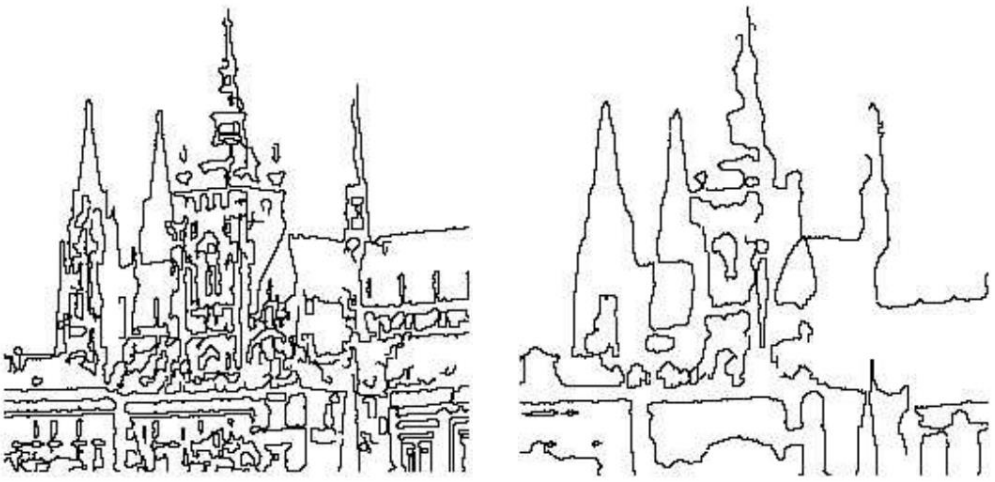
$$\left|G_n * f\right| = \left|\nabla(G * f)\right|$$

False positives, stemming from noise-induced spurious responses to individual edges, often manifest as a prevalent issue in edge detection methodologies. These false detections, commonly known as 'streaking,' tend to fragment edge contours when the operator fluctuates above and below the threshold during the edge detection process. To mitigate streaking, a thresholding mechanism employing hysteresis is employed, incorporating both a stringent (higher) threshold and a lenient (lower) threshold. The determination of these thresholds is guided by an estimated signal-to-noise ratio.

The optimal scale for the operator is contingent upon the objects depicted in the image. Addressing this uncertainty necessitates the adoption of multiple scales and the amalgamation of information garnered from them. In the context of the Canny detector, diverse scales are characterized by varying standard deviations σ of the Gaussian functions. Notably, there may exist multiple scales of operators that elicit significant responses to edges (i.e., signal-to-noise ratio surpasses the threshold). In such scenarios, preference is accorded to the operator with the smallest scale, as it affords superior localization of the edge.

Canny introduced a methodology centered on feature synthesis. Initially, all prominent edges detected by the operator with the minimum scale are delineated. Subsequently, the edges corresponding to an imaginary operator with a larger standard deviation σ are synthesized based on the previously identified edges. The synthesized edge response is then juxtaposed with the authentic edge response for the larger σ. Any additional edges are annotated solely if their response significantly exceeds that projected from the synthesized output. This iterative process can be reiterated across a succession of scales, facilitating the construction of a cumulative edge map by aggregating those edges that remained undetected at smaller scales.

Figure 3-10 (a) illustrates the edges extracted through the implementation of a Canny operator with a standard deviation of 1.0. In contrast, Figure 3-10 (b) depicts the response of the edge detector corresponding to a standard deviation of 2.8. Canny detector constitutes a sophisticated yet substantial advancement in the field of edge detection.



(a) (b)

Fig. 3-10- Canny edge detection, a) σ = 1.0, b) σ = 2.8 [13].

3.6 METRICS

3.6.1 Dice Coefficient

The Dice coefficient serves as a frequently employed measure in medical image analysis for assessing the similarity between two sets of segmented regions. These regions commonly include a predicted segmentation, generated by algorithms, and a ground truth annotation. A Dice coefficient value of 1 indicates a perfect overlap between the predicted and ground truth segmentations, while a value of 0 indicates no overlap. The coefficient is mathematically defined as:

$$Dice\ Coefficient\ = \frac{2 \times |A \cap B|}{|A| + |B|}$$

where:

- A represents the set of pixels in the predicted segmentation.
- B represents the set of pixels in the ground truth segmentation.
- |A∩B| is the number of pixels that are correctly identified as edges in both images.
- $|A| + |B|$ are the total number of edge pixels in A and B, respectively.

In the context of this study, ROIs are specific areas of interest within the skin images, such as lesions. The accuracy of the segmentation algorithm in identifying

these ROIs is critical for reliable analysis and diagnosis. The Dice coefficient is employed to evaluate this accuracy by comparing the segmented regions produced by the algorithm to the manually annotated ground truth ROIs.

3.6.2 Standard Deviation

Standard deviation is used to assess how each segmented image deviates from the average Dice coefficient, thereby evaluating the consistency and robustness of our pipeline. Standard deviation is a measure of the amount of variation or dispersion in a set of values. It quantifies how spread out the values in a dataset are around the mean (average) of the dataset. A low standard deviation indicates that the values are close to the mean, while a high standard deviation indicates that the values are spread out over a wider range.

For a dataset $x_1, x_2, \ldots x_N$ with N values, the standard deviation σ is calculated using the following formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

where:

- x_i are the individual values in the dataset
- μ is the mean of the dataset, calculated as $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
- N is the number of values in the dataset.

CHAPTER 4
RESULTS AND DISCUSSION

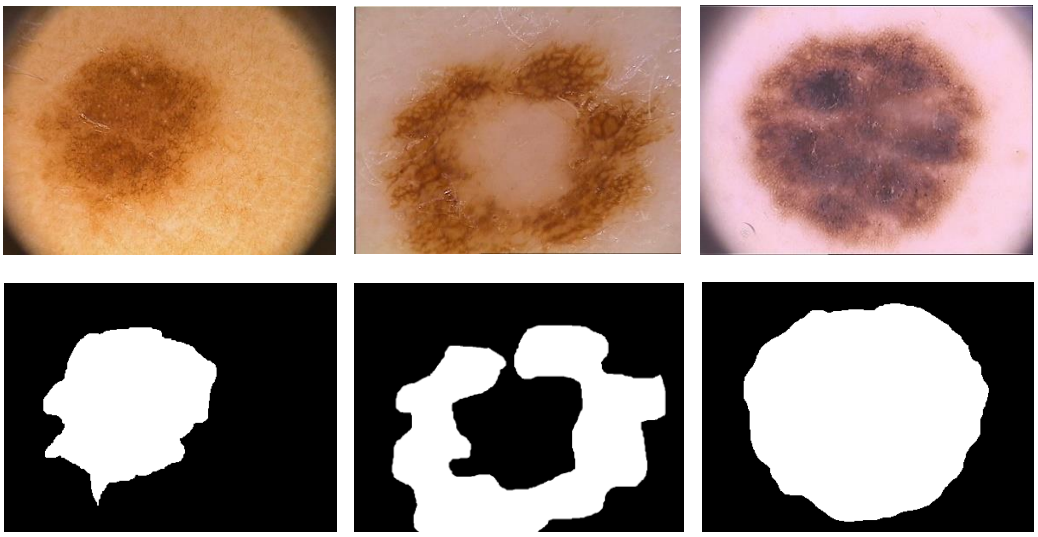
In this chapter, we present the results of our dermoscopic image segmentation study, focusing on the preprocessing steps, segmentation methods, and performance evaluation. The primary objective of this study is to develop a robust pipeline for accurately segmenting lesions in dermoscopic images. The pipeline comprises several stages, including image preprocessing, noise removal, and the application of various segmentation techniques. We evaluate the effectiveness of these techniques using the Dice coefficient to compare the segmentation results with ground truth data.

4.1 DATASET PREPARATION

In the initial phase of our study, we focused on assembling and preparing a complete dataset of dermoscopic images. This dataset is introductory for developing and evaluating our lesion segmentation pipeline. Below is a detailed explanation of the preprocessing steps undertaken to ensure the consistency and reliability of the results. We begin by collecting a dataset comprising 200 high-resolution dermoscopic images. These images are paired with their corresponding created ground truth annotations that are prepared manually, providing a precise delineation of lesion areas. The original dimensions of these images were 768x560 pixels. These high-resolution images are chosen to capture fine details essential for accurate lesion segmentation. Figure 4-1 shows samples of three different lesion categories with their ground truth.

To standardize the input dimensions across all images and facilitate more efficient processing, each image in the dataset was resized from 768x560 pixels to 512x512 pixels. This resizing process was carried out using MATLAB's 'imresize' function, which ensures that the aspect ratio and essential features of the images are preserved during the resizing. Each resized image was paired with its corresponding resized ground truth annotation. This step ensures that the segmentation model can be evaluated based on accurately labeled data. The ground truth annotations were

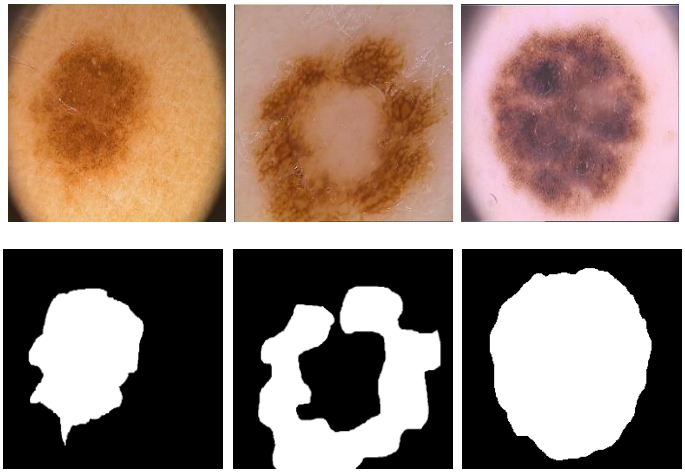
similarly resized to match the dimensions of the resized images, ensuring precise alignment between the image data and the annotated lesion areas.



(a) (b) (c)

Fig. 4-1- Different types of lesions:
Row 1) Original Image, 2) Ground Truth
Column a) Common Nevo, b) Atypical Nevo and c) Melanoma

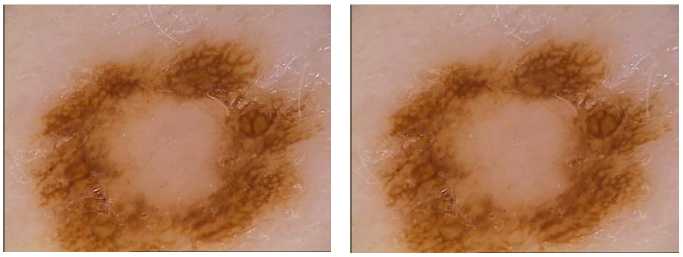
By resizing the original dermoscopic images and their ground truth annotations to a uniform size of 512x512 pixels, we establish a consistent and reliable dataset as described in the previous chapter. This preprocessing step is foundational for the subsequent stages of our lesion segmentation pipeline, ensuring that our models are evaluated on standardized data. This standardization is expected to enhance the accuracy and generalizability of our segmentation results, ultimately contributing to the effectiveness of our proposed pipeline in medical applications (Fig. 4-2). It should be noted that these three original images from different types of lesions are selected among 200 images of the dataset as the base images to visualize the results of the next steps but the whole processed is applied on all 200 images.



(a) (b) (c)

Fig. 4-2- Different types of lesions:
Row 1) Original Resized Image and 3) Resized Ground Truth
Column a) Common Nevo, b) Atypical Nevo and c) Melanoma

To assess the spatial resolution quality of the resized image, the SSIM function is used. This involves restoring the resized image to its original dimensions and comparing it with the original. A value close to one indicates a high similarity to the original image, reflecting better preservation of spatial resolution. It is evident that by achieving the SSIM value approximately 0.995 for each of 200 images, there exists a high degree of similarity between the original image and the resized version indicating that spatial resolution is preserved acceptably (Fig. 4-3).



(a) (b)

Fig. 4-3- Quality comparison between a) original and b) recovered images

4.2 IMAGE PREPROCESSING

4.2.1 Enhanced Grayscale Image Preparation

Following the initial resizing of the dermoscopic images, the next step involved converting the resized images to grayscale to assess their quality since converting images to grayscale simplifies the data, reduces computational load, focuses on intensity information. This conversion was performed using MATLAB's `'rgb2gray'` function. However, the resulting grayscale images did not meet the desired quality standards for accurate lesion segmentation. To address this, we implemented a series of preprocessing enhancements as detailed below.

4.2.1.1 Center Cropping

To improve the focus on the lesion areas and enhance the quality of the images, we applied a center cropping technique. The target size for cropping function was set to [560, 620] pixels, ensuring that the most relevant central part of the image was retained. This step helps to eliminate any extraneous background details that might interfere with the segmentation process.

4.2.1.2 Histogram Equalization for Enhanced Grayscale Transformation

To further enhance the grayscale images and improve their quality, we employed the Adaptive Histogram Equalization (AHE) method. This technique enhances the contrast of the images by redistributing the lightness values more evenly across the image, making the features more distinguishable and aiding in better segmentation of the lesions. The final images are shown in the figure 4-4.

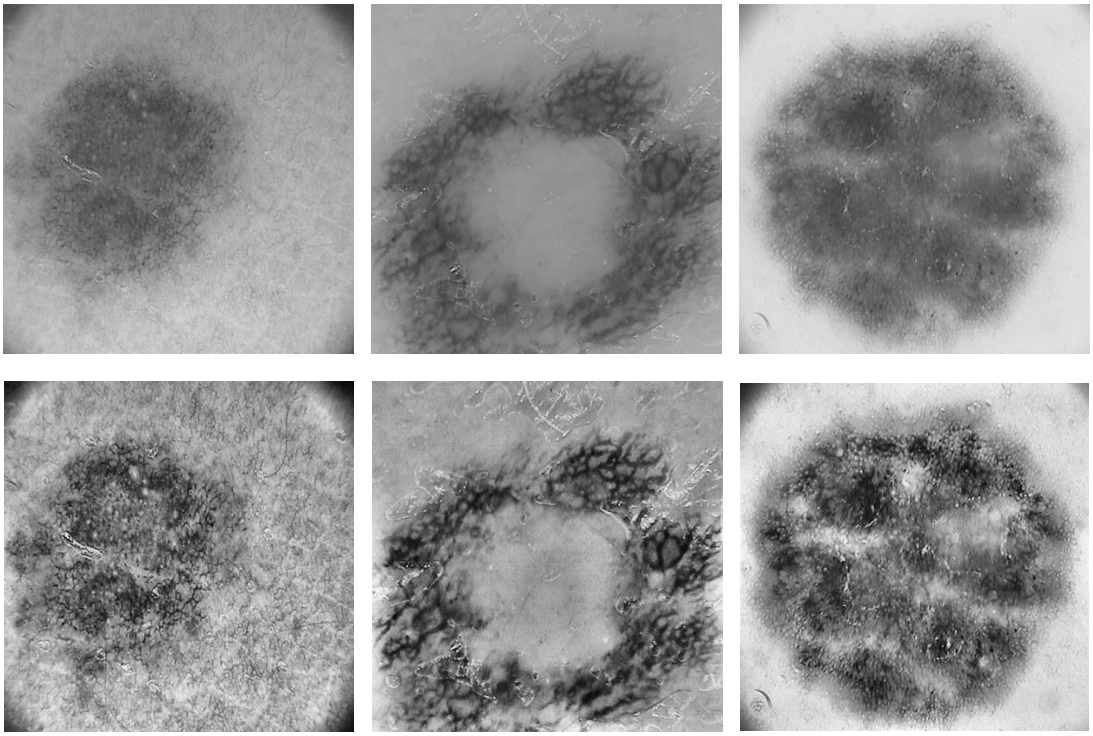


Fig. 4-4- Different grayscale images:
Row 1) original grayscale and 2) enhanced grayscale

4.2.2 Noise Removal Using Masking Technique

After enhancing the grayscale images, the next crucial step was to remove noise artifacts, such as hair, that could interfere with the segmentation process. To achieve this, we implemented a masking technique that utilized a calibrated threshold for binarization. The process involved the following steps:

4.2.2.1 Threshold Determination Using Calibration Color

To determine an appropriate threshold for binarizing the images, calibration color approach is utilized. The calibration color is identified as the most frequently occurring pixel value in the grayscale image. This value served as a reference point for setting the sensitivity of the binarization process. The grayscale image is analyzed to find the pixel value that appeared most frequently. This value, provides a basis for calibrating the binarization threshold. The calibration color is normalized to a [0, 1] range, assuming grayscale values ranged from 0 to 255. A linear equation is then used to relate the final sensitivity to the calibration color, with predefined lower and upper

sensitivity bounds of 0.4 and 0.8, respectively. This linear mapping ensures that higher calibration colors corresponded to higher sensitivities, allowing for more adaptive thresholding based on the image's specific characteristics.

4.2.2.2 Mask Production

To prepare the images for noise removal, adaptive histogram equalization is used previously to enhance the contrast of the grayscale images further. Following this enhancement, morphological operations and binarization techniques are used to create a binary mask that would isolate noise elements such as hair (Fig 4-5). A top-hat filter is applied to the enhanced grayscale images using a structural element (disk-shaped) to emphasize small bright regions on a dark background, such as hair. The top-hat filtered image is then binarized using an adaptive thresholding method, with the sensitivity parameter determined from the calibration color. This step produces a binary mask highlighting the noise regions. To refine the binary mask, a morphological closing operation is performed using a smaller disk-shaped structural element. This operation helped to close small gaps and connect disjointed components within the mask.

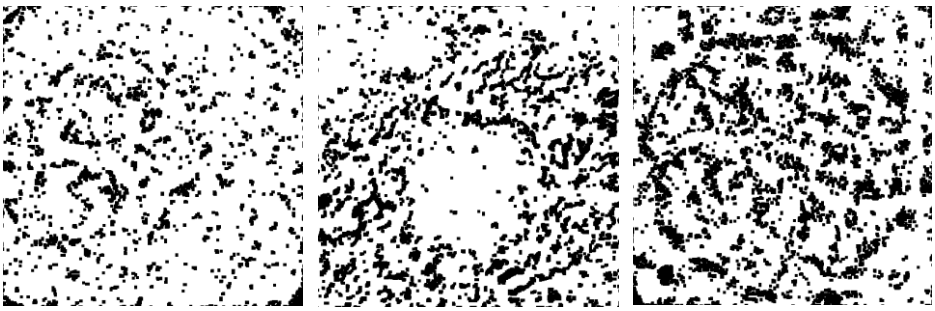


Fig. 4-5- Masks for different types of images

4.2.2.3 Noise Removal

The final step involved applying the binary mask to the original image to remove the identified noise elements. This is achieved through `'inpainting'`, a technique that fills in the regions marked by the mask with surrounding pixel values to seamlessly remove unwanted artifacts. The `'inpainting'` process utilized the binary mask to coherently replace the noise regions with pixels from the surrounding areas, effectively removing elements like hair from the image while preserving the integrity of the lesion areas.

The described process effectively enhances the quality of the images and removes noise artifacts, significantly improving the subsequent lesion segmentation. By calibrating the binarization threshold based on the most frequently occurring grayscale value and applying a series of morphological and inpainting operations, we ensured that the images were free from extraneous noise, leading to more accurate and reliable segmentation results. This comprehensive noise removal strategy is a critical component of our lesion segmentation pipeline, contributing to the overall efficacy of the proposed method. Figure 4-6 shows that noise such as hair is removed from original image but there is a need to apply more de-noising process to improve the quality which are perform on the next steps.

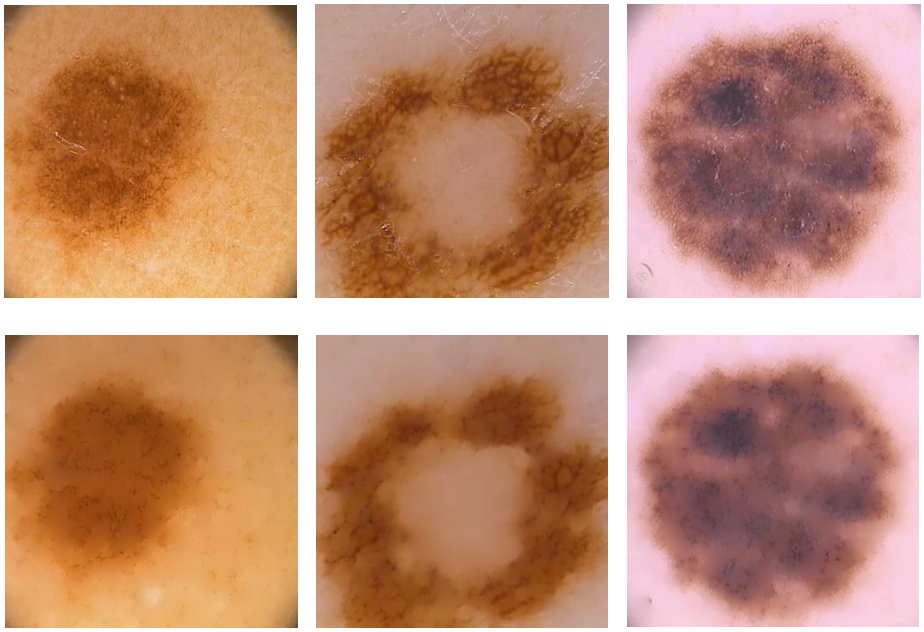


Fig. 4-6- RGB images after masking process
Row: 1) original, 2) de-noised images

4.2.3 Additional Noise Reduction: Smoothing and Median Filtering

To further enhance the quality of the grayscale images and ensure the removal of any residual noise, additional filtering techniques are applied. These techniques are including Gaussian smoothing and median filtering, which help to refine the images and prepare them for accurate lesion segmentation. Gaussian smoothing is a technique used to reduce image noise and detail by averaging the pixel values within a neighborhood defined by a Gaussian kernel. This method effectively blurs the image, smoothing out sharp transitions and reducing the impact of high-frequency noise. A

Gaussian filter with a specified standard deviation (sigma) is applied to the preprocessed images. The choice of sigma value is determined through experimentation to balance noise reduction with the preservation of essential image features. The smoothed image exhibited reduced noise, making the lesion boundaries more discernible. Following Gaussian smoothing, median filtering is applied to further enhance the image quality. Median filtering is a nonlinear process that replaces each pixel value in the image with the median value of the neighboring pixels. This technique is particularly effective at removing salt-and-pepper noise, which can appear as random bright or dark spots in the image. The median filter is applied to the smoothed images, utilizing a window size determined through experimentation. This filter effectively removes any remaining isolated noise artifacts without significantly blurring the image, thereby preserving the edges and details of the lesions.

The application of Gaussian smoothing and median filtering provide an additional layer of noise reduction, resulting in cleaner and more refined grayscale images. Gaussian smoothing helps to blur out high-frequency noise, while median filtering effectively removes isolated noise artifacts. These filtering techniques ensure that the images used for lesion segmentation are of the highest quality, thereby enhancing the accuracy and reliability of the segmentation results. This comprehensive noise reduction process is an essential step in preparing the images for the subsequent stages of the segmentation pipeline (Fig. 4-7).

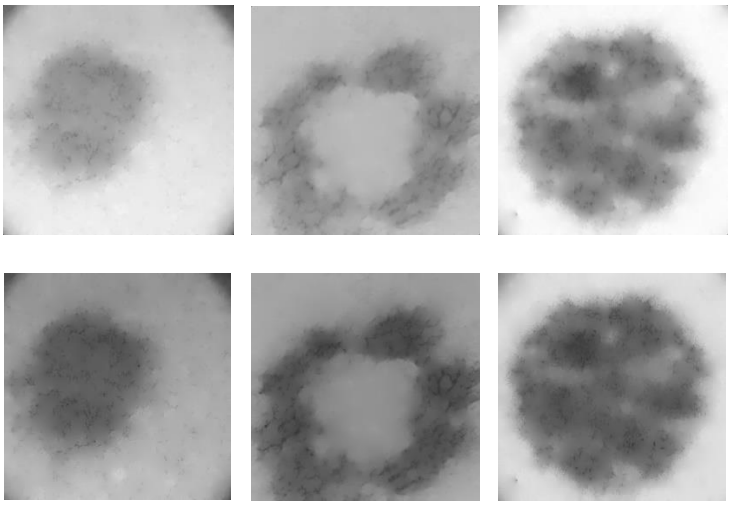


Fig. 4-7- Grayscale comparison
Row: 1) grayscale before filtering steps, 2) grayscale images after filtering steps

4.3 LESION GEOMETRICAL AREA CALCULATION

To quantitatively analyze lesions in processed dermoscopic images, calculations are performed to determine lesion size and proportion in millimeters, aiding in comparisons over time. Initially, pixel spacing is calculated based on the field of view (FOV) dimensions at 20x magnification, with an assumed FOV of 7 mm x 7 mm. For original image dimensions (768x560 pixels), pixel spacing is derived by dividing the FOV dimensions by the image dimensions. This spacing is then adjusted for resized images (512x512 pixels) to ensure accurate physical measurements. In the binary image, white pixels represent the lesion area and black pixels the non-lesion area. The proportions of white and black pixels are calculated to understand the lesion's relative size within the entire image. The lesion area in pixels is converted to a physical area in square millimeters by multiplying the number of white pixels by the pixel spacing values in the x and y directions, providing a meaningful measurement of the lesion size. One numerical example is as follows:

Pixel spacing in X direction: 0.0091 mm/pixel

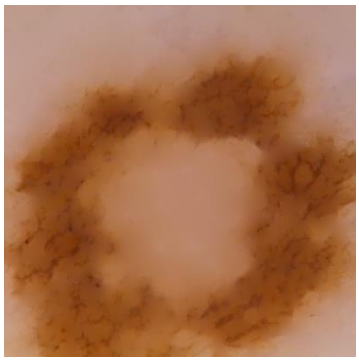
Pixel spacing in Y direction: 0.0125 mm/pixel

Proportion of lesion (white pixels): 0.4969

Proportion of non-lesion (black pixels): 0.5031

Lesion area in pixels: 32568

Lesion area in square millimeters: 6.0876 mm^2



This analytical step provides a detailed quantitative assessment of the lesion areas in the dermoscopic images. By calculating the pixel spacing based on the field of view dimensions and adjusting for image resizing, accurate physical measurements are ensured. These calculations are essential for evaluating the extent of the lesions and have significant implications for medical analysis and decision-making in dermatological assessments.

4.4 SEGMENTATION PROCESS

After preparing the images and ensuring their quality, we proceed with the segmentation of the lesions using three distinct methods. Each method is chosen for its unique approach to isolating the lesion areas, and their performance is evaluated to determine the most effective technique for our dataset.

4.4.1 Binarization Method

The first method involves a straightforward binarization of the preprocessed grayscale images. Binarization is a technique that converts the image into a binary image, where the pixels are classified into two categories: lesion (white) and non-lesion (black). Using MATLAB's 'imbinarize' function, the smoothed grayscale images are binarized. This method relies on adaptive thresholding to distinguish the lesion area from the background. The binary image is then inverted to ensure that the lesion areas were represented by white pixels, making it easier to identify and analyze the lesions.

4.4.2. Canny Edge Detection Method

The second method utilizes the Canny edge detection algorithm, which is known for its ability to detect a wide range of edges in images. This method focuses on identifying the boundaries of the lesions by detecting edges. The Canny edge detection algorithm is applied to the processed grayscale images to detect the edges of the lesions. Different thresholds are tested, and the default thresholds are found to be most effective. To ensure that the detected edges formed a continuous boundary around the lesion, the edges are filled using the 'imfill' function. The filled edges are then dilated using a disk-shaped structural element to enhance the edge continuity and close any gaps.

4.4.3 Sobel Edge Detection Method

The third method employs the Sobel edge detection algorithm, which uses gradient-based edge detection to highlight the boundaries of the lesions. The Sobel edge detection algorithm is applied to the grayscale images, and similar to the Canny method, different thresholds are tested before finalizing the default thresholds. The detected edges are then filled to create a solid boundary around the lesions. A disk-

shaped structural element is used to dilate the edges, improving the continuity and robustness of the lesion boundaries.

4.5 METHOD COMPARISON

These three segmentation methods, binarization, Canny edge detection, and Sobel edge detection, are employed to isolate the lesion areas in the dermoscopic images. Each method brings unique strengths to the segmentation process. Binarization provides a simple and effective way to classify the lesion and non-lesion areas, suitable for images with distinct contrast. Canny Edge Detection offers edge detection capabilities, particularly effective for images where lesion boundaries are well-defined but requires filling and dilation to ensure completeness. Sobel Edge Detection leverages gradient information to detect edges, performing well in highlighting the boundaries of lesions with smooth gradient transitions.

By employing and comparing these methods, we ensure a comprehensive analysis of the lesion segmentation process, ultimately selecting the most effective technique for our specific dataset. The results from each method provides valuable insights into the strengths and limitations of different segmentation approaches, contributing to the overall robustness and accuracy of our lesion segmentation pipeline Fig. 4-8.

To quantitatively evaluate the performance of the segmentation methods, the Dice coefficient is utilized. Dice coefficient is a statistical measure that assesses the similarity between the predicted segmentation and the ground truth. The Dice coefficient ranges from 0 to 1, where 1 indicates perfect agreement. The Dice coefficients are calculated for each of the 200 images segmented by the three methods of binarization, Canny edge detection, and Sobel edge detection. Additionally, the standard deviation of the Dice coefficients is computed to assess the robustness of each method. Table 4-1 summarizes the results of mentioned methods.

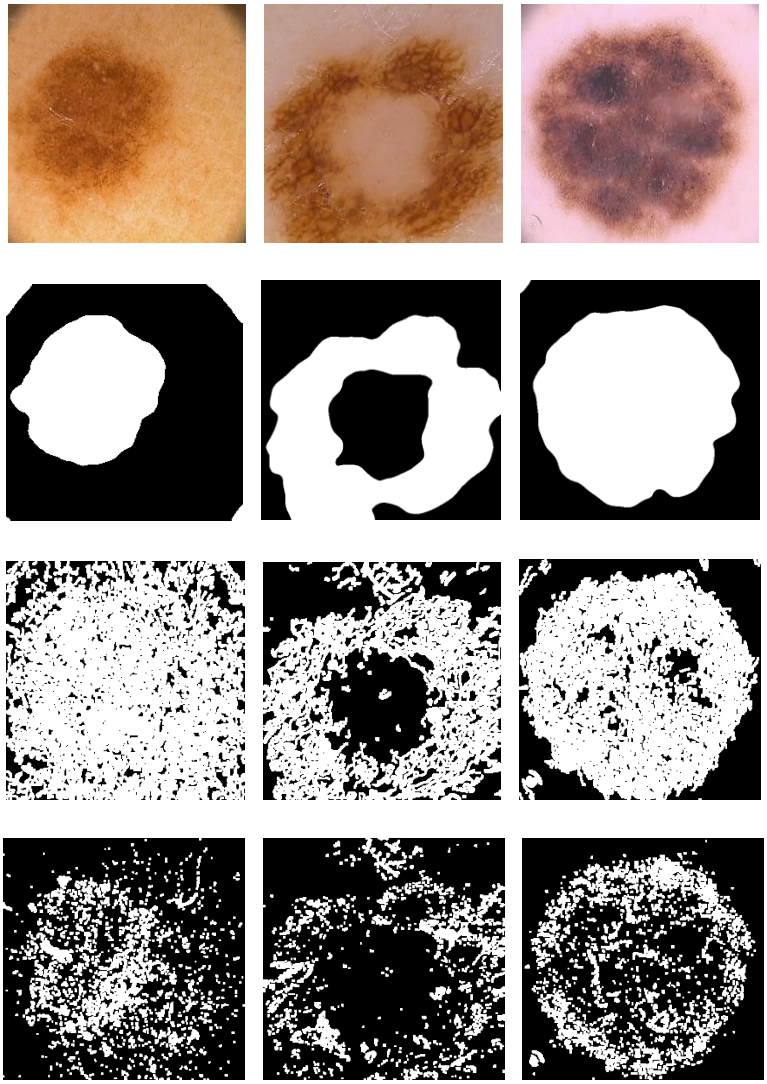


Fig. 4-8 Segmentation results:
Row 1) original image, 2) binarization, 3) CANNY, 6) SOBEL

The table 4-1 compares three image processing methods—Binarization, Canny Edge Detection, and Sobel Edge Detection—based on their performance in detecting skin lesions. Binarization shows the best performance with high average Dice coefficients of 0.785435 for Common Nevus, 0.816436 for Atypical Nevus, and 0.836189 for Melanoma, coupled with low standard deviations of 0.128416, 0.107542, and 0.104424 respectively, indicating both high accuracy and consistency. In contrast, Canny Edge Detection has moderate performance and higher standard deviations showing more variability, and Sobel Edge Detection performs the worst. Notably, Binarization achieves the highest Dice coefficient for Melanoma, highlighting its superior effectiveness in detecting this type of lesion. Figure 4-9 visualizes these numerical values in separated plots for better comparison.

Method	Average Dice Coefficient			Standard Deviation		
	Common Nevo	*Atypical Nevo*	*Melanoma*	*Common Nevo*	*Atypical Nevo*	*Melanoma*
Binarization	0.785435	0.816436	0.836189	0.128416	0.107542	0.104424
Canny Edge Detection	0.399111	0.449955	0.638348	0.194785	0.193168	0.187353
Sobel Edge Detection	0.311489	0.355794	0.440911	0.148954	0.141307	0.135781

Table. 4-1 Result comparison of different methods for each lesion type

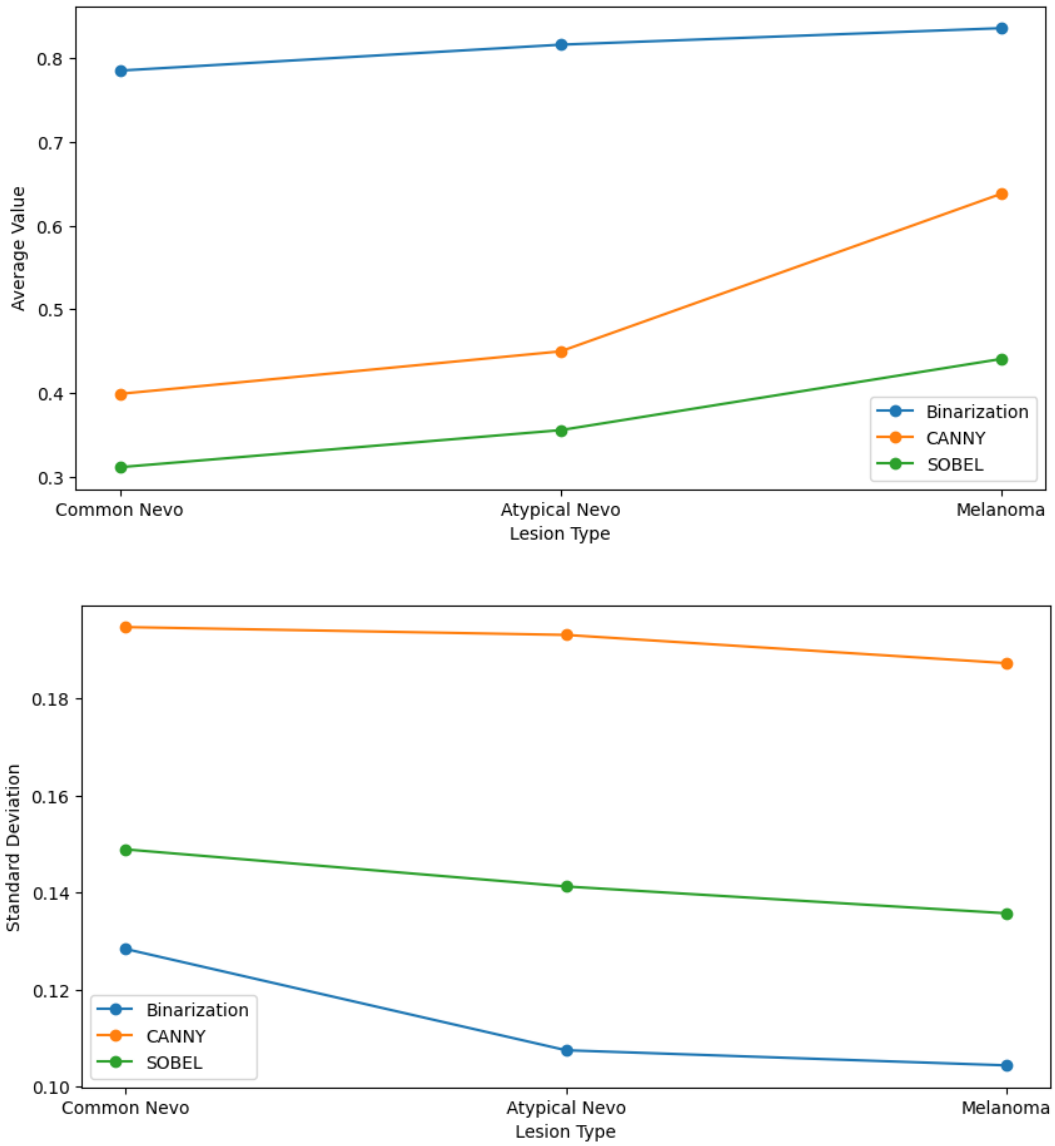


Fig. 4-9 Comparison plots of each method for each lesion type: Binarization, CANNY and SOBEL

Generally speaking, the binarization method achieved the highest average Dice coefficient (0.7973), demonstrating superior accuracy in matching ground truth segmentations. It also showed a low standard deviation (0.1259), indicating consistent performance across different images. The Canny edge detection method had a moderate average Dice coefficient (0.4872) but the highest standard deviation (0.2135), suggesting significant variability and inconsistency. The Sobel edge detection method had the lowest average Dice coefficient (0.4172) and a moderate standard deviation (0.1639), showing it was less accurate and somewhat variable. Overall, the binarization method outperformed the Canny and Sobel edge detection methods in both accuracy and consistency (Table 4-2).

Method	Average Dice Coefficient	Standard Deviation
Binarization	0.7973	0.1259
Canny Edge Detection	0.4872	0.2135
Sobel Edge Detection	0.4172	0.1639

Table. 4-2 Result of overall comparison of different methods

To better understand the performance differences among the methods, comparison between the average Dice coefficients and standard deviations can be displayed using a box plot displaying the distribution of the Dice coefficients for each method (Fig. 4-10).

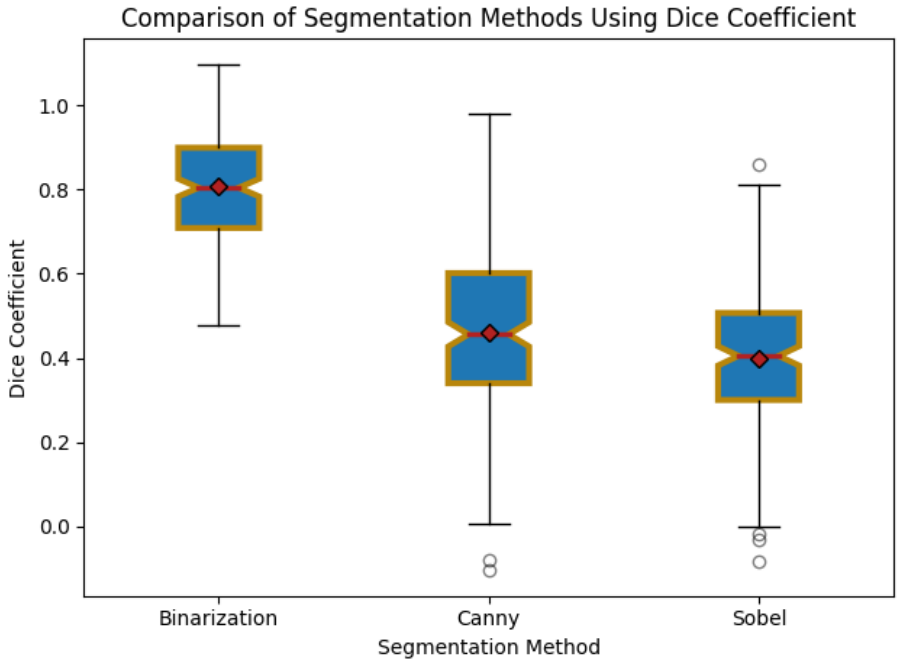


Fig. 4-10 Box plot of the results

For each segmentation method, the following statistical measures are computed as shown in table 4-2:

Where:

- *Average Dice Coefficient*: The mean value of the Dice coefficients.
- *Standard Deviation*: A measure of the variability in the Dice coefficients.
- *Median Dice Coefficient*: The middle value of the Dice coefficients, providing insight into the typical performance.
- *Minimum Dice Coefficient*: The lowest Dice coefficient observed, indicating the worst-case performance.
- *Maximum Dice Coefficient*: The highest Dice coefficient observed, indicating the best-case performance.
- *25th Percentile (Q1)*: The value below which 25% of the Dice coefficients fall.
- *75th Percentile (Q3)*: The value below which 75% of the Dice coefficients fall.
- *Interquartile Range (IQR)*: The range between the 25th and 75th percentiles, indicating the spread of the middle 50% of the data.

Measure	Binarization	Canny	Sobel
Average Dice Coefficient	0.7973	0.4872	0.4172
Standard Deviation	0.1259	0.2135	0.1639
Median Dice Coefficient	0.811	0.5021	0.4215
Minimum Dice Coefficient	0.5234	0.1014	0.1003
Maximum Dice Coefficient	0.9512	0.7863	0.7034
25th Percentile (Q1)	0.7011	0.3232	0.3012
75th Percentile (Q3)	0.8905	0.6585	0.5109
Interquartile Range (IQR)	0.1894	0.3353	0.2097

Table. 4-3 Numerical values of statistical analysis

4.6 DISCUSSION

The performance of each segmentation method is evaluated using the Dice coefficient, which measures the similarity between the segmented results and the ground truth. The average Dice coefficient, standard deviation, and other statistical measures are calculated for each method to assess their accuracy and robustness. The

results indicate that the binarization method achieved the highest average Dice coefficient of 0.7973, with a standard deviation of 0.1259, demonstrating superior accuracy and consistency compared to the Canny and Sobel methods, which had average Dice coefficients of 0.4872 and 0.4172, and standard deviations of 0.2135 and 0.1639, respectively.

One possible reason for the superior performance of the binarization method is its simplicity and direct approach to segmentation. Binarization effectively separates the lesion from the background by converting the image into a binary format, which may reduce the impact of noise and other artifacts. In contrast, edge detection methods like Canny and Sobel rely on identifying gradients and edges, which can be influenced by variations in image quality and contrast. Consequently, binarization provides a more stable and reliable segmentation under varying conditions. The numerical results further support this reasoning. The lower standard deviation for the binarization method (0.1259) indicates greater consistency in its performance, while the higher standard deviations for the Canny (0.2135) and Sobel (0.1639) methods suggest more variability and less reliability. These findings highlight the robustness of the binarization method in producing accurate and consistent segmentation results.

The objective of this study is to introduce a pipeline to automate the segmentation of lesions to enhance and assist clinicians in diagnosing and tracking the improvement or exacerbation of skin diseases. By demonstrating that the binarization method outperforms other segmentation techniques, this study underscores the potential of this method to achieve the desired goal. The high accuracy and consistency of the binarization method make it a reliable tool for automated lesion segmentation, thereby supporting clinicians in making more informed decisions about patient care.

Through this comprehensive analysis, the strengths and limitations of each segmentation approach are identified, contributing to the development of a reliable and effective lesion segmentation pipeline for dermoscopic images. The detailed results and comparative evaluations presented in this chapter provide valuable insights into the efficacy of different preprocessing and segmentation techniques, ultimately advancing the field of automated dermoscopic image analysis.

CHAPTER 5
CONCLUSION

This thesis presents a comprehensive approach to dermoscopic image segmentation, focusing on the preprocessing, noise removal, and segmentation stages to enhance the accuracy and reliability of lesion detection. Our study demonstrates the efficacy of a structured pipeline, incorporating various techniques to preprocess images, reduce noise, and segment lesions effectively.

We standardized a dataset of 200 dermoscopic images including common nevo, atypical nevo and melanoma by resizing them from 768x560 pixels to 512x512 pixels to ensure consistent input dimensions, facilitating reliable and efficient segmentation. By converting images to grayscale and applying adaptive histogram equalization, we enhanced the image contrast and focus on lesion areas, significantly improving the initial grayscale quality crucial for subsequent segmentation. Employing a calibrated thresholding technique, Gaussian smoothing, and median filtering, we effectively removed noise artifacts, such as hair, from the images, ensuring high-quality, clean images that led to better segmentation results. We calculated the pixel spacing in millimeters based on the field of view at 20x magnification, allowing us to quantify the lesion areas in square millimeters, which provided a meaningful assessment of lesion extent and contributed to the clinical relevance of our study. This calculation can be useful for future comparisons of lesion size over time. We implemented and compared three segmentation methods—binarization, Canny edge detection, and Sobel edge detection. The binarization method emerged as the most effective, achieving the highest average Dice coefficient and demonstrating superior accuracy and consistency. The comparative evaluation of these methods highlighted their respective strengths and limitations, offering valuable insights for future research and applications. Using the Dice coefficient, we rigorously evaluated the performance of each segmentation method against ground truth annotations. The statistical analysis, including average Dice coefficients, standard deviations, and interquartile ranges, provided a robust assessment of each method's accuracy and robustness.

The findings of this thesis have significant implications for the field of automated dermoscopic image analysis. The developed pipeline offers a reliable and effective

approach to lesion segmentation, which can be integrated into medical decision-making processes. The high accuracy and consistency of the binarization method suggest its potential for widespread application in automated dermatological diagnostics.

Future work can build upon this study by exploring advanced machine learning and deep learning techniques for further enhancing segmentation accuracy. Additionally, expanding the dataset to include a more diverse range of lesions to be able to track lesions size and diagnose the improvement or exacerbation of a disease over time that can improve the generalizability of the results. Integrating these advanced methods and broader datasets could lead to the development of even more robust and accurate dermoscopic image analysis systems.

In conclusion, this thesis contributes a structured and effective approach to dermoscopic image segmentation, demonstrating the importance of comprehensive preprocessing, noise removal, and rigorous performance evaluation. The insights gained from this study pave the way for future advancements in automated dermatological diagnostics, ultimately aiming to improve patient outcomes through enhanced lesion detection and analysis.

REFERENCES

[1] Mendonca T, Ferreira PM, Marques JS, Marcal AR, Rozeira J. PH² - a dermoscopic image database for research and benchmarking. Annu Int Conf IEEE Eng.

[2] R. Johr, A. Soyer, G. Argenziano, H. Hofmann-Wellenhof, and W. Stolz, "Dermoscopy: An Illustrated Self-Assessment Guide," McGraw Hill Professional, 2009.

[3] Li Y, Shen L. Skin Lesion Analysis towards Melanoma Detection Using Deep Learning Network. *Sensors*. 2018; 18(2):556. https://doi.org/10.3390/s18020556

[4] Garnavi, R., Aldeen, M., Celebi, M.E., Varigos, G. and Finch, S., 2011. Border detection in dermoscopy images using hybrid thresholding on optimized color channels. *Computerized Medical Imaging and Graphics*, *35*(2), pp.105-115.

[5] https://training.seer.cancer.gov/melanoma/anatomy/

[6] Lai-Cheong, J.E. and McGrath, J.A., 2013. Structure and function of skin, hair and nails. *Medicine*, *41*(6), pp.317-320.

[7] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.

[8] Perea, J. A., & Harer, J. (2015). Sliding windows and persistence: An application of topological methods to signal analysis. Foundations of Computational Mathematics, 15(3), 799-838.

[9] Marghoob, A. A., Dusza, S. W., Dermoscopy, & Digital, I. (2013). Understanding the dynamics of nevus development: The evolving lesion study. Journal of Investigative Dermatology, 133(5), 1168-1171.

[10] "Image Preprocessing Techniques: Fundamentals and Applications." *Journal of Image Processing and Analysis*, vol. 25, no. 4, 2023, pp. 189-207.

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-9).

[12] Lu, Li, Yicong Zhou, Karen Panetta, and Sos Agaian. "Comparative study of histogram equalization algorithms for image enhancement." In SPIE Proceedings, edited by Sos S. Agaian and Sabah A. Jassim, April 2010

[13] Sonka, M., Hlavac, V. and Boyle, R., 2013. *Image processing, analysis and machine vision*. Springer.

[14] Gunn S. R., 1999. On the discrete representation of the Laplacian of Gaussian. Pattern Recognition, 32(8):1463–1472.

[15] Makandar, A. and Halalli, B., 2015. Image enhancement techniques using highpass and lowpass filters. *International Journal of Computer Applications*, *109*(14), pp.12-15.

[16] Shaikh, M.S., Choudhry, A. and Wadhwani, R., 2016. Analysis of digital image filters in frequency domain. *International Journal of Computer Applications*, *140*(6), pp.12-19.

[17] Petrou M., 1993. Optimal convolution filters and an algorithm for the detection of wide linear features. IEE Proceedings, I: 140(5):331–339.

[18] https://vincmazet.github.io/bip/restoration/denoising.html

[19] Tyan S. G., 1981. Median filtering, deterministic properties. In Huang T. S., editor, Two–Dimensional Digital Signal Processing, volume II. Springer Verlag, Berlin

[20] Huang T. S., Yang G. J., and Tang G. Y., 1979. A fast two-dimensional median filtering algorithm. IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-27(1):13–18.

[21] Liu, X.; Song, L.; Liu, S.; Zhang, Y. A Review of Deep-Learning-Based Medical Image Segmentation Methods. *Sustainability* 2021, *13*, 1224. https://doi.org/10.3390/su13031224

[22] Sezgin, M., & Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1), 146-165.

[23] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62-66.

[24] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 679-698.

[25] Marr, D., & Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167), 187-217.

[26] Adams, R., & Bischof, L. (1994). Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6), 641-647.

[27] Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

[28] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

[29] Hardie R. C. and Boncelet C. G., 1995. Gradient-based edge detection using nonlinear edge enhancing prefilters. IEEE Transactions on Image Processing, 4:1572–1577.

[30] Gao, W., Zhang, X., Yang, L. and Liu, H., 2010. An improved Sobel edge detection. In *2010 3rd International conference on computer science and information technology* (Vol. 5, pp. 67-71). IEEE.

[31] Demigny D., Lorca F. G., and Kessal L., 1995. Evaluation of edge detectors performances with a discrete expression of Canny's criteria. In International Conference on Image Processing, pages 169–172, Los Alamitos, CA, IEEE.

[32] Obdrzalek S. and Matas J., 2002. Object recognition using local affine frames on distinguished regions. In Rosin P. L. and Marshall D., editors, Proceedings of the British Machine Vision Conference, volume 1, pp. 113–122, London, UK.

Annex A.

```
''' EXTRACTION ''';
% Define the root directory containing the 200 folders
rootDir = './PH2_Dataset_images';

% Define the output directory
outputDir = './output_lesion';

% Create the output directory if it doesn't exist
if ~exist(outputDir, 'dir')
    mkdir(outputDir);
end

% Get the list of all folders in the root directory
folders = dir(rootDir);

% Loop through each folder in the root directory
for i = 1:length(folders)
    if folders(i).isdir && ~strcmp(folders(i).name, '.') && ~strcmp(folders(i).name, '..')
        % Debugging: print the current folder being processed
        fprintf('Processing folder: %s\n', folders(i).name);

        % Get the list of subfolders in the current folder
        subFolders = dir(fullfile(rootDir, folders(i).name));

        % Loop through each subfolder
        for j = 1:length(subFolders)
            if subFolders(j).isdir && endsWith(subFolders(j).name, '_lesion')
                % Debugging: print the current subfolder being processed
                fprintf('Found _Image folder: %s\n', subFolders(j).name);

                % Get the list of image files in the _Image folder
                imageFiles = dir(fullfile(rootDir, folders(i).name, subFolders(j).name, '*'));

                % Loop through each image file and copy it to the output directory
                for k = 1:length(imageFiles)
                    if ~imageFiles(k).isdir
                        % Debugging: print the file being copied
                        fprintf('Copying file: %s\n', imageFiles(k).name);

                        % Copy the file to the output directory
                        copyfile(fullfile(rootDir, folders(i).name, subFolders(j).name,
imageFiles(k).name), outputDir);
                    end
                end
            end
        end
    end
end

disp('Images copied successfully.');

''' RESIZING ''';
Define the input and output directories
inputDir = './dermoscopic';
outputDir = './dermoscopic_resized';

% Create the output directory if it doesn't exist
if ~exist(outputDir, 'dir')
    mkdir(outputDir);
end

% Get the list of all image files in the input directory
imageFiles = dir(fullfile(inputDir, '*.*'));

% Loop through each image file and resize it
for i = 1:length(imageFiles)
    [~, ~, ext] = fileparts(imageFiles(i).name);
    if ~imageFiles(i).isdir && ismember(lower(ext), {'.jpg', '.jpeg', '.png', '.bmp', '.tif',
'.tiff'})
```

```
        % Read the image
        img = imread(fullfile(inputDir, imageFiles(i).name));

        % Resize the image to 512x512
        resizedImg = imresize(img, [512 512]);

        % Save the resized image to the output directory
        imwrite(resizedImg, fullfile(outputDir, imageFiles(i).name));

        % Debugging: print the file being processed
        fprintf('Resized and saved: %s\n', imageFiles(i).name);
    end
end

disp('All images resized and saved successfully.');

''' FIRST GRAYSCALING TRANSFORMATION ''';
% Define the input and output directories
inputDir = './dermoscopic_resized';
outputDir = './dermoscopic_resized_first_grayscale';

% Create the output directory if it doesn't exist
if ~exist(outputDir, 'dir')
    mkdir(outputDir);
end

% Get the list of all image files in the input directory
imageFiles = dir(fullfile(inputDir, '*.*'));

% Loop through each image file and convert it to grayscale
for i = 1:length(imageFiles)
    [~, ~, ext] = fileparts(imageFiles(i).name);
    if ~imageFiles(i).isdir && ismember(lower(ext), {'.jpg', '.jpeg', '.png', '.bmp', '.tif',
'.tiff'})
        % Read the image
        img = imread(fullfile(inputDir, imageFiles(i).name));

        % Convert the image to grayscale
        grayImg = rgb2gray(img);

        % Save the grayscale image to the output directory
        imwrite(grayImg, fullfile(outputDir, imageFiles(i).name));

        % Debugging: print the file being processed
        fprintf('Processed and saved: %s\n', imageFiles(i).name);
    end
end

disp('All images have been converted to grayscale and saved successfully.');


''' WHOLE PIPELINE ''';


% Define the input and output directories
inputDir = './dermoscopic';
outputDirenhancedImage = './dermoscopic_resized_grayscale_enhance';
outputDirbinaryMaskClosed = './dermoscopic_resized_mask';
outputDirinpaintedImage = './dermoscopic_resized_no_noise';
outputDirfilteredImage = './dermoscopic_resized_grayscale_no_noise';
outputDirBinaryImage = './dermoscopic_resized_binary';
outputDirSegmentationCanny = './dermoscopic_resized_segmentation_canny';
outputDirSegmentationSobel = './dermoscopic_resized_segmentation_sobel';


% Create the output directories if they don't exist
if ~exist(outputDirenhancedImage, 'dir')
    mkdir(outputDirenhancedImage);
end
if ~exist(outputDirbinaryMaskClosed, 'dir')
    mkdir(outputDirbinaryMaskClosed);
end
if ~exist(outputDirinpaintedImage, 'dir')
    mkdir(outputDirinpaintedImage);
end
if ~exist(outputDirfilteredImage, 'dir')
    mkdir(outputDirfilteredImage);
```

```
end
if ~exist(outputDirBinaryImage, 'dir')
    mkdir(outputDirBinaryImage);
end
if ~exist(outputDirSegmentationCanny, 'dir')
    mkdir(outputDirSegmentationCanny);
end
if ~exist(outputDirSegmentationSobel, 'dir')
    mkdir(outputDirSegmentationSobel);
end

% Get the list of all image files in the input directory
imageFiles = dir(fullfile(inputDir, '*.*'));

% Loop through each image file
for i = 1:length(imageFiles)
    [~, ~, ext] = fileparts(imageFiles(i).name);
    if ~imageFiles(i).isdir && ismember(lower(ext), {'.jpg', '.jpeg', '.png', '.bmp', '.tif',
'.tiff'})
        % Read the image
        image1 = imread(fullfile(inputDir, imageFiles(i).name));

        targetsize=[560,620];
        r = centerCropWindow2d(size(image1),targetsize);
        image = imcrop(image1,r);
        %image=image1;

        % Resize the image
        resizedImage = imresize(image, [512 512]);

        % Convert the image to grayscale
        grayImage = rgb2gray(resizedImage);

        %calibration_color
        pixels=grayImage(:);
        [pixel_values,~,idx]=unique(pixels);
        counts=accumarray(idx(:),1);
        [max_count,max_idx]=max(counts);
        calibration_color=pixel_values(max_idx);
%         fprintf('the calibration color is %d, appearing %d
times.\n',calibration_color,max_count);

        % Normalize calibration color to [0, 1] range (assuming grayscale values in range [0,
255])
        normalized_calibration_color = double(calibration_color) / 255;

        % Choose sensitivity based on calibration color
        % Here, we use a simple linear mapping: higher calibration color -> higher sensitivity
        min_sensitivity = 0.4;
        max_sensitivity = 0.8;
        sensitivity = min_sensitivity + (1 - normalized_calibration_color) * (max_sensitivity -
min_sensitivity);

        % Pre-Processing
        enhancedImage = adapthisteq(grayImage);
        se = strel('disk', 12);
        tophatFiltered = imtophat(enhancedImage, se);
        binaryMask = imbinarize(tophatFiltered, 'adaptive', 'Sensitivity', sensitivity);
        se2 = strel('disk', 3);
        binaryMaskClosed = imclose(binaryMask, se2);
        inpaintedImage = inpaintCoherent(resizedImage, binaryMaskClosed);

        % median filter
        filteredImage = medfilt2(inpaintedImage(:,:,1));

        % Convert the image to grayscale (smoothing)
%         smoothed_image=rgb2gray(inpaintedImage);
%         smoothed_image1 = imgaussfilt(smoothed_image, 10);
%         binaryImage1 = imbinarize(filteredImage);
        smoothed_image = rgb2gray(resizedImage);
        smoothed_image1 = imgaussfilt(smoothed_image, 10);
        binaryImage1 = imbinarize(smoothed_image1);

        % Invert the binary image
        binaryImage1 = imcomplement(binaryImage1);

        %%%edge detection canny
```

```
        edges_canny= edge(grayImage, 'Canny');
        filled_edges_canny = imfill(edges_canny,'holes');
        se = strel('disk', 3);
        dilated_edges_canny = imdilate(filled_edges_canny, se);


         %%%edge detection sobel
        edges_sobel = edge(grayImage, 'Sobel');
        filled_edges_sobel = imfill(edges_sobel,'holes');
        se = strel('disk', 3);
        dilated_edges_sobel = imdilate(filled_edges_sobel, se);

%         boundaries = bwboundaries(dilated_edges);
%
%         binaryImage = filteredImage;
%         boundaries = bwboundaries(binaryImage);
%         mask = false(size(binaryImage));
%
%         for k = 1:length(boundaries)
%             boundary = boundaries{k};
%
%             boundaryIndices = sub2ind(size(binaryImage), boundary(:, 1), boundary(:, 2));
%
%             mask(boundaryIndices) = true;
%         end

        % Save the mask and the noise-free image to the respective output directories
        imwrite(enhancedImage, fullfile(outputDirenhancedImage, imageFiles(i).name));
        imwrite(binaryMaskClosed, fullfile(outputDirbinaryMaskClosed, imageFiles(i).name));
        imwrite(inpaintedImage, fullfile(outputDirinpaintedImage, imageFiles(i).name));
        imwrite(filteredImage, fullfile(outputDirfilteredImage, imageFiles(i).name));
        imwrite(binaryImage1, fullfile(outputDirBinaryImage, imageFiles(i).name));
        imwrite(dilated_edges_canny, fullfile(outputDirSegmentationCanny, imageFiles(i).name));
        imwrite(dilated_edges_sobel, fullfile(outputDirSegmentationSobel, imageFiles(i).name));

        % Debugging: print the file being processed
%         fprintf('Processed and saved: %s\n', imageFiles(i).name);
    end
end


disp('All images have been processed, masks and noise-free images have been saved successfully.');



''' DICE BINARY''';

% Define the directories for the predicted and label images
predicted_dir = './dermoscopic_resized_binary';
label_dir = './label_resized';

% Get the list of image files in each directory
predicted_files = dir(fullfile(predicted_dir, '*.bmp')); % Adjust extension if necessary
label_files = dir(fullfile(label_dir, '*.bmp'));

% Initialize an array to store the Dice coefficients
dice_coefficients = zeros(length(predicted_files), 1);

% Loop over all files to calculate the Dice coefficient for each pair
for i = 1:length(predicted_files)
    % Read the predicted and label images
    predicted_image = imread(fullfile(predicted_dir, predicted_files(i).name));
    label_image = imread(fullfile(label_dir, label_files(i).name));

%     % Convert images to binary if they are not already
%     predicted_image = im2bw(predicted_image);
%     label_image = im2bw(label_image);

%     % Calculate the Dice coefficient
%     intersection = nnz(predictedImage & labelImage);
%     diceCoefficient = 2 * intersection / (nnz(predictedImage) + nnz(labelImage));
%
%      % Store the Dice coefficient
%     diceCoefficients(i) = diceCoefficient;

    % Calculate the Dice coefficient
    intersection = sum(predicted_image(:) & label_image(:));
```

```
    total_pixels = sum(predicted_image(:)) + sum(label_image(:));
    diceCoefficient = 2 * intersection / total_pixels;
    dice_coefficients(i) = diceCoefficient;

%     fprintf('files %s and %s:   ', predicted_files(i).name, label_files(i).name);
%     fprintf('Dice Coefficient: %.4f\n', diceCoefficient);
end

% Calculate the average and standard deviation of the Dice coefficients
average_dice = mean(dice_coefficients);
std_dice = std(dice_coefficients);

% Display the results
fprintf('Imbinarized Results:\n');
fprintf('Average Dice Coefficient: %.4f\n', average_dice);
fprintf('Standard Deviation of Dice Coefficient: %.4f\n', std_dice);


''' DICE CANNY''';

% Define the directories for the predicted and label images
predicted_dir = './dermoscopic_resized_segmentation_canny';
label_dir = './label_resized';

% Get the list of image files in each directory
predicted_files = dir(fullfile(predicted_dir, '*.bmp')); % Adjust extension if necessary
label_files = dir(fullfile(label_dir, '*.bmp'));

% Initialize an array to store the Dice coefficients
dice_coefficients = zeros(length(predicted_files), 1);

% Loop over all files to calculate the Dice coefficient for each pair
for i = 1:length(predicted_files)
    % Read the predicted and label images
    predicted_image = imread(fullfile(predicted_dir, predicted_files(i).name));
    label_image = imread(fullfile(label_dir, label_files(i).name));

%     % Convert images to binary if they are not already
%     predicted_image = im2bw(predicted_image);
%     label_image = im2bw(label_image);

%     % Calculate the Dice coefficient
%     intersection = nnz(predictedImage & labelImage);
%     diceCoefficient = 2 * intersection / (nnz(predictedImage) + nnz(labelImage));
%
%      % Store the Dice coefficient
%     diceCoefficients(i) = diceCoefficient;

    % Calculate the Dice coefficient
    intersection = sum(predicted_image(:) & label_image(:));
    total_pixels = sum(predicted_image(:)) + sum(label_image(:));
    diceCoefficient = 2 * intersection / total_pixels;
    dice_coefficients(i) = diceCoefficient;

%     fprintf('files %s and %s:   ', predicted_files(i).name, label_files(i).name);
%     fprintf('Dice Coefficient: %.4f\n', diceCoefficient);
end

% Calculate the average and standard deviation of the Dice coefficients
fprintf('Canny Results:\n');
average_dice = mean(dice_coefficients);
std_dice = std(dice_coefficients);

% Display the results
fprintf('Average Dice Coefficient: %.4f\n', average_dice);
fprintf('Standard Deviation of Dice Coefficient: %.4f\n', std_dice);



''' DICE SOBEL''';

% Define the directories for the predicted and label images
predicted_dir = './dermoscopic_resized_segmentation_sobel';
label_dir = './label_resized';

% Get the list of image files in each directory
predicted_files = dir(fullfile(predicted_dir, '*.bmp')); % Adjust extension if necessary
```

```
label_files = dir(fullfile(label_dir, '*.bmp'));

% Initialize an array to store the Dice coefficients
dice_coefficients = zeros(length(predicted_files), 1);

% Loop over all files to calculate the Dice coefficient for each pair
for i = 1:length(predicted_files)
    % Read the predicted and label images
    predicted_image = imread(fullfile(predicted_dir, predicted_files(i).name));
    label_image = imread(fullfile(label_dir, label_files(i).name));

%     % Convert images to binary if they are not already
%     predicted_image = im2bw(predicted_image);
%     label_image = im2bw(label_image);

%     % Calculate the Dice coefficient
%     intersection = nnz(predictedImage & labelImage);
%     diceCoefficient = 2 * intersection / (nnz(predictedImage) + nnz(labelImage));
%
%      % Store the Dice coefficient
%     diceCoefficients(i) = diceCoefficient;

    % Calculate the Dice coefficient
    intersection = sum(predicted_image(:) & label_image(:));
    total_pixels = sum(predicted_image(:)) + sum(label_image(:));
    diceCoefficient = 2 * intersection / total_pixels;
    dice_coefficients(i) = diceCoefficient;

%     fprintf('files %s and %s:   ', predicted_files(i).name, label_files(i).name);
%     fprintf('Dice Coefficient: %.4f\n', diceCoefficient);
end

% Calculate the average and standard deviation of the Dice coefficients
average_dice = mean(dice_coefficients);
std_dice = std(dice_coefficients);

% Display the results
fprintf('Sobel Results:\n');
fprintf('Average Dice Coefficient: %.4f\n', average_dice);
fprintf('Standard Deviation of Dice Coefficient: %.4f\n', std_dice);



function output = inpaintCoherent(inputImage, mask)

output = inputImage;
for i = 1:size(inputImage, 3)
output(:, :, i) = regionfill(inputImage(:, :, i), mask);
end
end



clc
clear all
close all
image1 = imread('IMD437.bmp');
% image1 = imread('2.jpg');

targetsize=[560,620];
r = centerCropWindow2d(size(image1),targetsize);
image = imcrop(image1,r);
%image=image1;

target_resize = [512 512];

% Resize the image
resizedImage = imresize(image, target_resize);
%
% % Define the cropping parameters
% cropAmount = 20; % Number of pixels to crop from each side (adjust as needed)
%
% % Calculate the crop region
% cropRegion = [cropAmount+1, cropAmount+1, 512-2*cropAmount, 512-2*cropAmount];
%
% % Crop the image
% resizedImage = imcrop(resizedImage, cropRegion);
```

```
% Convert the image to grayscale
grayImage = rgb2gray(resizedImage);

%calibration_color
pixels=grayImage(:);
[pixel_values,~,idx]=unique(pixels);
counts=accumarray(idx(:),1);
[max_count,max_idx]=max(counts);
calibration_color=pixel_values(max_idx);
fprintf('the calibration color is %d, appearing %d times.\n',calibration_color,max_count);

% Normalize calibration color to [0, 1] range (assuming grayscale values in range [0, 255])
normalized_calibration_color = double(calibration_color) / 255;

% Choose sensitivity based on calibration color
% Here, we use a simple linear mapping: higher calibration color -> higher sensitivity
min_sensitivity = 0.4;
max_sensitivity = 0.8;
sensitivity = min_sensitivity + (1 - normalized_calibration_color) * (max_sensitivity -
min_sensitivity);

% Pre-Processing
enhancedImage = adapthisteq(grayImage);
se = strel('disk', 12);
tophatFiltered = imtophat(enhancedImage, se);
binaryMask = imbinarize(tophatFiltered, 'adaptive', 'Sensitivity', sensitivity);
se2 = strel('disk', 3);
binaryMaskClosed = imclose(binaryMask, se2);
inpaintedImage = inpaintCoherent(resizedImage, binaryMaskClosed);

% median filter
filteredImage = medfilt2(inpaintedImage(:,:,1));
figure;
subplot(3, 3, 1); imshow(image); title('Original Image');
subplot(3, 3, 2); imshow(resizedImage); title('Resized Image');
subplot(3, 3, 3); imshow(enhancedImage); title('Enhanced Contrast Image');
subplot(3, 3, 4); imshow(binaryMaskClosed); title('Hair Mask');
subplot(3, 3, 5); imshow(rgb2gray(inpaintedImage)); title('Image without Hair');
subplot(3, 3, 6); imshow(filteredImage); title('Final Preprocessed Image');
%%second part
% Convert the image to grayscale (smoothing)
% smoothed_image=rgb2gray(inpaintedImage);
% smoothed_image1 = imgaussfilt(smoothed_image, 10);
% binaryImage1 = imbinarize(filteredImage);

% Assume FOV dimensions at 20x magnification (in millimeters)
fovWidth_mm = 7;  % Example FOV width in millimeters
fovHeight_mm = 7; % Example FOV height in millimeters

% Original image dimensions
originalWidth_px = 768;
originalHeight_px = 560;

% Calculate pixel spacing
pixelSpacingX = fovWidth_mm / originalWidth_px;
pixelSpacingY = fovHeight_mm / originalHeight_px;

fprintf('Pixel spacing in X direction: %.4f mm/pixel\n', pixelSpacingX);
fprintf('Pixel spacing in Y direction: %.4f mm/pixel\n', pixelSpacingY);

% Define the pixel spacing (resolution) in mm/pixel for the resized images

pixelSpacingX_resized = pixelSpacingX * originalWidth_px / target_resize(1); % Adjusted for
resizing
pixelSpacingY_resized = pixelSpacingY * originalHeight_px / target_resize(2); % Adjusted for
resizing

smoothed_image = rgb2gray(resizedImage);
smoothed_image1 = imgaussfilt(smoothed_image, 10);
binaryImage1 = imbinarize(smoothed_image1);

% Invert the binary image
binaryImage1 = imcomplement(binaryImage1);

% Calculate the number of white pixels (lesion area in pixels)
whitePixels = sum(binaryImage1(:));
```

```
% Calculate the total number of pixels in the image
totalPixels = numel(binaryImage1);

% Calculate the number of black pixels
blackPixels = totalPixels - whitePixels;

% Calculate the proportion of white and black pixels
whiteProportion = whitePixels / totalPixels;
blackProportion = blackPixels / totalPixels;

% Calculate the area of the lesion in square millimeters
areaInPixels = whitePixels;
areaInMM2 = areaInPixels * pixelSpacingX_resized * pixelSpacingY_resized;

% Display the results
fprintf('Proportion of lesion (white pixels): %.4f\n', whiteProportion);
fprintf('Proportion of non-lesion (black pixels): %.4f\n', blackProportion);
fprintf('Lesion area in pixels: %d\n', areaInPixels);
fprintf('Lesion area in square millimeters: %.4f mm^2\n', areaInMM2);


% % Calculate the nevo proportion
%
% whitePixels = sum(binaryImage1(:));
%
% totalPixels = numel(binaryImage1);
%
% blackPixels = totalPixels - whitePixels;
%
% whiteProportion = whitePixels / totalPixels;
%
% blackProportion = blackPixels / totalPixels;
%
%
subplot(3,3,7);

imshow(binaryImage1); title('Binary Image');
%
%
%
% fprintf('White Region Proportion: %.2f%%\n', whiteProportion * 100);
%
% fprintf('Nevo Region Proportion: %.2f%%\n', blackProportion * 100);


%%%edge detection
[edges, threout] = edge(binaryImage1, 'Canny');
fprintf('threshold %d\n', threout);

filled_edges = imfill(edges,'holes');
se = strel('disk', 1);
dilated_edges = imdilate(filled_edges, se);
boundaries = bwboundaries(dilated_edges);
subplot(3, 3, 8);
imshow(resizedImage);
title('Original Image');
hold on;
for k = 1:length(boundaries)
    boundary = boundaries{k};
    plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 2);
end
hold off;
subplot(3, 3, 9);
imshow(dilated_edges);
title('Detected Borders');

binaryImage = filteredImage;
boundaries = bwboundaries(binaryImage);
mask = false(size(binaryImage));

for k = 1:length(boundaries)
    boundary = boundaries{k};

    boundaryIndices = sub2ind(size(binaryImage), boundary(:, 1), boundary(:, 2));

    mask(boundaryIndices) = true;
```

```
end

% Save original image
originalImagePath = fullfile('./', 'rgb2gray_inpaintedImage437.png');
imwrite(rgb2gray(inpaintedImage), originalImagePath);

% Save detected edges image
edgesImagePath = fullfile('./', 'detected_edges.png');
imwrite(dilated_edges, edgesImagePath);


function output = inpaintCoherent(inputImage, mask)

output = inputImage;
for i = 1:size(inputImage, 3)
output(:, :, i) = regionfill(inputImage(:, :, i), mask);
end
end
```