University of Thessaly

Faculty of Engineering

Department of Electrical and Computer Engineering

Volos, Greece

Skin Cancer Detection using Image Processing Techniques in Reconfigurable Hardware

DIPLOMA THESIS

In partial fulfillment of the requirements for the degree of

Diploma in Electrical and Computer Engineering

Author:

Christodoulou Dimitrios

Supervisor:

Professor Bellas Nikolaos

University of Thessaly

Faculty of Engineering

Department of Electrical and Computer Engineering

Volos, Greece

Skin Cancer Detection using Image Processing Techniques in Reconfigurable Hardware

DIPLOMA THESIS

CHRISTODOULOU DIMITRIOS

Supervising committee

Supervisor	Co-Supervisor	Co-Supervisor
Bellas Nikolaos	Katsaros Dimitrios	Lalis Spyros
Professor	Associate Professor	Professor

Volos, September 2020

Πανεπιστήμιο Θεσσαλίας

Πολυτεχνική Σχολή

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Βόλος, Ελλάδα

Εντοπισμός Καρκίνου του Δέρματος με Τεχνικές Επεξεργασίας Εικόνας Χρησιμοποιώντας Επαναδιατασσόμενες Αρχιτεκτονικές

Διπλωματική Εργασία

Χριστοδούλου Δημήτριος

Επιβλεπων Καθηγητής	Δεύτερο μέλος Επιτροπής	Τρίτο μέλος Επιτροπής
Μπέλλας Νικόλαος	Κατσαρός Δημήτριος	Λάλης Σπύρος
Καθηγητής	Αναπληρώτης Καθηγητής	Καθηγητής

Βόλος, Σεπτέμβριος 2020

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Μπέλλας Νικόλαος** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Κατσαρός Δημήτριος** Αναπληρώτης Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Λάλης Σπύρος** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 01-10-2020

Dedicated to

my family

Acknowledgements

At the end of this diploma thesis, I would like to thank everyone who have stood by my side all these years and contributed to the effort I have made. First of all, I would like to deeply thank Prof. Bellas Nikolaos for the supervision of this diploma thesis. His support and guidance have been dominant for the completion of this Thesis. Moreover, I would like to thank to Prof. Katsaros Dimitrios and Prof. Lalis Spyros for their evaluation and contribution on my thesis.

Furthermore, I would like to express my gratitude to my family for encouraging me to pursuit my dream and providing me with the unconditional support to my life.

Lastly, I would like to thank my friends for assuming this journey during the past five years, making wonderful memories that will last a lifetime.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

Ονοματεπώνυμο Φοιτητή
Ημερομηνία

ABSTRACT

Nowadays, Skin cancer is one of the most common cancers with low survival rate among humans. However, early and fast detection of skin cancer can save the patient's life. In this thesis, an alternative method to detect and classify a lesion by using the current technology is developed, to avoid patients the painful and time-consuming process of Biopsy Method. The Skin Cancer Detection systems are beneficial for humans, since the lesion can be classified within some seconds. The methodology consists of basic components and different methods are used in each stage. First of all, different filters are applied in order to reduce noise and enhance the image, and to extract correctly the necessary features, which are used to train the classifiers. The ABCD rule and GLCM texture method provide the necessary features to categorize a lesion in benign or malignant. Different classification algorithms are trained with the extracted features to fit more appropriate the input data. Furthermore, to increase the performance of the system, the image process stage is implemented in hardware.

ΠΕΡΙΛΗΨΗ

Στην εποχή μας, ο καρκίνος του δέρματος είναι ένας από τους πιο συνηθισμένους καρκίνους με χαμηλό ποσοστό επιβίωσης μεταξύ των ανθρώπων. Ωστόσο, η γρήγορη ανίχνευση του καρκίνου του δέρματος σε πρώιμο στάδιο μπόρει να σώσει τη ζωη του ασθενή. Σε αυτήν την διπλωματική εργασία, αναπτύσσεται μια εναλλακτική μέθοδος για τον εντοπισμό και την κατηγοροποίηση ενός μώλωπα χρησιμοπιώντας την τρέχουσα τεχνολογία, για να αποφύγουν οι ασθενείς την επώδυνη και χρονοβόρα διαδικασία της βιοψίας. Οι εφαρμόγες για τον εντοπίσμο του καρκίνου του δέρματος είναι χρήσιμες για τον άνθρωπο, καθώς μπορούν να κατηγοροποιήσουν τον μώλωπα μέσα σε λίγα λέπτα. Η μέθοδος αποτελείται από καποία βασικά τμήματα, για την υλοποίηση των οποίων χρησιμοποιούνται διαφορετικές προσεγγίσεις. Πρώτα απ 'όλα, εφαρμόζονται διάφορα φίλτρα για τη μείωση του θορύβου και την βελτιστοποίηση της εικόνας, προκειμένου να εξαχθούν σωστά τα απαραίτητα χαρακτηριστικά, τα οποία χρησιμοποιούνται για την εκπαίδευση των classifiers. Οι μέθοδοι ABCD και GLCM texture παρέχουν τα απαραίτητα χαρακτηριστίκα για την κατηγοριοποίηση μιας βλάβης σε καλοήθη ή κακοήθη. Διαφορετικοί αλγόριθμοι ταξινόμησης εκπαιδεύονται με τα προηγούμενα χαρακτηριστικά. Επιπλέον, για να αυξήσουμε την απόδοση του συστήματος, το στάδιο της επεξεργασίας εικόνας υλοποιείται και σε Hardware.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In our diploma thesis, Skin Cancer Detection using Image Processing is implemented. Additionally, the Image processing part of our implementation, which is the most demanding and time-consuming part, is accelerated in Hardware. In these days and age, the advent of technology can facilitate all aspect of our life and improve our living conditions. Health is the discipline that the technology can provide the necessary equipment in order to detect and overcome diseases. There are many technological fields which involved in a large variety of project related with health. One demanding and useful project is skin cancer detection, which implemented in this project, since the early detection of a malignant lesion it is easier to cure.

In our project, machine learning algorithms to classify the lesion as malignant or benign are used. Different types of algorithms are used in order to compare the result and select one with higher accuracy. To reduce the burden and the size of our predicting model, specific features from dermoscopic images are extracted. These features used to train and classify the lesion instead the use of the whole image. In order to extract this features two well-known techniques, the ABCD rule and the GLCM texture extraction, are implemented. We investigate which of both had the most precision metrics and leading to a model with high accuracy. Also, trying to achieve a model with high accuracy the two methods are combined in order to avoid misclassification since a minimal error to the values can have serious impact to the result

Moreover, to improve the correctness of the extracted values it is dominant to minimize the noise from the input image. The input images are captured with microscope as a result small noise such as hair is enlarged. Furthermore, the lighting conditions can affect the image, creating darker parts. To accomplish this task, different filters are used, in order to blur, adjust the contrast and remove small objects of the initial image. To extract correctly the value of border irregularity, the prevention of the lesion edges is required, as a consequence the selections of blur algorithm are critical. In bibliography, the median filter is represented as the most suitable filter for blurring the image. Applying this filter, the removal of hairs and edges prevention are achieved.

1.2 Motivation

The detection of cancer, located in different part of human body, is not a new problem. From the previous decades, doctors use innovate machines to capture the potentially part of body in which can exist abnormality. Doctor examine the abnormality and determine if it is cancer using their knowledge and experience. To detect skin cancer, which is one of the most common type of cancer, doctors examine the lesion and decide the type (malignant or benign). If the lesion evaluated as malignant, the patient follows the biopsy process, which is a painful process. Its trigger our mind to think how the current technology and especially machine learning can provide humans an easily access application which classify the lesion given an affordable second option before follow the infallible biopsy process.

Skin is the largest organ of the integumentary system and consisted of seven layers of tissues. The skin is the outer part of the human body forming a shield which helps to cover the different system of human structure such as the muscular, skeletal (bones and ligaments) and internal organ system [1]. Skin cancer is one of the most common cancer in the world, and the medical record deduce that skin cancer is the most common type of cancer in many countries. Sun can easily harm our skin and as a result skin cancer is primarily detected on areas of skin which highly exposed to the sun, but it is also formed on areas that seldom see the light of day. The functions of skin are very important for the survival of human, so the development of abnormal cells on any skin layer can be harmful for the organism and might affect other parts of the body since it is detached to other organs and tissues and cancer can be spread more easily.

Compared to the other two types of skin cancer, melanoma is the deadliest cancer although it is the least common type. Melanoma reported as the deadliest form of skin cancer since it is responsible for 75% of all skin cancer deaths [6]. The early detection of the melanoma is essential, since it can be treated and finally cured in its early stages. However, if the diagnosis becomes late, it is difficult to cured and can be hazardous to the health, since it grows deeper into the skin and spread to other tissues and organs. So, it is important to examine our existing moles, freckles, bumps, and birthmarks to detect possible abnormality in early stage. This lead us to invest more time to improve skin cancer detection algorithms, which is more advantageous to patients compared to the traditional biopsy method.

The cases of this disease show that skin cancer can affect people of all races, including those with darker complexions. Current research in the discipline of skin cancer detection using artificial intelligence has a limitation affiliated with the non-white population. This restriction leads us to expand the current skin cancer classification models using dermoscopic images from people with different skin tones to train the models. This improvement generates an application which appeal to the whole population.

Skin cancer is a very difficult problem with a lot of challenges since it is associated with visual media and medical science. The possible solutions are limited by the current technology and the deficit of dermoscopic images. In the future, the new knowledge in the field of computer vision and machine learning algorithms, and also the expansion of the skin cancer dermoscopic image dataset can help us tackle the problem more efficiently and accurately, making right and proper decisions. But the enrich of image dataset is the most challenging task since these record are privacy so our model need to fulfill this need and does not reveal any information which can be used to identify the person.

In this diploma thesis, we deal with the problem of the Skin Cancer detection by examining different approaches related with image preprocessing, feature extractions, and classification models. Our approach includes image filters to reduce image noise, ABCD rule [5] and GLCM Texture Feature [4] to extract the necessary features for our predicted model, Support-Vector-Machine and decision trees to classify the input images.

1.3 Related Work

The problem of the skin cancer exists since the last decades, and as result many researchers have been working on the Computer Vision approach for skin cancer detection. There are different methods to detect and classify skin lesion in recent research. Some approaches use dermoscopic images, without any preprocessing stage, as input to large model, which comprise Deep Learning or Convolutional Neural Network (CNN). Other approaches use image processing to reduce noise and extract features to classify lesion using machine learning algorithms. In our project, we use the second approach in order to tackle the problem of skin cancer detection.

It is obvious that, dermoscopic images contain noisy, such as hairs and some darker or lighter parts due to the light conditions. In bibliography, there many different approaches for image noise reduction, includes different filters to blur image and remove small objects. Many of these approaches use Gaussian blur or median filter as a typical pre-processing step to remove hairs and smoother the represent lesion. Among them, median filter is the most popular since it preserves edges while removing the existing noise. The preservation of edges is very important for this approach since the regularity of the lesion border has big impact to the final decision. Additionally, many approaches use histogram equalize in order to enhance the contrast of the image in order to identify easier the location of lesion. The localization of the lesion can be accomplished using thresholding techniques or machine learning algorithms.

The most important part is the extraction of the features. From the literature, it can clearly distinguish two different approaches that are used. The first method called ABCD rule (Asymmetry, Border, Color, Diameter). With the implemented of this approach, researchers try to categorize the lesion using these attributes. The second approach called GLCM method (Gray-Level Co-Occurrence Matrix). The GLCM is a method of examining texture of the image and extracting statistical measures. These statistics provide information about contrast, correlation, energy and homogeneity.

Extracted data from both method is the input data for classification algorithms. Due to the lack of large datasets related with skin cancer, in bibliography simple classification models are used. Two different types of models are widely used, SVM (Support Vector Machine) and Decision Trees. The expirimental results show that the accuracy stays below the 95% [7].

Chapter 2

Skin Cancer Information

2.1 General

By skin cancer we mean the three commonest types, basal cell (BCC), squamous cell (SCC) carcinomas of the skin and cutaneous malignant melanoma (melanoma). Skin cancer [2] [3] occurs when mutations develop in the DNA of skin cells. This abnormal most often develops on areas of skin which is highly exposed to the sun. However, the sun is not the only factor that contribute to create these common types of skin cancer, as a result can develop on areas of human skin which do not expose directly to sunlight. Since the exposure to sun is the primary factor to develop cancerous cells, the risk of skin cancer can eliminate by avoiding UV radiation. The regular checking of our skin for suspicious changes can help to early detection of skin cancer and combined with a propel treatment help to successfully overcome this situation.

2.2 Types of Skin Cancer

- Basal cell (BCC) [10] is the most common type of skin cancer (Figure 2-1). BCC normally appear in people with fair skin and often develops after years of regular sun exposure. This type of cell can form anywhere on the body including head, neck, arms, chest, abdomen, and legs. It often looks like a flesh-colored round growth, pearl-like bump, or a pinkish patch of skin.

FIGURE 2-1: BASAL CELL CARCINOMA [10].

- The second most common type of skin cancer is Squamous cell carcinoma (SCC) (Figure 2-2) [10]. This type of cancer can be formed in all race, irrespective of the skin color. SCC tends to form on areas that frequent exposure to the sun such as the ear, face, neck, arms, chest, and back which. Squamous cells can grow deep into the skin, causing damage to skin functionality and spread to other organs and areas of the body. SCC often looks like a red firm bump, scaly patch, or a sore that heals and then re-opens.

FIGURE 2-2: SQUAMOUS CELL CARCINOMA [10].

- Since melanoma [10] can lead to death, it is the most serious skin cancer (Figure 2-3). Melanoma can affect people of any skin tone. Melanoma can develop within a mole that you already have on your skin or appear suddenly as a dark spot on the skin that looks different from the rest. People with darker skin, melanoma tends to develop on the palms or soles, or under the fingernails, Early diagnosis and treatment are crucial for patient health.

FIGURE 2-3: MELANOMA [10].

2.3 Risk Factors

Skin cancer [2] [9] begins in the epidermis which is the outer layer of human skin. The epidermis is a thin layer and contains the three main types of cell, which are described meticulous above and involved in skin cancer. Nominally Squamous cells, Basal cells, and Melanocytes.

The evidence from a combination of experimental and epidemiological data proves that the main source of skin cancer is the directly exposure of UV the sun since only the UV radiation can damage DNA leading to abnormality. The key factor to prevent skin cancer is the controlling of sun exposure.

Factors that may increase the risk to develop skin cancer described analytically below.

- Fair skin: Anyone can develop skin cancer, regardless of skin color. Although, people with fair skin, indicates the fact that having less melanin,

can more easily to wounding from UV radiation than a person with darker skin.

- Excessive sun exposure: Spending a vast amount of time under the sun can increase the risk to develop skin cancer.
- A family history of skin cancer.
- A weakened immune system: People with weakened immune systems are more likely to form skin cancer.
- Exposure to radiation.
- A history of sunburns.

Chapter 3

Project Requirements

3.1 Software

Programming Language

Our project is developed using python programming language [11] [12]. Python is a clear and powerful object-oriented programming language. By using Python, many common programming tasks can easily be developed, since a large standard library is supported. These standard libraries facilitate the creation of different programs related with image processing manipulations like cropping, image segmentation, classification and features extractions, which are predominant tasks for our implementation. An additional appealing feature is that, many new and advanced tools, related to image processing, are available for free. All these reasons make the python programming language an appropriate choice for developing image processing and machine learning task.

Image preprocessing Libraries.

In order to implement this project, the first goal is the reduction of noise and the adjustment of contrast. Many python image preprocessing libraries, like Scikit image, Numpy, PIL/Pillow and OpenCV are used. Especially, OpenCV library is used for both image processing and feature extraction stage. A briefly overview of the usage and functionality of these libraries is made below:

Scikit Image

Scikit-image [12] [13] consist of many algorithms for image processing. It is an open source Python package that primarily works with Numpy arrays. By using the available algorithms of this library, applications used in different fields such as research, education and industry can be developed. Scikit-image focuses on being the reference library for scientific analysis related with image processing. It is a fairly simple and straightforward library. Every function comes with detailed documentation and examples that describe how each function is used in an application. This code is of high-quality and peer-reviewed. Algorithm's codes are written by volunteer developers, who are members of the active community. This library is of high-quality and peer-reviewed since all the included algorithms are tested by core developers to ensure correctness. A primary mission is to care for users' data, meaning that the code doesn't modify input arrays unless explicitly directed to do by the user. The package is imported as skimage and most functions are found within the submodules.

Numpy

Numpy [12] [14] is one of the fundamental libraries in Python programming and provides support for arrays. It is independent on any other Python packages and it only depends on an accelerated linear algebra library. This library uses MATLAB-style syntax to perform numerical operations. It is widely used for image processing applications, since an image is essentially an array structure, consisting of data points that represent pixels. Therefore, by using basic Numpy operations, such as slicing, masking and fancy indexing, the pixel values of an image can be easily modified and the desire output image can be achieved. Numpy is commonly used to mask an image. In order to use this image library, it is mandatory to load the initial image by using skimage library and display the preprocessing image with matplotlib library.

PIL/ Pillow

PIL (Python Imaging Library) [12] [15] is a free library included in python programming language which supports opening, manipulating, and saving many different image file formats, with the ability to create new file decoders to expand the

library. However, its development has stagnated, with its last release in September 2009 which supports Python 1.5.2-2.7. The following years, the Pillow project has forked the PIL repository and added Python 3 support. As a consequence, the original PIL has been replaced and is available for all major operating systems (Windows, Mac OS X, and Linux). This library contains basic image processing functionalities, including per-pixel operations, masking and transparency handling, filtering with a set of built-in convolution kernels, and color space conversions.

OpenCV

OpenCV (Open Source Computer Vision Library) [12] [16] is one of the most widely used libraries for computer vision applications. The use of this library requires the OpenCV-Python which is the python API for OpenCV. OpenCV-Python can be referred as a fast library since the codes are written in C/C++. Additionally, OpenCV-Python is a highly optimized library, while it uses Numpy, which is an optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. It is an advantageous choice to perform computationally intensive computer vision programs, as it a fast and easy library to code and deploy. It is available on Linux, MacOS, Windows, iOS and Android.

Chapter 4

Methodology

4.1 General

Before starting the analysis of our project, a diagram of the process which is followed in order to complete our design, is represented. The following diagram (Figure 4-1) can help us to describe more clearly the functions that were implemented at each stage to reach our goal.

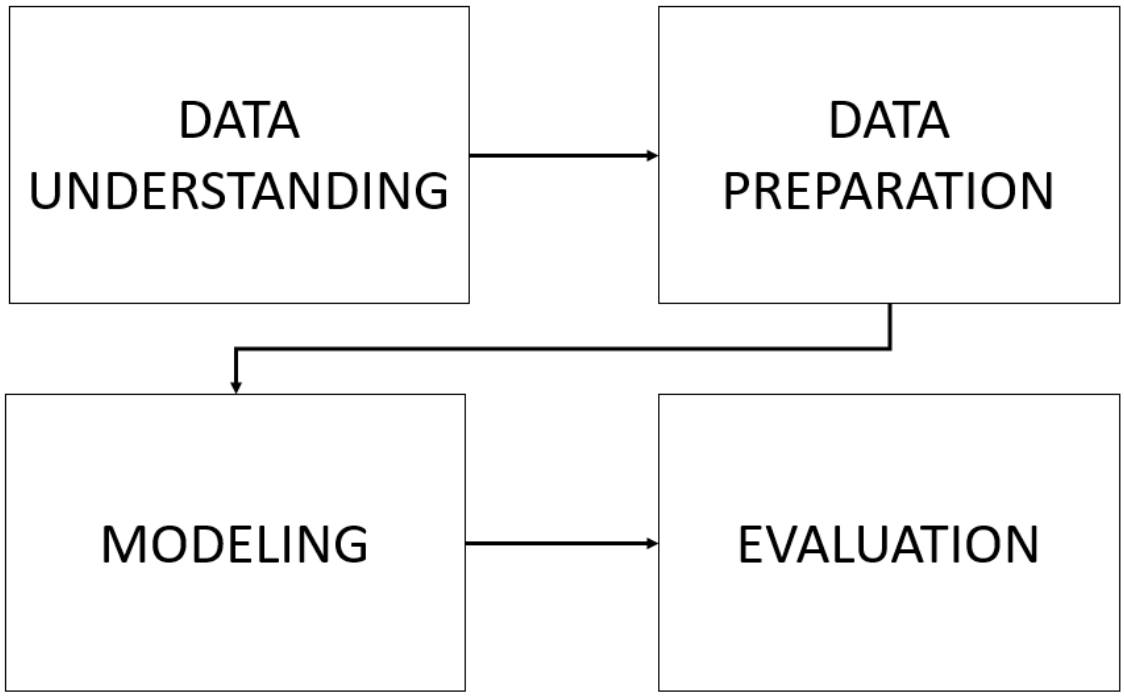

FIGURE 4-1: PROCESS DIAGRAM

It is obvious, that the stage of Data Understanding is the most vital. Data Understanding combines the subdivided stages of Data Requirements, Data Collection and Data Understanding. Before starting the collection of data, we need to clearly understand the problem that is needed to tackle, and specify all the requirements for our project. These requirements help us to determine the sources and the methods that we have to use in order to collect the appropriate data. At this first stage of the process, the creation of the database is started, by collecting all data needed, which consist of the initial data. Following this, we have to clearly understand the content of the input data and identify noise that may be included. For our implementation, it is obvious that dermoscopic images are the necessary input that must be obtained in order to move to the next stage. It is important that, these images should be labeled and confirmed from expert as benign and malignant to ensure the correctness of the project. Furthermore, these images needed to be captured with a microscope or a high resolution camera in order to depict the lesion in detail. It is an evidence that these images can contain noise. The most common type of noise, is hairs, which exist in many part of human skin.

Data Preparation is the most necessary stage of the process, since the input image is transformed to useable data. Once the data has been collected, we need to process the data, to reduce the existing noise and extract information, to create the final database for modeling stage. The data preparation is a very time-consuming procedure since the implementation of different algorithms needed to improve and transform the initial database. In this stage some features are extracted, in order to classify the lesion and as a consequence, existing noise can corrupt the final decision. It is prerequisite to build an efficient image processing stage, stacking many image filters to reduce the noise.

Modeling refers to the purpose of data mining. After the stage of data collection and preparation, the data must be handled by an appropriate model. Depending on the data mining problem a different appropriate model exists. The selection of the appropriate model is a demanding task, while the usage of the final result depends on it. The correct models reveal patterns and structures that provide meaningful insights and new knowledge. For our project, these patterns and structures can lead to a classification of the lesion into two categories: benign and malignant.

Evaluation is the last and most important stage. During this stage, the selected model is tested by a preselected part of the database. This, allow one to see the effectiveness and the accuracy of the model. Results from this stage, are used to determine if it is mandatory to return in one of the previous stages in order to follow a different approach and alter the requirements of our project. This feedback can lead to more appropriate results. The results of this stage lead us to spend more time in data preprocessing stage applying additional filter to reduce noise and to improve the accuracy of the models.

Chapter 5

Data Understanding

5.1 General

Images related to skin cancer are necessary to complete the implementation of the project. The first step, is finding the appropriate dataset, which contains melanoma and non-melanoma images with a high percentage of correct classification. In order to achieve this, we need to select a database that includes images which are confirmed from a specialist. Due to the lack of databases related to the skin cancer problem, and some standards for dermatologic images, the quality and usefulness of skin lesion image is undermined and the selection of the database is being difficult. After research, we finally choose ISIC-ARCHIVE, which contains over 23.000 dermoscopic images of skin lesion. It is worth mentioning, that most of these images have been assayed from a dermatologist. Since, the classification of the most dermoscopic images are validated, the data specialists have the opportunity to develop a model with a highly correctness. Another important factor, which contributes to this selection is that the ISIC (International Skin Imaging Collaboration) contains the Melanoma Project, which is an academic and industry partnership, designed to facilitate the application of digital skin imaging to help reduce melanoma mortality. ISIC is being developed by proposing standards to address the technologies, techniques, and terminology used in skin images. Additionally, special attention to the issues of privacy and interoperability is given, since the issue of privacy is really important, especially for medical record like those contained in this database. Besides, ISIC developed and is still expanding an open-source public access archive of skin images to test and validate the proposed standards

and boost the research on the automated diagnosis of dermatoscopic images. This serves as a public resource for distinguishing the differences between the two categories, referred above.

Training of neural networks for automated diagnosis of pigmented skin lesions is hampered by the small size and lack of diversity of the available datasets. The ISIC tackles this problem by collecting dermatoscopic images from different populations. The result of this effort, is that the ISIC-ARCHIVE contains over 23.000 images that are categorized as benign and malignant. However, some images haven't classified yet. The ISIC has also categorized some of the images using clinical attributes, so as facilitate data specialists with a vast amount of information, which can be used to extract new knowledge for skin cancer. There are 14 attributes. The most common of there are named below: approximate age, clinical size, type of diagnosis, family history of melanoma, sex, and melanoma class.

5.2 Input Dataset

The ISIC-ARCHIVE [8] contains some ready databases with a different number of images and proportion of benign and malignant lesions (Figure 5-1). After the

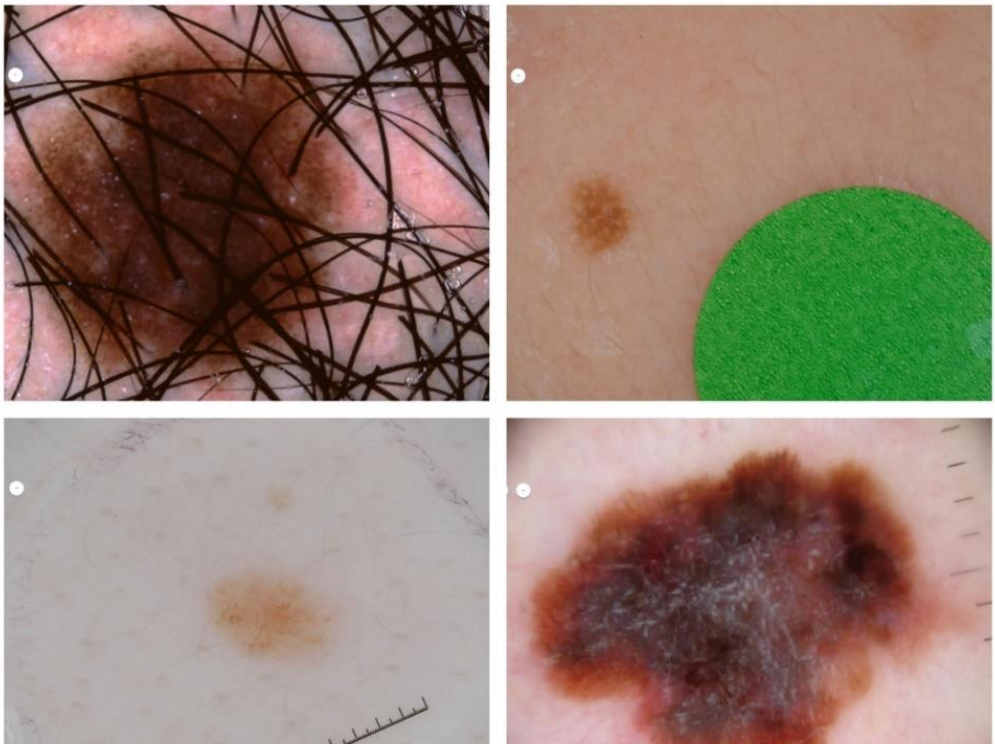

FIGURE 5-1: DATABASE CATEGORIES TOP-LEFT: MSK-1, TOP-RIGHT: SONIC, DOWN-LEFT: MSK-3, DOWN-RIGHT: UDA-2 [8].

examination of these databases, it is concluded, that the grading occurs from the content of the image.

For our project, we use 500 images, selected in random, to make our project more general. To be more specific, the database consists of 400 melanomas and 100 benign images. It is difficult to extract correctly the features of the images, needed for our implementation, due to some specific characteristics of the images. Some images contain noise related to hair, while some others have a black frame or darker corners due to the photography method and lighting conditions. Furthermore, the images have a wide range of resolution (444-600,2199x1922) so dataset images cannot be cropped in a specific size. When cropping them to specific dimensions, valuable information is lost and accuracy decreases, since some images represent only the lesion without containing a skin outline. As a result, the appropriate and accurate measures for extracted features are not taken. All of these, make it harder to implement a model for achieving high accuracy, since a small variation in a feature value can lead to misclassification.

5.3 Privacy

The ISIC-ARCHIVE images have metadata which contains some information related to the lesion that is depicted. To be more precise, the metadata contains clinical information, such as age, sex, melanocytic, lesion category (benign or malignant), and the anatomy site. It also contains some other information, such as dimensions, image ID, and the dataset name in which it belongs. The fact that, we only use the lesion category from metadata to create the training and testing dataset, special attention to privacy from the database creators is given and The personal information is prevented, due to the fact that in our implementation we only use the lesion category from metadata to create the training and testing dataset and special attention to privacy is given from the database creators. As a result, it is hard to expose any private information to connect the lesion and the owner.

Chapter 6

Data Preparation

6.1 General

Data preparation is the most important stage for our diploma thesis. This stage combines both image preprocessing and feature extraction. Image preprocessing is a highly demanding procedure, since many filters need to be applied in sequence to the whole original image, in order to dissociate the useful content from the noise and enhance some important image features for our implementation. This stage involves four main parts, grayscale conversion, noise removal, image enhancement and thresholding. It is clearly understood that, in the noise removal stage, different types of noise are necessary to be removed, before moving to the feature extraction stage. To be more precise, two common types of noise are identified, hairs and dark colored segments, due to the lighting conditions related with visual media. The implementation of our project uses two essential filters, median filter and open morphological filter. Open morphological filters have also the ability to facilitate the thresholding process, since the use of this filter erases hairs and small dots which occur after thresholding, related with dark colored areas. The image enhancement stage is an essential part before thresholding, since adjusts the image contrast and distinguish the lesion area. After preprocessing, the improved image is used for extracting the necessary features in order to generate the final database for the training of our machine learning model. For the extraction of features, two popular approaches are used, the ABCD rule and the GLCM texture features. In the following diagram the data preparation process is described (Figure 6-1).

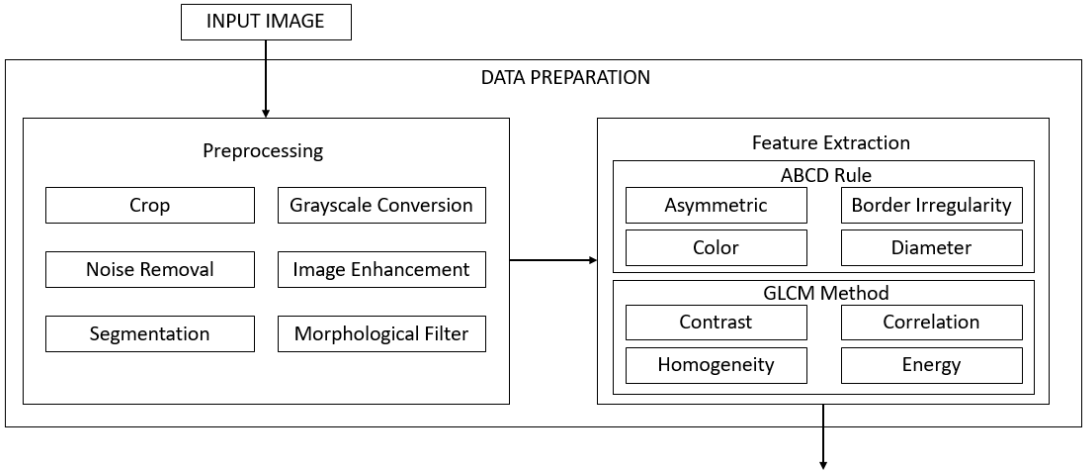

FIGURE 6-1: DATA PREPARATION SYSTEM ARCHITECTURE.

6.2 Crop

Observing the input images, it can be concluded that there is some noise in the corners of the images. This noise occurs from the lighting conditions and the machine that is used for capturing the lesion. Cropping a part of the initial image, the removal of this noise can be achieved and the quality of the final image increases.

6.3 Grayscale conversion

Grayscale image has a continues range from 0 to 255 of gray values for each pixel, where 0 is equivalent to black color and 255 is equivalent to white color. These values can represent the 256 level of light in image. Grayscale image contains brightness information while measuring the light intensity. The final image can differ in brightness graduation depending on the different equations existing in order to convert a colored image to grayscale.

From the bibliography occurs that, the two different methods used in the feature extraction stage, do not need colored image as input. Especially, the ABCD rule uses a binary image as input to extract the features and the GLCM method uses the equalized image. Taking this factor under consideration, it can be understood that, the colored image can be converted to grayscale without losing valuable information. Colored image can be converted to grayscale by using a weighted sum method. One of the most commonly used equation is represented below (6.1):

$$\text{Grayscale intensity} = 0.299R + 0.587G + 0.114B \quad (6.1)$$

The conversion of the initial image to grayscale can improve the performance of the image processing stage. This type of image is easier and faster to edit than colored. So, all filters are applied on grayscale image.

6.4 Noise Removal

This stage focuses on the removal of noises which arisen in the process of capture and the hairs that exist on human body. The detection and removing of unwanted noise is a highly demanding process since an algorithm is needed for separating the real features of an image and the features which are create the noise. The noise can corrupt true information of the image, having serious affect to final result. In bibliography, different filters to reduce noises from dermoscopic images are used [20]. The first filter called Gaussian blur and smooths an image using a Gaussian function [17]. It is a low-pass filter which removes the noise level and insignificant details by blurring the image. This is achieved be applying a Gaussian kernel. The second filter is called Median filter which is widely used in projects related with skin cancer since it can prevent edges [65]. In these projects, the Median Filter is selected, since edges invariant is a key factor to extract correct features from ABCD rule. The median filter is a non-linear filter and it is commonly used in image processing [18]. The appealing characteristic of this filter is the conservation of the edges and the ability of reducing impulse noise. The median filter sorts the pixel's values and take the median value under the kernel area and replace the central pixel with this median value (Figure 6-2).

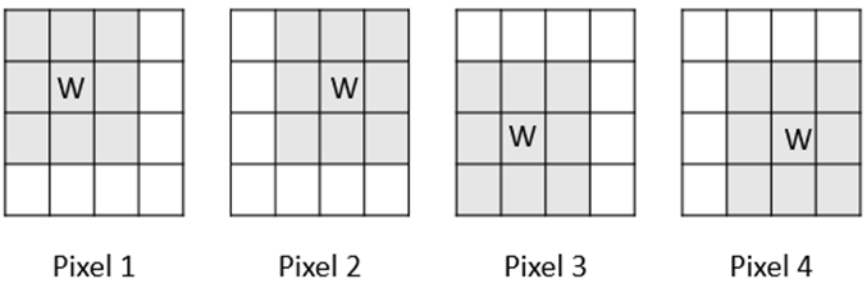

FIGURE 6-2: MEDIAN FILTER USING 3X3 KERNEL SIZE.

Different approaches use kernel with different size and shape since the reduction of noise depends on these. This is an important disadvantage. If the number of pixels

which is characterize as a noise is greater than N(N+N)/2 and the kernel size is NxN, then the noise cannot be deleted, as a result we need to increase the size of the kernel [19]. On contrast, if the number of noise pixels is smaller than N(N+N)/2 and the kernel size is NxN, then the pixel's values can corrupt since the algorithm assign irrelevant values, as a result we need to decrease the size of the kernel. To clearly understand the algorithm, we illustrate below an analytic example of a median filter using a 3x3 kernel.

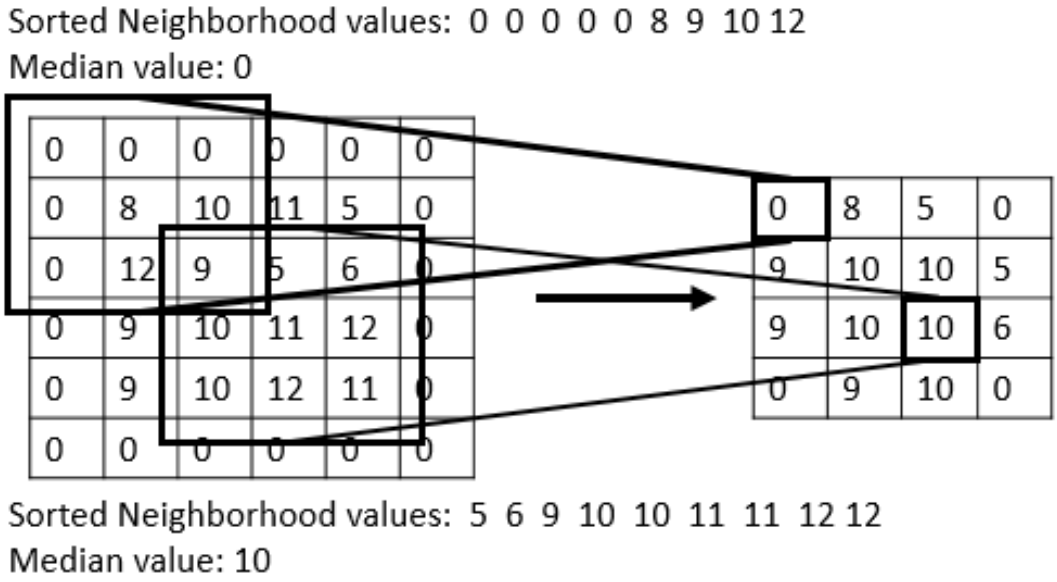

FIGURE 6-3: MEDIAN FILTER EXAMPLE.

According to the above example (Figure 6-3), it can be easily understood the implementation of the median filter algorithm. We simplify our example expanding the grayscale image matrix with one-pixel size border to see the analytically the corner cases. So the input image has (N+2)x(N+2) dimension and the output has the initial NxN dimension. As it can be seen the 3x3 kernel size run through each pixel in order to replace the value of the central pixel of the kernel area with the median value of the neighborhood. To be more precise, the first pixel that the algorithm calculates, is located at the point (1,1) of the input image and saved in the output image at the point (0,0) to reconstruct the image to initial dimensions. Applying the median filter kernel to this pixel we marked an area of neighborhood pixels and sort these pixel's values in order to take the median value. As a result, the sorted neighborhood values are 0, 0, 0, 0, 0,8, 9, 10, 12 and the selected median value is 0.

6.5 Image Enhancement

The purpose of this stage is to increase visibility of the feature of interest, based on our project, the adjustment and enhancement of the contrast is needed [23]. Image enhancement techniques change the pixels' intensity of the image so as the output images to looks better. It is an important step to prepare the image for threshold stage and the GLCM method. The result of this stage is to make more sharpen the image border and improve the brightness. After applying this filter, it is easier to separate the background and the object of interest which is depicted. Nowadays, there are many image enhancement techniques suitable for different tasks with specific requirements. The most used methods are classified into Linear and Non-Linear contrast enhancement techniques [21]. In our implementation, the histogram equalization method to improve the image enhancement is selected. The above images (Figure 6-4) are an example where the result of this filter to the histogram of the image can be observed.

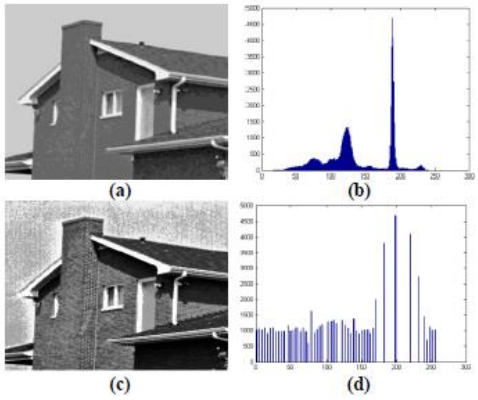

FIGURE 6-4: HISTOGRAM EQUALIZATION: (A) INPUT IMAGE, (B) HISTOGRAM OF THE OUTPUT IMAGE, (C) OUTPUT IMAGE BY THE HISTOGRAM EQUALIZATION TECHNIQUE, (D) HISTOGRAM OF THE OUTPUT IMAGE [25].

The histogram equalization is the most common method for image enhancement [22], since its simplicity and better performance with all types of images, compared to other existing methods. Histogram equalization replaces the output image with uniform distribution of pixel and it flattens and stretches the image [25]. The operations of the Histogram Equalization are performed by remapping the gray levels using the cumulative distribution of the input gray levels. The step of the implementation of this algorithm is simple. The first step is to calculate the histogram of the image. After that we use the cumulative distribution function and the normalization of this function gives the output image. The general histogram equalization formula is the following (6.2):

$$h(v) = round\left(\frac{cdf(v) - cdf(min)}{M * N - cdf(min)}\right) * (L - 1) \quad (6.2)$$

where cdf(min) is the minimum value which occur from the cumulative distribution function, M*N is equal to the pixel of image and the L is equal to the number of gray levels. In our case the L take the value of the 256 since the initial image converted to grayscale and the pixels' value range is from 0 to 256. The equalization formula scales the values of the grayscale pixels from 0 to 255.

6.6 Image Segmentation

The purpose of image segmentation stage is to separate the background and the lesion by converting the image to binary and depict the area of interest. This stage is dominant for our implementation, since the binary image is used for extracting the most features from the ABCD rule. There are many approaches to implement this filter. Some of them combine machine learning algorithms in order to locate the region of interest, separating from the background and other more simply methods, called thresholding algorithms, segment the image. There are many thresholding algorithms. Among them, the Otsu's [26] and global thresholding method [27] have been distinguished. The main difference between these two approaches, is that the Otsu's algorithm performs automatic image thresholding. It searches for the most efficient thresholding value that minimizes the intra-class variance instead of the global algorithm which requires a predefined value of the threshold. Finding the global thresholding value could be an easy task, if the background and the area of interest have distinguished values of brightness in the grayscale image. The global thresholding method is quite easy for implementation. In this method the value of each pixel is checked and replaced with a white pixel if the intensity is greater than the global thresholding value or with a black pixel if the pixel's value is less.

From the thresholding process the output is a white background and a black area for the lesion. Another essential step in the stage of segmentation, is the inversion of the pixel's values of the image. The inverted image is necessary for the implementation of the morphological filter. In order to extract some of the features of the ABCD rule, some functions of the OpenCV library are used. Black background to locate the region of interest with white color is required by these functions.

6.7 Morphological Filter

The purpose of this stage is to reduce any noise from the segmentation image, that have not been removed from the previous image preprocessing step. The morphological filters follow the thresholding algorithms to improve the final result. Morphological filters typically work with binary images. The filter which is more suitable for this task is the open filter. This filter is the combination of two other filters, the erosion and dilation. To be more specific, open filter is implemented as the process that requires an erosion filter followed by a dilation filter. This sequence can help as to erase white small object without losing information for the region of interest. The implementation of the open filter requires a kernel, the size of which differs regarding the size of the unwanted object.

So what happens when applying the erosion filter in a binary image? While the boundary of the white object is decreased, small or thickness white object will be discarded. It has to be mentioned that, the decrease of the white objects depends on the kernel size. The kernel is applied to each pixel and replaces the initial value with 1 when all the pixels under the kernel area is 1, and with 0 for every other case [29]. On the following example (Figure 6-5), the effect of the erosion filter using a 3x3 kernel can be noticed.

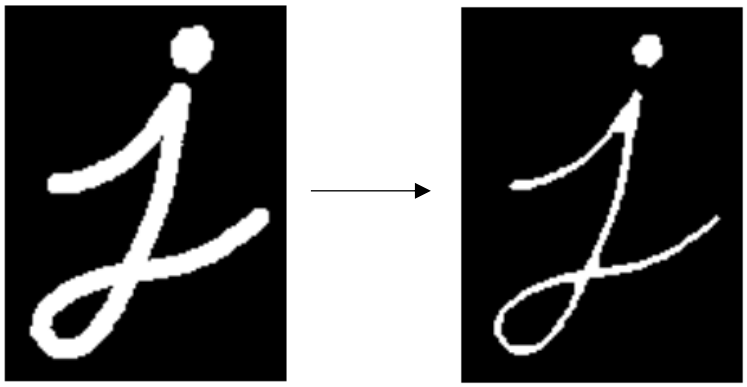

FIGURE 6-5: A EXAMPLE IMAGE OF THE EFFECT OF EROSION FILTER. THE KERNEL SIZE IS 3X3 [29].

The dilation filter has the same properties as the erosion filter with the only difference that, a pixel has value 1, when at least one pixel with value 1 under the kernel size exists. From the figure (Figure 6-6) is obvious that, applying this filter the area of white object is increasing. The white area of the lesion has decreased due to the previous filter as a consequence this filter can reconstruct the lesion to the initial size which is

important to calculate the features from the ABCD rule correctly. An example of dilation, in order to understand the functionality of the filter is proposed.

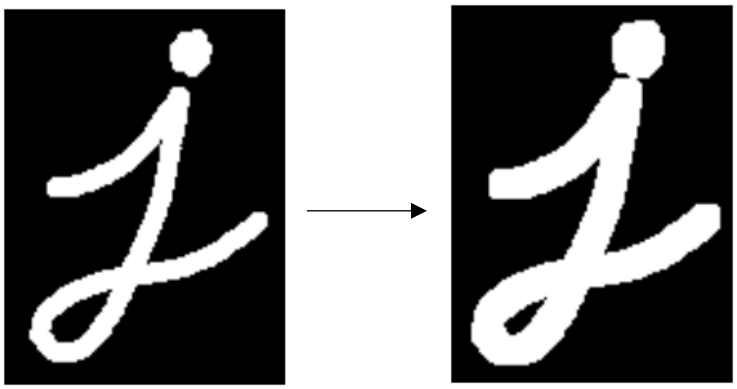

FIGURE 6-6: A EXAMPLE IMAGE OF THE EFFECT OF DILATION FILTER. THE KERNEL SIZE IS 3X3 [29].

6.8 ABCD Rule

The ABCD rule is one of the methods used in feature extraction stage. The ABCD rule (Asymmetry, Border Irregularity, Color and Diameter) of dermoscopy was the first algorithm which has been developed for helping scientists separating benign from malignant tumors. This algorithm was described by Stolz and created to answer the critical question in desmoscopy of whether a skin lesion is benign or malignant [33]. This method is quite simply to learn and to apply since it is based on four criteria for the classification of the lesion. The ABCD method is widely used in scientific research, while the meticulous study of this method can improve the diagnostic performance of lesion evaluation. In order to extract the final result, this technique combine four main criteria which are asymmetry, border, color and diameter. However, there are some variations related with the criteria of this method in order to fit the project requirement, since the feature extraction is the main part to categorize the lesion to benign and melanoma.

Asymmetry (A): From the medical records occur that the melanomas are developed in an anarchic way, while benign lesions are symmetrical [31]. In order to compute the asymmetry index, we find the difference between the lesion image and its horizontal flip and the difference between the lesion image and its vertical flip. After that, we calculate the ratio between each difference and the total lesion region and keep

their average values. The asymmetry can be representing by the following mathematic equation (6.3).

$$Asymmetry = \frac{\frac{(image_{area} - hflip_{area})}{image_{area}} + \frac{(image_{area} - vflip\_area)}{image_{area}}}{2} \quad (6.3)$$

Border Irregularity (B): According to dermatologist, melanomas are characterized by much border irregularity, on the other benign tumors are typical defined by clear border [33]. The evaluation of the border related with the existence of a sharp pigment at the periphery of the lesion. The border irregularity is computed by the following equation (6.4).

$$Border\ Irregularity = \frac{Perimeter^2}{4 * \pi * image_{area}} \quad (6.4)$$

Color (C): Malignant lesion are represented by several colors. If a lesion is represented by five to six color may be characterized as melanoma. The verified colors are: Black, Dark Brown, Light Brown, Red, Blue and White [33].

Diameter (D): The diameter is one of the four criteria. A lesion with diameter of more than 6-7 mm is typically defined as a melanoma. We calculate the diameter as the diameter of a circle with lesion area. Additionally, we convert the diameter in mm using the fact that 1 pixel is equal to 0.265 mm.

6.9 GLCM Method

The Gray Level Co-Occurrence Matrix is a texture analysis method. The texture analysis computes the features from the statistical distribution of observed combinations of intensities at standard locations. The number of intensity points in each combination, statistics are categorized into 3 subcategories: first-order, second-order and higher-order statistics [36]. According to the previous classification, the GLCM method is included to the second-order statistical texture features. This approach is commonly used in many applications, since the higher-order textures try to identify the relationships among three or more pixels which needed more execution time and the implementation is difficult.

The GLCM is a popular method for extracting texture features, since it can calculate many features and it is easy to be implemented. Using the co-occurrence matrix, fourteen characteristic texture features can be extracted from the probability matrix. These features are Angular Second Moment (ASD), Entropy, Sum Entropy, Difference Entropy, Contrast, Correlation, Information Measures of Correlation, Maximal Correlation Coefficient, Sum of Squares, Inverse Difference Moment (IDM), Sum Average, Sum Variance, Difference Variance [37]. From these features, only four texture features are used in our experiment: Correlation, Homogeneity, Energy and Contrast were extracted from the enhancement images.

The Correlation describes the relationship between pixels in each row and column of the image. This relationship that occurred, measures the joint probability occurrence of pairs pixels. The correlation can be computed using the following formula (6.5).

$$Correlation = \sum_{i,j=0}^{N-1} [\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}] \tag{6.5}$$

The Homogeneity measures the closeness of the distribution of the specific pair of pixel in the GLCM. To compute the Homogeneity, the following equation is used (6.6).

$$Homogeneity = \sum_i \sum_j \frac{1}{1+(i-j)} * P_{i,j} \tag{6.6}$$

The Energy expresses the repetition of pixel pairs in the image and calculated as the square root of the sum of squared pixels (6.7) [34].

$$Energy = \sqrt{\sum_{i,j} P_{i,j}^2} \tag{6.7}$$

The Contrast measures the local variations present in the image, so higher values identify large local variations (6.8).

$$Contrast = \sum_i \sum_j (i-j)^2 * P_{i,j} \tag{6.8}$$

The purpose of the feature extraction using the ABCD rule and the GLCM techniques is to compress the original image set to specific features which is used to classify the different lesion image.

Chapter 7

Modeling

7.1 General

Nowadays, the advent technology, especially in the field of the artificial intelligent, facilitates the research in all disciplines of science. The use of the AI contributes in developing many applications related with every-day tasks. Additionally, applying different AI algorithms can detect patterns, in order to extract new knowledge. Using these patterns, demanding tasks can be solved, while the extracted knowledge can lead us to solve many other new problems. It is an evidence that, the use of the AI helps society to overcome many crucial problems. AI algorithms are used in many applications related with health. For example, different methods are used in vaccines testing to observe and identify potential side effects of different chemical substances. AI algorithms are also implemented in order to identify the type of disease by observing the symptoms. The most important applications are those which are related with different types of cancer, since cancer is a high deadly disease and the early detection can help the treatment. Applications related with skin cancer can be used by everyone, since medical equipment is not required for capturing the lesion. A camera is the only mandatory tool. According to bibliography, depending on the type of the input dataset, different models can be used, in order to classify a lesion. When the input is a full image, applications with CNN networks are being developed, for classifying a lesion. Otherwise, when inputs are some extracted features of the lesion image, simpler models, such as Support Vector Machine (SVM) are being applying. In our thesis, these kind of inputs have been selected for implementation. SVM decision boundary and

some other modified models are used, to investigate the appropriateness of each model based on the input type. To be more specific, we select as training models the following: Support Vector Machine with RBF kernel, NuSVM, Linear SVM, Multi-layer Perceptron Classifier, Decision Tree Classifier and Random Forest Classifier.

7.2 Data Format

Before starting analyzing the training models used in our project, the way that the extracted features have been saved, is being described. From the stage of feature extraction occurs that, the extracted features are ten from both methods ABCD and GLCM. In order to evaluate the utility of its method, and finally create the most accurate model, which is the combination of the two methods, three different inputs can be created. The first, contains only the six values, which are extracted from the ABCD rule, while the second one contains the four values from the GLCM texture method. The third one contains all the ten values from both methods. The three inputs database are formed as tabular database. After the features extraction step, the initial database of images is converted into a tabular database consisting of the extracted features. Tabular data consist of columns and rows and they are structured into rows. Each row contains the values of the features for each lesion. The inputs databases are saved as Comma Separated Values (CSV) files. The SCV file is a text file consisting of a list of data, and its structure is quite simple, since the data are separated by commas. The first row describes the features that are contained in its column, while the other rows contain the value of the features which are extracted from each image. Another important thing is that, csv files can be supported in spreadsheet programs, which make them easier to read and observe the output.

7.3 Support Vector Machine

Support Vector Machine (SVM) is a linear model, which is used for regression and classification problems. This model is one of the most popular machine learning algorithms and was developed in 1990 [39]. It is a supervised algorithm and requires large process, in order to extract the final result. As a consequence, it is used with small

dataset. The main idea of the SVM is to generate a line which separates the input data into two different classes. To be more precise, this line refers to a 2-D classification problem. When referring to a 3-D problem a plane is generated and for higher dimension, a hyperplane [38]. Using this model, data related with linear and no-linear problems can be separated (Figure 7-1). Although there are many potential separating lines, the model must predict only one line. After that, the question how the model selects the most appropriate line, occurs. To select the most suitable line, the model calculates the distance between the separating line and the closest points to the line. The closest points are called "support vectors" and the distance is called "margin". The target of this model is to maximize the margin, since the probability to correctly separate the points in their target classes or region is higher, when the support vector has large distance from the line. The use of the margin indicates that the data is linearly separable, but in many cases this is not a feasible hypothesis. As a result, an update of this hard approach, which is called soft margin SVM, has been developed, in order to skip few outliers to separate the input data. One more important update is that, different types of kernel can be used, when using the SVM. These kernels, it can be used to classify a little bit more irregular data. We can have linear kernel, polynomial kernel and radial basis function kernel. The formula when using kernel type is being described below (7.1) and the parameter "K" represents the function of the kernel.

$$\max_{a} \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j K(X_i^T . X_j) \tag{7.1}$$

The linear and polynomial kernels are less time consuming, since less computations are needed and in most cases less accuracy is achieved, in contrast to other kernel types, like the RBF. The Radial Basis Function is a very popular kernel in Support Vector Classifier. The RBF kernel function is the following equation (7.2):

$$K(x, x') = e^{\left(-\frac{||x-x'||^2}{2\sigma^2}\right)} \tag{7.2}$$

where $||x - x'||^2$ is the Euclidean distance between the two points x and x'. The Sigmoid kernel is commonly used in SVM and is being described from equation (7.3):

$$K = \frac{1}{1 + e^{-x}} \tag{7.3}$$

The SVM classifier requires two other parameters: the gamma and C. The gamma parameter affects the curve of the decision boundary. Small values of gamma parameter mean small curve of the decision boundary. The C parameter is related with the tolerance of the misclassified points. Small values of C indicate large tolerance to misclassified points [41].

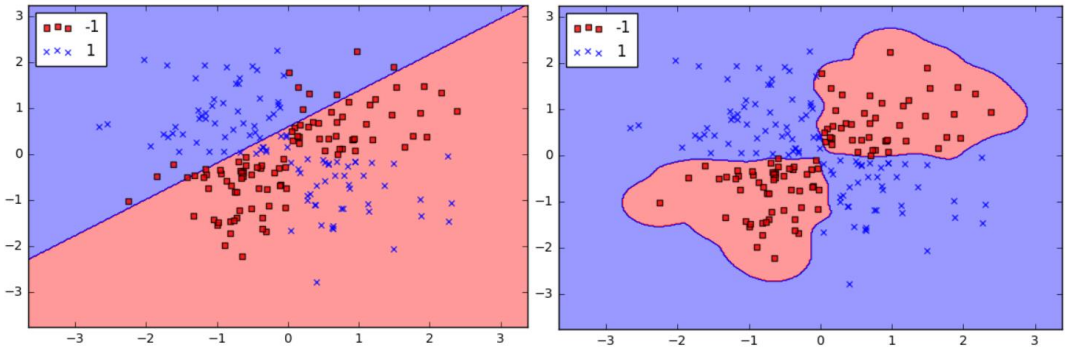

FIGURE 7-1: THE LEFT DIAGRAM DEPICTS A SVM WITH LINEAR KERNEL. THE RIGHT DIAGRAM DEPICTS A SVM WITH RBF KERNEL [41].

Additionally, the Nu-SVM classifier is a parameterization of the C-SVM classifier. The nu parameter is equal to 0.5 by default, and it is a boundary on the fraction of margin errors and support vectors, especially an upper bound for margin errors and lower for support vectors [42]. This parameter can take values in the range of 0 to 1 (interval (0,1]).

7.4 Multi-Layer Perceptron Classifier

Multi-Layer Perceptron is a supervised machine learning algorithm and is included to feedforward artificial neural network (ANN). The MLP [43] uses the training data to generate a function, which converts the R^i to R^o, where i is the dimension of the input and o is the dimension of the output. This algorithm uses Backpropagation which is a supervised learning method and differs from logistic regression. The MLP consists of at least three layers, the input layer, the output layer and an intermediate activation layer called hidden layer. Every layer consists of a certain number of neuron. It is fully connected (Figure 7.2); each neuron is connected to every neuron of the following layer with certain weights. Each neuron in the hidden layer performs a weighted linear summation with the values of the previous layer

followed by an activation function. The output transforms the values of the previous layer into output values.

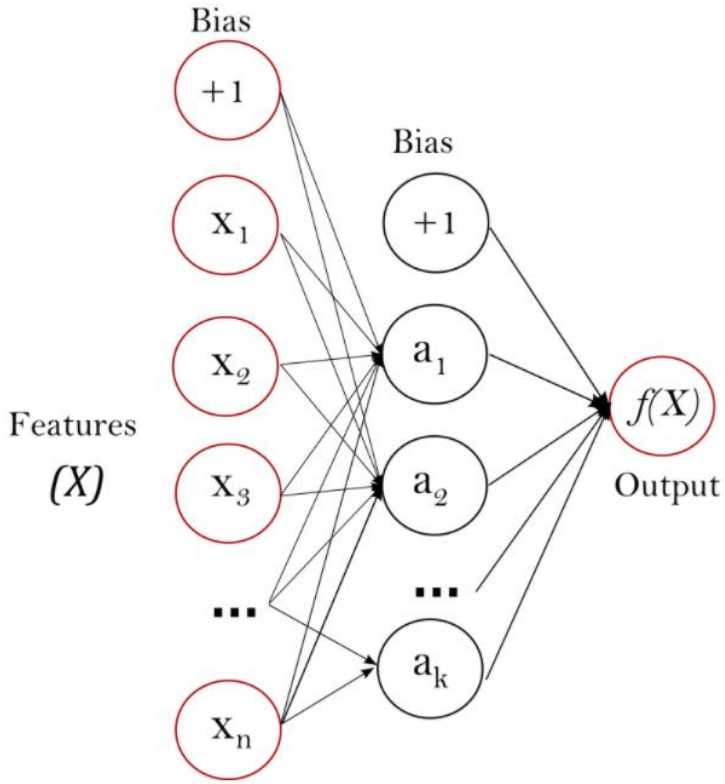

FIGURE 7-2: ONE HIDDEN LAYER MULTI-LAYER PERCEPTRON [43].

In our implementation, the alpha parameter for regularization, is used. This parameter helps in avoiding overfitting. Rectified Linear Unit (ReLU) [44], logistic sigmoid and hyperbolic tangent, are used as activation Functions. The sigmoid function is very popular, but sometimes the tanh function is more preferred. Recently, the ReLU is the most used activation function. The range of the output values is zero to infinity $[0, \infty)$ (Figure 7-3). The output value is equal to zero, if the input value is less than zero, and equal to input, if the input value is above to zero. The disadvantage of this activation function is that the negative values turn into zero. As a result, the model can fail to fit the data properly. To avoid this limitation, the leaky ReLU helps to increase the range of the output values (Figure 7-3). This activation function multiplies the input value with a small number α when the input value is less than zero. We can notice the functionality of these activations function in the following figure.

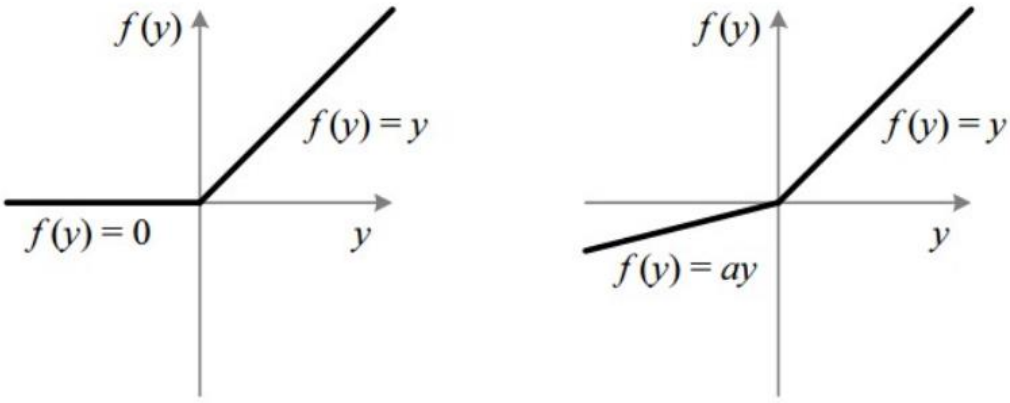

FIGURE 7-3: IN THE LEFT DIAGRAM IS THE RELU ACTIVATION FUNCTION. IN THE RIGHT DIAGRAM IS LEAKY RELU ACTIVATION FUNCTION [44].

The logistic sigmoid function is one of the other two kernel types [45]. It is a common S-shaped curve (Figure 7-4). The logistic sigmoid is the standard logistic function which is described from (7.4):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{7.4}$$

The hyperbolic tan function is a kernel type that has been selected for our project. The tanh function is a shifted function of the sigmoid. The sigmoid values range from 0 to 1 and the tanh values range from -1 to 1 (Figure 7-4). The tanh is described from the (7.5).

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{7.5}$$

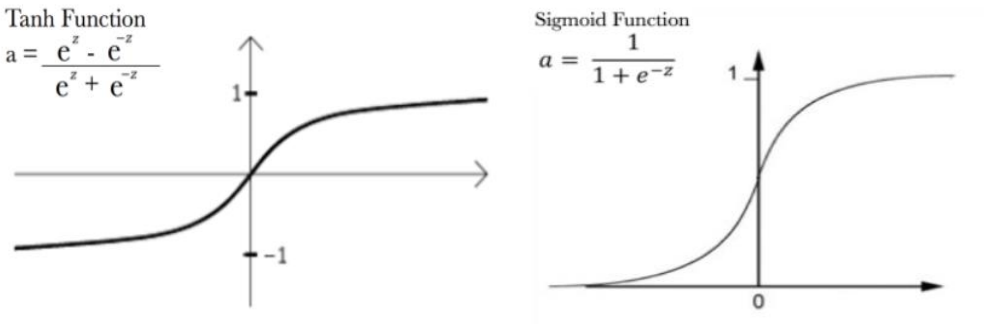

FIGURE 7-4: IN THE LEFT DIAGRAM IS THE TANH ACTIVATION FUNCTION. IN THE RIGHT DIAGRAM DISPLAY THE SIGMOID ACTIVATION FUNCTION [45].

7.5 Decision Tree Classifier

Decision tree classifier is a non-parametric supervised learning method used for classification problems and it is widely used in data mining. Decision trees is a simple classifier. Goal of this model is to separate the input data into different classes using the classification features. To categorize the given set of data, a combination of mathematical and computational methods, as shown in Figure 7.5, is used.

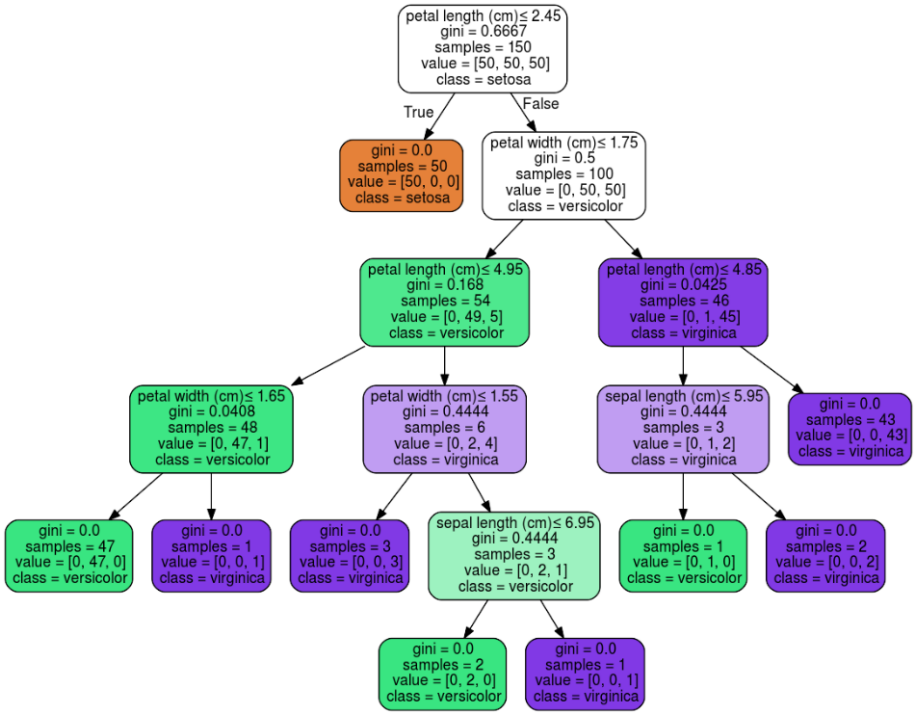

FIGURE 7-5: THE STRUCTURE OF A DECISION TREE CLASSIFIER [47].

The decision trees consists of three different parts: the nodes for the value of the classification parameters, edges that connect the current node with the next layer node, based on the result, and the leaves that represented the target values, which are the final output [46]. There are two distinct categories of the decision trees related to classification problems or regression problems. The Decision tree classifier procedure is to separate the training data into smaller and smaller subset and gradually formed the decision tree. The above figure (7-5) represents the structure of a Decision Tree Classifier. The complex of the tree is related to the depth. More complex trees can fit the training model more appropriate. This model spilt the data on the feature in order to achieve the largest information gain. There are different methods to calculate the information gain such as Entropy (7.6), Misclassification error (7.7) and Gini index (7.8), the equations of them are given below:

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t) \tag{7.6}$$

$$Classification\ Error = 1 - \max\left[p_i(t)\right] \tag{7.7}$$

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2 \tag{7.8}$$

The decision tree starts from the initial point which is called root node. This node contains the whole training dataset. The algorithm uses different metrics for measuring the best selection of the classification feature to split the data into smaller groups of data. These metrics are applied to each subset and the result indicates the quality of the split. This step is being repeated until the data is separated into the target categories which is the leaves of the decision tree.

7.6 Random Decision Forest

Random Decision Forest [54] is an ensemble method of learning that can be used for classification and regression problems. Tim Kam Ho [51] has introduced the first algorithm at his paper in 1995 for the random decision forest as an alternative classifier, instead of the traditional decision trees classifiers that require a very high execution time, but present a limitation related to the complexity of the models. The purpose of an ensemble algorithm such as Random Forest algorithm is to combine the predictions of several models with different attributes, in order to improve the estimator. The ensemble methods can be distinguished into two categories: averaging methods which contain the random decision forest and boosting methods [50]. It can be assumed that each classifier in random forests is a decision tree classifier and the total number of the classifiers form a forest (Figure 7-6). Each decision tree is developed by using random selection of parameters at each node; as a consequence, the split is unique. Each tree depends on random values of attributes and with the same distribution for the whole tree [48]. The decision trees are highly dependent on the

training data. As a result, small changes related to the input dataset can modify the tree structure. The returned tree is the most voted in our prediction. The basic idea behind random forest is the large number of relatively uncorrelated trees and this low correlation is the benefit of the model. The final predicted model is more accurate than the individual, since every tree protects the other from their errors. Some trees can make a wrong prediction, while others can make a correct. So, the trees, as a group, are able to follow a path for the correct direction [49]. A key concept to random decision forest is to generate strength individual classifiers with very correlation, since the accuracy of the RDF depends on the strength of each tree.

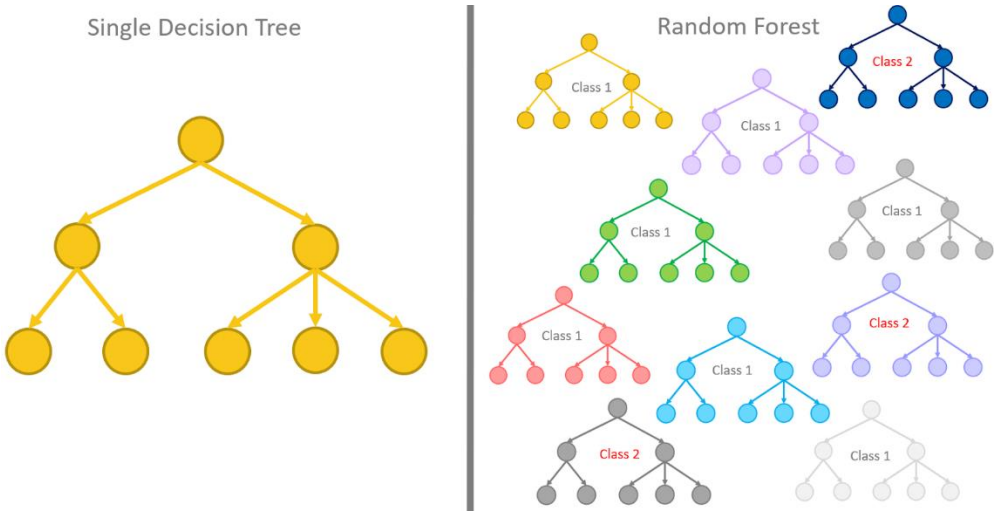

FIGURE 7-6: DIFFERENCE BETWEEN A SINGLE DECISION TREE AND A RANDOM FOREST [51].

Additionally, there is a modified random forest which uses random linear combinations of the training attributes. The difference between the other random forests, is the creation of new features, which occurs from the linear combination of the existing attributes, instead of selecting a random attribute at each node to create multiply trees. The generated attributes help the algorithm to split the data in a more appropriate way, increasing the accuracy of the final model. This form of random forest is suitable for training data with a few attributes and the creation of new features can maintain the correlation between individual trees in a low range.

Chapter 8

Results

8.1 General

This chapter presents and analyzes the results of our software implementation. First of all, the results of the data preparation stage are presented; the results of image processing and features extraction methods are presented and commented. In addition, the filters which are used for the image processing stage and their specific characteristics are described in detail and the results of the modeling stage are provided. The selection of each classifier and the effect of the different model parameters to the accuracy are described. Finally, a short discussion to explain the results is provided.

8.2 Data Preparation Results

The first step is cropping the input images so as to reduce the noise that is displayed at the corners. After some experiments, it is concluded that the correct percentage of cropping is 8% for each dimension. To be more precise, 4% of the initial pixels are cropped of each side. After cropping the image, it is necessary to convert the RGB image to Grayscale image, Figure 8.1-B. For this conversion the equation (6.1), which is described in data preparation chapter, is used. To reduce the noise, the Median Filter with a 3x3 kernel size is selected. This selection is based on the ability of Median filter to prevent the edges. Observing the Figure 8.1-C, it is noticed that the hairs are reduced.

The input image for the GLCM method is the results of the histogram equalization filter. The histogram equalization filter increases the adjustment of the image. The goal of this filter is to scale the pixels' values in the range [0,255]. After this step the area of lesion is emphasized. The results image is the D image that is displayed in Figure 8.1. The next step is to segment the image, using thresholding techniques. The Otsu's method is firstly selected, but after some experiments is concluded that, the global thresholding method achieves the same results, since the selection of thresholding value is an easy process, due to the fact that, the pixel values of the lesion are distinct from the background. To be more specific, the thresholding value is 100. As a result, the ability of the Otsu's method to find the propel thresholding value is unnecessary. The binary image is shown on the Figure 8.1-E. It can be seen that, the background is white and the lesion area is black. Some functions of the OpenCV, which are used to extract some features values, in order to work properly, require a black background and a white lesion area. One of the most important function is the cv2.findContours which is used to locate the lesion area. So, the inverted image is necessary for our implementation. The result image after this stage is represent at Figure 8.1-F.

From the inverted image, it is noticed that some noise is still existing, so a further image process is required. To improve the thresholding stage and reduce the existed noise, the open morphological filter is added to the process. The selected kernel size for open filter is 5x5. As already described in Chapter 6, the open filter uses an erode filter to reduce the size of the white objects included the lesion. To reconstruct the lesion to the initial size, the dilation filter is applied. Figure 8.1-G depicts that the noise of the previous picture is almost removed.

The ISIC_0000022 is the input image for the image preparation process. This image is the initial RGB image which is displayed in the Figure 8.1-A. This image is classified as malignant.

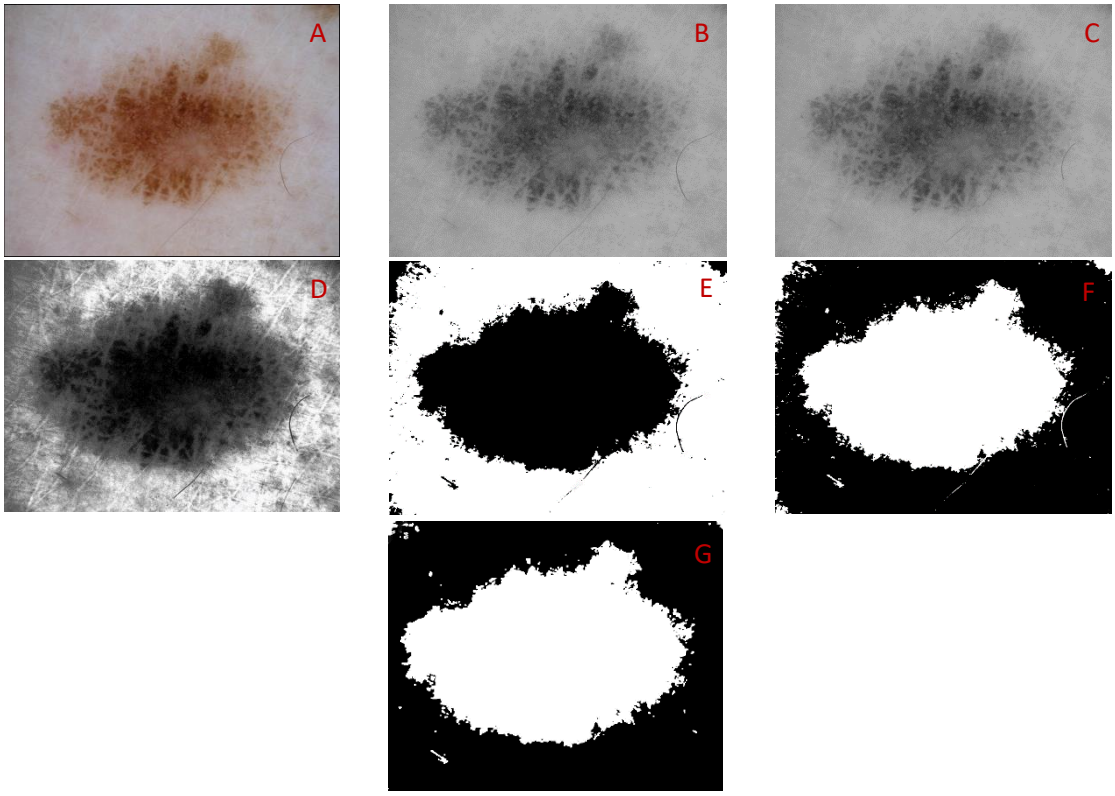

FIGURE 8-1: IMAGE PROCESS ISIC_0000022 [8].

One of the most important steps in our project is the values of the extracted features. With this features the initial database of images is converted to the final tabular database, which is used to train the classifications models. The values of the extracted features of the two different images are represented below. One of these images is classified as a malignant, while the other image classified as benign. By using the ABCD rule and the GLCM method, six and four features are extracted correspondingly. To extract the ABCD rule, the OpenCV library is mainly used. The cv2.findcontours function returns the boundary of the lesion. Using this contour, we can calculate the centroids and the perimeter of the lesion. Also, with contours we can calculate the area of the lesion and the diameter. The border irregularity is calculated by using the perimeter and the area. To calculate the asymmetric, functions from the numpy library are used. Color variegation values are calculated by using the initial RGB image, instead of other extracted features that use either grayscale enhancement image or binary image. To extract the GLCM features we use the greycomatrix function which is provided by the skimage library. Furthermore, to extract each features from the result of the previous function we use the greycoprops which is also provided by the skimage library.

The extracted features values for both ABCD rule and GLCM method are represented below. The extracted feature values for malignant lesion is extracted from the ISIC_0000013 and for the benign lesion using the ISIC_0000000 image. From the extracted features, it can be concluded that the malignant lesion diameter is significant higher than the benign lesion diameter. Also, the color variegations of the benign lesion have almost the same values, instead of the malignant color variegation. In addition, the border irregularity of the malignant lesion must be higher, since the borders are not clear. Furthermore, the malignant lesions are formed in anarchic way and the asymmetric feature take small values. From the extracted features of the GLCM method, the correlation, homogeneity, and energy values are greater for malignant than benign lesions. The contrast extracted value is significant small for the malignant lesions.

The ABCD rule results:

Malignant Lesion:

Asymmetric: 0.11

Border Irregularity: 11.462

Color Variegation Red: 0.075

Color Variegation Green: 0.158

Color Variegation Blue: 0.23

Diameter mm: 568.13

Benign Lesion:

Asymmetric: 0.435

Border Irregularity: 4.88

Color Variegation Red: 0.12

Color Variegation Green: 0.135

Color Variegation Blue: 0.132

Diameter mm: 149.68

The GLCM Texture Method results:

Malignant Lesion:

Correlation: 0.998

Homogeneity: 0.961

Energy: 0.233

Contrast: 0.078

Benign Lesion:

Correlation: 0.981

Homogeneity: 0.824

Energy: 0.181

Contrast: 0.783

8.3 Modeling Results

In our project, six different Classification Models are built, to compare the accuracy of each model before selecting the model with the higher correctness. Before

starting improving the models, the selection the appropriate size of database is needed. From the images that are provided by the ISIC Archive, four different datasets consisting of different number of benign and malignant images are formed. The first database was developed with 200 images, more detailed this database contains 100 benign images and 100 malignant images (Database A). In order to improve the accuracy of the models, the number of the database is increased to 800 images, 400 benign images and 400 malignant images (Database B). Since the accuracy of the models, is maintained in low level, an unbalanced dataset, to notice the behavior of the models, is used. When training the models with an unbalanced dataset, it can be noticed that the accuracy of the most models is increased. These databases contain 500 images. In the first case, the database contains 100 malignant images and 400 benign images (Database C) and in the second case the dataset consists of 400 malignant images and 100 benign images (Database D). In the following table the accuracy on each case is represented. From these table (Table 8-1, 8-2, 8-3), the most suitable input database is the last form which contains 400 malignant and 100 benign images.

TABLE 8-1: ABCD THE ACCURACY OF EACH CLASSIFIER RELATED WITH THE FORMATION OF THE DATABASE.

Training Model	M:100/B:100	M:400/B:400	M:100/B:400	M:400/B:100
	Accuracy	Accuracy	Accuracy	Accuracy
Support Vector Machine	0.73	0.74	0.8	0.86
Nu-SVM	0.65	0.7	0.72	0.81
Linear SVM	0.51	0.52	0.8	0.74
Multi-layer Perceptron Classifier	0.5	0.47	0.8	0.25
Decision Tree Classifier	0.68	0.68	0.68	0.86
Random Forest Classifier	0.73	0.74	0.78	0.87

TABLE 8-2: GLCM THE ACCURACY OF EACH CLASSIFIER RELATED WITH THE FORMATION OF THE DATABASE.

Training Model	M:100/B:100	M:400/B:400	M:100/B:400	M:400/B:100
	Accuracy	Accuracy	Accuracy	Accuracy
Support Vector Machine	0.6	0.6	0.8	0.74
Nu-SVM	0.56	0.44	0.58	0.22
Linear SVM	0.61	0.57	0.8	0.73
Multi-layer Perceptron Classifier	0.56	0.55	0.78	0.82
Decision Tree Classifier	0.46	0.57	0.66	0.7
Random Forest Classifier	0.6	0.59	0.77	0.8

TABLE 8-3: THE ACCURACY OF EACH CLASSIFIER USING ABCD AND GLCM METHOD RELATED WITH THE FORMATION OF THE DATABASE.

Training Model	M:100/B:100	M:400/B:400	M:100/B:400	M:400/B:100
	Accuracy	Accuracy	Accuracy	Accuracy
Support Vector Machine	0.79	0.77	0.81	0.86
Nu-SVM	0.79	0.75	0.76	0.82
Linear SVM	0.63	0.55	0.8	0.72
Multi-layer Perceptron Classifier	0.54	0.4	0.78	0.75
Decision Tree Classifier	0.69	0.7	0.71	0.82
Random Forest Classifier	0.8	0.75	0.81	0.91

The three previous tables display the result of the two different extracted methods and the combination of them. Observing Table 8-1 and Table 8-2, it can be noticed that, these training models achieve lower accuracy with symmetric dataset than asymmetric. Using database D, which is the most suitable, it is obvious that the models can fit more appropriate in the training examples, leading to a more accurate classifier. When input database consists of the features that have been extracted from both methods, we can conclude that the accuracy is not be affected in great degree from the input dataset. However, the Database D can generate a more accurate model. In addition, the training examples of the Database C, are linear separated, since the linear SVM generates a classifier with higher accuracy.

After the previous analysis, it is concluded that the combination of the ABCD rule and the GLCM method lead to a more accurate model. Also, it is easily understood that the most appropriate dataset for our project, is the database that contains 100 benign and 400 malignant images. All models are parameterized, as a consequence, to improve the accuracy, the parameters of the models are modified. From the previous discussion about the classification algorithms, it can be easily understood that there are different approaches that can be used to fit more appropriate each model to the training data. In order to describe the effect of these modifications to the parameter of each model, the last database is used, since the other databases have lower accuracy.

First of all, the three types of the Support Vector Machine algorithms are being analyzed. The three parameters that can seriously affect the model are changed. The first parameter, is the kernel type that can be used in the method. Another important

parameter is called gamma, which affects the curve of the decision boundary, and the final parameter is the C regularization parameter.

TABLE 8-4: THE ACCURACY OF SVM RELATED WITH THE MODEL PARAMETERS.

SVM Parameters	Accuracy
Kernel type: RBF, C: 1, gamma: auto	0.81
Kernel type: Sigmoid, C: 1, gamma: auto	0.75
Kernel type: RBF, C: 1, gamma: scale	0.85
Kernel type: RBF, C: 1000, gamma: auto	0.82
Kernel type: RBF, C: 1000, gamma: scale	0.86

From the above Table 8-4, it can be observed that the most appropriate kernel for the SVM, is the RBF, instead of Sigmoid. When the parameter gamma takes the scale value (1/n_features*X.var()), it leads to a higher accuracy than auto, which is used the type: 1/n_features. Also, the increase of the C value leads to the decrease of tolerance of the errors. As a result, the accuracy of the mode increases.

In the other two types of SVM, only one parameter can be modified. In the Nu-SVM, only the kernel type can be changed. The RBF kernel leads to a better model. On the other hand, in the Linear SVM, only the regularization parameter C, can be changed. Same as the SVM, the increase of the C parameter, generates a model with higher accuracy. The following table shows the accuracy of each model (Table 8-5):

TABLE 8-5: THE ACCURACY OF NU-SVM AND LINEAR SVM RELATED WITH THE MODEL PARAMETERS.

Nu-SVM	Accuracy	Linear SVM	Accuracy
Kernel type: Sigmoid	0.74	C: 1	0.66
Kernel type: RBF	0.82	C: 1000	0.72

From the results of the three previous models, it can be noticed that the most appropriate kernel type is RBF instead of sigmoid and linear type. The use of different kernel, provides different accuracy, since the selection of kernel plays a dominant role to the correctness of the model. Also it is obvious that, large values of the C parameter, lead to more accurate models, since the models have less tolerance to the errors. From the results, it can be observed that, the higher accuracy that can bed achieve is 0.86.

Although it is a good accuracy, is not the appropriate for a medical application. The SVM is effective in cases where the features are more than training examples. In bibliography, the training examples are small and as a result SVM achieves higher accuracy. In our case, the dataset is quite large and as a result accuracy is smaller than this in bibliography. It is obvious that, we need to implement some other classification models to examine the appropriateness of these in our dataset. Another important information which can be extracted from this result is that, the linear model has the lowest accuracy. This can verify the fact that the linear models are rarely used in practice since linear SVM needs to have a well separated point to work efficiently.

TABLE 8-6: THE ACCURACY OF MLPC RELATED WITH THE MODEL PARAMETERS.

MLPC Parameters	Accuracy
Kernel type: ReLU, solver: lbfgs, hidden layer: 100	0,25
Kernel type: tanh, solver: lbfgs, hidden layer: 100	0.76
Kernel type: logistic, solver: lbfgs, hidden layer: 100	0.86
Kernel type: logistic, solver: lbfgs, hidden layer: 200	0,82
Kernel type: logistic, solver: adam, hidden layer: 100	0,75
Kernel type: tanh, solver: adam, hidden layer: 100	0,753
Kernel type: ReLU, solver: adam, hidden layer: 100	0,73

From the previous Table 8-6, we observe that the logistic kernel type is the most suitable from the training data. Also, using different number of layer in hidden layer, it is concluded that the most appropriate number of hidden layers is 100. The 0.86 accuracy is good but not satisfactory so it is necessary to use another model to fit the training data.

TABLE 8-7: THE ACCURACY OF DTC REGARDING THE DIFFERENT INFORMATION GAIN METHODS.

DTC Parameters	Accuracy
Gini	0,82
Entropy	0,92

According to the result of the classification methods, it can be concluded that one of the most appropriate model for our project is the decision tree classifier. We can see that changing the information gain method, from gini to entropy, generates a model that can classify our training example more accurate (Table 8-7). Our training examples

contain information related to 10 features; as a consequence, a decision tree can separate in a more appropriate way the training examples. Each node of the decision tree selects the most appropriate feature to split the data so as to generate the tree.

TABLE 8-8 THE ACCURACY OF RFC REGARDING THE DIFFERENT INFORMATION GAIN METHODS.

RFC Parameters	Accuracy
Gini	0,91
Entropy	0,943

Comparing the accuracy of the decision tree classifier with the random forest classifier, can be noticed that the second model achieves higher accuracy (Table 8-8). The fact that the RFC method achieves higher accuracy than DTC, was expected, since random forest implements additional functions to generate the most appropriate tree.

Finally, from the whole previous analysis, it is concluded that the higher accuracy that can be achieved is 94.3%. It can be understood that, the input dataset plays a dominant factor for generating a model to fit with high accuracy. The selected database consists of the features values that are extracted from both methods. Also, the most appropriate dataset regarding the size is database D which contains 400 malignant and 100 benign images. The classification method that was used to achieve this accuracy, is the random forest classifier. To split the data at each node the classification uses the entropy method. Entropy is used at each node to calculate the information gain for each features. Using the entropy, the algorithm selects the most appropriate feature to split the data into subtrees.

Chapter 9

Hardware Implementation

9.1 General

The implementation of our project in hardware provide us the opportunity to accelerate some part, that contains intensive computations to reduce the execution time. To achieve this, FPGA board is used, in which some parts are executed directly to the hardware through the generated IPs and the other parts run at the ARM processor. By creating IPs, we can easily increase the hardware resources, that used in our project, so as to reduce the execution time. In addition, we can parallelize the calculation of the computations to decrease the execution time. In our project, the hardware resources are used to accelerate the image processing stage, which is a demanding task, due to many computations and a vast amount of time to be completed are required. Using hardware, the operations are usually computationally faster since logic block is used.

9.2 FPGA

A field-programmable gate array (FPGA) (Figure 9-1), is an integrated circuit designed to be re-configurable by the consumer, after its manufacturing stage [53]. It's ability of being reconfigurable, differs FPGA from the Application-Specific Integrated Circuit (ASIC) chips. The functionality of the ASIC chips is limited to specific applications, which is pre-defined by the manufacturer. This restriction, makes the FPGA to be widely used in research, related with many problems. The FPGA can be

reconfigured by using a hardware description language (HDL). It consists of an array of programmable logic blocks, and a hierarchy interconnects of blocks that can be wired together. Logic blocks can be reconfigured in order to execute complex combinational functions, or simple logic gates such as AND and XOR. In addition, logic blocks contain memory elements like Flip-Flop (FF) or more complex memory structures like BRAMs and ROMs. All the above, make FPGAs a suitable choice for many applications, especially those that are related with embedded systems, since software and hardware can be concurrently executed.

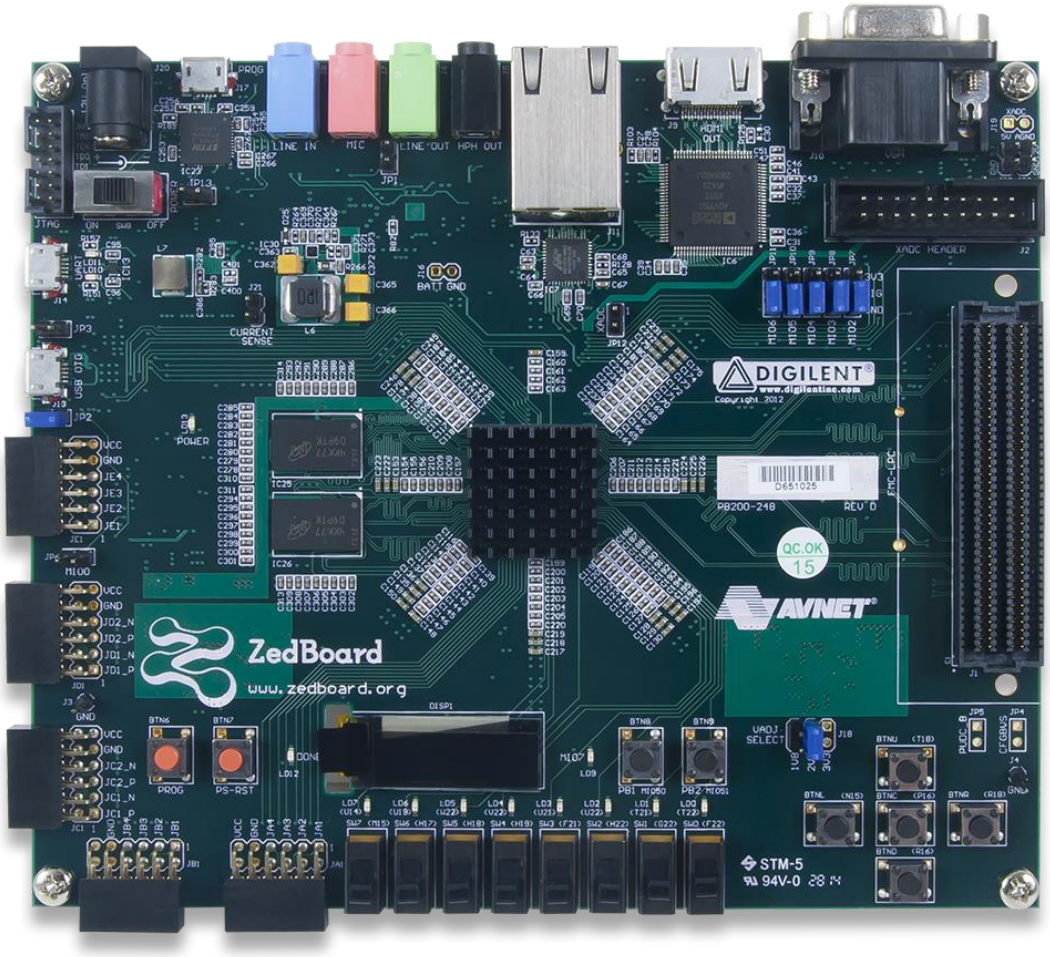

FIGURE 9-1: ZEDBOARD [54].

9.2.1 Xilinx Zedboard Zynq-7000

Zedboard is a low-cost development board for the Xilinx Zynq-7000 SoC. The Zedboard [55] is an evaluation and development board, based on reconfigurable

devices. The Zynq device combines two parts, a Processing System (PS) and a Programmable Logic (PL). To be more precise, the chip contains a dual Corex-A9, a core processor and a Xilinx 7-series FPGA. The use of the Zynq-7000 provides developers the ability to calculate intensive computations on FPGA, and to control parts on the processor. The intercommunication between the PS and PL (Figure 9-2) is known as Programmable Logic [56]. The Processing System mainly consists of the ARM Cortex-A9, DDR3 controller for external memory, and UART for serial communication. On the other hand, the Programmable Logic is formed by standard FPGA structures, and AXI interface is used for connection between PS and PL.

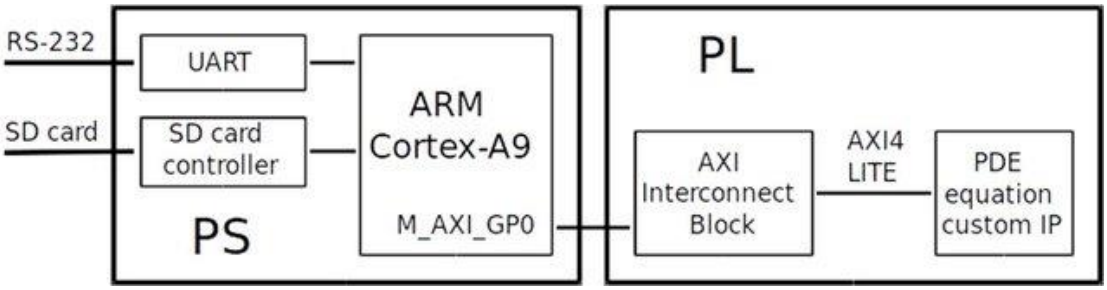

FIGURE 9-2: COMMUNICATION BETWEEN PS AND PL THROUGH AXI4-LITE INTERFACE [57].

9.3 Application Design Tools

9.3.1 Vivado HLS

Vivado High-Level Synthesis (HLS) tool [58] is a successor to the Xilinx ISE Design Tool Suite. HLS tool is included in all Vivado HLx Editions and facilitates the process of IP creations. The HLS is used for the design and the execution of IPs by using C/C++/System C programming language, and provides tools that optimize the hardware, since the programming language is converted to RTL without the necessity of knowing or using Hardware Design Languages (HDL). To optimize the hardware for achieving the timing, area and hardware resources requirement, we can develop program or use directives. Also, HLS contains tools that are able to run Synthesis and Simulation so as to check the functionality of our IP (Figure 9-3). Using the Synthesis and Simulation results, the programmer's possible mistakes can be checked and the function of the program at each clock period can be observed by using the waveforms.

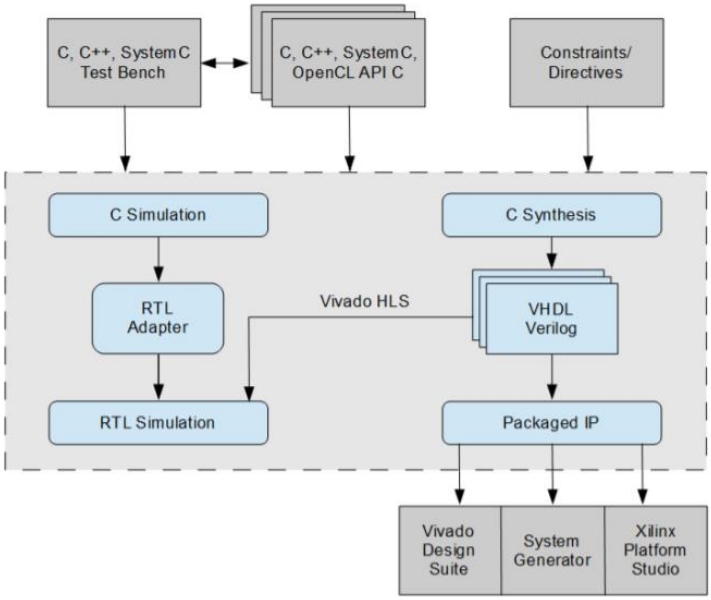

FIGURE 9-3: VIVADO HLS DESIGN FLOW [59].

9.3.2 Vivado Design Tool

The Vivado [60] [61] provides a powerful intuitive graphical user interface and place and route tools that accelerate the design implementation. Vivado [56] allows us to develop very easy and quick base embedded processor systems and applications, since creating a design does not require HDL codes. Vivado provides a large library with standard IP blocks. Each IP block is a box with pins for inputs and outputs. It is a very useful tool, since the user can have an overview of the design and the pins are connected with buses instead of writing any code. Also, a double click in the IP block generates a GUI panel which is mandatory for the user to modify the functionality of the IP. When the system is validating, we can generate the bitstream to export hardware for SDK which is executed from Vivado. Additionally, if parts of software codes are not included in the program, we can download the hardware directly to the board in order to verify the operation. On the other hand, the Software development take place in SDK. By using the SDK, the board and the processor can be programmed.

Since Vivado is used to create embedded applications, it requires many IPs that communicate with the Processing System Units. This communication is achieved through a bus interface, which is called AXI. AXI interface can be separated in three different types. The AXI4-Lite bus, which is used for writing and reading small data

such as status register. The AXI4 bus interface is used for sending and receiving large data, like images, so as to achieve better performance. The AXI4-Stream is the third and final interface, which is used for continuous data like the frame preprocessing of a video. To achieve the communication between the Processing System and Programmable Logic, the standard AXI4 bus is used. The memory address for each communication needs to specified. The address editor which is contained in Vivado, can automatically assign the memory address.

9.4 Image Format

Before developing the code that have to be executed on ARM and the SDK, the format of the image has to be defined. As a consequence, the image format is important to correctly read and write the image. The selected image format is the BMP file [62]. The BMP file format is able to save two-dimensional digital images, either black and white or colored image, in various color depths. The BMP file format is formed with fixed-size and variable-size structures in a specified order. Many of these structures required in the file while others are optional. The need of these structures occur that the BMP file is typically a long size file. The Bitmap File Header and the DIB header, which follow the Bitmap File Header, have fixed-sized and are not optional. The other required structure is the pixel array, which has fixed size depending on the size of the image. The BMP File Header stores general information so as to validate the correctness of the file. The size of this structure is only 14 bits and contains the size of the BMP File in bytes, the starting address of the pixel array and other information related with the creation way. The DIB header consists of information about the image and there are 7 different versions of this structure. To be more precise, the DIB header contains information about the size of this header, the width, the height, the resolution, and the number of bits per pixels. The fact that, the number of bits per pixel can be various, better performance by using the hardware can be achieved. The number of bits per pixel are typically 1, 4, 8, 16, 24, and 32.

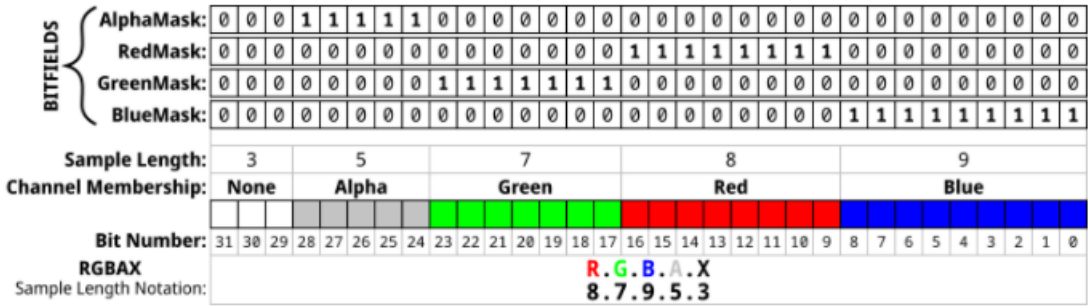

FIGURE 9-4: MECHANISM FOR A 32-BIT PIXEL [62].

Analyzing the above figure (Figure 9-4), it can be observed that the number of bits that represent the color are not fixed and can be modified by using the BITFIELDS mechanism. Using the color mask, someone can easily change the number of bits which represent each color. Also, the order of the color is specified by the BITFIELDS. The most common color order is Blue, Green, Red.

9.5 Hardware Optimization Overview

Improving the performance of an application, is an easy process when the application is developed in hardware using Vivado. The ability of using the fully potentiality of the existing hardware resources can reduce the execution time. The Vivado HLS tool provides the user the ability to optimize its code by using some predetermined optimizations. These optimizations, called pragma, are inserted in the code and explicitly determine the way of Synthesis. Using these HLS Pragmas, user can improve the memory access, the way of data transfer, and the reduction of the execution time of a function or loop.

9.5.1 HLS ARRAY PARTITION

Array partition [63] is a common technique for separating an array into smaller arrays or even into individual elements. This technique segments the memory. As a result, the number of available ports which are used to read and write the data to storage, is increasing. This means that, the program can read and write to memory multiple data at each clock period by using the available different ports. The array partition technique increases the number of memory instances and registers and enhance the throughput of

the design. The Vivado HLS tool provide the pragma array partition, which can be inserted to the source code in the function where the array is defined.

```
#pragma HLS array_partition variable=<name> \
<type>  factor=<int>  dim=<int>
```

The above definition contains the required arguments for the HLS partition. The first argument specifies the variable which is needed for partition. The argument type specifies the partition type which is one of the following: cyclic, block, and the default complete type. The complete partition type separates the original array variable into individual elements. The block partition type separates the initial array into smaller continuously block. The cyclic partition type separates cyclically the element of the original array into the smaller arrays. The cyclic and block type require the definition of the factor argument. The factor indicates the number of the smaller arrays. The last dimension argument specifies the dimension in which the partition takes place. The following example (Figure 9-5) displays an array of 9 elements and how the previous types affect the formation of the smaller arrays and the content.

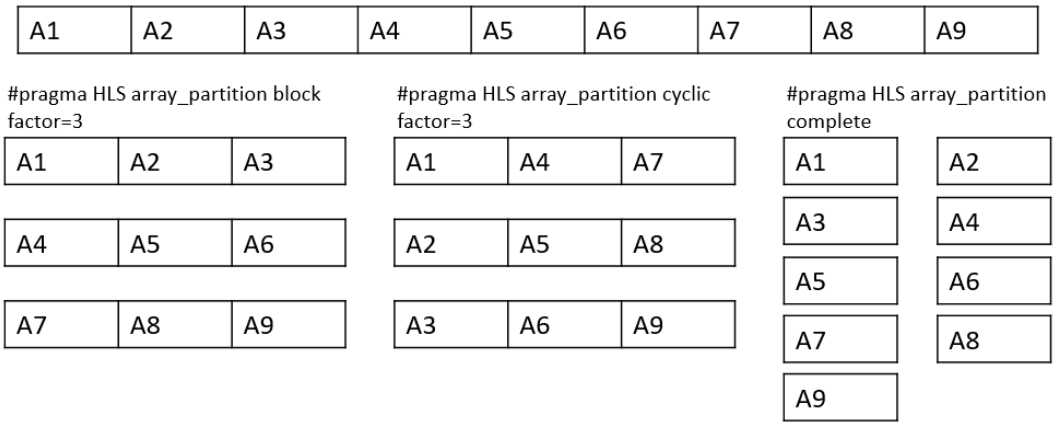

FIGURE 9-5: AN EXAMPLE OF ARRAY PARTITION TYPES.

9.5.2 HLS Data Pack

Data pack pragma [63] is used to pack the data fields of a struct into a single scalar with more extensive word width. The use of this pragma permits reading and writing all the information of the struct at the memory, concurrently. Also, the packing of the struct into a single scalar reduces the size of the memory for the struct. One of

the most common implementation of this pragma is struct of pixels, which contains information for the values of the RGB colors. In addition, this pragma adjusts the member of the struct to 8-bit boundaries, although bit accurate data leads to smaller and faster operations. If the connection between the PS and PL is achieved by using an AXI4 interface, the protocol requires to align the data to 8-bit boundaries. It can be concluded that, this pragma improves the memory performance and as a result, reduces the number of cycles, required for transferring the data.

9.5.3 HLS Loop Merge

Loop Merge pragma [63] is used for joining consecutive loops into a single loop. Merging the loops, a parallel execution is occurred, when it is feasible. The parallel execution of the loops decreases the number of clock cycles needed for the loop body. The pragma merges all the sub-loop into the loop that is defined. There are some rules that need to be satisfied before using the loop_merge pragma. If the loop limits are variables, then the iterations must be the same. Otherwise, if the loop bounds are both variables and constants, the loop_merge pragma cannot be applied. Moreover, the code should compute the same output and not contain FIFO reads, since this pragma can spoil the order of the reads.

9.5.4 HLS PIPELINE

The PIPELINE pragma [63] is based on the simultaneous execution of operations to decrease the initiation interval within the boundary of a function or a loop in which is placed. This pragma gives the ability on a function or a loop to execute new data every N clock period, where N is equal to the initiation interval (II). The best performance is achieved when the initiation interval is equal to 1. The default value of the initiation interval is 1, which means that the function or loop starts a new iteration every clock cycle. The following figure (Figure 9-6), shows the sequential execution of the loop and the affection that the pipeline pragma has, when the initiation interval equals to 1. The pipelined loop starts the following iteration on the next clock cycle.

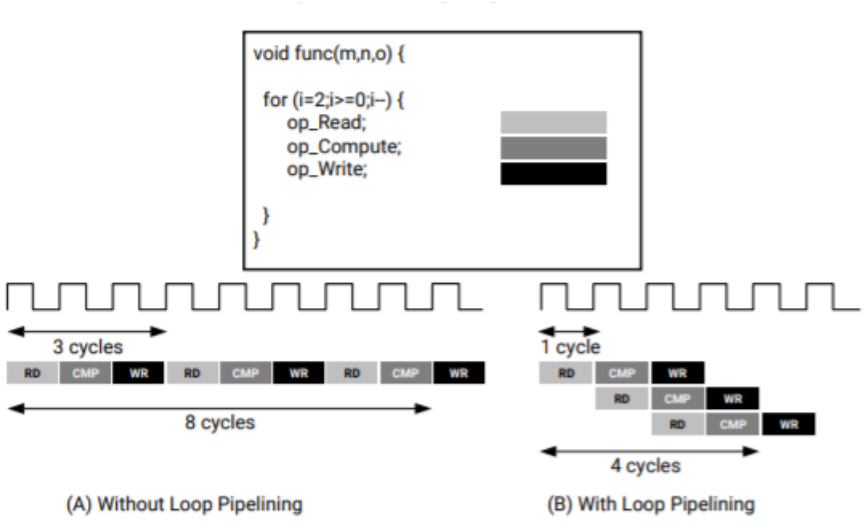

FIGURE 9-6: LOOP PIPELINE [63].

9.5.5 HLS UNROLL

The Unroll pragma [63] generates multiple independent copies of the loop. These copies can be executed in parallel. We can discrete two different categories, the partial and the fully unrolled. The type of unroll is specified by the factor argument which is equal to number of iterations, by default. The unroll pragma generates a copy for each iteration by default, and as a result, the whole loop can be executed simultaneously. When defining a value N for the factor argument, the unroll pragma generates N copies in order to reduce the execution time and the number of the loop iterations.

9.6 ARM Implementation

In order to execute our code in the ARM preprocessor of the Zedboard, the SDK tool, which is provided by the Vivado Design Suite, is used. Using the SDK, the code is written in C language in order to implement the filters from the image preprocessing stage. To implement the image processing, is necessary to write code for the filters: RGB to GRAYSCALE, Median Filter, Histogram Equalization, Thresholding, Invert Binary Image, and the Open Morphological Filter which consists of Erosion and Dilation Filters. To compute the value of each pixel, double nested for loops is needed, to access the whole image.

9.6.1 Code Analysis

In this part of our thesis, the code analysis of each filter is discussed. The functionality of each filter is being descried more analytical in Chapter 6. The development of these codes are based on this information. In the below section we describe the algorithm for each filter in detail.

Algorithm 1: Convert RGB to Grayscale image

```
Result: Grayscale Image
for i = 2; i < WIDTH − 3; i++;
    for j = 2; j < HEIGHT − 3; j++;
        gray_pixel = 0.299 ∗ red + 0.587 ∗ green + 0.114 ∗ blue;
    end
end
```

Algorithm 2: Median Filter

```
Result: Image after median filter
mask = 3x3 kernel size;
for i = 2; i < WIDTH − 3; i++;
    for j = 2; j < HEIGHT − 3; j++;
        sort values in the mask;
        select the middle value of the sorted list;
        replac the pixel value with median;
    end
end
```

Algorithm 3: Histogram Equalization

```
Result: Image after Histogram Equalization
hist[256] = 0;
cdf[256] = 0;
for i = 2; i < WIDTH − 3; i++;
    for j = 2; j < HEIGHT − 3; j++;
        hist[pixel_value]++;
    end
end
cdf[0] = hist[0];
for i = 1; i < 256; i++;
    cdf[i] = cdf[i − 1] + hist[i];
end
for i = 0; i < 256; i++;
    hist[i] = ((cdf[i] − 1) ∗ 255)/num_pixels;
end
for i = 2; i < WIDTH − 3; i++;
    for j = 2; j < HEIGHT − 3; j++;
        pixel_value = hist[pixel_value];
    end
end
```

Algorithm 4: Thresholding

```
Result: Binary Image
for i = 2; i < WIDTH − 3; i++;
    for j = 2; j < HEIGHT − 3; j++;
        if pixel_value > 99 then
            pixel_value = 255;
        else
            pixel_value = 0;
        end
    end
end
```

Algorithm 5: Invert Binary Image

```
Result: Invert Image
for i = 0; i < WIDTH − 1; i++;
    for j = 10; j < HEIGHT − 1; j++;
        if pixel_value = 255 then
            pixel_value = 0;
        else
            pixel_value = 255;
        end
    end
end
```

Algorithm 6: Erosion Filter

```
Result: Image after erosion filter
mask = 5x5 kernel size;
for i = 2; i < WIDTH − 3; i++;
    for j = 2; j < HEIGHT − 3; j++;
        sum kernel values
        if sum = 6375 then
            pixel_value = 255;
        else
            pixel_value = 0;
        end
    end
end
```

Algorithm 7: Dilation Filter

```
Result: Image after dilation filter
mask = 5x5 kernel size;
for i = 2; i < WIDTH − 3; i++;
    for j = 2; j < HEIGHT − 3; j++;
        sum kernel values;
        if sum > 0 then
            pixel_value = 255;
        else
            pixel_value = 0;
        end
    end
end
```

It can be noticed that the above algorithms have different boundaries, related to the filter that is implemented. The range of the loop is changed, in order to avoid multiply "if statement", for the corner cases. By doing this, we can achieve better performance, since the evaluation of expressions in if statements require much more time. Since the lesion is located in the center of the image, the two-pixels border, which are not processing, cannot corrupt the final result. To achieve the desirable output, we need to initialize the two-pixels border with white pixels. After inverting the binary stage, these pixels are colored with black, which is the color of the background at the final image. The inversion image algorithm, edits the whole image, in order to convert the border, instead of the other filters. The other filters do not process the two-pixel border, since loop boundaries skip the two first and the two last pixels from both width and height.

9.6.2 Experimental Results

After the code analysis for the filters that used in our implementation, it is important to measure the execution time and to observe the performance. In the following chart (Figure 9-7), each filter's run time is displayed for the initial and the optimization stage.

	INITIAL TIME	INVERT THE ORDER OF LOOP	O3 OPTIMITAZION
DILATION	0.73	0.71	0.07
EROSION	0.72	0.69	0.07
INVERT	0.1	0.094	0.017
THRESHOLDING	0.1	0.095	0.017
HISTOGRAM	0.25	0.2	0.05
MEDIAN FILTER	3.15	3.13	0.25
RGB2GRAY	0.39	0.39	0.14

FIGURE 9-7: ARM EXECUTION TIME OF EACH FILTER.

From the above chart, we conclude that when inverting the order of loops for each image filter, the execution time remains almost the same. Changing the order of loops, the result is computed row by row, instead of column by column. This way, better memory performance is achieved, since the values are saved and read in rows. Also, when we use the –O3 optimization, the execution time rapidly decreases. In addition, it can be noticed that, the most timing consuming filter, is the median filter. This is obvious, since the implementation of the median requires sorting the data in the kernel neighborhood, by using a sorting algorithm. The sorting algorithm requires time to execute due to complexity. The erosion and dilation filters, which consist the open filter, is the second more demanding process of our project. The implementation of these filters demand to compare the values of the neighborhood pixels in order to calculate the final value of the binary. The conversion of RGB image to Grayscale image has intensive computations and as a result, this process needs some time to calculate the final value.

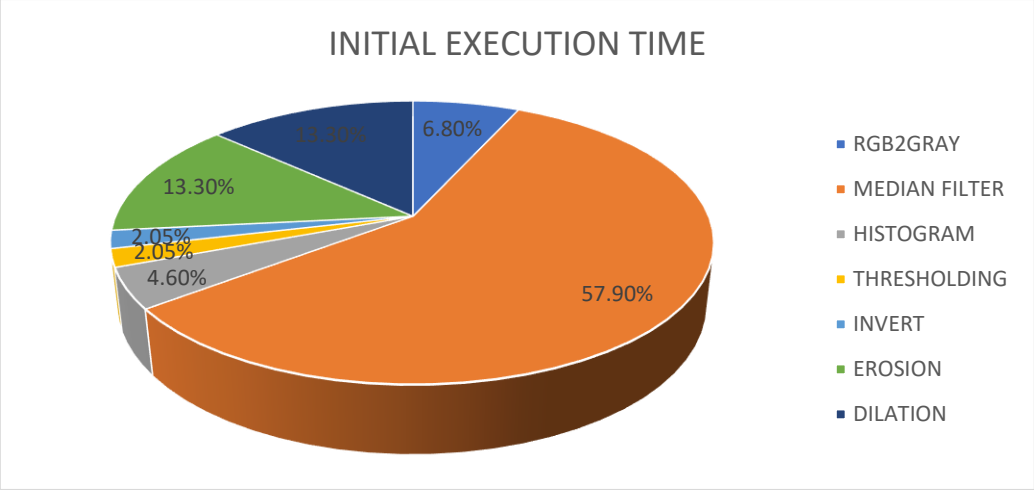

FIGURE 9-8: INITIAL EXECUTION TIME (PERCENTAGE).

To be more precise, from Figure 9-8, it can be noticed that, at the initial execution time the median filter has the 57,9% of the total execution time. The open morphological filter has the 26,6%, the color conversion process has the 7.2%, the histogram equalized filter has the 4,6%, and the thresholding and inverting of the binary image stages, which are the least demanding tasks, have the 3,7% of the total execution time. After the –O3 optimization we can extract the following percentage for each filter: The Median filter has 40,7%, the color conversion has 22,8%, the open morphological filter has 22,8%, the histogram equalized filter has 8,2%, and the other two filters have 5,5% of the total execution time. We can observe that the percentage of the median filter execution time is reduced at the optimized stage.

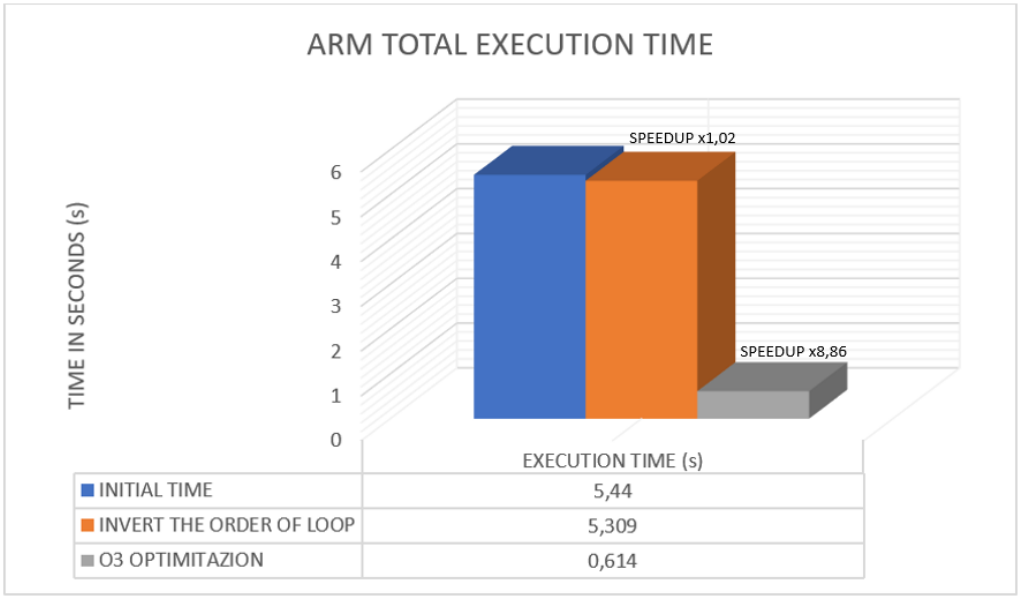

FIGURE 9-9: ARM TOTAL EXECUTION TIME.

The execution time for the initial image processing is 5,44 seconds (Figure 9-9). Inverting the computation from column-column to row-row the execution time decreases to 5,309 seconds. A minimal decrease in the run time is noticed, since the speedup of this optimization is equal to x1,02. The use of the –O3 optimization reduces the time to 0,614 seconds. We observe that this optimization leads to a significant speedup, which is equal to x8,86.

9.7 Hardware Implementation

The hardware implementation requires the generation of the HDL for the filters. The HLS tool facilitates this process, since the C/C++ code can easily be converted to HDL code. The HLS generates an IP block, which can be used from the Vivado and is executed in Hardware. The histogram equalization filter processes the whole image in order to create the enhancement image. Also, the dilation filter follows the erosion filter, starts only if the erosion filter is completed. As a result, the implementation of the image preprocessing requires the development of four different kernels. Before starting analyzing the optimizations that are used to increase the performance of each kernel, it is needed to describe and understand the functions that are implemented in each kernel.

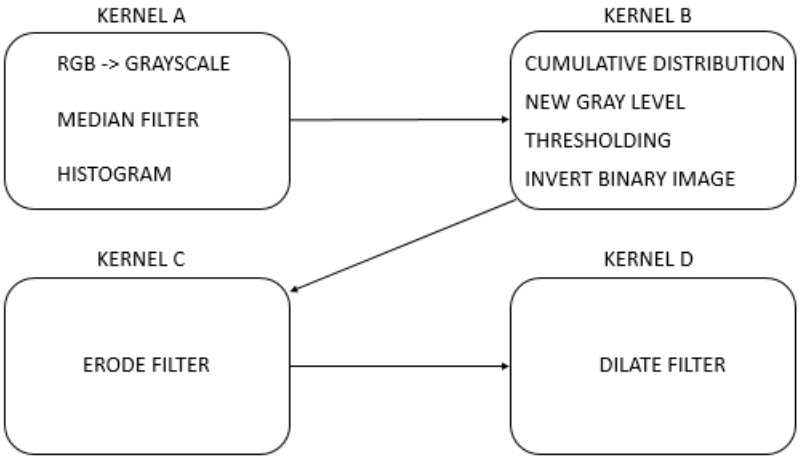

FIGURE 9-10: KERNELS.

From the above diagram (Figure 9-10), we can easily understand that the image processing stage needs four kernels. To be more precise, the implementation of image

conversion from RGB to GRAYSCALE, the median filter, and the first part of the histogram equalization filter, which is the computation of image histogram, are included in Kernel A. The completion of Histogram Equalization filter, the update of pixel's values with the new histogram equalization values, the stage of thresholding, which generates the binary image, and the process of inversion of binary image, which is necessary for the following stage, are included in Kernel B. The erode filter consists in Kernel C and the dilate filter is contained in Kernel D, which is the final Kernel.

9.7.1 Kernels Input and Output

Since each kernel implements a different functionality, the input and output size differs, based on the selected kernel size of each filter. The initial goal of each kernel is to compute one line of the image at each time. Kernel A contains the Median Filter, the implementation of which requires a 3x3 kernel. As a consequence, the kernel has 3 rows as input, in order to calculate one row (Figure 9-11). The kernel B has one row as input and one row as output, since the calculation of each pixel is independent of the other. Kernel C and D require 5 rows as input to compute the required one row (Figure 9-12).

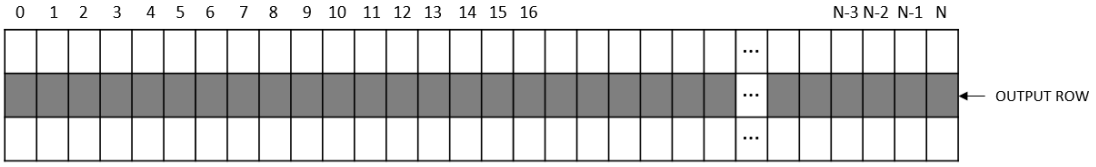

FIGURE 9-11: KERNEL A INPUT.

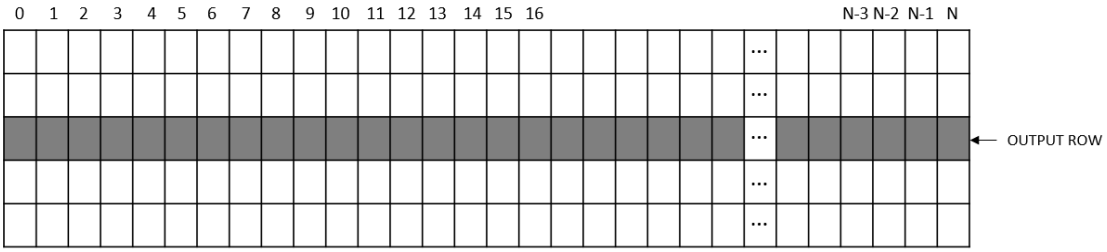

FIGURE 9-12: KERNEL C AND KERNEL D INPUT.

9.7.2 Implementation of Kernel A

One of the Kernel's A filter is the median filter, which is the most time consuming process. This is validated by the metrics of the execution time in ARM processor. Also, the conversion of the RGB to grayscale image is implemented in Kernel A. It is not as demanding as the median and the two morphological filters, but the contribution to the total execution time cannot be omitted. In addition, Kernel A implements the histogram calculation, which is part of the histogram equalization filter. From the functions that are implemented by Kernel A, as shown in fig.9-10, is concluded that this kernel is the most time consuming. So, the implementation of Kernel A in hardware can increase the performance.

Initial Stage – Interface: At the initial stage of the kernel development process, the interface that is used to communicate the accelerator with the processor unit is defined. The kernel's interface is specified by the m_axi HLS pragma and the s_axilite HLS pragma. Applying these pragmas, the port uses the AXI4-Lite interface. This makes the commination easier, since the Zynq processor system uses the AXI4-Lite in order to control everything.

Local Variable: The first optimization aims to reduce the access to the external memory. To achieve this, the grayscale pixels are not saved into the host DDR memory, but into local variable. The percentage of BRAM used memory resources, increases from 1% to 4%, since BRAM_18K memory is used to save the local variable (Table 9-1). The grayscale pixels are used from the median filter, so transferred data from the external memory is avoided. The median filter uses each pixel nine times, since the kernel size is 3x3, as a result the number of multiply transfer from memory of the same data is reduced significantly. Doing this, the execution time of the kernel decreases from 11,4 secs to 8,54 secs, by accelerating the kernel x 1,33.

Unroll Sorting Algorithm: The next optimization is related with the sorting function, which is used from the median filter. The sorting algorithm is the bubble sort algorithm. Bubble sort is selected since the HLS restricts the use of recursive functions. The bubble sort is a demanding function since the average complexity is $O(n^2)$ and it is implemented with a double nested loop. To improve the performance of this part of code the Unroll HLS pragma, which is previously described, is used. The result is the concurrent execution of each iteration of loops for reducing the execution time. The speed up after this step is x 1,05 and the execution time is equal to 8,11 secs.

Pipeline: To increase the performance of kernel, the parallelism of the filters is necessary. The pipeline HLS pragma is used to achieve this goal. This pragma is used in the three loops that implement histogram, median filter and gray conversion. Since the gray conversion is implemented with a double nested for loop, iterating in the three input row pixel by pixel, the HLS pipeline pragma is defined in the inner loop. The initiation interval differs for each loop, for gray conversion II=3, for median filter II=5, and for histogram II=15. From the diagram (Figure9-13), this optimization leads to x 7,11 speedup, which is the bigger value that can be observed. The execution time falls to 1,14 secs.

Struct – Datapack: Since each pixel is represented by three values (Red, Green, Blue), packing these values into a struct type, which contains the value of each color, is dominant. More data are transferred at every clock cycle using struct, improving the use of axi bus. The ability to read simultaneously the values of the struct member and transfer the data more efficient, is provided by the Datapack Pragma. Changing the input and output data type, the initiation interval of the inner gray conversion loop decreases to II=1. The speed up of this step is x 1,52 and the run time is 0,75 secs. Also, the required BRAM memory is reduced after this optimization, using only the 2% of the BRAM_18, instead of the 5% in the previous stage.

Combine Loops: The initiation interval of the median filter and histogram remain the same. In order to increase the performance, these loops are combined into one. The achieved initiation interval for the new loop is II=15. The speed up of this optimization is x 1,27 and the execution time is reduced to 0,59 secs.

Local Variable: From the synthesis analysis is extracted that the transfer of the histogram vector from host DDR requires time, since the same positions of the vector are read and written multiply times. To reduce the amount of transferred data, the histogram vector is saved in a local variable. After this step, the pipeline initiation interval decreases to II=5 from II=15 and the execution time falls to 0,2 which means a x 2,95 speed up.

Pipeline: To write the histogram vector back to external memory, a new for loop to send data to host DDR is required. To optimize this loop, we use pipeline pragma specified the initiation interval equal to 1. This optimization little improves the performance, since the run time decreases only 0,01 secs and is equal to 0,19 secs.

Separate the Input: In order to have more reading ports, the initial input is separated to three inputs. Each input contains information for one row only. After this

step, the double nested for loop for grayscale conversion is replaced by a single for loop. This can be achieved since the three row input can be converted in parallel. The execution time after this optimization is 0,1.

Fabric clock frequency: The final optimization is to increase the frequency. The goal is to increase the performance of the design. The clock period of the IP that is extracted from HLS is specified at 10ns. Increasing the frequency from 100MHz to 175MHz in our design, the achieved speed up is 1,44 and the execution time is 0,069 secs. The 175MHz is the largest value in which our design can work correctly.

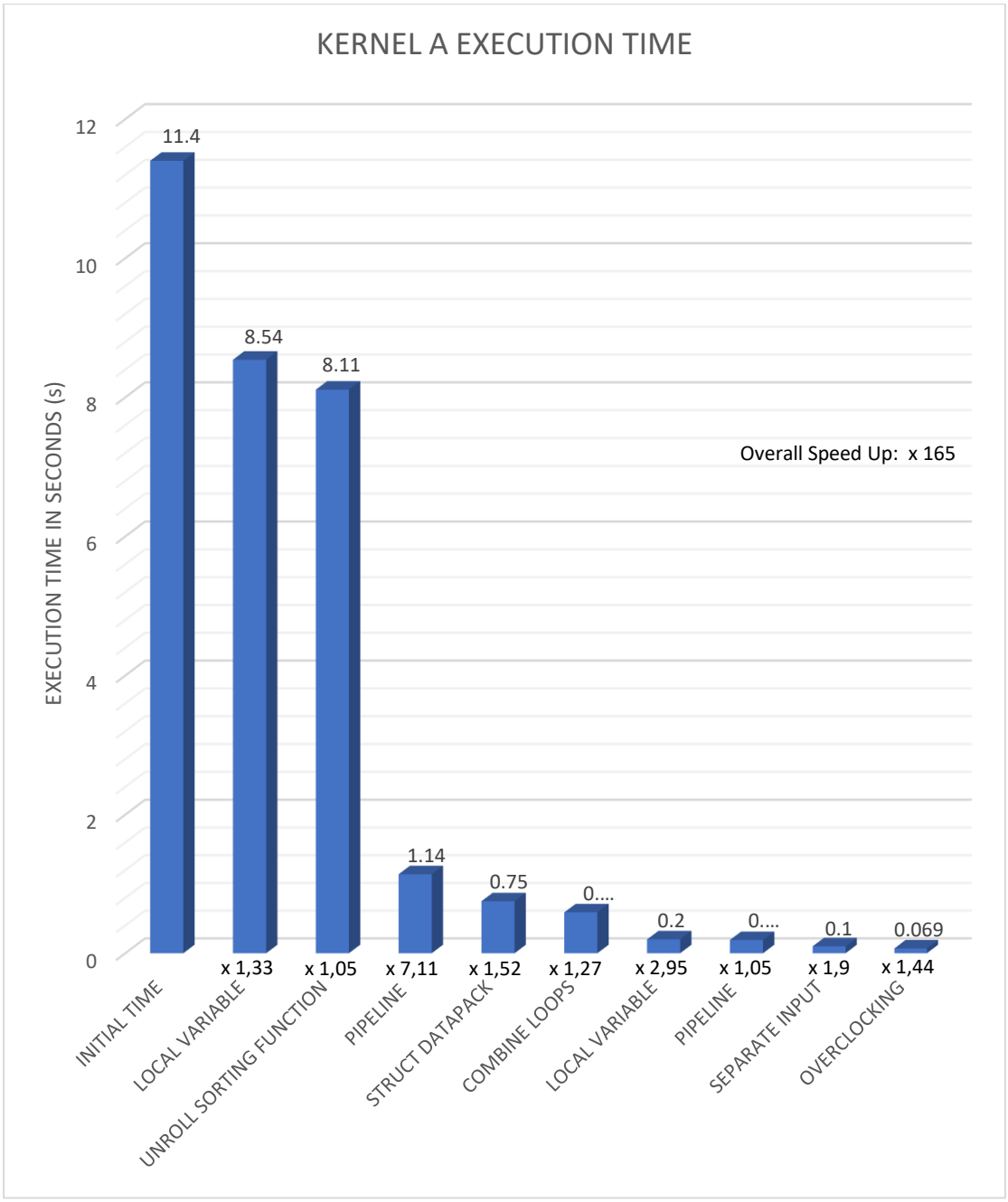

FIGURE 9-13: KERNEL A IMPLEMENTATION.

From the above diagram (Figure 9-13), it is noticed that the kernel initial time is 11,4 secs and after the applied optimizations is 0,069 secs. The overall speed up is equal to x 165. Additionally, the Table 9-1 describes the Utilization. From this table is concluded that the kernel's final design requires 3% of the available BRAM_18K, 12% of the available flip-flops (FF), and the 42% of the available Look Up Table (LUT). Almost the half available LUT of the used board are used by the implementation of Kernel A.

TABLE 9-1: ZEDBOARD KERNEL A – UTILIZATION AND LATENCY(CLOCK CYCLES).

	BRAM_18K	FF	LUT	LATENCY
INITIAL TIME	3 (1%)	4357 (4%)	7798 (14%)	507981
LOCAL VARIABLE PIXEL	13 (4%)	4468 (4%)	7285 (13%)	462885
UNROLL SORTIN FUNCTION	14 (5%)	4544 (4%)	7891 (14%)	452371
PIPELINE	14 (5%)	5348 (5%)	8227 (15%)	43670
STRUCT DATAPACK	8 (2%)	5543 (5%)	10864 (19%)	34643
COMBINE LOOPS	8 (2%)	6689 (6%)	10983 (20%)	27091
LOCAL VARIABLE HISTOGRAM	9 (3%)	6866 (6%)	11175 (21%)	13364
PIPELINE TRASNFER OF HISTOGRAM	9 (3%)	6875 (6%)	11228 (21%)	12600
SEPARATE INPUT	10 (3%)	113085 (12%)	22536 (42%)	5086
AVAILABLE RESOURCES	280	106400	53200	

9.7.3 Implementation of Kernel B

The filters with the least requirement regarding the execution time are implemented by Kernel B. From the previous analysis related with the kernels functionality, this kernel is used to calculate the cumulative distribution, assign the new gray level for each pixel that generated after the completion of the histogram equalization filter, thresholding and finally invert the binary image. The implemented optimizations are quite same with these from the previous kernel. The development of this kernel in hardware can reduce the execution time.

Initial Stage – Interface: At this stage, the interface that used to communicate the PS and PL is defined. Similar to Kernel A, the interface is specified by the m_axi HLS pragma and the s_axilite HLS pragma, which facilitate the communication through the AXI4_Lite interface.

Pipeline: The first optimization that used to increase the performance is the pipeline HLS pragma. This pragma is defined in the three main for loops. To be more accurate, the loop that assign the new gray level, the thresholding loop, and the loop that inverts the binary image are pipelined. The synthesis report shows that the two first loops achieve initiation interval II=2, and the loop that inverts the binary image achieves initiation interval II=4. The speed up of the pipeline pragma is equal to x 3,83 and the execution time falls from 1,15 to 0,3.

Struct – Datapack: Converting the input data type to struct and using the Datapack HLS pragma, the necessary information that is needed in BMP file, are packed into a single scalar. Each packed data can be read and written concurrently with this optimization. As a consequence, the initiation interval of the thresholding loop and loop that invert image decreases to 1. After this optimization, the run time is 0,064 secs which defines a 4,68 speed up.

Local Struct: Since the new gray values are used from the thresholding function, saving these values to a local variable can improve the performance. The execution time fall from 0,064 to 0,053 due to the number of reading from memory are reduced.

Pipeline: Since the new gray levels are calculated at each kernel the first thought is to be executed one time on ARM. Moving this computation to software the execution time increases. Since the calculation of the new gray levels is more efficient in hardware, applying the pipeline pragma can increase the performance. The speed up is x 1,76 and the execution time is equal to 0,03.

Calculate 2 Rows: The new gray level is computed for every row. To reduce the number that this computation is performed, 2 rows are calculated at each kernel. As a consequence, the number of new gray level computation is reduced to half. The execution time decreases to 0,021 with a x 1,43 speed up.

Fabric clock frequency: The final optimization is to increase the frequency. Increasing the frequency from 100MHz to 175MHz, the achieved speed up is x 2,1 and the optimized execution time is 0,01.

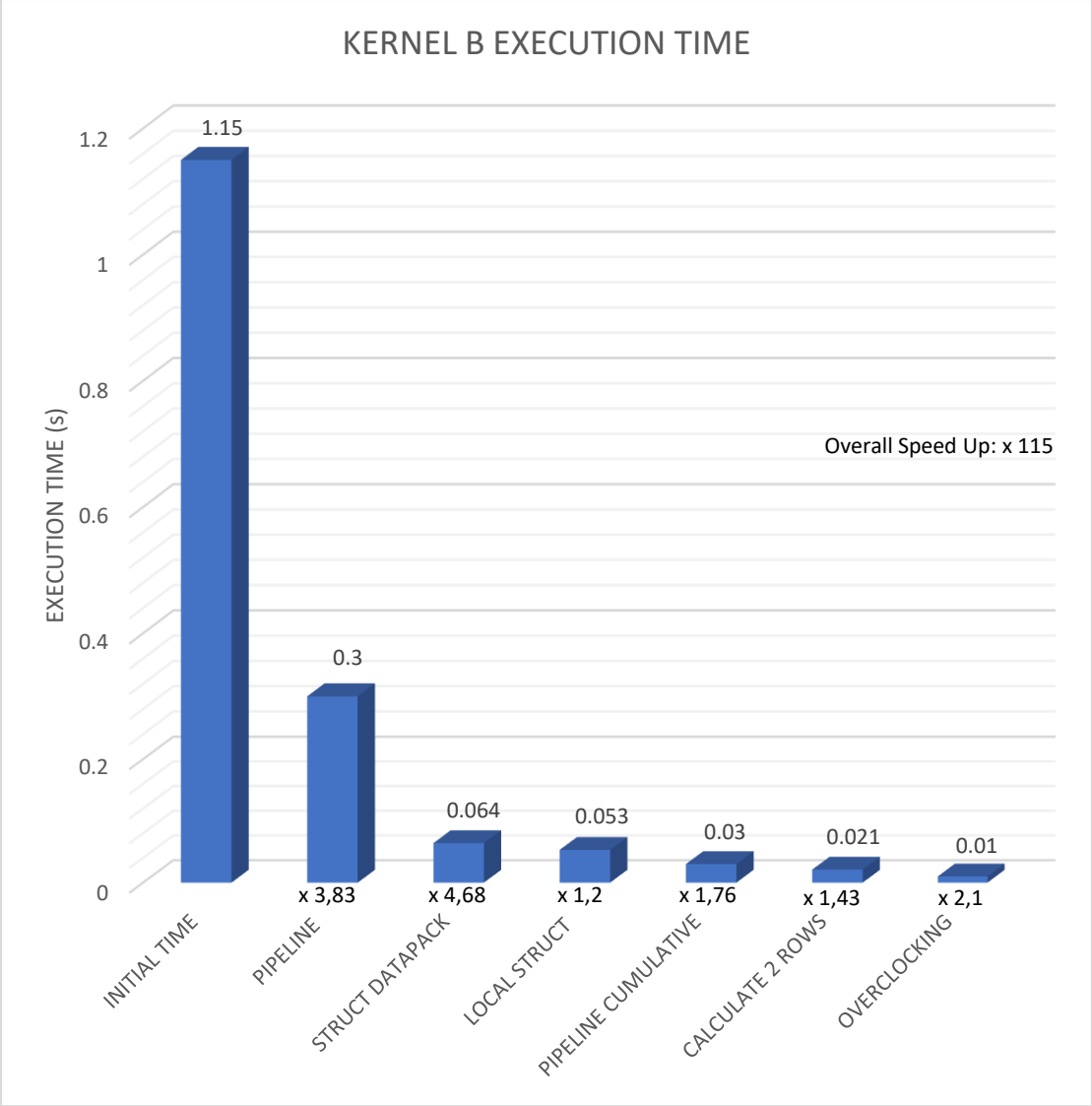

FIGURE 9-14: KERNEL B IMPLEMENTATION.

From the above diagram (Figure 9-14), it is noticed that the kernel initial time is 1,15 secs and the final time is 0,01 secs and the overall speed up is equal to x 115. Moreover, the Table 9-2 describes the Utilization of the board. From the table is extracted that the kernel's final design requires 6% of the available BRAM_18K, 3% of the available flip-flops (FF), and the 9% of the available Look Up Table (LUT).

TABLE 10: ZEDBOARD KERNEL B – UTILIZATION AND LATENCY(CLOCK CYCLES).

	BRAM_18K	FF	LUT	LATENCY
INITIAL TIME	13 (4%)	2867 (2%)	5007 (9%)	46986
PIPELINE	13 (4%)	2840 (2%)	4839 (9%)	21436
STRUCT DATAPACK	11 (3%)	2094 (1%)	3842 (7%)	11284
LOCAL STRUCT	11 (3%)	2644 (2%)	4255 (7%)	9147
PIPELINE CUMULATIVE	14 (5%)	2665 (2%)	4288 (8%)	5322
CALCULATE 2 ROWS	16 (6%)	3383 (3%)	5205 (9%)	5322
AVAILABLE RESOURCES	280	106400	53200	

9.7.4 Implementation of Kernel C and Kernel D

The Kernel C implements the erosion filter and the Kernel D implements the dilation filter. These filters are also time consuming, as occurred from the software analysis. The implementation of these Kernels in hardware can increase the performance. The only difference between these filters is the condition in if statement, so the implementation is almost the same. Same optimizations are applied in these kernels.

Initial Stage – Interface: At this stage, the definition of the interface, that used to communicate the PS and PL, is necessary. Similar to Kernel A implementation, the interface is specified by using the m_axi HLS pragma and the s_axilite HLS pragma, which facilitates the communication through the AXI4_Lite interface.

Memcpy: Since this kernel requires to use multiply times the value of each pixel, the memory requirements are quite high. To increase the performance, it is needed to save input data to local variables in the BRAMs. The Memcpy is used to store the data to local memory from the host DDR memory. From the kernel's Utilization table, it is noticed that the BRAM_18K resources increase to 7%. Using the BRAMs instead of host DDR a x 3,52 speed up is occurred, reducing the execution time to 0,46 secs.

Pipeline: The previous step provides the opportunity to parallelize the kernel's function to increase the performance. Using the pipeline pragma, the achieved initiation interval is II=4. Moreover, the number of clock cycles, related to the computation, is reduced, since a new iteration starts every 4 clock cycles. This optimization decreases the execution time from 0,46 secs to 0,28 secs, a x 1,64 speed up.

Calculate 2 Rows: Four additional rows are required to calculate the one output row since the kernel size is 5x5. It is clearly understood that, two sequential rows use 4 same rows as shown in the figure below (Figure 9-15). This conclusion leads us to modify the kernel, in order to calculate 2 output rows every time. By doing this, the number of computation operation and the amount of transferred data for each row between PS and PL is reduced. To extract the final result for each row, the 4 intermediate rows are compared with the upper input row to generate the first output row and with the last input row to generate the second output row. The execution time at this stage falls to 0,23 secs and the corresponding speed up is x 1,21.

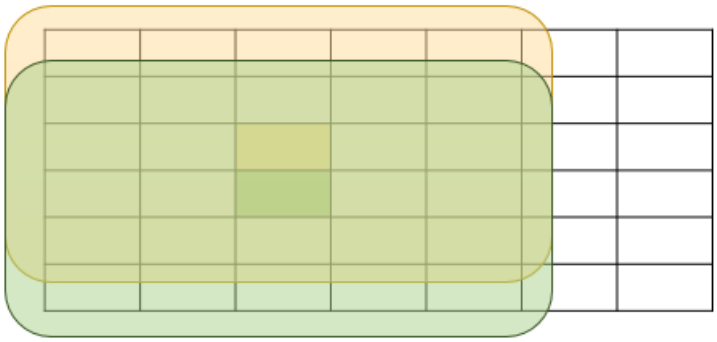

FIGURE 9-15: KERNEL NEIGHBORHOOD FOR THE EACH PIXEL.

Merge Loop: The merge loop HLS pragma is used to reduce the number of clock cycles which is required to copy the data from the host DDR to the BRAM. Since the memcpy functions have the same size, this pragma can execute these function simultaneously. The input contains information for 6 rows, and the use of two different bundles (gmem0 and gmem1) leads to the reading of 2 rows simultaneously. Since in this stage only two different ports are used, the execution time is reduced slightly. The run time after this stage is 0,19 secs.

Struct – Datapack: Converting the input data type to struct and using the Datapack HLS pragma, the 3 required values for each pixel are packed into a single scalar that can be read simultaneously. After this optimization, the run time is 0,083 secs which defines a x 2,29 speed up.

Unroll: The Unroll HLS pragma is used in kernel function to further improve the performance of the kernel. The unroll factor is equal to 10 and as a consequence the execution of N=10 iterations of loops concurrently reduces the execution time. The speed up after this optimization is x 1,31 and the execution time is equal to 0,063 secs.

Separate Input: In order to have more reading ports, the input data is divided to six inputs. Each input contains data for only one row. After the division of the input, the data can be saved in local variable simultaneously. To achieve this, HLS merge loop is used to execute the six memcpy in parallel. Also, the initiation interval of the kernel is II = 5. The execution time after this optimization is 0,047.

Fabric clock frequency: The final optimization is to increase the frequency of the design using Vivado. Increasing the frequency from 100MHz to 175MHz, the achieved speed up is x 2,13 and the optimized execution time is 0,022.

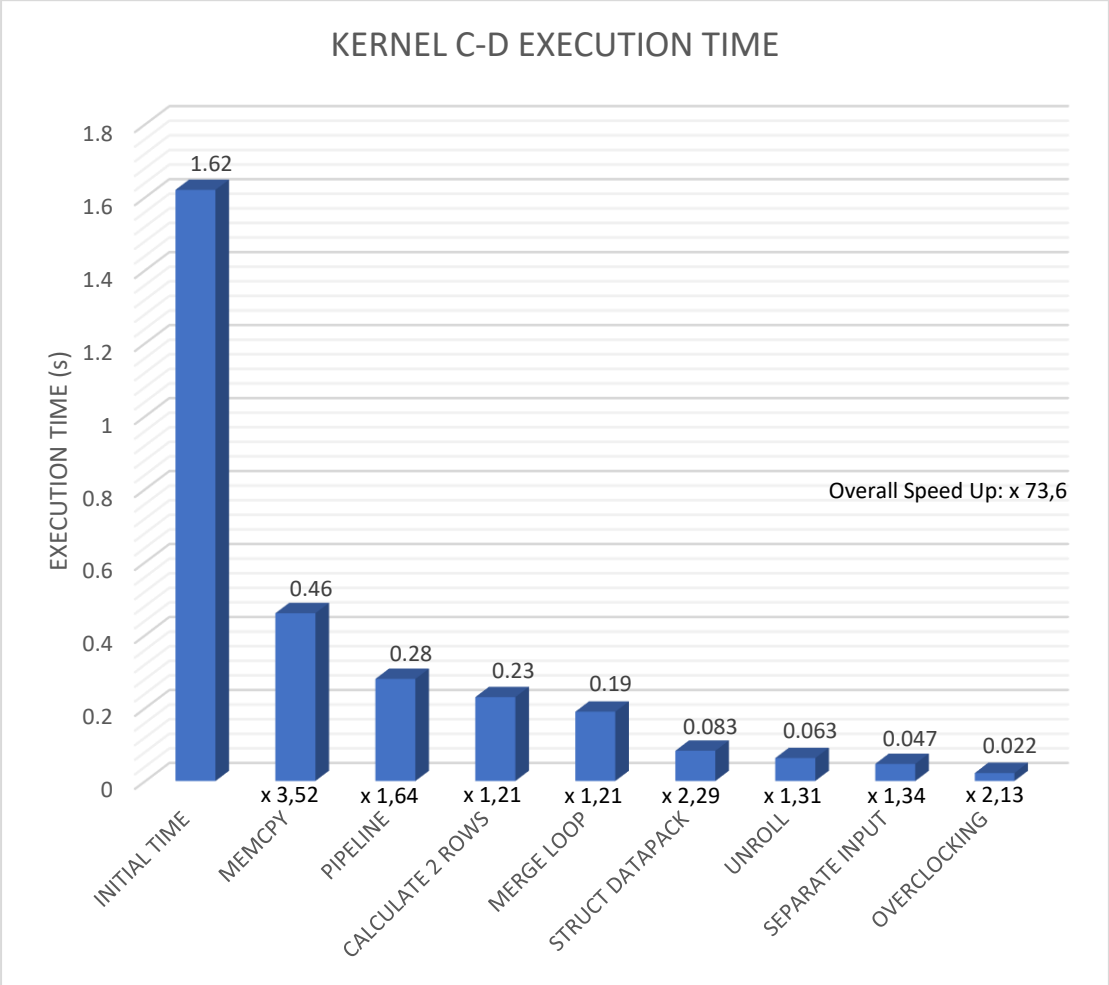

FIGURE 9-16: KERNEL C-D IMPLEMENTATION.

From the above diagram (Figure 9-16), it is observed that the kernel initial time is 1,62 secs and the final time is 0,022 secs. The overall speed up is equal to x 73,6. Additionally, the below Table 9-3 describes the Utilization of the board. The kernel's final design requires 6% of the available BRAM_18K, 5% of the available flip-flops (FF), and the 14% of the available Look Up Table (LUT).

TABLE 11: ZEDBOARD KERNEL C-D – UTILIZATION AND LATENCY(CLOCK CYCLES).

	BRAM_18K	FF	LUT	LATENCY
INITIAL TIME	2 (0%)	2048 (1%)	3292 (6%)	64501
MEMCPY	22 (7%)	1524 (1%)	3012 (5%)	42111
PIPELINE	22 (7%)	1661 (1%)	2773 (5%)	28621
CALCULATE 2 ROWS	26 (9%)	1905 (1%)	3271 (6%)	39143
MERGE LOOP	30 (10%)	2971 (2%)	4471 (8%)	25557
STRUCT DATAPACK	42 (4%)	2096 (1%)	3027 (6%)	9052
UNROLL	15 (5%)	3958 (3%)	4987 (9%)	6039
SEPARATE INPUT	18 (6%)	5376 (5%)	7911 (14%)	3025
AVAILABLE RESOURCES	280	106400	53200	

9.7.5 Final Design Overview

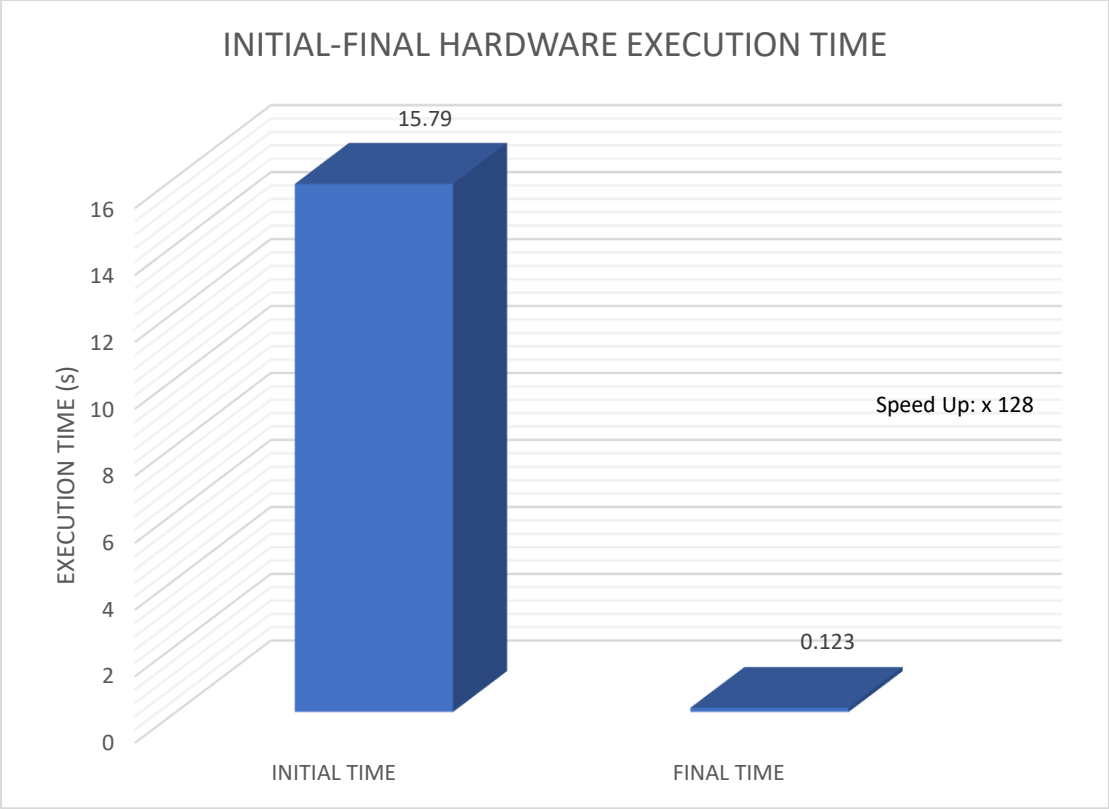

FIGURE 9-17: INITIAL AND OPTIMIZED HARDWARE EXECUTION TIME.

To clearly understand the effect that the optimizations have to the image process execution time, the summarization of the above analysis for each kernel is needed. To extract the run time for the whole image process, the execution time of each kernel is added up. The result for the initial and final time are displayed in the above diagram

(Figure 9-17). The execution time without any optimization is 15,79 seconds and the optimized execution time is equal to 0,123 seconds. The speed up that is achieved is equal to x 128. This speedup increases of the hardware resources to decrease the run time is significantly reduced due to the increment of hardware resources.

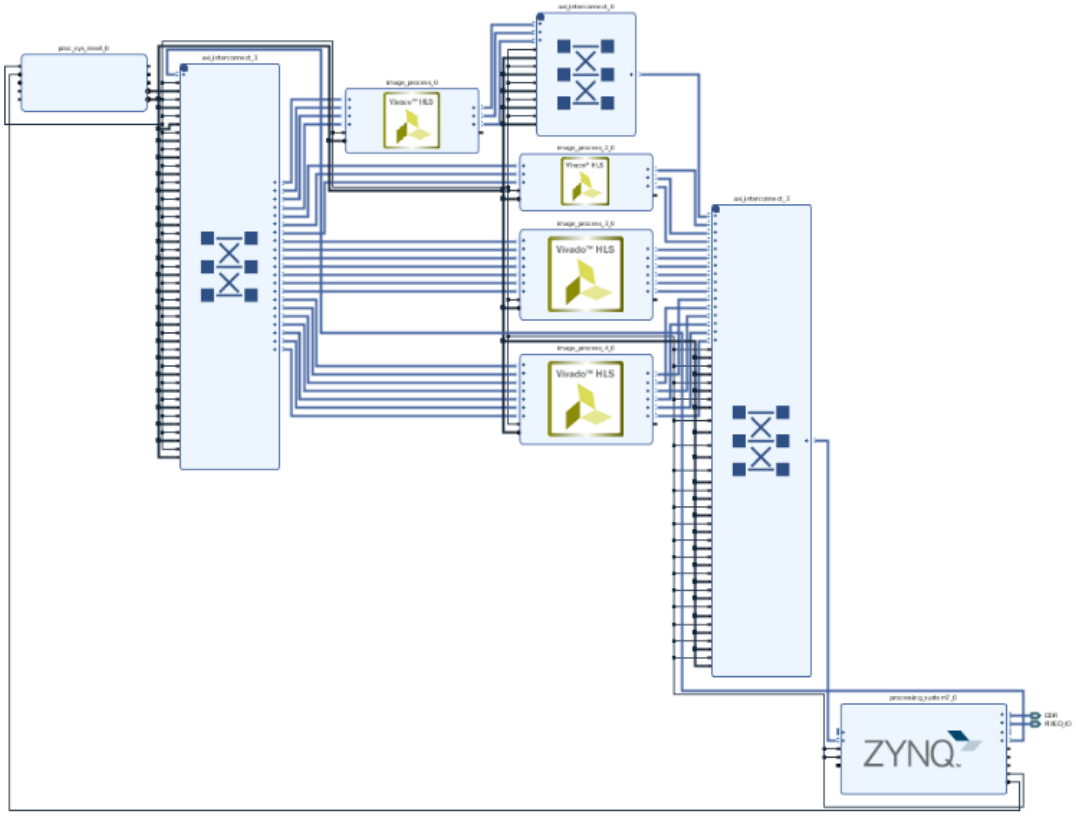

FIGURE 9-18: SYSTEM BLOCK DESIGN

The Figure 9-18 displays the block design, which is implemented at Vivado in order to extract the hardware. This extracted hardware is used from SDK tool to develop and execute our project. The block design consists of the four IP that are generated by HLS and correspond to each kernel. The connection between the kernels ports and Zynq process are achieved by using axi interconnector. The connection of axi interface and Zynq processor are implemented through the axi high performance slave. Also, in the Figure 9-19 is displayed the resources of the board that are used. It is clearly understood that the design requires almost the whole resources of the board. Due to the lack of available resources, the design cannot be expanded using multiply IP of each kernel to segment the image and assign to them specific part of the image.

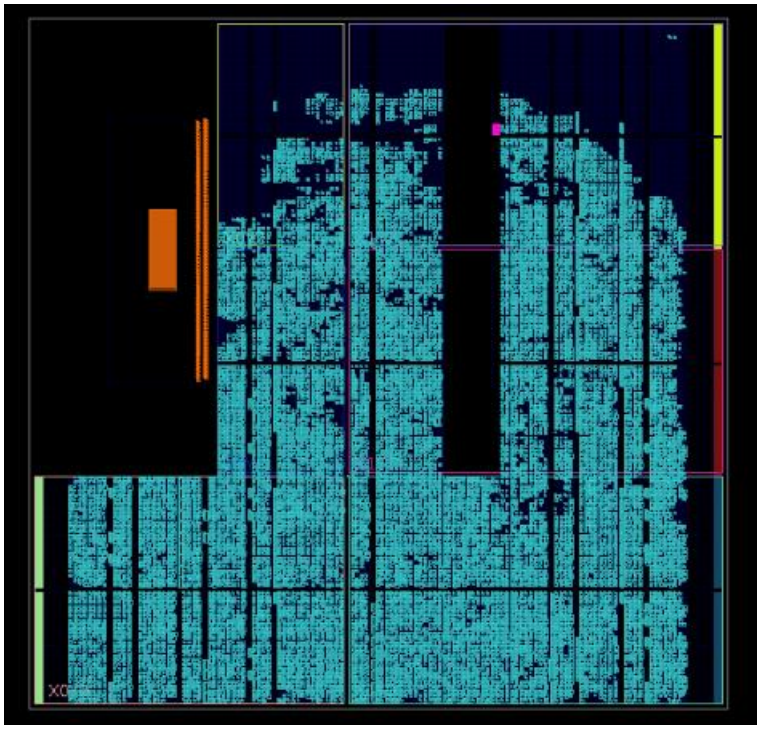

FIGURE 9-19: SYSTEM HARDWARE RESOURCES.

9.7.6 Software vs Hardware Implementation

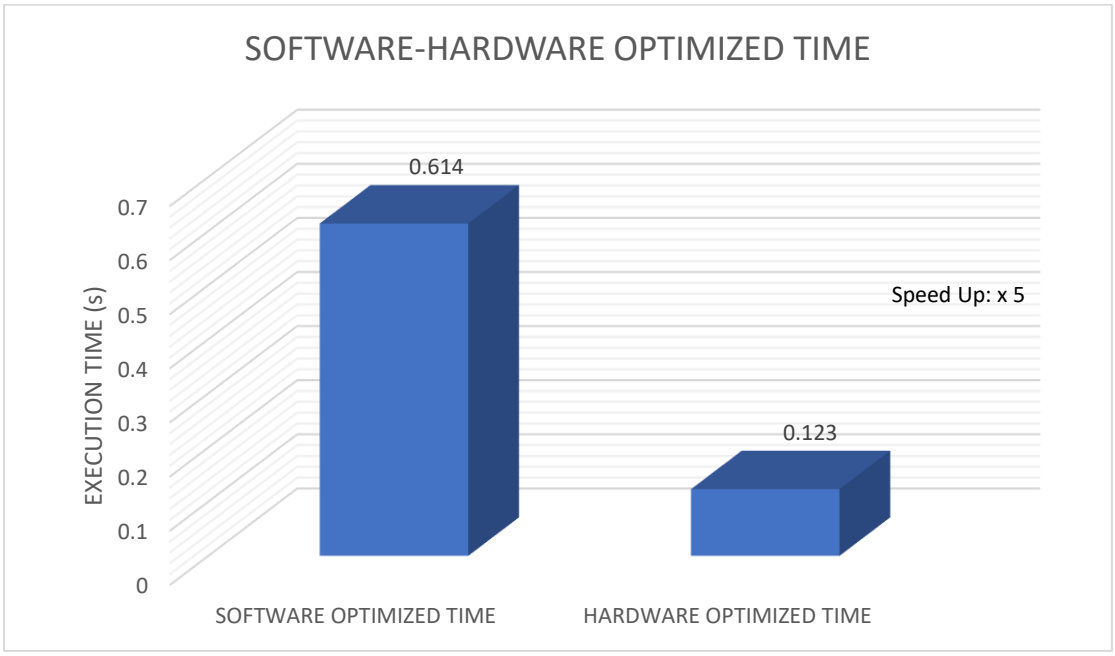

FIGURE 9-20: SOFTWARE AND HARDWARE OPTIMIZED EXECUTION TIME.

Using the available hardware resources, the image preprocessing stage can be accelerated and reduce the execution time that is needed. In the above diagram (Figure

9-20), the software and hardware run time after the implementation of the optimizations are displayed. The final software execution time is 0,614 seconds and the final hardware execution time is 0,123 seconds. The achieved speed up is equal to 5, which means that the hardware implementation is 5 times faster than the software.

9.8 Second Hardware Implementation

9.8.1 HLS OpenCV Library

The second hardware implementation is based on the pre-built OpenCV libraries that are supported by Vivado HLS on different boards. HLS Video Library [64] is a library which is developed in C/C++. It is used for computer vision and image processing problems, which are implemented with Vivado HLS and are accelerated on FPGA. To facilitate the process of creating a new application, based on the HLS Video Library, two header files are provided. The first one is hls_video.h, which is included in top design file and provides the necessary data structures and functions needed to build the application. Also, the content of this library is synthesizable. On the other hand, the hls_opencv.h is included in testbench file in order to use the pre-built OpenCV libraries in HLS and most of these functions cannot be synthesized.

Some basic data structures are provided by HLS Video library to represent images and pixels. These basic data structures types are Matrix, Scalar, Window, and LineBuffer. The most used type for images is the Matrix. In the FPGA, only the parts of the algorithms that need to be accelerated are implemented, since the other algorithm is executed on the processor. Also, the OpenCV interface ensures the transfer of data from the PS to PL and backwards. The communication between the processor and the hardware is achieved by using the AXI4 Steaming protocol. Furthermore, the Video Library contains functions which convert the AXI4-Stream data type to the basic data structures of HLS Video library.

The HLS Video Library contains many accelerated OpenCV functions. These functions are mainly used for video images. They can implement simple operations, such as the min and max functions, which calculate the minimum or the maximum of the two inputs images pixel by pixel. Also, by using these libraries, other more complicated filters such as Sobel, Gaussian Blur, and Harris Corner, can be implemented.

To implement the image process stage, seven function are used. The first function is the CvtColor for converting the RGB image to grayscale. Since the Median Filter is not contained, the Gaussian Blur is used. The Gaussian Blur is also used in the bibliography, so the change is valid. The next filter is EqualizeHist to enhance the image before the Thresholding stage, in order to extract the binary image and invert it. The last filters are the Erode and Dilate which consists the open morphological filter. Before the first filter, it is needed to convert the stream data type to matrix since the filter requires matrix as input and output. After the image process is completed, the output matrix is converted to stream data so as to be sent back to the processor.

9.8.2 Experimental Results.

To implement the whole image process, Vivado HLS tool is used. The initial specification is the board that is used and the target clock period. After experimentation, the target clock period is 14,5 ns and the estimated clock period is 14,028 ns.

One important parameter is the latency. From the synthesis report the latency is 1708958 clock cycles (Figure 9-21). The most demanding filter is the Gaussian Blur with 1708943 clock cycles latency. The overall latency is almost the same, since the dataflow is used. When a pixel is completed, it is moved to the next filter without waiting the end of the process of the other pixels.

Latency		Interval		
min	max	min	max	Type
1708958	1708958	1708944	1708944	dataflow

FIGURE 9-21: LATENCY IN CLOCK CYCLES.

The extracted information for the Utilizations of the process is displayed in the Table 9.4. It is noticed that, the implementation has low percentage of hardware resources. From the BRAM_18K only the 4% is used. The used flip-flop is 4% and the used look up table is 19%.

TABLE 12: ZEDBOARD OPENCV – UTILIZATION.

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	32	-
FIFO	0	-	100	434	-
Instance	12	10	4321	9717	0
Memory	-	-	-	-	-
Multiplexer	-	-	-	36	-
Register	-	-	6	-	-
Total	12	10	4427	10219	0
Available	280	220	106400	53200	0
Utilization (%)	4	4	4	19	0

It can be concluded that the first approach is better than the second which is just analyzed. It is an evidence that the implementation of the filters in first approach are adjusted in the project requirements. To be more precise, in the first approach the corner cases are not checked and as result, the use of if statements are significantly reduced, instead of the second approach. Also, the first implementation uses almost the whole hardware resource. The two previous factors make the first implementation the more suitable selection.

Chapter 10

Conclusion

The advent of technology can facilitate the developmental process of different applications, which can help to overcome many challenging problems related with different scopes of the human life. Our diploma thesis is based on this direction, since we try to tackle the problem of the skin cancer detection, which is a serious problem with high mortality rate. Our implementation focuses on creation of a useful tool, that can be used by everyone, so as to examine their lesions and to detect the existence of melanoma in early stage without visiting a dermatologist.

Our implementation is based on image processing techniques, features extraction methods, and data mining techniques to classify the input lesion image. In our thesis, we try to tackle to different problems that are engaging in current research. The first is based on the existing gap in the research, related with the human color skin. To be more specific, most of the researches based on white color skin population, neglecting the other colored human population, due to the lack of database in these populations. However, the ISIC contains some classified lesion of colored human, that gives the ability to expand the utility of our project. The second problem is based on the requirement to improve the performance of the existing methods. In our project, to accomplish this requirement, the image process stage is accelerated by implementing in hardware.

In the process that is implemented in bibliography, some additional features are used to improve the results of our implementation. Since the image process, which is

used in other implementations, does not remove the whole existing noise, we add the open morphological filter as last step of the process to clear the input image. Furthermore, the thought to combine the results of the ABCD rule and GLCM method generates a classifier with higher accuracy. Also, an additional classifier is used instead of the support vector machine and decision tree classifier, which are commonly used in research. This classifier is called random forest classifier, which is an improvement of DTC and achieves a 94.3% accuracy. The other classifiers have less accuracy as occurred from the analysis of the experimental results in Chapter 8. So, the final implementation uses the ABCD rule and GLCM method to extract the features values of the inputs image. These features generate a new tabular database which is used to train the random forest classifier. The random forest tree is selected since this classifier makes a more correctly prediction. The final implementation has a 5.7% error, due to the fact that the classify of the lesion is only based on ten features. As a consequence, a small error to the value of one of these features can mislabel the input image. Finally, regarding the hardware implementation of the image preparation, it is observed that the performance of this stage is improved by a x 5 speedup.

10.1 Future Work

Some ideas for further investigation are provided in this section. These ideas refer to both software and hardware. First of all, since the size of the database currently expands, the use of more complex structures to generate accurate classifiers may be required. The convolutional neural network (CNN) or recurrent neural network (RNN) may be a suitable solution.

Obviously, the implementation of the feature extraction and modeling stage in hardware may help to increase the performance, although these stages do not require heavily computations, as the image processing stage. Also, an approach based on internet of things may be useful, since the lesion can be transferred through the internet to classify the lesion.

Since the first hardware implementation of the image preparation requires almost the whole resources of the Zedboard Board, the use of a larger board may give the opportunity to increase the parallelism and the performance. Boards, such as Zynq UltraScale, provide more hardware resources and may increase the performance by

using multiply IPs to process the image parallel, assign a segment of the input image to each IP. Also, the m-axi and s-axilite interfaces are used to transfer the data between PS and PL. The combination of the AXI-streaming protocol and AXI Direct Memory Access may increase the performance.

References

[1]. Wikipedia, “Human Skin” URL: https://en.wikipedia.org/wiki/Human_skin#cite_ref-Hcare_1-2

[2]. Mayo Clinic, “Skin Cancer” URL: https://www.mayoclinic.org/diseases-conditions/skin-cancer/symptoms-causes/syc-20377605

[3]. Skin Cancer Foundation, “Skin Cancer Information” URL: https://www.skincancer.org/skin-cancer-information/

[4]. Sumithra R, Suhil M, Guru DS (2015) Segmentation and classification of skin lesions for disease diagnosis. Procedia Computer Science 45:76–85

[5]. Guerra-Rosas E., Álvarez-Borrego J., “Methodology for diagnosing of skin cancer on images of dermatologic spots by spectral analysis,” Biomed. Opt. Express 6(10), 3876–3891 (2015).10.1364/BOE.6.003876

[6]. Jain S., Pise N., “Computer aided melanoma skin cancer detection using image processing”, Procedia Computer Science, 48 (2015), pp. 735-740

[7]. Garg N., Sharma V., Kaur P. (2018) Melanoma Skin Cancer Detection Using Image Processing. In: Urooj S., Virmani J. (eds) Sensors and Image Processing. Advances in Intelligent Systems and Computing, vol. 651. Springer, Singapore. https://doi.org/10.1007/978-981-10-6614-6_12

[8]. ISIC Archive, URL: https://www.isic-archive.com/

[9]: B.K. Armstrong, A. Kricker, “The epidemiology of UV induced skin cancer”, J Photochem Photobiol B, 63 (2001), pp. 8-18

[10]: American Academy of Dermatology Association, “Types of Skin Cancer” URL: https://www.aad.org/public/diseases/skin-cancer/types/common

[11]: Python, “Beginners Guide” URL: https://wiki.python.org/moin/BeginnersGuide/Overview

[12]: Parul Pandey (2019, May 26), “10 Python image manipulation tools”, Towards Data Science, URL: https://towardsdatascience.com/image-manipulation-tools-for-python-6eb0908ed61f

[13]: Scikit-image, URL: https://scikit-image.org/

[14]: Numpy, URL: https://numpy.org/

[15]: Pillow, URL: https://pillow.readthedocs.io/

[16]: OpenCV, URL: https://opencv.org/\

[17] Chapter 10 - Machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking. Machine Learning for Subsurface Characterization 2020, Pages 289-314

[18]: Youlian Zhu, Cheng Huang, “An Improved Median Filtering Algorithm for Image Noise Reduction,” Physics Procedia, Vol. 12 (2012), pp. 609-616

[19]: Y. Hu and H. Ji, "Research on Image Median Filtering Algorithm and Its FPGA Implementation," 2009 WRI Global Congress on Intelligent Systems, Xiamen, 2009, pp. 226-230, doi: 10.1109/GCIS.2009.130.

[20]: OpenCV, “Smoothing Images”, URL: https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

[21]: Azadeh Noori Hoshyar, Adel Al-Jumaily and Afsaneh Noori Hoshyar, “The Beneficial Techniques in Preprocessing Step of Skin Cancer Detection System Comparing”, in Procedia Computer Science, vol. 42, pp.25-31, 2014, doi: https://doi.org/10.1016/j.procs.2014.11.029

[22]: H. Ibrahim and N. S. Pik Kong, "Brightness Preserving Dynamic Histogram Equalization for Image Contrast Enhancement," in IEEE Transactions on Consumer Electronics, vol. 53, no. 4, pp. 1752-1758, Nov. 2007, doi: 10.1109/TCE.2007.4429280.

[23]:OpenCV, “Histogram Equalization”, URL: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html

[24]: G. Yadav, S. Maheshwari and A. Agarwal, "Contrast limited adaptive histogram equalization based enhancement for real time video system," 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), New Delhi, 2014, pp. 2392-2397, doi: 10.1109/ICACCI.2014.6968381.

[25]: Ravindra Pal Singh, Manish Dixit, “Histogram Equalization: A Strong Technique for Image Enhancement”, International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.8, No.8 (2015), pp.345-352 http://dx.doi.org/10.14257/ijsip.2015.8.8.35

[26]: D. Liu and J. Yu, "Otsu Method and K-means," 2009 Ninth International Conference on Hybrid Intelligent Systems, Shenyang, 2009, pp. 344-349, doi: 10.1109/HIS.2009.74.

[27]: Lee, S.U., Chung, S.Y., Park, R.H., 1990. A comparative performance study of several global thresholding techniques for segmentation. Comput. Vis. Graph Image Process 52 (2), 171-190.

[28]: Serra, J., Vincent, L. An overview of morphological filtering. Circuits Systems and Signal Process 11, 47–108 (1992). https://doi.org/10.1007/BF01189221

[29]: OpenCV, “Morphological Transformations”, URL: https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html

[30]: H. Kittler, M. Seltenheim, M. Dawid, H. Pehamberger, K. Wolff, M. Binder. Morphologic changes of pigmented skin lesions: a useful extension of the ABCD rule for dermatoscopy J Am Acad Dermatol, 40 (4) (1999), pp. 558-562

[31]: A. Gola Isasi, B. García Zapirain, A. Méndez Zorrilla. Melanomas non-invasive diagnosis application based on the ABCD rule and pattern recognition image processing algorithms Comput Biol Med, 41 (2011), pp. 742-755

[32]: L. Sai Vineela, G. Tarun Ravi Sankar, K. Nohan Prasanth, K. Vamsidhar. Skin Cancer Detection Using Region Based Segmentation IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 6 Issue 4, April 2019

[33]: ABCD rule. (2019, April 24). dermoscopedia. Retrieved 00:02, August 29, 2020 from https://dermoscopedia.org/w/index.php?title=ABCD_rule&oldid=15572.

[34]: Guang-ming Xian. An identification method of malignant and benign liver tumors from ultrasonography based on GLCM texture features and fuzzy SVM. Expert Systems with Applications, Vol. 37 Issue 10, October 2010, pp.6737-6741

[35]: Punal M. Arabi, Gayatri Joshi, N. Vamsha Deepa. Performance evaluation of GLCM and pixel intensity matrix for skin texture analysis. Perspectives in Science, Vol. 8, September 2016, pp. 203-206.

[36]: S. Kolkur and D. R. Kalbande, "Survey of texture based feature extraction for skin disease detection," 2016 International Conference on ICT in Business Industry & Government (ICTBIG), Indore, 2016, pp. 1-6. doi: 10.1109/ICTBIG.2016.7892649

[37]: A. Rosenfeld and A. Kak, 1982 Digital Picture Processing, New Serra: Academic Press, 1982.

[38]: Rushikesh Pupale, (2018, Jun 16), “Support Vector Machines(SVM) — An Overview”, Towards Data Science, URL: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

[39]: Ajay Yadav (2018, Oct 20), “SUPPORT VECTOR MACHINES(SVM)”, Towards Data Science, URL: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589

[40]: Pai-Hsuen Chen, Chih-Jen Lin and Bernhard Scholkopf. A tutorial on n-support vector machines. APPLIED STOCHASTIC MODELS IN BUSINESS AND INDUSTRY (2005), Vol 21, pp111-136. DOI: 10.1002/asmb.537

[41]: Chris Albon, (2017, Dec 20), “SVC Parameters When Using RBF Kernel”, URL:https://chrisalbon.com/machine_learning/support_vector_machines/svc_parameters_using_rbf_kernel/

[42]: Scikit learn, “sklearn.svm.NuSVC”, URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html

[43]: Scikit learn, “1.17. Neural network models (supervised)”, URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[44]: Sagar Sharma, (2017, Sep 6), “Activation Functions in Neural Networks”, Towards Data Science, URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[45]:Niklas Donges, (2019, Aug 2), “Activation Functions within Neural Networks”, Experfy, URL: https://www.experfy.com/blog/activation-functions-within-neural-networks/

[46]: Afroz Chakure, (2019, Jul 6), “Decision Tree Classification”, Towards Data Science, URL: https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac

[47]: Scikit learn, “1.10. Decision Trees”, URL: https://scikit-learn.org/stable/modules/tree.html#tree

[48]: Jiawei Han, Micheline Kamber, Jian Pei. 8 - Classification: Basic Concepts, Data Mining (Third Edition), The Morgan Kaufmann Series in Data Management Systems (2013), pp.327-391. https://doi.org/10.1016/B978-0-12-381479-1.00008-3

[49]: Tony Yiu, (2019, Jun 12), “Understanding Random Forest”, Towards Data Science URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[50]: Scikit learn, “1.11. Ensemble methods” , URL: https://scikit-learn.org/stable/modules/ensemble.html

[51]: Tin Kam Ho, "Random decision forests," Proceedings of 3rd International Conference on Document Analysis and Recognition, Montreal, Quebec, Canada, 1995, pp. 278-282 vol.1, doi: 10.1109/ICDAR.1995.598994.

[52]: Rosaria Silipo, (2019, Oct 1), “From a Single Decision Tree to a Random Forest”, Towards Data Science, URL: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147

[53]: Wikipedia, “Field-programmable gate array” , URL: https://en.wikipedia.org/wiki/Field-programmable_gate_array

[54]: Xilinx Team, “Zedboard”, URL: https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html

[55]: Digilentinc, “ZedBoard (Zynq™ Evaluation and Development) Hardware User’s Guide”, Version 1.1, August 1, 2012. https://reference.digilentinc.com/_media/zedboard:zedboard_ug.pdf

[56]: A. R. Kapgate and S. S. Agrawal, "Edge based data embedding steganography algorithm using ZedBoard," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2017, pp. 2203-2207, doi: 10.1109/RTEICT.2017.8256991.

[57]: Luis Castano-Londono and Gustavo Adolfo Osorio, “An approach to the numerical solution of one-dimensional heat equation on SoC FPGA”, Research Gate, June 2017, doi: 10.1234/rielac.v38i2.517

[58]: Xilinx, “Vivado High-Level Synthesis” URL: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html

[59]: So-logic, “Basic HLS Tutorial”, (V2019.2) May 2020, URL: https://www.so-logic.net/it/knowledgebase/fpga_universe/tutorials/Basic_HLS_Tutorial

[60]: Xilinx Team, “Vivado Design Suite User Guide, Using the Vivado IDE”, UG893(V2017.3) October 4, 2017. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug893-vivado-ide.pdf

[61]: Xilinx Team, “Vivado Design Suite User Guide Getting Started”, UG910 (v2020.1) August 17, 2020

[62]: Wikipedia, “BMP file format”, URL: https://en.wikipedia.org/wiki/BMP_file_format

[63]: Xilinx Team, “Vivado HLS Optimization Methodology Guide”, UG1270 (V2017.4) December 20, 2017.

[64]: Xilinx Wiki, (2018, September 24) “HLS Video Library”, URL: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841665/HLS+Video+Library

[65]: Uzma Bano Ansari and Tanuja Sarode, “Skin Cancer Detection Using Image Processing”, International Research Journal of Engineering and Technology, vol. 04 Issue 04, April 2017.