TÉCNICO LISBOA

Fractional Order Processing of Satellite Images

Manuel Maria Marques Adegas Bairrão Henriques

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Duarte Pedro Mata de Oliveira Valério
Prof. Mário Rui Melício da Conceição

January 2021

To my family and everyone who believed in me

Acknowledgments

Firstly, I would like to thank Instituto Superior Técnico for these five years of education. During these five years, Técnico has been my second home and provided me the tools to grow and become a better man and mechanical engineer in the future.

I am also thankful to my supervisors, Prof. Duarte Valério and Prof. Rui Melício, that since the beginning of this investigation showed great availability to help me every time I needed. Without their guidance it would not be possible to complete this dissertation. To Prof. Paulo Gordo from INFANTE project who was available to exchange valuable information throughout this study.

To my colleagues and lab mates, who teamed up with me in all group works in this path and to my friends that were always there for me. A special acknowledgment to my girlfriend who gave me all the strength I needed to complete this work.

Lastly but most important, a compliment to my family, specially my parents. They are the main reason I was able to study and provided me with everything I needed to finish my Master and become an engineer.

Resumo

Nesta dissertação, foi desenvolvido um algoritmo computacional para aplicar derivadas de ordem fracionária ao processamento de imagens de satélite de alta definição.

Um dos objetivos deste trabalho foi avaliar a performance de métodos de deteção de contornos, utilizando derivadas fracionárias. Para isso, seis métodos de deteção em escala de cinza foram utilizados. Todos os métodos implementados foram também adaptados para executarem deteção usando imagens de cor. Por fim, foi testada uma possível solução com parâmetros fixos que permitam a segmentação automática de costas nas mais variadas imagens de satélite.

O estado da arte nesta área conclui que os métodos que usam derivadas fraccionárias melhoram os resultados ao nivel de deteção de contornos e aumentam os nives de imunidade ao ruído do processamento de imagem. Também afirma que os métodos que utilizam imagens de cor são melhores. Contudo, nas aplicações já existentes houve sempre dificuldade em perceber quais os parâmetros que optimizam a detecção dos contornos pretendidos.

Todos os detectores implementados revelaram boa performance na segmentação pretendida, tendo a máscara de derivadas fracionárias obtido melhor desempenho geral. Os métodos fracionários igualaram ou melhoraram a performance dos métodos convencionais inteiros. As versões de cor dos detectores apresentaram em geral igual ou melhor desempenho do que os métodos em escala de cinza.

Por fim, foram descobertos pares de parâmetros que permitem a detecção de costas automática com uma média de desempenho acima de 90%, ligeiramente inferior à melhor solução com parâmetros variados.

Palavras-chave:

Satélite

Derivada Fracionária

Deteção Automática

Deteção Baseada em Cor

Deteção em Escala de Cinza

Processamento Fracionário

Abstract

In this dissertation, a computational algorithm was developed in order to apply fractional derivative image processing to high definition Satellite Images.

One objective of this work was to evaluate the use of fractional derivatives on edge detection in the aforementioned scope. For that, six grey-scale fractional detection detectors were used. All implemented methods were also adapted to perform color-based detection. Finally, a search for fixed parameters that allow the automatic detection of coasts in the broad range of types of satellite images was also performed.

The state of the art on this matter concludes that the fractional methods enhance the results of contour detection and improve immunity to noise in image processing. It also states that color based processing can achieve better results. However, in the already existent applications there is a difficulty in achieving fixed parameters that optimize the performance in edge detection.

All the implemented detectors presented good performance in the desired segmentation. The Fractional Derivative operator revealed the best overall performance. The fractional methods matched or enhanced the functioning of the conventional integer ones. The color-based versions of the detectors worked in general as well or better than the grey-scale ones.

Lastly, pairs of parameters were found which allow the automatic detection of coasts with an average performance above 90%, slightly inferior than the best solution with varying parameters.

Palavras-chave:

Satellite

Fractional Derivative

Automatic Detection

Color-Based Detection

Grey-Scale Detection

Fractional Processing

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Derivative order.
Γ	Gamma function.
λ	Eigenvalue corresponding to the largest discontinuity in the chromatic image function.
∇	Gradient.
σ	Standard deviation of the Gaussian function.
θ	Edge orientation.

Roman symbols

h	Spacing between pixels.
th	Threshold.

Subscripts

i, j, k	Computational indexes.
x, y, z	Cartesian components.

Superscripts

$^{\circ}$	Degrees.
T	Transpose.

Chapter 1

Introduction

This first chapter briefly describes the contents of the present dissertation. It includes a small text explaining the relevance of studying fractional image processing and applying it to the aeronautic field, an overview of what will be the studied topics, the objectives and its structure.

1.1 Motivation

Throughout the years, image processing has been an essential tool in the field of aerospace engineering, from inspection of aircraft parts until the navigation of these ships. In order to overcome bad weather's lack of visibility, pilots use processed images to know where they are and where to go. Also, the processed images taken by satellites allow to extract relevant information in different fields such as [1]:

- land monitoring;
- routine mapping;
- surveillance of the marine environment, including oil-spill monitoring;
- ship detection for maritime security;
- mapping to support humanitarian aid and crisis situations.

This list goes on, and everyday appears a new application for this type of images.

Tiago Bento [2] and José Gonçalves [3] already applied fractional edge detection to medical images and arrived to conclusions that show its great potential. It is now relevant to apply this fractional processing to other matters. In this dissertation, this knowledge is applied to the aerospace field by using satellite images acquired from Copernicus Open Access Hub [4]. Being Prof. Rui Melicio part of INFANTE project [5], it is interesting to apply this knowledge to the aerospace field. INFANTE project is an R&D project for the development and in-orbit demonstration of technology for a small satellite, precursor for Earth observation constellations. Thus, this dissertation also aims to help the project with insights that may benefit and enhance current investigations regarding satellite technology.

1.2 Objectives

The objectives of this work are the application of fractional derivatives to image processing in the aeronautical field to:

- Verify if fractional algorithms reveal the same potential in the processing of images outside of the medical area, namely satellite images;
- Compare grey-scale algorithms against color based methods;
- Check if it is possible to identify sets of parameters that allow an automatic treatment of the images, since that revealed impossible in medical applications.

1.3 Thesis Outline

This thesis is composed of five chapters and four appendices:

- **Chapter 1** is the present chapter and it is the introduction, where the objectives, motivation, outline and innovation of the investigation are presented.
- In **Chapter 2** the state of the art in the subject is handed over. This means that the theoretical formulations needed to understand the experiments performed are exposed as well as all the accomplishments in the matter so far.
- **Chapter 3** is where all the experimental implementation is explained. In this case, the work is computational so the implementations are the algorithms developed as well as a few considerations.
- In **Chapter 4** the results of the algorithms' testing are briefly presented and analysed. Extended results are presented in the appendices under the form of tables and plots.
- Finally **Chapter 5** presents conclusions on the results and objectives proposed in the introduction.

1.4 Contribution

In this work, the application of fractional processing of images is carried on [2, 3], only this time, the fractional derivatives are applied to satellite images. This means these are images with a much greater resolution (10980X10980p) which makes their processing much more heavy computationally, but, at the same time, due to this, edge gradients are very well defined. In order to detect coasts in the images, conventional integer algorithms are integrated with fractional derivatives' formulations.

Besides conventional grey-scale fractional processing, recent algorithms perform edge detection using color images. An already existent algorithm will be used in this work. Furthermore, its mathematical formulations will be the base to adapt other detectors to receive as input a color image. A different novel approach is also tested to adapt a zero-crossing fractional edge detector to color-based edge detection. This study intends to check if color based processing is useful in the application in question.

Finally, it is of great interest the automatic detection of coasts for a large and heterogeneous data set. The search for parameters that may present high performance in the detection of coasts in satellite images is also carried out in this work. A high performance solution with fixed parameters that almost matches results with varying parameters is presented. This result allows the algorithm to automatically detect coasts in different images without having to tune parameters.

Chapter 2

State of the Art

In this section, the state of the art relevant for this study is presented. The topics covered here are:

- Satellite Image Processing: section **2.1**
- Fractional Derivatives: section **2.2**
- Image Processing: section **2.3**
- Edge Detection Methods: section **2.4**

2.1 Satellite Image Processing

Satellite images are taken in digital form by artificial satellites. These images are then processed by computers to extract information. Statistical methods are applied to them and after processing pixel values are evaluated.

As mentioned in the introduction, the information retrieved by this processing has many different applications such as land monitoring or support humanitarian aid and crisis situations. Therefore this theme has been deeply studied and developed recently.

According to [6], satellite image processing is a kind of remote sensing which works on pixel resolutions to collect coherent information about the earth surface.

There are four types of resolutions related to satellite imagery [7]:

- **Spatial resolution**: it is determined by the sensors Instantaneous Field of View (IFoV) and is defined as the pixel size of an image that is visible to the human eye being measured on the ground;
- **Spectral resolution**: measures the wavelength internal size and determines the number of wavelength intervals that the sensor measures;
- **Temporal resolution**: defined as the time that passes between various imagery cloud periods;
- **Radiometric resolution**: provides the actual characteristics of the image. It gives the effective bit depth and records the various levels of brightness of the image.

On October 24, 1946, the first photo from space was taken. This grainy, black-and-white photo shown in Figure 2.1 was taken from an altitude of 105 kilometers by a 35-millimeter motion picture camera riding on a V-2 missile.

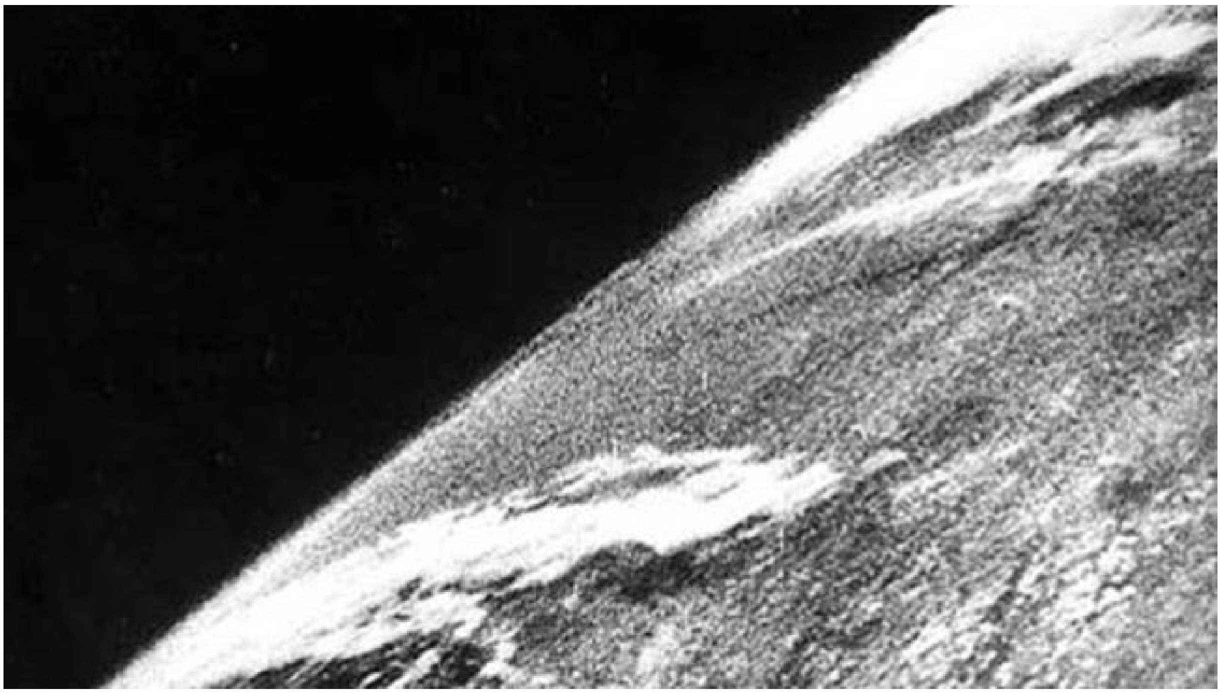


Figure 2.1: *First Photo taken from Space [8]*

Since the 60s, the earth observation field evolved significantly. A film sensitive to wavelengths was developed. This feature could be used to differentiate for example different types of vegetation.

TIROS 1, the first weather satellite, was put in orbit in 1960 [9]. It supplied the US Weather Bureau with daily images of cloud formation and represented a milestone in weather forecasting.

Non-photographic remote sensing technology grew rapidly after the first mapping satellite, Landsat 1, was sent into orbit in 1972 [10]. It was geared with a new type of sensor known as a multi spectral scanner (MSS). With this new technology, data was generated in the form of digital matrices enabling substantial advances in image processing. Today the scanner is a very important tool in remote sensing. It is used on land, in aircraft and on board satellites.

Another important tool used nowadays is the radar sensor [11]. A radar sensor system emits the radiation that it ultimately records and is therefore classified as an active sensor. In simple terms, the radar sensor releases pulses of energy down towards the surface of the Earth. A fraction of the energy is reflected and returns as an 'echo' signal. The power of the returned signal will depend on the roughness and moisture content of the surface and the degree and orientation of sloping in relation to the radar beam. The delay of the 'echo' reveals the distance to the reflecting surface. Radar sensors use energy emitted at longer wavelengths which can infiltrate clouds and haze effectively, and can therefore acquire imagery at night. This provides a significant lead over passive satellites that are hampered by clouds and require sunlight to acquire detailed imagery.

2.1.1 Sentinel-2

In this dissertation, images from the Copernicus Sentinel-2 mission are used [1]. This mission comprises two polar-orbiting satellites placed in the same sun-synchronous orbit, phased at $180°$ in relation to each other. It aims at monitoring variability in land surface conditions, and its wide swath width (290 km) and high revisit frequency will support monitoring of Earth's surface changes.

Each of the Sentinel-2 satellites weights approximately 1200 kilograms. Both have been put into orbit with the European launcher VEGA. The satellite lifespan is approximately 7 years. Batteries and propellants have been provided to accommodate 12 years of operations, including end of life de-orbiting operations.

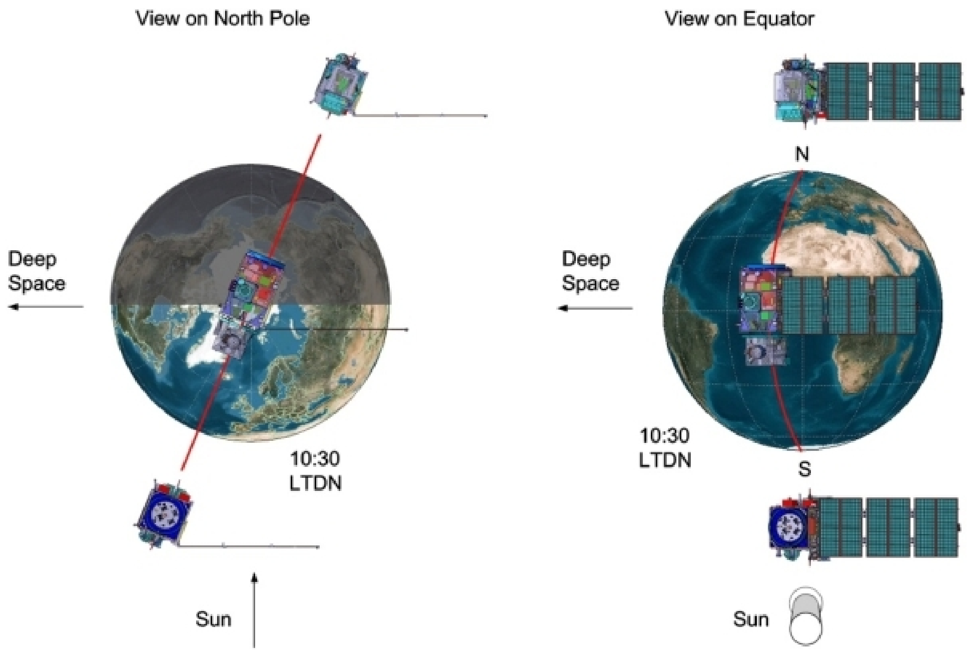


Figure 2.2: *Sentinel-2 Satellites [1]*

The mission objectives are the following [1]:

- systematic acquisitions of high-resolution, multi spectral images allied to a high revisit frequency;
- continuity of multi spectral imagery provided by the SPOT series of satellites and the USGS LANDSAT Thematic Mapper instrument;
- observation data for the next generation of operational products, such as land-cover maps, land-change detection maps and geophysical variables.

These high-level objectives will ensure that Sentinel-2 makes a significant contribution to Copernicus themes, such as climate change, land monitoring, emergency management, and security.

2.2 Fractional Derivatives

In this section, theoretical formulations regarding fractional Derivatives will be demonstrated.

2.2.1 Historic Background

Fractional calculus constitutes an augmentation of the conventional integer calculus that exists for over than three hundred years. Its study was launched by Leibniz and L'Hospital as a conclusion of a written conversation by letters in 1695 [12]. They wondered what would be the derivative if the order was one half rather than an integer order. That year is considered the birth year of fractional calculus.

The subject introduced by these two gentlemen was object of study during the following decades. There are reports from Euler (1730), Lagrange (1772), Laplace (1812), Lacroix (1819), Fourier (1822), Liouville (1832), Riemann (1847), Green (1859), Holmgren (1865), Grunwald (1867), Letnikov (1868), Sonini (1869), Laurent (1884), Nekrassov (1888), Krug (1890), Weyl (1919), among others.

In 1819 Lacroix [13], answered the problem raised by Leibnitz and L'Hospital for the first time. He said that $\frac{d^{1/2}x}{dx^{1/2}}$ was $2\sqrt{x/\pi}$. In [13], Lacroix developed (2.1) for the fractional derivative of $y = x^{\beta}$ with fractional order α :

$$D_x^{\alpha} x^{\beta} = \frac{\Gamma(\beta+1)}{\Gamma(\beta-\alpha+1)} x^{\beta-\alpha} \tag{2.1}$$

where Γ denotes the gamma function which is defined for $z > 0$ by:

$$\Gamma(z) = \int_0^{\infty} e^{-x} x^{z-1} dx \tag{2.2}$$

Substituting $\alpha = 1/2$ and $\beta = 1$ in (2.1) Lacroix's result ($2\sqrt{x/\pi}$) is obtained.

Using the linearity of fractional derivatives, this method may be applied to polynomials and series. However, this class of functions is small in order for the method to be considered a general rule. Nevertheless, this was a first development for practical applications on fractional calculus.

In 1823, Abel presented an application for the subject applying fractional calculus to the resolution of an integral equation of the tautochrone problem [14]. The tautochrone problem is the determination of a curve in the (x, y) plane such that the time needed for a particle to go down the curve until its lowest point under the force of gravity is independent of its initial position (x_o, y_o) on the curve.

The most relevant advances in the fractional calculus field occurred around 1832 when Joseph Liouville began to study fractional calculus after analysing Abel's solution and managed to apply his investigation to potential theory [14]. Liouville proposed two formulas for the fractional derivatives. However, these formulas were not accepted as general formulas for the same reason as Lacroix's formula. Their scope was a narrow group of functions.

One of the most useful breakthroughs in the history of fractional calculus might be due to an investigation paper written by Bernhard Riemann [15]. While attempting to generalize a Taylor series, Riemann obtained the expression that is the most-widely used modern definition of fractional integral.

The formulation that is now called the Riemann-Liouville definition is derived using the Cauchy's integral formula as a starting point. Laurent [16] used a contour given as an open circuit (known as Laurent loop) and arrived to today's definition.

At the same time, Grünwald and Letnikov suggested another definition of fractional derivative which is also frequently used today. Grünwald [17] (1867) adopted the definition of a derivative as the limit of a difference quotient. He arrived at definite-integral formulations for ordinary derivatives, proved that Riemann's integral had to have a finite lower limit and also that Liouville's definition had a $-\infty$ limit. Letnikov also showed that Grünwald-Letnikov (GL) definition coincides, under certain relatively soft conditions, with the Riemann-Louville definition. Nowadays, the GL definition is mainly used for derivation of many numerical methods.

Significant modern advances in fractional calculus were made by Caputo in 1967 [18]. One of the main disadvantages of the Riemann-Liouville definition was that unpractical initial conditions were required. Caputo reformulated this classical definition in order to use classical initial conditions, the same as integer order differential equations. GL, RL and Caputo definitions will be explored in the next subsections.

Fractional calculus has been deeply explored during the last three decades. With the theoretical formulations developed before, new applications to engineering and science problems were created using fractional derivatives. Applications such as fractional conservation of mass [19], tuning of PID controllers [20], groundwater flow problem [21], ultrasonic wave propagation in human cancellous bone modelling [22], modeling contaminant flow in heterogenous porous media [23], fractional dynamics in the trajectory control of redundant manipulators [20], time-space fractional diffusion equation modelling [24], heat diffusion [20], structural damping models [25] or even a variable-order fractional Schrödinger equation [26].

2.2.2 Grünwald-Letnikov Definition

The first order derivative of a function is given by (2.3):

$$D^1 f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.3}$$

Iterating, it is possible to arrive to the n^{th} derivative of a function. (2.4) can be deduced by induction:

$$D^n f(x) = \lim_{h \to 0} h^{-n} \sum_{m=0}^{n} (-1)^m \begin{pmatrix} n \\ m \end{pmatrix} f(x - mh) \tag{2.4}$$

where,

$$\begin{pmatrix} n \\ m \end{pmatrix} = \frac{n!}{m!(n-m)!} \tag{2.5}$$

The Grünwald-Letnikov derivative is a generalization of the derivative in (2.4). The idea behind is that h should approach 0 as n approaches infinity. But before going there, to extend this expression, binomial coefficients must cope with real numbers. For that, the Euler gamma function is used, instead of the factorials in (2.5):

$$\left(\begin{array}{c} a \\ b \end{array}\right) = \begin{cases} \frac{\Gamma(a+1)}{\Gamma(b+1)\Gamma(a-b+1)}, & \text{if } a, b, a-b \notin \mathbb{Z}^- \\ \frac{(-1)^b\Gamma(b-a)}{\Gamma(b+1)\Gamma(-a)}, & \text{if } a \in \mathbb{Z}^- \wedge b \in \mathbb{Z}_0^+ \\ 0, \text{ if } [(b \in \mathbb{Z}^- \vee b-a \in \mathbb{N}) \wedge a \notin \mathbb{Z}^-] \vee (a, b \in \mathbb{Z}^- \wedge |a| > |b|) \end{cases} \tag{2.6}$$

Combining (2.4) with (2.6) it is provided the main justification for the following extension (2.7) of the integer-order derivative to any real order $\alpha > 0$ which was proposed independently by Grünwald [17] and Letnikov [27]:

$$^{\mathrm{GL}}D^\alpha f(t) = \lim_{h\to 0} h^{-\alpha} \sum_{j=0}^{\infty} (-1)^k \left(\begin{array}{c} \alpha \\ k \end{array}\right) f(t-kh), \quad t \in \mathbb{R} \tag{2.7}$$

For practical reasons, a truncation of the expression above was introduced. With an initial value c, the truncated version of the derivative is often preferred since it can be applied to functions that are not defined (in the interval from -∞ to c):

$${}_cD_t^\alpha f(t) = \lim_{h\to 0} h^{-\alpha} \sum_{k=0}^{N} (-1)^k \left(\begin{array}{c} \alpha \\ k \end{array}\right) f(t-kh), \quad N = \left\lfloor \frac{t-c}{h} \right\rfloor, \quad t > c \tag{2.8}$$

Expression (2.8) constitutes the formulation that is going to be applied to the different edge detectors in order to perform fractional image processing.

2.2.3 Riemann-Liouville Definition

The Riemmann-Liouville fractional definition was obtained computing the direct generalization of Cauchy's formula for a n^{th} order integral:

$$\int_a^x dx_1 \int_a^{x_1} dx_2 \ldots \int_a^{x_{n-1}} f(x_n)\, dx_n = \frac{1}{(n-1)!} \int_a^x \frac{f(t)}{(x-t)^{1-n}} dt \tag{2.9}$$

Since, $(n-1)! = \Gamma(n)$, Riemann realized that (2.9) could have a meaning even when n took values that were not integers. So, by generalization, the Riemann-Liouville fractional integral of order α was derived:

$$I_{a+}^\alpha f(x) := \frac{1}{\Gamma(\alpha)} \int_a^x \frac{f(t)}{(x-t)^{1-\alpha}} dt \tag{2.10}$$

Due to the properties of this operator, the Riemann-Louville's fractional derivative can be defined as in (2.11) [28]:

$${}_cD_x^\alpha f(x) = \begin{cases} \int_c^x \frac{(x-t)^{-\alpha-1}}{\Gamma(-\alpha)} f(t)\mathrm{d}t, & \text{if } \alpha \in \mathbb{R}^- \\ f(x), & \text{if } \alpha = 0 \\ \frac{\mathrm{d}^{\lceil\alpha\rceil}}{\mathrm{d}x^{\lceil\alpha\rceil}} {}_cD_x^{\alpha-\lceil\alpha\rceil} f(x), & \text{if } \alpha \in \mathbb{R}^+ \end{cases} \tag{2.11}$$

2.2.4 Caputo's Definition

The Caputo derivative is important to model phenomena using classical initial conditions which are more practical than the initial conditions required by other definitions . It is stated in (2.12):

$$ {}^{C}D_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_0^t \frac{f^{(n)}(\tau)}{(t-\tau)^{\alpha+1-n}} d\tau \tag{2.12} $$

With $\alpha > 0$. If needed, an inferior limit for the derivative may be added:

$$ {}_{a}^{C}D_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(\tau)}{(t-\tau)^{\alpha+1-n}} d\tau \tag{2.13} $$

In contrast with the Riemann-Liouville's derivative, Caputo's derivative of a constant is equal to zero. Caputo's definition for all the $\mathbb{R}$ domain (2.14) can be found in [28]:

$$ {}_{c}D_t^{\alpha} f(t) = \begin{cases} \int_c^t \frac{(t-\tau)^{-\alpha-1}}{\Gamma(-\alpha)} f(\tau) \mathrm{d}\tau, & \text{if } \alpha \in \mathbb{R}^- \\ f(t), & \text{if } \alpha = 0 \\ {}_{c}D_t^{\alpha-\lceil\alpha\rceil} \frac{\mathrm{d}^{\lceil\alpha\rceil}}{\mathrm{d}t^{(\alpha)}} f(t), & \text{if } \alpha \in \mathbb{R}^+ \end{cases} \tag{2.14} $$

2.3 Image Processing

Image processing is a method in which one carries out some operations on an image, in order to get an enhanced image or to obtain some useful information from it. It is a type of signal processing in which input is a picture and output may be a picture or relevant information regarding that image.

But what is an image? In the context of the present dissertation, an image is considered a two-dimensional function $f(x,y)$, where x and y are the coordinates of the pixels within the area $(N \times M)$ of the picture. The value of f is the intensity of pixel (x,y). Computationally speaking, an image is a 2D array organized in rows and columns given by:

$$ f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix} \tag{2.15} $$

The applications of image processing range from medicine to entertainment, passing by geological processing and the main application for this work: **remote sensing**.

The processing of digital images can be divided into four categories:

- image enhancement
- image restoration
- image analysis
- image compression

2.3.1 Image Processing Historic Background

One of the first applications in image processing appeared in the early 1920s in the newspapers industry: the Bartlane System.

The name of the system comes from its two inventors, Bartholomew and Macfarlane, both from the Daily Mirror newspaper. This way of transmitting images within long ranges was invented in 1920 and the first trans-Atlantic cable picture was transmitted in 1921, between London and Halifax [29]. The cable system employs the telegraphic typewriter to convert the picture values into those that will fit the standard forms of communication. Figure 2.3 shows a picture transmitted with the Bartlane System [29].

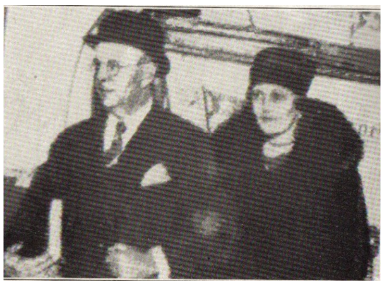


Figure 2.3: *Picture Transmitted and Reproduced by the Bartlane System [29].*

The following years along the decade of the 20s were spent improving the Bartlane System. At the end of the decade, the system retrieved higher quality images. New reproduction processes based on photographic techniques were introduced and the number of tones reproduced in the images increased. The following image presents a 15 tone digital image retrieved from the system at the end of the decade:

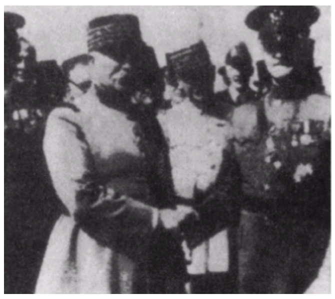


Figure 2.4: *Barltlane System's image at the end of the 1920s(15 tones) [30].*

In the 1960s , enhancements in computing technology and the space race led to a great development of digital image processing. In 1964, image processing techniques were used to improve the quality of images of the moon taken by the Ranger 7 probe. Later these techniques were continuously used in other space missions.

In the 70s, the processing of images started to be used in medical applications and in 1979 the tomography was invented. This invention allowed Sir Godfrey N. Hounsfield and Prof. Allan M. Cormack to win a shared Nobel Prize in medicine.

Since then, digital image processing has exploded and today is used in many different areas.

2.3.2 Image Processing Steps

The processing of digital images can be divided in eleven steps (Figure 2.5) [31]:

- **Image Acquisition**: It is simply capturing the image. This step may include pre-processing;
- **Image Enhancement**: Procedure of filtering image to improve quality (e.g. noise removal, contrast increase);
- **Image Restoration**:Process of enhancing appearance of an image by mathematical or probabilistic models (e.g. deblurring);
- **Color Image Processing**: Usage of colors of an image to extract features of interest;
- **Wavelets and Multi-Resolution Processing**: Representation of the image in different degrees of resolution;
- **Compression**: Technique to reduce storage needed to save an image or the bandwidth required to transmit it;
- **Morphological Processing**: Methods to extract image features that are useful in the representation and description of shape;
- **Segmentation**: Process to divide image components and isolate them. The more accurate the segmentation, the more likely recognition is to succeed.;
- **Representation and Description**: It involves representing an image in different ways:
  - Boundary Representation: it focuses on the external shape characteristics such as corners and inflections;
  - Regional Representation: it centers on internal properties such as texture and skeletal shape;
  - Description: it is also known as Feature Selection and helps extracting relevant information.
- **Recognition**: Process of allocating labels to an object rooted on its description;
- **Knowledge Base**: Detailing regions of an image where the relevant information is known to be located, therefore limiting the search that has to be performed in seeking that figures.

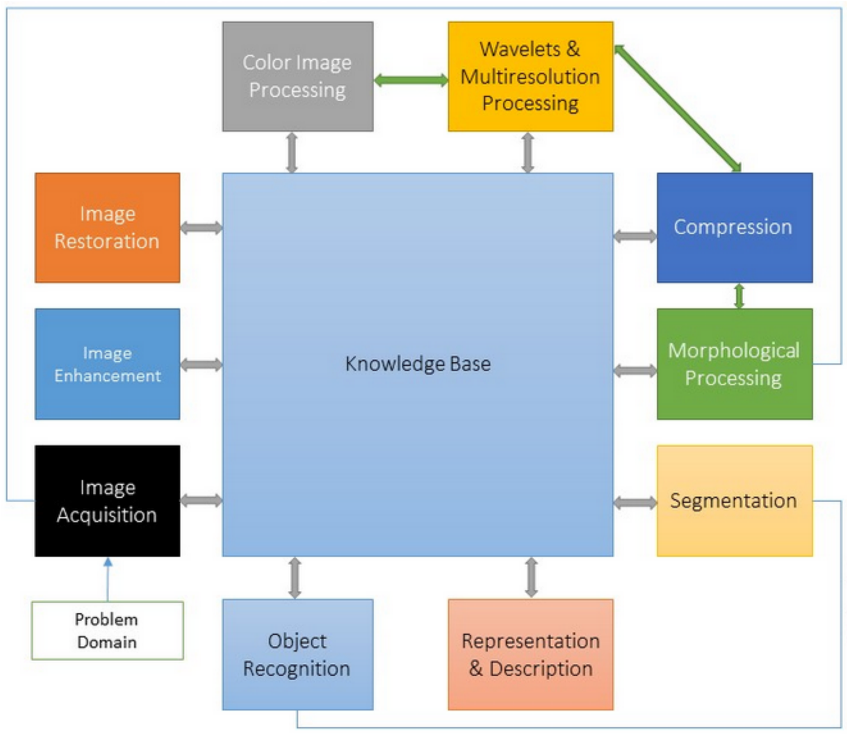


Figure 2.5: *Fundamental Steps of Digital Image Processing [32].*

2.3.3 Convolution with Masks

Convolution is the procedure of summing each element of an image to its local neighbors, weighted by a kernel or mask.

The convolution of an image can be represented by:

$$g(x, y) = h(x, y) * f(x, y) \tag{2.16}$$

where $f(x, y)$ and $h(x, y)$ are respectively the image and the kernel. Note that due to the commutative property these two can be swapped in (2.16).

The mask can be represented by a two-dimensional matrix, generally 1x1, 3x3, 5x5 or 7x7. Note that the mask has usually odd dimensions. That is due to the fact that being odd, the center of the mask can be found. This is important because normally the center corresponds to the target pixel that is assigned with the result of the convolution.

Convolution is composed of the following steps (Figure 2.6):

1. For each pixel in the input image, the mask is placed on top of the image with its origin lying on that pixel;
2. the values of each pixel under the mask are multiplied by the values of the corresponding kernel weights;
3. the results are summed to yield a single output value that is placed in the output image at the location of the pixel being processed on the input.

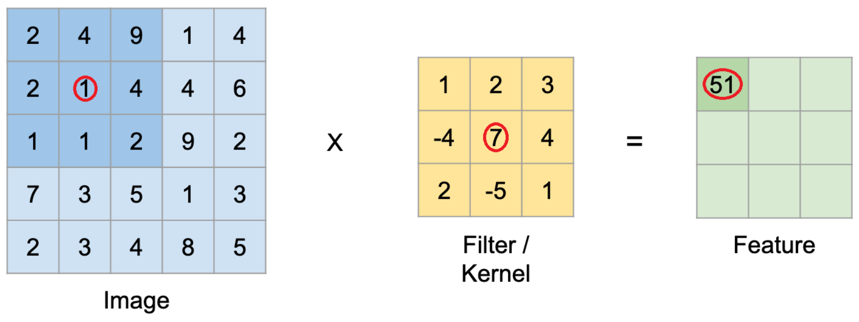


Figure 2.6: *Convolution with masks (Adapted from [33]).*

2.3.4 Derivative Filters

Derivative filters supply a quantitative measurement for the rate of change in pixel value within a digital figure. When this kind of filters are used, the result can be used to enhance contrast, detect edges and boundaries and to measure feature orientation. In this work, edge detection will be performed, thus these filters are of relevant matter.

Convolution of the specimen image with derivative filters is known as derivative filtering operation. In most of the times, there is a filter to each direction so convolution is performed twice. Using two dimensions, a gradient can be measured from the combination of the convolutions in x and y:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \tag{2.17}$$

The gradient presented above points in the direction of most rapid increase in intensity. The magnitude (2.18) and orientation (2.19) of this gradient are frequently used to combine the two convolutions in the processing of images with edges with different orientations.

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{2.18}$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \tag{2.19}$$

2.3.5 Image Segmentation

According to [34], ”Image segmentation is the process of partitioning an image into multiple segments”. It is used in image processing to locate objects and boundaries in images. Processing an entire image can be computationally heavy. Nevertheless, image segmentation may help identifying the regions of interest.

Image segmentation creates a pixel-wise mask for each object in the image. This technique gives us a deep understanding of the object(s) present in the scene.

There are two types of segmentation:

- **Semantic Segmentation**
- **Instance Segmentation**

To understand the difference between these two, let one take a look at Figure 2.7 and Figure 2.8:

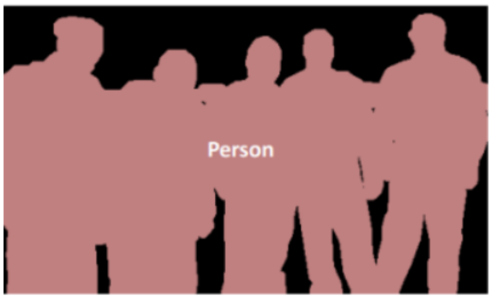


Figure 2.7: Semantic Segmentation [35].

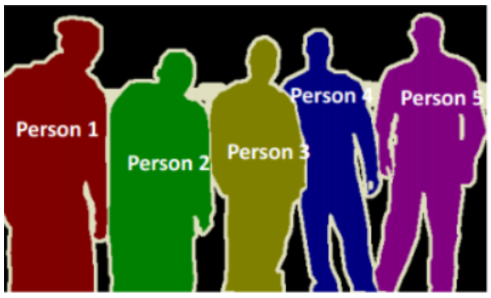


Figure 2.8: Instance Segmentation [35].

In Figure 2.7, every pixel belongs to one class of the two existent. All the pixels belonging to a particular class are represented by the same color. This is an example of Semantic Segmentation.

In Figure 2.8 each person is identified individually so the algorithm assigns a unique color for each individual. This is an example of Instance Segmentation.

One simple way to segment different objects is to use their pixel values. The pixel values will be different for the objects and the image's background if there is a sharp contrast between them. In this case it can be used a technique called thresholding.

2.3.6 Thresholding

In many image processing applications, it is helpful to be able to split the regions of the image corresponding to objects in which one is interested, from the regions that correspond to the background. Thresholding frequently provides a simple and appropriate way of performing this segmentation based on the different intensities of pixels in the foreground and background of an image.

The input of a thresholding operation is typically a gray-scale or color image. In the simplest implementation, the output is a binary image. Usually, black pixels correspond to background and white pixels correspond to foreground. In elementary applications, the segmentation is determined by a single parameter known as the intensity threshold. Each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the pixel is set to white in the output. If it is less than the threshold, it is set to black.

Therefore, an image $f(x, y)$ that is an output of a thresholding operation is defined mathematically by (2.20):

$$f(x,y) = \begin{cases} 0, & \text{pixel} < \text{threshold} \\ 1, & \text{pixel} \geq \text{threshold} \end{cases} \tag{2.20}$$

2.3.6.1 Otsu Threshold

Otsu's thresholding method requires iterating through all the possible threshold values and computing a measure of spread for the pixel levels in each side of the threshold. This means that the pixels that either fall within two classes: the **foreground** or the **background**. The goal of this method is to discover the threshold value where the sum of foreground and background spreads is at its minimum.

First, the method computes the histogram for the values of the pixels in the input image. Then it iterates the threshold value in order to find the optimal one. The metrics used to find it are:

- **Within-class variance**;
- **Between-class variance**.

From the following mathematical equation, variance (σ^2) can be explained as the distribution of the data. The higher the value of variance, the more dispersed the data is:

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N} \tag{2.21}$$

For two classes, within-class variance is given by the following expression:

$$V_w = W_1 * \sigma_1^2 + W_2 * \sigma_2^2 \tag{2.22}$$

where W_i is the number of pixels of class i over the total of pixels.

In terms of the above mentioned metric (V_w), the lower the value of V_w is, the less dispersed the data in each class is (background and foreground). In order to get the optimal threshold value one needs to find the minimal value of V_w.

As it stands, between-class variance is the variance between two classes. In the case of two classes, this metric is given by:

$$(V_b) = W_1 W_2 (\mu_1 - \mu_2)^2 \tag{2.23}$$

To get the suitable threshold value one shall find the maximal value of V_b. It is clear that this metric is computationally faster since it does not require the computation of the variance [36].

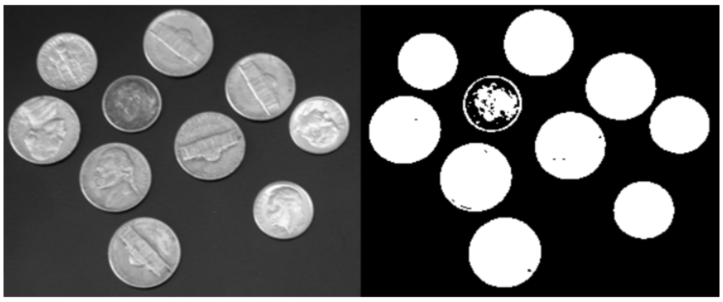


Figure 2.9: *Otsu's Thresholding [36].*

2.4 Edge Detection Methods

Edge detection is an image processing technique that aims to find the limits of objects within images. It works by detecting discontinuities in pixels intensities. It is a crucial tool in segmentation and other areas of image processing.

An edge consists in a set of pixels with high intensity variations in their neighbourhood. These variations can be used to resolve the depth, size orientation and surface properties of a digital image.

There are two types of edge detection: **Gradient** based and **Laplacian** based.

In the gradient based methods, edges are detected by taking the first order derivative of the image and computing the gradient as explained in subsection 2.3.4. The gradient magnitude (2.18) is used to calculate a measure of edge strength. Gradient orientation (2.19) helps to identify local edge orientation. Sobel, Canny, Prewitt and Roberts are examples of edge detection algorithms that use the gradient.

In the laplacian based methods, the second order derivative of the image which may have zero-crossings is computed. Generally, edges are found by searching zero-crossings of a non-linear differential expression. Usually, a Gaussian smoothing is performed before using this type of method. This is due to the fact that the 2^{nd} derivative is very sensitive to noise. The smoothing operation filters noise and allows better edge detection. The most relevant relevant second order derivative based detector is the Laplacian of Gaussian (LoG).

In the following subsections, the above mentioned algorithms for edge detection will be detailed.

2.4.1 Integer Edge Detection Methods

2.4.1.1 Gradient Based operators

2.4.1.1.1 Sobel operator

The Sobel operator performs a 2-D spatial gradient measurement on an image and so it highlights regions where there are sudden increases of pixel intensity which correspond to edges.

The operator consists of two masks: one for the gradient in x direction (G_x) and other for the gradient in y direction (G_y). Note that G_y is nothing more than G_x rotated 90 degrees [37].

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

Gx

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

Gy

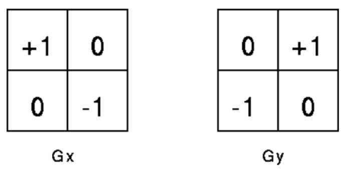
Figure 2.10: *Sobel convolution masks [37].*

These kernels are planned to respond maximally to edges that are vertical and horizontal relatively to the pixel grid, one kernel for each of the two perpendicular orientations. The masks can be applied separately to the input image, in order to produce different measurements of the gradient component in each orientation (G_x and G_y). These can then be combined together to find the magnitude of the gradient at each point and the orientation of that gradient with respectively (2.18) and (2.19).

2.4.1.1.2 Roberts Cross operator

The Roberts Cross operator [38] performs a simpler, quick way to compute, 2-D spatial gradient measurement on an image. It also summits regions with great variations in pixel intensity that correspond to edges. The input and output of the operator are grey-scale images. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.

The Roberts operator consists of a pair of 2x2 masks, again one for each direction. Here, the mask to compute one direction's gradient is the other one rotated 90 degrees as one can observe in Figure 2.11 [37].



Gx

+1	0
0	-1

Gy

0	+1
-1	0

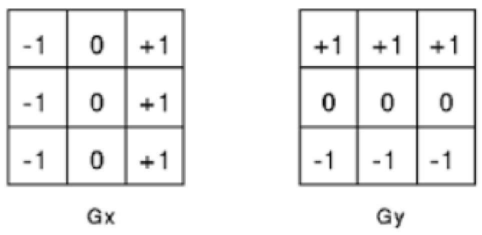
Figure 2.11: *Roberts convolution masks [37].*

The combination of the two gradients in order to find the magnitude and orientation is once more performed using the above mentioned expressions.

2.4.1.1.3 Prewitt operator

Prewitt operator is used for edge detection in order to find two types of edges: Horizontal and Vertical

Detection of edges is performed by using the difference between corresponding pixel intensities of an image. For this, again, a derivative mask is used. Two 3x3 operators, one for each direction (Figure 2.12) [39].



Gx

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

Gy

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

Figure 2.12: *Prewitt convolution masks [39].*

When these masks are applied individually on a image it outputs only the horizontal or vertical edges. As the center column/row is zero, it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. This increases the edge pixels' intensity and it becomes enhanced comparatively to the original image. Once more these two masks may be combined in order to try to detect all edges in one image.

2.4.1.1.4 Canny operator

The Canny edge detector is a very popular edge detection algorithm. It was developed in 1986 by John F. Canny [40]. The algorithm is composed of the following steps:

1. Noise Reduction;
2. Gradient Calculation;
3. Non-maximum Suppression;
4. Hysteresis Thresholding.

The first step of the Canny algorithm is **Noise Reduction**. Since image processing is always vulnerable to noise, it is important before processing to remove or reduce it. This is possible convolving the image with a Gaussian Filter. Then, a simple 2-D first derivative operator (in the case of the algorithm used in this work is the derivative of the Gaussian function used to smooth the image) is applied to the smoothed image to highlight regions of the image with high first spatial derivatives (2.24) and (2.25).

$$\nabla g(x,y) = \nabla(G(x,y,\sigma) * f(x,y)) = \nabla(G(x,y,\sigma) * f(x,y) \tag{2.24}$$

$$E_x = \frac{\partial G(x,y,\sigma)}{\partial x} * f(x,y) \quad E_y = \frac{\partial G(x,y,\sigma)}{\partial y} * f(x,y) \tag{2.25}$$

This step of the process is called **Gradient Calculation**. In (2.24) and (2.25), E_x and E_y are the gradients in each direction and $G(x,y,\sigma)$ the Gaussian filter defined as:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2.26}$$

After computing gradient magnitude and orientation with (2.18) and (2.19) , a full scan is performed in order to remove any unwanted pixels which may not constitute edges (**Non-maximum Suppression**). In order to do this, for every pixel, it is checked if the point is a local maximum in its neighborhood, in the direction of the gradient. Let one check the following example in Figure 2.13 [41, 42].

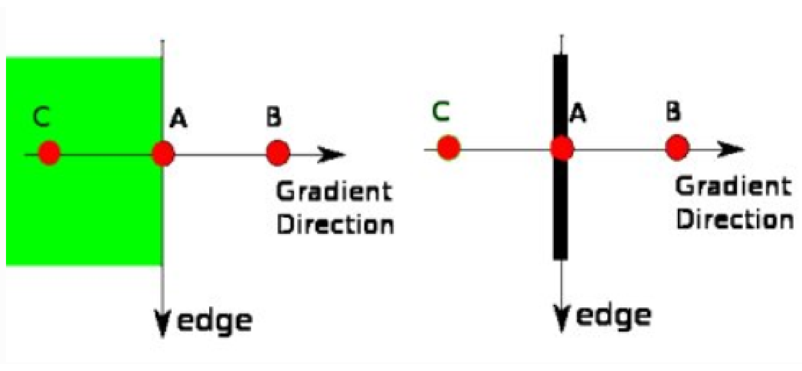


Figure 2.13: *Canny's Non Maximum Supression [41].*

Observing Figure 2.13, one can see that point A is on the edge (vertical edge). Gradient orientation is perpendicular to the edge. Points B and C are in the gradient direction. So the value for the pixel in point A is compared with the values from points B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). Therefore, in this case all the points in the direction of A are suppressed, including C and B. In conclusion, the result obtained from this phase is an intensity image with edges with a thickness of one pixel.

The final step of the algorithm is the **Hysteresis Thresholding**. In this phase, the algorithm decides which edges are suitable for the output image. Each edge has an intensity proportional to the magnitude of the gradient. For this, two threshold values are defined, the minimum and maximum values. All gradients higher than the maximum threshold are considered ”sure-edges”. In contrast, the gradients that are lower than the minimum threshold are considered ”non-edges”. For the gradients that are between the two values two instances may occur [43]:

- if the pixels in question are connected to “sure-edge” pixels, they are considered to be part of edges;
- otherwise, these pixels are also discarded and considered non-edges.

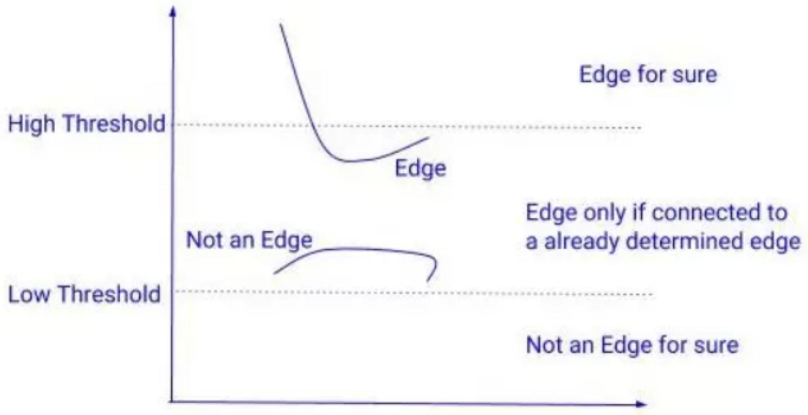


Figure 2.14: *Canny’s Hysteresis Thresholding [42].*

This stage also removes small pixels noises on the assumption that edges are long lines. After this, the algorithm retrieves an output image with the edges identified by the detector.

The effect of the Canny operator is determined mostly by two parameters [44]: the width of the Gaussian kernel used in the smoothing phase (σ), and the upper and lower thresholds used in the hysteresis thresholding phase. Augmenting the width of the Gaussian kernel reduces the detector’s sensitivity to noise, at the expense of losing some of the finer detail in the image. Usually, the maximum threshold can be set quite high, and the lower threshold quite low for good results. Setting the lower threshold too high can cause important edges not to be detected. Setting the maximum threshold too low can lead to detection of edges that are actually residual noise that were not eliminated with the smoothing operation.

2.4.1.2 Laplacian Based operators

2.4.1.2.1 Laplacian of Gaussian operator

In image processing, the Laplacian is a measure of the 2^{nd} spatial derivative of an image. It allows the identification of regions with a rapid change of pixels intensity and is thus often used for edge detection. The Laplacian is frequently used in an image that has first been smoothed with a Gaussian smoothing filter in order to reduce noise intensity. The operator generally takes a single grey-scale image as input and produces another grey-level image as output. The Laplacian of a 2D Image is given by the following expression:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{2.27}$$

Since the image is a representation of discrete pixels, we have to find a discrete convolution mask that can approximate the second derivatives in the formulation of the Laplacian. In the discrete domain, the simplest approximation to the continuous Laplacian is to compute the difference of slopes along each axis:

$$\frac{\partial^2 f}{\partial x^2} = f(i,j+1) - 2f(i,j) + f(i,j-1) \tag{2.28}$$

$$\frac{\partial^2 f}{\partial y^2} = f(i+1,j) - 2f(i,j) + f(i-1,j) \tag{2.29}$$

Substituting (2.28) and (2.29) in (2.27), the first kernel of (2.30) is obtained. The second, a non-separable eight-neighbor Laplacian defined by the gain-normalized impulse response array, was suggested by Prewitt. The mask on the right is a separable eight-neighbor version of the Laplacian [45].

$$\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & -8 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -1 & 2 & -1 \\ \hline 2 & -4 & 2 \\ \hline -1 & 2 & -1 \\ \hline \end{array} \tag{2.30}$$

Using one of these three kernels, the Laplacian can be calculated convolving them with the image. Since these kernels are approximating a second derivative on the image, they are very sensitive to noise. To tackle this, the image is Gaussian smoothed before applying the Laplacian filter reducing high frequency noise.

The smoothing filter can also be convolved first with the Laplacian kernel and only then convolve the result with the input image. This process presents two main advantages:

- since both the Gaussian and the Laplacian kernels are normally smaller than the input image, this method usually requires less arithmetic operations;
- The LoG kernel can be pre-calculated in advance so only one convolution needs to be performed saving time in processing.

The Laplacian of Gaussian operator of a 2D Image is defined by (2.31) and is illustrated in Figure 2.15 [37]:

$$\mathrm{LoG}(x,y) = -\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.31}$$

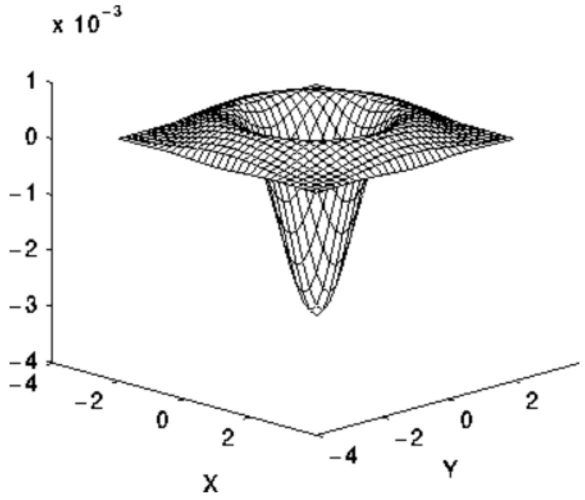


Figure 2.15: *2-D Laplacian of Gaussian operator [37].*

2.4.2 Fractional Edge Detection Methods

2.4.2.1 Fractional Roberts Operator

Nowadays, fractional calculus attracts as much attention as the integer-order differential algorithms. As for the definition of fractional derivatives, there are a few. Three of the most popular definitions are detailed in section 2.2: G-L, R-L and Caputo's definitions.

The authors of [46] present the application of Grünwald-Letnikov definition to the integer Roberts edge detector and arrive to a kernel for a fractional order operator. It is known that the Roberts expression for the gradients stands:

$$g(x,y) = |\nabla f(x,y)| = \left\{\ [f(x,y+1)-f(x+1,y)]^2+[f(x+1,y+1)-f(x,y)]^2\ \right\}^{\frac{1}{2}} \tag{2.32}$$

Combining (2.8) with (2.6) the authors arrived at expressions for the gradient's components:

$$\begin{aligned}&\tfrac{\partial^\alpha f(x,y)}{\partial x^\alpha} \approx f(x,y)+(-\alpha)f(x-1,y)+\tfrac{(-\alpha)(-\alpha+1)}{2}f(x-2,y)\\ &+\cdots+\tfrac{(-1)^{n-1}\Gamma(\alpha+1)}{(n-1)!\Gamma(\alpha-n+2)}f(x-n,y)\end{aligned} \tag{2.33}$$

$$\begin{aligned}&\tfrac{\partial^\alpha f(x,y)}{\partial y^\alpha} \approx f(x,y)+(-\alpha)f(x,y-1)+\tfrac{(-\alpha)(-\alpha+1)}{2}f(x,y-2)\\ &+\cdots+\tfrac{(-1)^{n-1}\Gamma(\alpha+1)}{(n-1)!\Gamma(\alpha-n+2)}f(x,y-n)\end{aligned} \tag{2.34}$$

Referring to (2.33) and (2.34), the 3×3 fractional differential mask can be constructed in the eight central symmetric directions, which are negative x-coordinate, negative y-coordinate, positive x-coordinate, positive y-coordinate, left downward diagonal, right upward diagonal, left upward diagonal, and right downward diagonal. The sum of the eight directional masks yields:

$\frac{\alpha^2-\alpha+2}{2}$	$\frac{\alpha^2-\alpha+2}{2}$	$\frac{\alpha^2-\alpha+2}{2}$
$\frac{\alpha^2-\alpha+2}{2}$	-8α	$\frac{\alpha^2-\alpha+2}{2}$
$\frac{\alpha^2-\alpha+2}{2}$	$\frac{\alpha^2-\alpha+2}{2}$	$\frac{\alpha^2-\alpha+2}{2}$

(2.35)

Note that the mask in (2.35) with $\alpha = 1$ corresponds to the center Laplacian mask in (2.30).

Combining the fractional mask to the Roberts operator defined by (2.32), the authors arrived at a solution for edge detection in which the texture of the image is enhanced and small edges are also detected. (2.36), (2.37) and (2.38) represent the mathematical formulation for this combination:

$$D^{\alpha}[g(x,y)] = \frac{\partial^{\alpha} g(x,y)}{\partial x^{\alpha}} + \frac{\partial^{\alpha} g(x,y)}{\partial y^{\alpha}} \tag{2.36}$$

$$\begin{aligned}
&\tfrac{\partial^2 g(x,y)}{\partial x^2} \approx g(x,y) + (-\alpha)g(x-1,y) + \tfrac{(-\alpha)(-\alpha+1)}{2} g(x-2,y) \\
&+\cdots+ \tfrac{(-1)^{n-1}\Gamma(\alpha+1)}{(n-1)!\Gamma(\alpha-n+2)} g(x-n,y) \\
&= \left\{ \begin{array}{l} [f(x,y+1)-f(x+1,y)]^2 \\ +[f(x+1,y+1)-f(x,y)]^2 \end{array} \right\}^{\frac{1}{2}} + \\
&(-\alpha)\left\{ \begin{array}{l} [f(x-1,y+1)-f(x,y)]^2 \\ +[f(x,y+1)-f(x-1,y)]^2 \end{array} \right\}^{\frac{1}{2}} + \\
&\tfrac{(-\alpha)(-\alpha+1)}{2}\left\{ \begin{array}{l} [f(x-2,y+1)-f(x-1,y)]^2 \\ +[f(x-1,y+1)-f(x-2,y)]^2 \end{array} \right\}^{\frac{1}{2}} +\cdots+ \\
&\tfrac{(-1)^{n-1}\Gamma(\alpha+1)}{(n-1)!\Gamma(\alpha-n+2)}\left\{ \begin{array}{l} [f(x-n,y+1)-f(x-n+1,y)]^2 \\ +[f(x-n+1,y+1)-f(x-n,y)]^2 \end{array} \right\}
\end{aligned} \tag{2.37}$$

$$\begin{aligned}
&\tfrac{\partial^{\alpha} g(x,y)}{\partial y^{\alpha}} \approx g(x,y) + (-\alpha)g(x,y-1) + \tfrac{(-\alpha)(-\alpha+1)}{2} g(x,y-2) \\
&+\cdots+ \tfrac{(-1)^{n-1}\Gamma(\alpha+1)}{(n-1)!\Gamma(\alpha-n+2)} g(x,y-n) \\
&= \left\{ \begin{array}{l} [f(x,y+1)-f(x+1,y)]^2 \\ +[f(x+1,y+1)-f(x,y)]^2 \end{array} \right\}^{\frac{1}{2}} + \\
&(-\alpha)\left\{ \begin{array}{l} [f(x,y)-f(x+1,y-1)]^2 \\ +[f(x+1,y)-f(x,y-1)]^2 \end{array} \right\}^{\frac{1}{2}} + \\
&\tfrac{(-\alpha)(-\alpha+1)}{2}\left\{ \begin{array}{l} [f(x,y-1)-f(x+1,y-2)]^2 \\ +[f(x+1,y-1)-f(x,y-2)]^2 \} ^{\frac{1}{2}} +\cdots+ \end{array} \right. \\
&\tfrac{(-1)^{n-1}\Gamma(\alpha+1)}{(n-1)!\Gamma(\alpha-n+2)}\left\{ \begin{array}{l} [f(x,y-n+1)-f(x+1,y-n)]^2 \\ +[f(x+1,y-n+1)-f(x,y-n)]^2 \end{array} \right\}^{\frac{1}{2}}
\end{aligned} \tag{2.38}$$

where $f(x,y)$ is the the original image, and $g(x,y)$ is the processed image using an integer Roberts operator.

From the experimental results in [46], it was concluded that the improved algorithm has the advantages of Roberts, that is, obtaining thinner edges, besides allowing edge enhancement as one can observe in Figure 2.16.

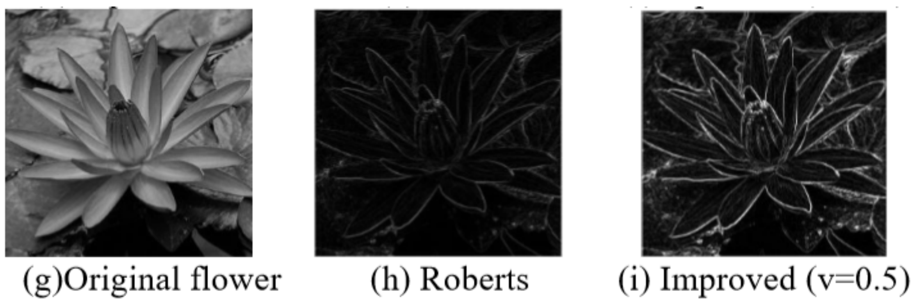


Figure 2.16: *Roberts operator combination with fractional operator [46].*

2.4.2.2 Fractional Sobel Operator

Following the same line of thought, Charles Yaacoub [47] presented a fractional Sobel operator.

Sobel's approximations to the first order numerical derivative applied in two-dimensional space result in the filter masks shown in Figure 2.10. Thus, the resulting image gradient components can be expressed as:

$$\begin{aligned} G_x = &-f(x-1,y-1) - 2f(x-1,y) - f(x-1,y+1) \\ &+ f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1) \end{aligned} \tag{2.39}$$

$$\begin{aligned} G_y = &-f(x-1,y-1) - 2f(x,y-1) - f(x+1,y-1) \\ &+ f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1) \end{aligned} \tag{2.40}$$

By applying the G-L definition (2.8) to G_x expressed in (2.39), the fractional α-th order derivative of G_x yields:

$$\begin{aligned} D^\alpha G_x = &\frac{\alpha(-\alpha+1)(-\alpha+2)}{12} \cdot [f(x-4,y-1) + 2f(x-4,y) + f(x-4,y+1)] \\ &+ \frac{\alpha(-\alpha+1)}{4} \cdot [f(x-3,y-1) + 2f(x-3,y) + f(x-3,y+1)] \\ &+ \left[\frac{\alpha}{2} - \frac{\alpha(-\alpha+1)(-\alpha+2)}{12}\right] \cdot [f(x-2,y-1) + 2f(x-2,y) + f(x-2,y+1)] \\ &+ \left[-\frac{1}{2} - \frac{\alpha(-\alpha+1)}{4}\right] \cdot [f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)] \\ &- \frac{\alpha}{2}[f(x,y-1) + 2f(x,y) + f(x,y+1)] \\ &+ \frac{1}{2}[f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)] \end{aligned} \tag{2.41}$$

The gradient in (2.41) can be obtained by convolving the image $f(x,y)$ with the filter mask presented in (2.42):

$\frac{\alpha(-\alpha+1)(-\alpha+2)}{12}$	$\frac{\alpha(-\alpha+1)(-\alpha+2)}{6}$	$\frac{\alpha(-\alpha+1)(-\alpha+2)}{12}$
$\frac{\alpha(-\alpha+1)}{4}$	$\frac{\alpha(-\alpha+1)}{2}$	$\frac{\alpha(-\alpha+1)}{4}$
$\frac{\alpha}{2} - \frac{\alpha(-\alpha+1)(-\alpha+2)}{12}$	$\alpha - \frac{\alpha(-\alpha+1)(-\alpha+2)}{6}$	$\frac{\alpha}{2} - \frac{\alpha(-\alpha+1)(-\alpha+2)}{12}$
$-\frac{1}{2} - \frac{\alpha(-\alpha+1)}{4}$	$-1 - \frac{\alpha(-\alpha+1)}{2}$	$-\frac{1}{2} - \frac{\alpha(-\alpha+1)}{4}$
$-\frac{\alpha}{2}$	$-\alpha$	$\frac{\alpha}{2}$
$\frac{1}{2}$	1	$\frac{1}{2}$

(2.42)

Since the mask has an even number of rows, the origin is not centered. In the mask above the origin is considered to be located on the 5^{th} row and 2^{nd} column.

A similar reasoning can be applied to the y-direction and the conclusion that the mask in y is the mask in x transposed can be withdrawn.

According to the authors, the proposed edge detector was able to reduce the number of false edge pixels while presenting thinner edges, compared to the conventional Sobel-based edge detection.

2.4.2.3 Fractional LoG Operator

In 2014, the authors of [48] presented a fractional adaptation for the first operator in (2.30) (using the symetric mask).

In a discrete function (f), the operator corresponds to the approximation in (2.43):

$$G(f) = -f(x-1,y) - f(x,y-1) + 4f(x,y) - f(x,y+1) - f(x+1,y) \tag{2.43}$$

Decomposing and noting that for this case $h = 1$, (2.44) may be derived:

$$G(f) = \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y} - \frac{\partial f(x,y+1)}{\partial y} - \frac{\partial f(x+1,y)}{\partial x} = -\frac{\partial^2 f(x+1,y)}{\partial x^2} - \frac{\partial^2 f(x,y+1)}{\partial y^2} \tag{2.44}$$

By generalizing the order from integer to fractional, a fractional-order differential form of the Laplacian operator can be obtained:

$$G^\alpha(f) = -\frac{\partial^\alpha f(x+1,y)}{\partial x^\alpha} - \frac{\partial^\alpha f(x,y+1)}{\partial y^\alpha} \tag{2.45}$$

Using the G-L definition for the fractional order derivative as it was used for the other operators, one may arrive to (2.46):

$$\begin{aligned} G^a(f) &= -\sum_{k=0}^{K-1}(-1)^k C_k^a f(x+1-k,y) - \sum_{k=0}^{K-1}(-1)^k C_k^a f(x,y+1-k) - \\ &= \left[f(x+1,y) - \alpha f(x,y) + \frac{\alpha^2-\alpha}{2} f(x-1,y) + \ldots + (-1)^{K-1} C_{K-1}^\alpha f(x+2-K,y)\right] \\ &- \left[f(x,y+1) - \alpha f(x,y) + \frac{\alpha^2-\alpha}{2} f(x,y-1) + \ldots + (-1)^{K-1} C_{K-1}^\alpha f(x,y+2-K)\right] \end{aligned} \tag{2.46}$$

With the definition above, the mask that performs the calculation of the fractional laplacian may be constructed:

0	$\dots$	0	$(-1)^{\kappa}C^{\alpha}_{K-1}$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$\dots$	0	$\left(\alpha-\alpha^2\right)/2$	0
$(-1)^{\kappa}C^{\alpha}_{K-1}$	$\dots$	$\left(\alpha-\alpha^2\right)/2$	2α	-1
0	$\dots$	0	-1	0

(2.47)

Experiments with (2.47) show that, the larger the order of differentiation is the better the image feature is preserved, but the more noise appears too.

2.4.2.4 CRONE Operator

In 2002, Benoît Mathieu wanted to prove that an edge detector based on fractional differentiation could improve edge detection and detection selectivity in the case of parabolic luminance transitions.

He started by analysing the detection power of fractional differentiation. To do so, a fractional derivative was applied to a step type parabolic transition and the abscissa of its maximum compared to the inflexion point of the transition (Figure 2.17) [49].

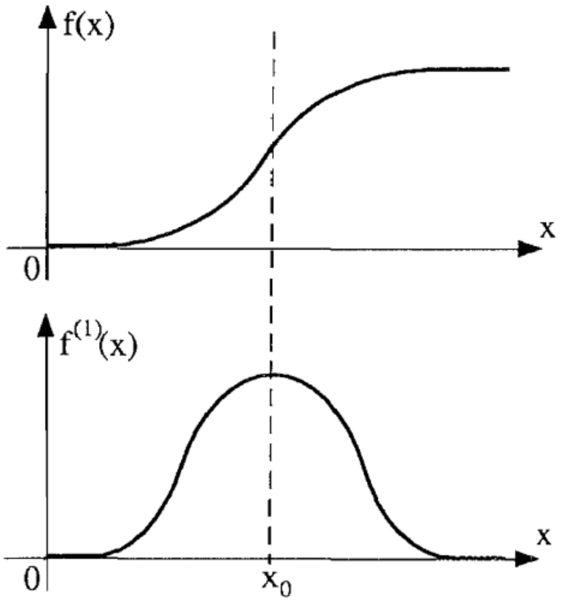


Figure 2.17: *Parabolic Step-type transition and its derivative [49].*

The analysis comprises two intervals for the derivative order: initially values between 0 and 1, and then between 1 and 2. This division is relevant because each of these intervals defines a different shape for the derivatives of the transition curve.

For $\alpha \in]0, 1[$, there is a shift between the inflexion point of the function and its derivative. The closer α is from zero, the greater the shift is (Figure 2.18) [49]. This means the maximum of the derivative for these orders is always on the right of the inflexion point.

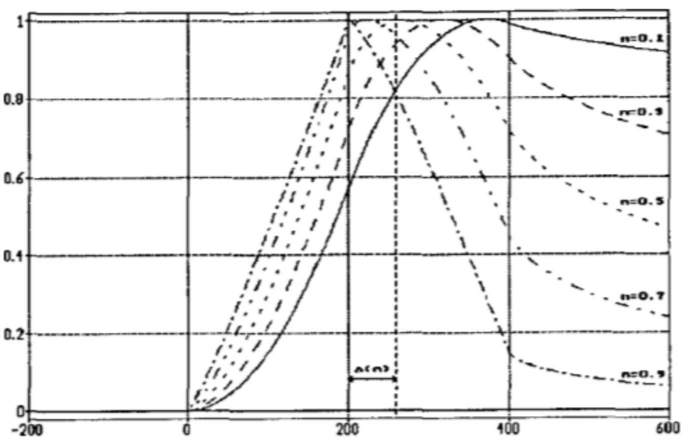


Figure 2.18: *Normalized modulus of $f^{(n)}(x)$ for $\alpha \in]0,1[$ [49].*

For $\alpha \in]1,2[$, the abscissa of the derivative's maximum corresponds to the inflexion point of the parabolic step curve. It is also noted that there is an infinite slope to the right of the inflexion point. This allows not only the detection of the transition inflexion point but also high selectivity.

With the aim of achieving better detection selectivity, an operator which presents an infinite slope in both sides of the inflexion point was developed. This was achieved by using the derivative with orders of $\alpha \in]1,2[$. The solution steps are the following:

1. take the opposite of the derivative function calculated with decreasing x (Figure 2.19b);
2. add the derivative function calculated with increasing x (Figure 2.19a).

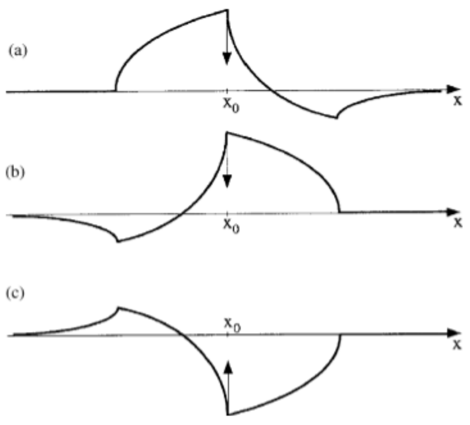


Figure 2.19: *(a) obtained $f^{(n)}(x)$ for increasing x, (b) inverted $f^{(n)}(x)$ for decreasing x, (c) obtained $f^{(n)}(x)$ for decreasing x [49].*

The spatial operator which defines CRONE ("Contour Robuste d'Ordre Non Entier"), therefore subtracts $f^{(n)}(x)$ calculated using decreasing x, from $f^{(n)}(x)$ calculated using increasing x. The operator is written as $\underset{\leftrightarrow}{D}^n$ and is illustrated in Figure 2.20 [49].

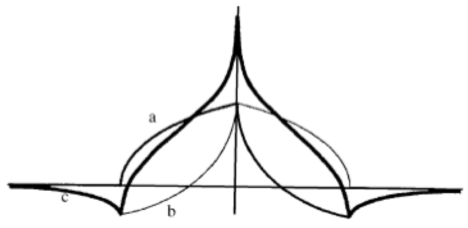


Figure 2.20: *(a) obtained $f^{(n)}(x)$ for increasing x, (b) inverted $f^{(n)}(x)$ for decreasing x, (c) CRONE detector [49].*

2.4.2.4.1 Mathematical formulation of $\underset{\leftrightarrow}{D}^n$

The first derivative of a function $f(x)$, calculated with increasing x can be defined by:

$$\underset{\rightarrow}{D}^n f(x) = \frac{f(x) - f(x-h)}{h} \tag{2.48}$$

and with decreasing x:

$$\underset{\leftarrow}{D}^n f(x) = \frac{f(x) - f(x+h)}{h} \tag{2.49}$$

being h infinitely small.

It is relevant to introduce a shift operator q which is defined by:

$$\begin{aligned} qf(x) &= f(x+h) \\ q^{-1}f(x) &= f(x-h) \end{aligned} \tag{2.50}$$

Using the shift operator on the directional derivative yields:

$$\begin{aligned} \underset{\rightarrow}{D} f(x) &= \frac{1-q^{-1}}{h} f(x) \\ \underset{\leftarrow}{D} f(x) &= \frac{1-q}{h} f(x) \end{aligned} \tag{2.51}$$

From the expressions above it is clear that:

$$\begin{aligned} \underset{\rightarrow}{D} &= \frac{1-q^{-1}}{h} \\ \underset{\leftarrow}{D} &= \frac{1-q}{h} \end{aligned} \tag{2.52}$$

Generalizing to a order n, $\underset{\rightarrow}{D}^n$ and $\underset{\leftarrow}{D}^n$ can be defined as:

$$\underset{\rightarrow}{D}^n = \left(\frac{1-q^{-1}}{h}\right)^n \tag{2.53}$$

$$\underset{\leftarrow}{D}^n = \left(\frac{1-q}{h}\right)^n \tag{2.54}$$

As explained before, the bidirectional detector can be constructed by a composition of the two unidirectional operators using the following expression:

$$\underset{\leftrightarrow}{D}^n = \underset{\rightarrow}{D}^n - \underset{\leftarrow}{D}^n = \frac{1}{h^n}\left[\left(1-q^{-1}\right)^n - (1-q)^n\right] \tag{2.55}$$

Expanding $(1-q^{-1})^n$ and $(1-q)^n$ using Newton's binomial formula, the expression above can be rewritten:

$$\begin{aligned}\underset{\leftrightarrow}{D}^n =& \frac{1}{h^n}\sum_{k=0}^{\infty}(-1)^k \frac{n(n-1)\cdots(n-k+1)}{k!} \\ & \times \left(q^{-k}-q^k\right)\end{aligned} \tag{2.56}$$

Applying the operator to a function, for example the transition studied before:

$$\underset{\leftrightarrow}{D}^n f(x) = \frac{1}{h^n}\sum_{k=0}^{\infty} a_k[f(x-kh)-f(x+kh)] \tag{2.57}$$

where

$$\begin{aligned}a_k &= (-1)^k \begin{pmatrix} n \\ k \end{pmatrix} \\ &= (-1)^k \frac{n(n-1)\cdots(n-k+1)}{k!}\end{aligned} \tag{2.58}$$

In order to detect edges on images, the formulated detector must be designed in two dimensions. It must be considered then a vectorial operator consisting in two independent components, x and y. Each of the components is a truncated single dimension CRONE detector. The horizontal and vertical component masks are shown respectively in (2.59) and (2.60).

$$\begin{bmatrix} +a_m & \cdots & +a_1 & 0 & -a_1 & \cdots & a_m \end{bmatrix} \tag{2.59}$$

$$\begin{bmatrix} +a_m \\ \vdots \\ +a_1 \\ 0 \\ -a_1 \\ \vdots \\ -a_m \end{bmatrix} \tag{2.60}$$

The detector was experimented in artificial and real images and performance compared with Prewitt operator's. In all cases CRONE detector showed better immunity to noise.

2.4.3 Color-Based Edge Detection

Nowadays, the methods using grey-scale images yet described are the ones most often used in edge detection. Nevertheless, detectors that use as input the color information of images also exist. The main difference between coloured images and gray-level images is that, in a colour image, a vector (which generally consists of the three components of the RGB space) is assigned to a pixel, while a scalar grey-level magnitude is assigned to a pixel of a grey-level image. This means that in color-based image processing, the image functions are measured as vectors instead of scalar values.

The techniques used in color-based edge detection can be subdivided on the basis of their principle procedures into two classes [50]:

- **Monochromatic-based techniques**: treat values from the color channels first separately and then combine them together;
- **Vector-valued techniques**: treat the color information as color vectors in a vector space provided with a vector norm.

Until now, the majority of the existent color edge detection methods are monochromatic-based techniques, which produce, in general, better results than grey-level edge detection [51].

While in grey-level images a great variation in the intensity is considered as an edge, the term “color edge” has not been permanently defined for color images. Throughout the years few definitions have been proposed. One of the elder ones is that an edge exists precisely in the color image if the intensity image contains an edge [52]. However, this definition ignores that discontinuities in the hue and saturation channels may occur. For example, the edges of objects that are next to each other and that have the same value but different colors (Hue) might not be recognized using this definition. A second definition for a color edge states that it exists if at least one of the color components contains an edge. But merging the edge detection results from the different color components may lead to some localization inaccuracies of edges in the individual color channels. A third monochromatic-based definition for color edges is rooted on the computation of the sum of absolute values of the gradients for the three channel components. According to this definition, a color edge exists if the summation of the magnitudes of the gradients exceeds a threshold value .

All definitions ignore the relationship between vector components. Since a color image represents a vector-valued function, a discontinuity of chromatic information can and should also be defined in a vector-valued way [51]. In subsection 2.4.3.1, an adaptation of the Canny algorithm to color image processing using a vector-valued technique is presented.

2.4.3.1 Canny operator using Color image processing

In 1987, Kanade introduced an extension of the Canny operator [53] for color edge detection. The operator is based on the same steps as the conventional Canny but the computations are now vector based. This means that the algorithm determines the first partial derivatives of the smoothed image in both x and y directions.

A three-component color image is represented by a function that maps a point in the image plane to a three-dimensional (3-D) vector in the color space. In the RGB space, the function is $C = (R, G, B)$. It is possible now to define the Jacobian, which is the matrix that contains the first partial derivatives for each component of the color vector:

$$\mathbf{J} = \begin{pmatrix} R_x & R_y \\ G_x & G_y \\ B_x & B_y \end{pmatrix} = (\mathrm{C}_x, \mathrm{C}_y) \tag{2.61}$$

The indexes x and y represent the partial derivatives:

$$R_x = \frac{\partial R}{\partial x} \quad \text{and} \quad R_y = \frac{\partial R}{\partial y} \tag{2.62}$$

The direction in the image along which the largest variation in the chromatic image function occurs is represented by the eigenvector of $J^T J$ corresponding to the largest eigenvalue.

$$\mathbf{J^T J} = \begin{pmatrix} J_x & J_{xy} \\ J_{yx} & J_y \end{pmatrix} \tag{2.63}$$

$$\begin{aligned} J_x &= R_x^2 + G_x^2 + B_x^2 \\ J_y &= R_y^2 + G_y^2 + B_y^2 \\ J_{xy} = J_{yx} &= R_x R_y + G_x G_y + B_x B_y \end{aligned} \tag{2.64}$$

In order to calculate the magnitude one has to compute $\det(J^T J - \lambda I) = 0$ which yields:

$$\lambda = \frac{J_y + J_x \pm \sqrt{(J_y + J_x)^2 - 4(J_x J_y - J_{xy}^2)}}{2} \tag{2.65}$$

The orientation θ of a color edge is determined in an image by:

$$\tan(2\theta) = \frac{2 \cdot C_x \cdot C_y}{\|C_x\|^2 - \|C_y\|^2} \tag{2.66}$$

After the magnitude is determined for each edge, non-maximum suppression is used, based on a threshold value in order to eliminate broad edges.

According to experiments in Carnegie Mellon University [51], the color edges describe object geometry in the scene better than the intensity edges, although over 90% of the edges are identical.

Chapter 3

Implementation

In this chapter, the implementation of the algorithms for edge detection in satellite images is presented. This work was developed in a Windows environment using MATLAB as programming workspace and Excel as data analysis assistant. The detectors implemented for grey-scale edge detection were Fractional Canny, Roberts, Sobel, LoG, CRONE and the Fractional Derivatives Mask in (2.35) . Versions of each operator using color-based image processing were also implemented. All the detectors were introduced in the same code that loads the images, performs pre and post-processing and performance analysis. MATLAB custom functions were used, as well as functions adapted from colleagues' work in medical images [2] [3].

3.1 Introduction

In order to check fractional derivatives' power in the processing of satellite images, more specifically the identification of coasts in satellite images, the conventional edge detectors were adapted to perform fractional detection. The implementation was totally performed using MATLAB in a Windows environment.

In the main script, basic sections of code were written for the natural processing of images including the loading, pre-processing, edge detection , post-processing (which includes morphological operations on the processed image) and finally performance assessment.

Two types of detectors were used:

- **Grey-scale image edge detectors**: Five fractional state-of-the-art detectors (Canny, Sobel, Roberts, CRONE and LoG) were used and one Fractional Derivative mask introduced in [46] but not used in that study as direct edge detector;

- **Color-based image edge detectors**: One already existent color-based Canny adapted to perform fractional processing, four novel color-based fractional detectors (Sobel,Roberts,Fractional Mask and CRONE) adapted with the already existent canny formulations and one novel zero-crossing color-based fractional detector (LoG).

In this study, 43 random images with low nebulosity from the Sentinel-2 satellite retrieved from ESA's "Copernicus Open Access Hub" [4] were used. The images retrieved from the website were analysed and a ground truth was manually taken using GIMP [54].

3.2 Ground Truth Definition

The HQ satellite pictures collected from ESA with a JPEG2000 format were stored and filtered to check if there were no anomalies within. In order to perform the identification of coasts in the images, a manual segmentation was made in advance. Using an open-source software called GIMP and its intelligent selection tools, it was possible to go through all images and select all coasts within them.

The process to obtain the ground truth for an image was selecting the areas of the image along the shores and then apply a threshold in which the land became white and the water black. The result was stored in the *.jpg* format in the same folder as the original images.

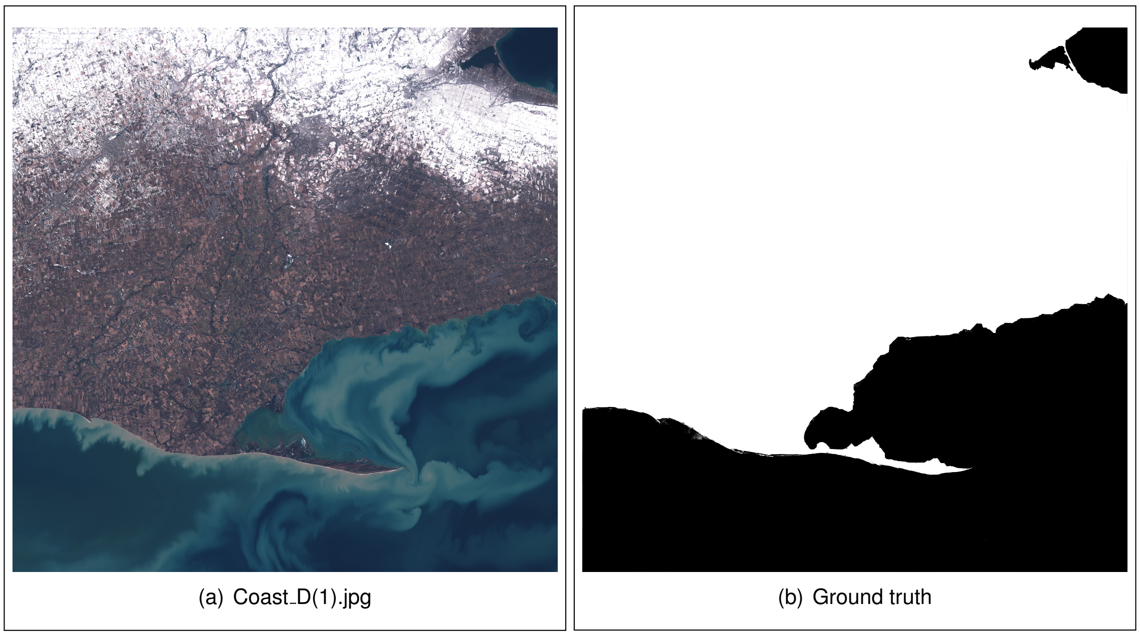
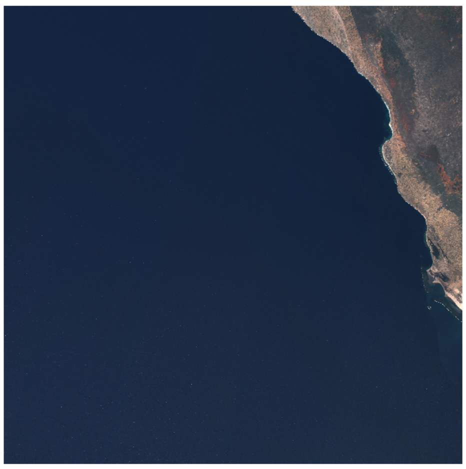

(a) Coast_D(1).jpg (b) Ground truth

Figure 3.1: Original image and Ground Truth of Coast_D(1).jpg.

3.3 Main Algorithm

In this section, the main algorithm will be presented. This code is presented in the form of a script of written code (*IdentCoastWithoutNeb.m*) and it includes the following steps:

- The image to be processed is read from the images folder;
- A pre-processing step is performed where the figure is scanned for blue color pixels. A mask is formed attributing value zero to all blue pixels found. This mask is later used in post-processing (Figure 3.2);


(a) Coast23

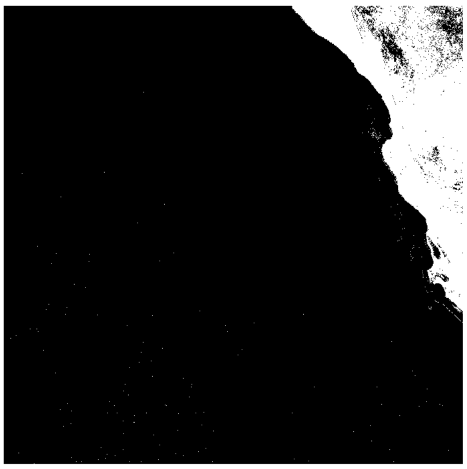
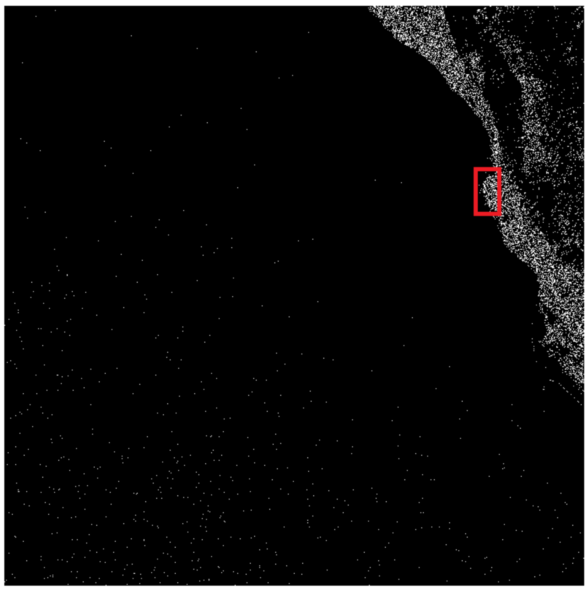

(b) Blue Mask

Figure 3.2: Original image and Blue Mask of Coast_D(23).jpg.

- The fractional edge detection operator is applied to the image. If it is a grey-scale detection, before using the operator, the RGB picture is converted to intensity (Figure 3.3);


(a) Edge detection result

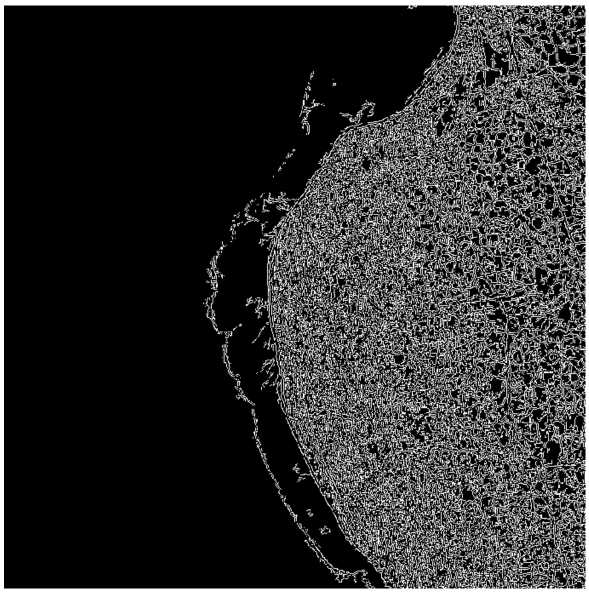

(b) Detail

Figure 3.3: Output of an edge detection operation.

- After contour detection it is necessary to close the image, so that in the end segmentation of land and water is evident. For that, several closing morphological operations are performed. Firstly, using lines in 0, 45, 90 and 135 degrees directions a closing operation is carried out in order to close pixel gaps between nearby edges that the detector did not encounter;
- Next, in order to fill the parts of land that were not yet detected by the operator and the closing operations, the custom function *imfill* is used;

- The blue mask obtained in the pre-processing is then applied to the result of the previous operations. As one can see in Figure 3.3, after the edge detection there is still a lot of noise in the water. The blue mask serves as a filter that eliminates this noise and guarantees better performance in segmentation;

- One last closing operation is performed. This is due to the fact that sometimes the color filter identifies small areas of land as blue. Using this closing operation, these small areas are set to white and are correctly identified. This step finalizes the processing part of the algorithm. An example of a processed image is shown in Figure 3.4;

- After post-processing, performance analysis is done. For this, the ground truth *.jpg* image is read and converted to a logical array. Then, by comparison between the resultant processed image and the ground truth, four metrics of performance are computed. The results of this analysis are stored in an array and are ready for analysis.

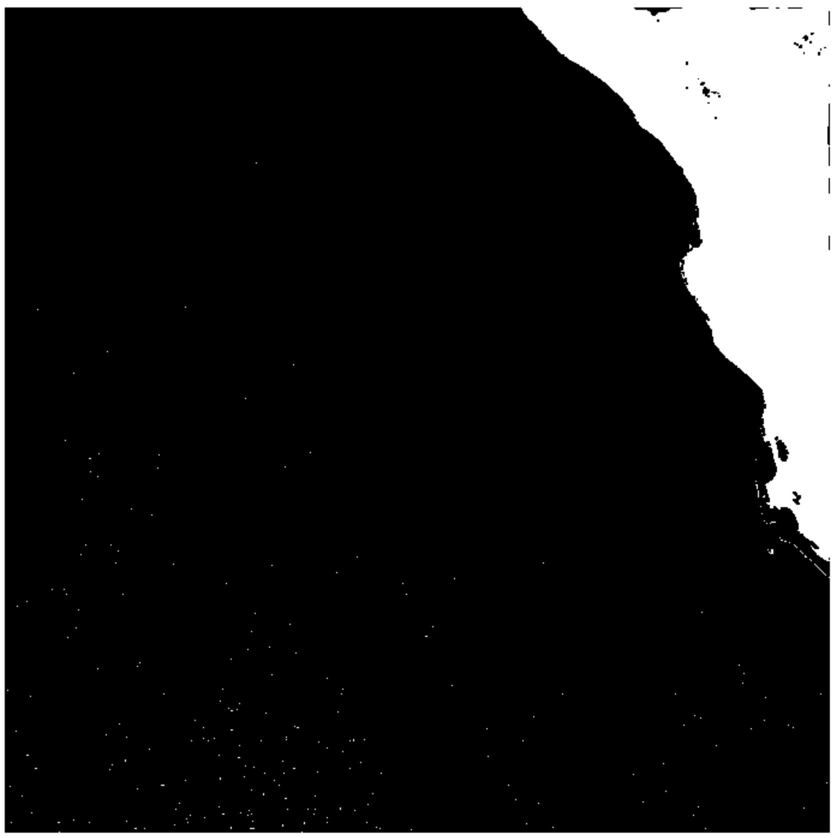


Figure 3.4: *Totally Processed Image*

3.4 Grey-Scale Detectors Adaptation

It is important now to detail the implementation of the featured grey-scale edge detectors. For the Canny, Sobel, Roberts and LoG detectors, the MATLAB custom function *edge()* was adapted. The CRONE detector was implemented adapting the CRONE function featured in [2]. The Fractional Derivative mask was implemented by substituting the Roberts core in the custom function by the desired one. The different detectors that relied on integer derivatives were substituted by the fractional versions introduced in subsection 2.4.2. The following subsections present relevant specifications on the implementation of the above mentioned detectors.

3.4.1 Fractional Canny detector

The fractional Canny operator as said above is an adaptation of the integer canny presented in subsection 2.4.1.1.4.

When adapting the *edge* function, the only change was that instead of calculating the first order gradient of the Gaussian kernel, a function was introduced that applies Grunwald-Letnikov derivatives to the Gaussian function with the desired α order. This means that for each point of the Gaussian mask its fractional derivative is obtained:

$$ {}_cD_t^\alpha f_G(t) = \lim_{h\to 0} h^{-\alpha} \sum_{k=0}^{N} (-1)^k \begin{pmatrix} \alpha \\ k \end{pmatrix} f_G(t-kh), \quad N = \left[\frac{t-c}{h}\right], \quad t > c \tag{3.1}$$

where $\begin{pmatrix} \alpha \\ k \end{pmatrix}$ was programmed as a separated function using Equation 2.6.

After computing the fractional derivative, the algorithm follows the same steps of the conventional Canny including non-maximum suppression and hysteresis thresholding.

3.4.2 Fractional Roberts, Sobel and LoG detectors

In subsections 2.4.2.1 and 2.4.2.2, the already existent formulations of the Roberts and Sobel fractional operators are presented. The masks that find the derivative of order α of a figure are also exhibited.

The original function *edge* performed this operation by using the conventional masks of these operators. Here, the only change applied to the *edge* function was the introduction of the fractional masks where the integer ones took place. The original magnitude computation and post-processing of the integer method was kept.

The LoG fractional operator presented in subsection 2.4.2.3 was implemented using the same logic. An instance where α is not integer was created and the fractional mask added to the featured function. The rest of the algorithm was maintained where edges are found searching for zero-crossings of the second order derivative obtained by convolving the image with the fractional LoG mask.

3.4.3 Fractional CRONE detector

An adaptation of the CRONE detector mathematically formulated in subsection 2.4.2.4.1 was performed using a function implemented in [2]. Some minor modifications had to be made in order to integrate the function in the main algorithm. The function that implements combinations (2.6) used in the definition of fractional derivative was substituted and some extra simplifications were performed.

Note that the Otsu method presented in subsection 2.3.6.1 is used in this function to transform the input image to binary using the normalized gradients obtained convolving the figure with the CRONE detector.

3.4.4 Fractional Derivative detector

The mask formulated in subsection 2.4.2.1 was already used in the fractional Roberts. The image was convolved with a Roberts operator and consecutively with the fractional mask in (2.35). It was implemented also this mask convolving directly the images in the database instead of using a Roberts convolution prior to the fractional derivative operation.

3.5 Color Image Edge Detection Algorithms

Since it was clear that great color variations were present in the satellite images, the good performance that color image processing techniques had been showing [51] could be useful. For that, the Canny algorithm using RGB images presented in subsection 2.4.3.1 was implemented. Moreover, color versions for the other gradient based operators were developed based on the same concepts.

The *edge* function was again the basis for this implementation. The function was modified in order to accept as input an RGB image and proceed accordingly.

In the gradient-based color algorithms, the only difference is in the computation of the gradient. Each of the color channels of the image is convolved with the respective kernel in order to obtain the derivative of each component. Then the problem reduces to find the greatest eigenvalue of the jacobian transposed times the jacobian itself. The already formulated expressions for this were copied to the function and the algorithm tested. The last steps of the algorithm are maintained as in the original one.

For the Laplacian of Gaussian color-based detector, the same algorithm could not be applied since this second order operator, in the *edge* function, detects edges based on the search for zero-crossings in the result of the convolution. In order to find edges in color images, the definition from [55] was considered in which a pixel of a colored image is considered part of an edge if zero-crossings are found in any of the color channels. The function used for the grey-scale LoG was therefore transformed into three convolutions of the LoG fractional mask with the respective color channels and search for zero-crossings in each convolution result. The pixels identified as zero-crossings are considered part of edges and the final result is a logical array in which every pixel identified as zero-crossing in at least one channel is set to one (white).

3.6 Performance Assessment

In this section the metrics that measure the performance of the segmentation algorithm will be presented. In order to check how well the algorithm can differentiate land from water, quantification has to be made. For that, the ground truth extracted as explained on section 3.2 is compared to the output of the algorithm by scanning all pixels within the figure. One of four instances may occur when pixels are compared:

- **True Positive**: When both the processed image pixel and the ground truth's are both set to white. Corresponds to a part of land correctly identified;
- **False Positive**: When the processed image pixel is set to white but in the ground truth it is set to black. Corresponds to a part of water wrongly identified by the algorithm as land;
- **True Negative**: When both the processed image pixel and the ground truth's are set to black. Corresponds to a part of water correctly identified by the algorithm as water;
- **False Negative**: When the processed image pixel is set to black but in the ground truth it is set to white. Corresponds to a part of land erroneously identified by the algorithm as water;

Table 3.1 summarizes the above presented instances. In the following subsections, the different metrics used in the algorithm for performance assessment are presented.

		Processed Image	
		0	1
Ground Truth	0	TN	FP
	1	FN	TP

Table 3.1: Performance Instances.

3.6.1 Jaccard similarity coefficient

The Jaccard similarity coefficient, introduced by Paul Jaccard in 1901 [56], is a standard used for measuring the similarity of sets. It is defined as the size of the intersection divided by the size of the union of the sample sets (3.2):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.2}$$

Being A the set of pixels in the processed image, and B the set of pixels in the ground truth, the intersection is given by the pixels that are one (white) in both sets. The union is the set of pixels that yields one at least in one set. The expression above can be substituted with the instances presented before as demonstrated in (3.3):

$$J(A,B) = \frac{TP}{FP + TP + FN} \tag{3.3}$$

This metric is very important because it will be the main performance indicator that will be used in the results analysis.

3.6.2 Dice similarity coefficient

A simple spatial overlap index is the Dice similarity coefficient (DSC), first proposed by Dice. Dice similarity coefficient is a spatial overlap index and a reproducibility validation metric. The original formula was supposed to be applied to discrete groups. Given two sets, X and Y, it is defined as:

$$DSC = \frac{2 \times |X \cap Y|}{|X| + |Y|} \tag{3.4}$$

where $|X|$ and $|Y|$ represent the number of elements in each set. So the coefficient yields twice the number of elements in common to both sets divided by the sum of the number of elements in each set. Translating this expression to Boolean data it becomes:

$$DSC = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3.5}$$

3.6.3 Sensitivity

Sensitivity is a measure that checks whether the algorithm is valid or not. In this work, it corresponds to the probability of the algorithm correctly classify a pixel as land. The formula for this index holds:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{3.6}$$

If the sensitivity is 1 it means that every pixel identified as land is for sure land.

3.6.4 Specificity

Specificity also measures the validity of the algorithm. The ability of the algorithm to correctly classify a pixel as water (black pixel) is called the algorithm's specificity. It is given by (3.7):

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{3.7}$$

If the specificity is 1 it means that every pixel identified as water is indeed water.

One should always use both sensitivity and specificity metrics in order to ensure that both water and land pixels are well identified.

Chapter 4

Results and Analysis

In this chapter, the results of the implementation of the algorithm using the different edge detectors are presented and its analysis conducted. Comparison between methods and between integer and fractional orders is also carried out. In addition, the processing of images using color detection is analysed in order to check if it is an upgrade to the conventional grey-scale edge detection. An overall analysis of the detectors performance will be performed. Finally, results that aim to understand if it is possible to find optimal parameters that allow automatic detection will be presented. All these steps are filled in the following sections with the aim of understanding the impact of using fractional derivatives in the contour detection in satellite images.

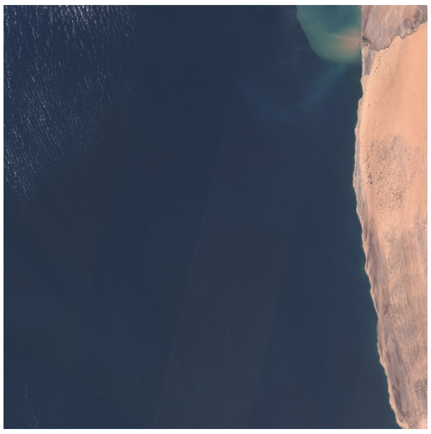
4.1 Image Database

In this study, 43 HQ satellite JPEG 2000 images were processed. These images were collected from ESA's Copernicus Open Access Hub [4]. All these images present coasts where land and water are well defined areas and there is very low nebulosity.

4.2 Performance Analysis of Fractional vs Integer Edge Detection

The first step of performance analysis is to understand in each edge detection algorithm, if a fractional order based adaptation thereof outperforms the conventional integer version. Therefore, all the algorithms formulated in chapter 3 were tested for a set of parameters and the results will be presented in the following subsections. The performance was evaluated for all 43 images in the database using the metrics introduced in section 3.6. For space reasons, from the forty three figures, four were chosen to carry out individual performance assessment in this document. The selected figures were number 12, 18, 27 and 38 presented with their corresponding ground truth respectively in Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4. These figures were selected because they present different levels of performance when processed with the algorithm.

Appendix A presents plots of the Jaccard coefficient with varying parameters for the selected images. The extended results of all methods for all images are available online in [57]


(a) Coast_D(12)

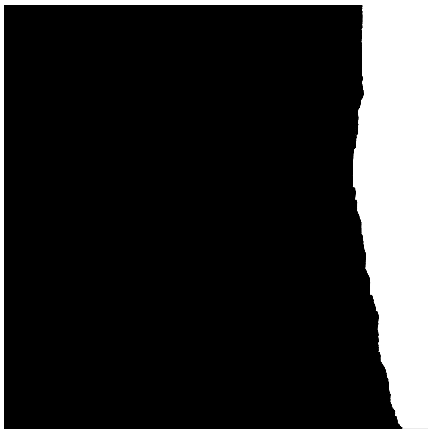
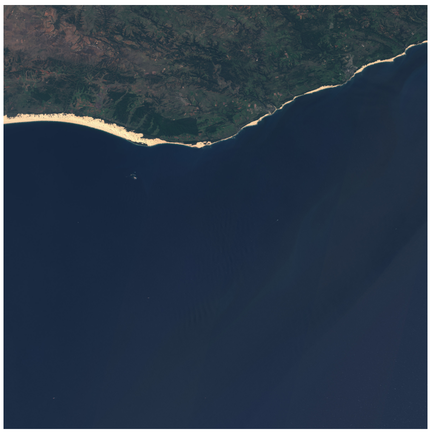

(b) Ground Truth defined for Coast_D(12)

Figure 4.1: Figure number 12 to be evaluated with corresponding ground truth.


(a) Coast_D(18)

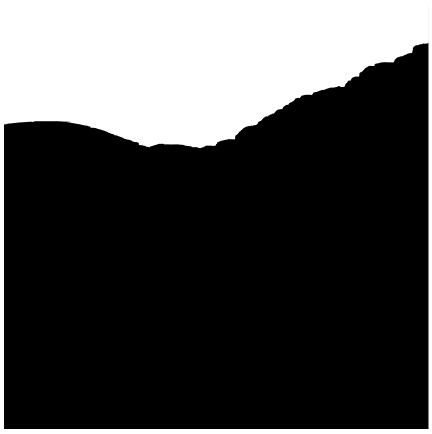
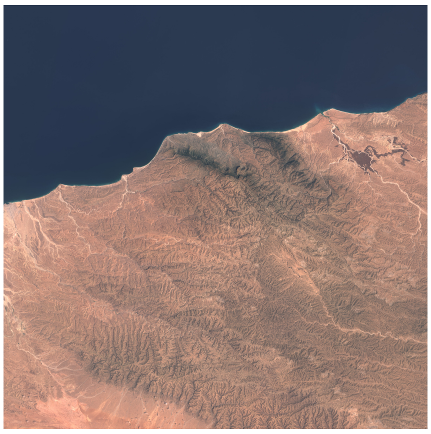

(b) Ground Truth defined for Coast_D(18)

Figure 4.2: Figure number 18 to be evaluated with corresponding ground truth.


(a) Coast_D(27)

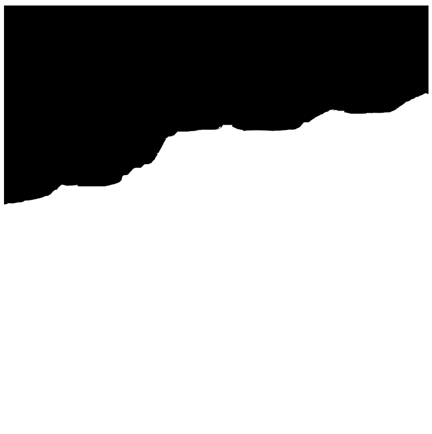

(b) Ground Truth defined for Coast_D(27)

Figure 4.3: Figure number 27 to be evaluated with corresponding ground truth.

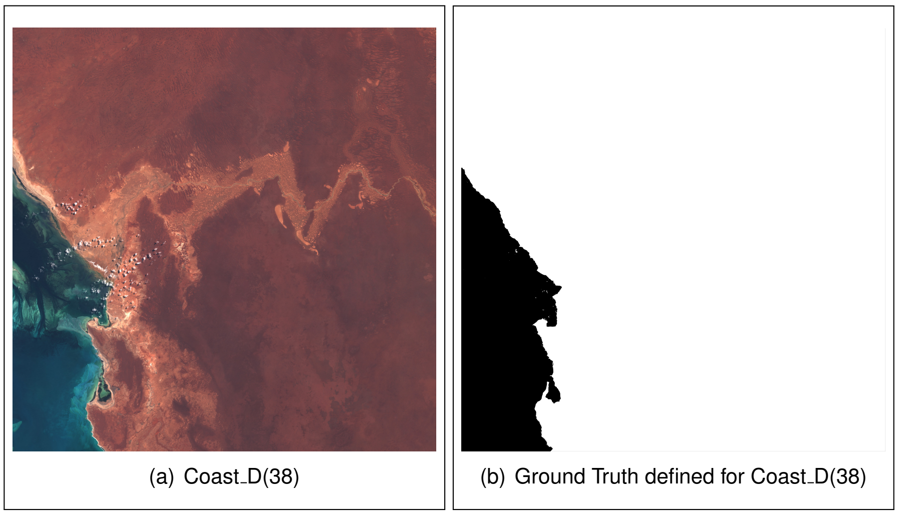

(a) Coast_D(38) (b) Ground Truth defined for Coast_D(38)

Figure 4.4: Figure number 38 to be evaluated with corresponding ground truth.

4.2.1 Canny vs Fractional Canny

The parameters that may influence the performance in the Canny algorithm are the order of the derivative of the gaussian function (α) and its standard deviation (σ).

Alpha was ranged initially between -2 and 2 because the literature states that values outside this interval may lead to unstable results. However, after a first analysis it was checked that high performance was obtained near the upper limit order. Thus, the range was extended to -2 up to 3 with a step of 0.1. The sigma values were tested in the interval from 0.2 until 2 with a step of 0.2.

The best performances for the selected figures are presented in Table 4.1, The first four lines for each image correspond to the result which presented the highest value for each performance metric. The fifth line represents the best overall result, corresponding to the processing parameters that achieved the highest mean regarding the four performance metrics. The last line corresponds to the same as in the fifth but for the conventional integer algorithm using the original *edge* function.

From Figure 4.5 and Table 4.1 it is possible to draw some conclusions. Fractional Canny shows major immunity to the high level of noise in the water (mainly in the top left corner) which allows it to present higher specificity. However, the fractional algorithm shows some difficulties in closing the coast, in contrast with the integer one. Nevertheless, the fractional algorithm showed slightly better performance.

In the 18^{th} image of the database, the global performance of the two types of Canny is approximately the same. The Sensitivity is lower and the Specificity higher in the fractional algorithm such as in the previous selected image. The performance here is low due to the blueish tones (check Figure 4.2) in the coast that are set to black by the color filter in the post-processing.

Both algorithms present exceptional performance in the identification of the coast in figure 27. The fractional one presents slightly higher overall measures.

It is clear from Table 4.1 and Figure 4.8 that the fractional detector improved the detection in image 38. The Jaccard, Dice and sensitivity measures increased significantly. Smooth areas of land in this image make it hard for the detectors to find edges in these areas. Fractional Canny adaptation improves the identification but it is still not optimal. It still identifies areas of water as land and the opposite.

Image #		σ	α	J	D	Sensitivity	Specificity
12	Best J	1.4	2.2	**0.8885**	0.9410	0.9945	0.9804
	Best D	1.4	2.2	0.8885	**0.9410**	0.9945	0.9804
	Best Sens	0.8	1.2	0.6116	0.7590	**0.9996**	0.8956
	Best Spec	1.8	2.3	0.8791	0.9357	0.9705	**0.9829**
	Best Overall	**1.4**	**2.2**	0.8885	0.9410	0.9945	0.9804
	Best Integer	**2.0**	**-**	0.8685	0.9296	0.9960	0.9758
18	Best J	1.8	-0.1	**0.6600**	0.7952	0.6741	0.9928
	Best D	1.8	-0.1	0.6600	**0.7952**	0.6741	0.9928
	Best Sens	0.6	1.2	0.6592	0.7946	**0.6772**	0.9909
	Best Spec	2.0	-1.2	0.6597	0.7949	0.6725	**0.9935**
	Best Overall	**1.8**	**-0.1**	0.6600	0.7952	0.6741	0.9928
	Best Integer	**1.6**	**-**	0.6598	0.7950	0.6767	0.9914
27	Best J	0.6	2.6	**0.9956**	0.9978	0.9984	0.9943
	Best D	0.6	2.6	0.9956	**0.9978**	0.9984	0.9943
	Best Sens	1.4	2.0	0.9954	0.9977	**0.9984**	0.9937
	Best Spec	1.0	2.4	0.9955	0.9978	0.9981	**0.9946**
	Best Overall	**0.6**	**2.6**	0.9956	0.9978	0.9984	0.9943
	Best Integer	**0.8**	**-**	0.9952	0.9976	0.9986	0.9932
38	Best J	0.6	2.4	**0.7356**	0.8477	0.7600	0.7359
	Best D	0.6	2.4	0.7356	**0.8477**	0.7600	0.7359
	Best Sens	0.6	2.4	0.7356	0.8477	**0.7600**	0.7359
	Best Spec	1.4	2.6	0.2826	0.4406	0.2904	**0.7780**
	Best Overall	**0.6**	**2.4**	0.7356	0.8477	0.7600	0.7359
	Best Integer	**0.2**	**-**	0.5180	0.6825	0.5365	0.7167

Table 4.1: Table with best performance results using Canny edge detectors on selected images

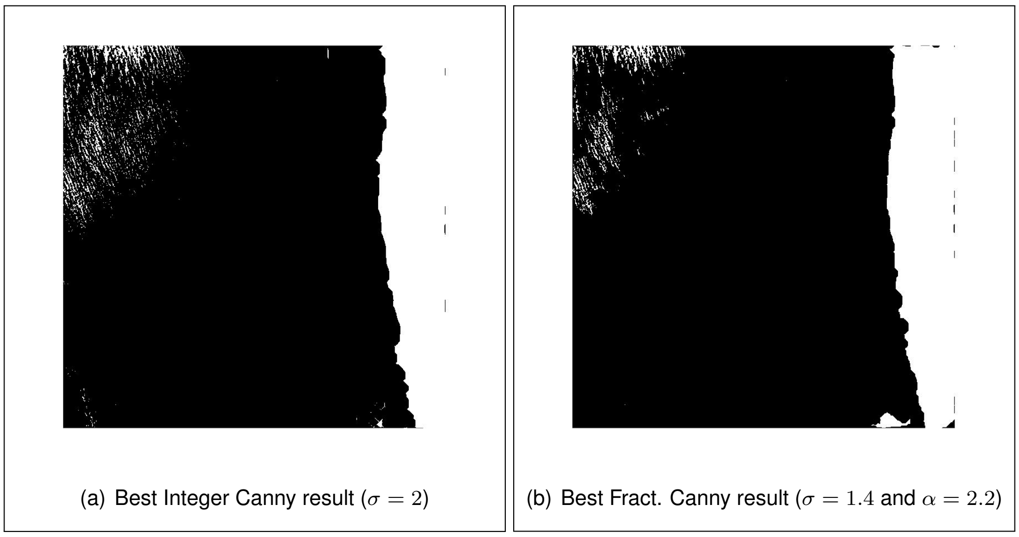
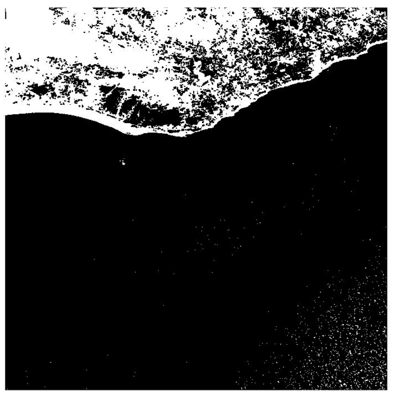

(a) Best Integer Canny result ($\sigma = 2$)
(b) Best Fract. Canny result ($\sigma = 1.4$ and $\alpha = 2.2$)

Figure 4.5: Best results for Figure 12 processed using Canny algorithms.


(a) Best Integer Canny result ($\sigma = 1.6$)

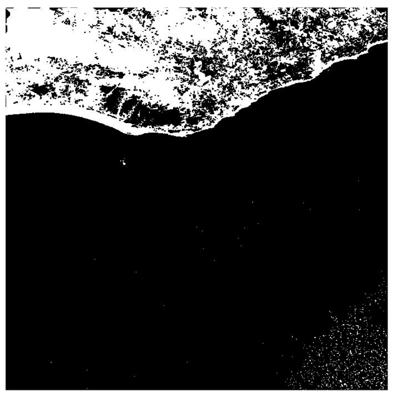

(b) Best Fract. Canny result ($\sigma = 1.8$ and $\alpha = -0.1$)

Figure 4.6: Best results for Figure 18 processed using Canny algorithms.

(a) Best Integer Canny result ($\sigma = 0.8$)

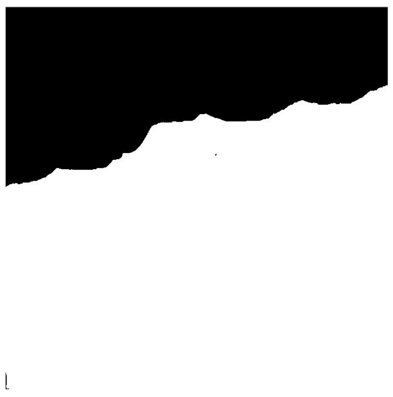
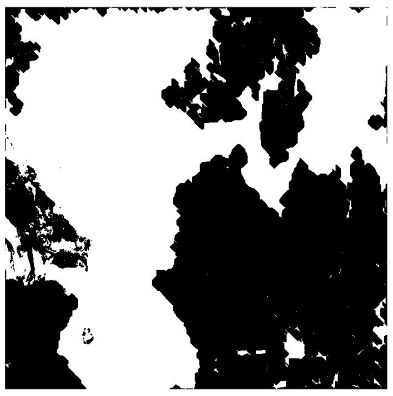

(b) Best Fract. Canny result ($\sigma = 0.6$ and $\alpha = 2.6$)

Figure 4.7: Best results for Figure 27 processed using Canny algorithms.


(a) Best Integer Canny result ($\sigma = 0.2$)

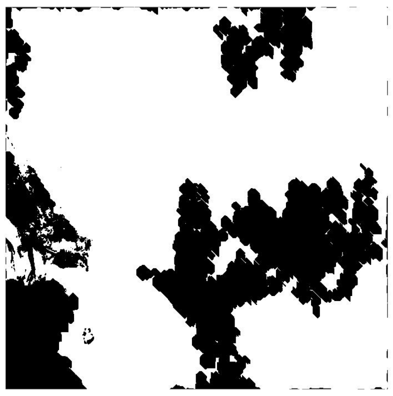

(b) Best Fract. Canny result ($\sigma = 0.6$ and $\alpha = 2.4$)

Figure 4.8: Best results for Figure 38 processed using Canny algorithms.

4.2.2 Sobel vs Fractional Sobel

The Sobel Fractional mask formulated in subsection 2.4.2.2 does not depend only on the order of the derivative (α) that one wants to apply. The threshold used to select the gradients that present edges also influences the performance of this method.

The value of alpha was varied between -2 and 3 with a step of 0.1. The threshold values were tested in the interval from 0.1 until 0.9 with a step of 0.2.

The best performances for the selected figures are shown in Table 4.2.

Image #		threshold	α	J	D	Sensitivity	Specificity
12	Best J	0.9	0.7	**0.5620**	0.7196	0.9999	0.8717
	Best D	0.9	0.7	0.5620	**0.7196**	0.9999	0.8717
	Best Sens	0.7	3.0	0.5618	0.7194	**1.0000**	0.8716
	Best Spec	0.3	1.0	0.0004	0.0008	0.0004	**0.9999**
	Best Overall	**0.9**	**0.7**	0.5620	0.7196	0.9999	0.8717
	Best Integer	**0.1**	**-**	0.5043	0.6705	0.5104	0.9758
18	Best J	0.9	0.6	**0.6584**	0.7940	0.6774	0.9903
	Best D	0.9	0.6	0.6584	**0.7940**	0.6774	0.9903
	Best Sens	0.9	0.6	0.6584	0.7940	**0.6774**	0.9903
	Best Spec	0.9	0.6	0.6584	0.7940	0.6774	**0.9903**
	Best Overall	**0.9**	**0.6**	0.6600	0.7952	0.6741	0.9928
	Best Integer	**0.1**	**-**	0.1419	0.2486	0.1445	0.9940
27	Best J	0.9	1.4	**0.9948**	0.9974	0.9987	0.9919
	Best D	0.9	1.4	0.9948	**0.9974**	0.9987	0.9919
	Best Sens	0.1	-2.0	0.9936	0.9968	**0.9989**	0.9891
	Best Spec	0.9	1.4	0.9948	0.9974	0.9987	**0.9919**
	Best Overall	**0.9**	**1.4**	0.9948	0.9974	0.9987	0.9919
	Best Integer	**0.1**	**-**	0.8435	0.9151	0.8440	0.9988
38	Best J	0.9	0.5	**0.9600**	0.9796	0.9997	0.6710
	Best D	0.9	0.5	0.9600	**0.9796**	0.9997	0.6710
	Best Sens	0.1	-2.0	0.9582	0.9786	**0.9999**	0.6529
	Best Spec	0.9	0.5	0.9600	0.9796	0.9997	**0.6710**
	Best Overall	**0.9**	**0.5**	0.9600	0.9796	0.9997	0.6710
	Best Integer	**0.1**	**-**	0.0768	0.1427	0.0775	0.9271

Table 4.2: Table with best performance results using Sobel edge detectors on selected images

From Table 4.2 and Figure 4.9 it is possible to conclude that the methods present very different results for figure 12. Fractional Sobel could not deal with the great noise in the top left corner. It is also clear that the coast line was not well defined by this operator. The integer algorithm shows some difficulties in closing the coast in spite of the fact that it eliminates the high level of noise. None of these two presents a decent solution for the problem at hand.

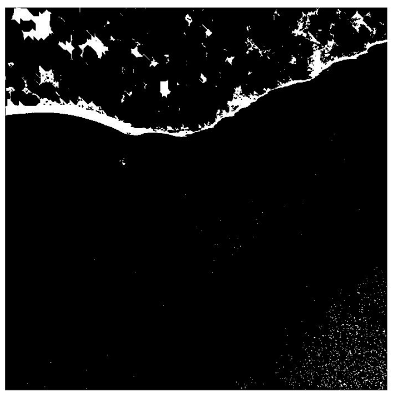
The fractional algorithm clearly outperforms the integer method for image 18 (Figure 4.10) since it is more sensitive to gradients. The noise in the bottom right corner is not mitigated by any of the two processing algorithms.

As one can observe in Figure 4.11, the fractional algorithm can successfully close the contours of the coast that the integer algorithm struggles to identify. This is confirmed by the performance metrics in Table 4.2.

(a) Best Integer Sobel result ($th = 0.1$)

(b) Best Fract. Sobel result ($th = 0.9$ and $\alpha = 0.7$)

Figure 4.9: Best results for Figure 12 processed using Sobel algorithms.


(a) Best Integer Sobel result ($th = 0.1$)

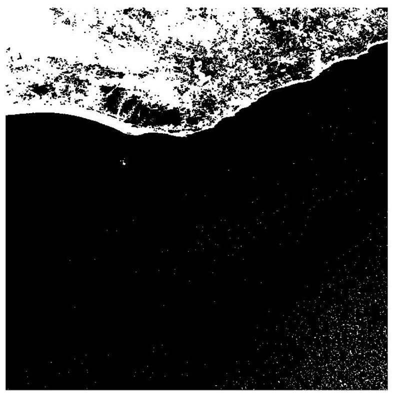
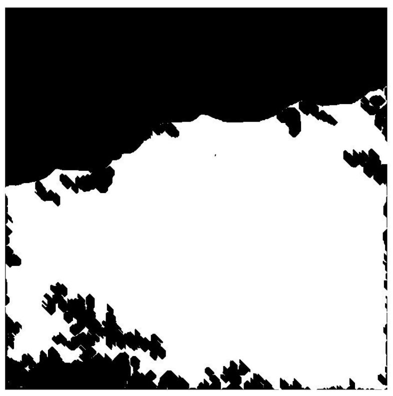

(b) Best Fract. Sobel result ($th = 0.9$ and $\alpha = 0.6$)

Figure 4.10: Best results for Figure 18 processed using Sobel algorithms.


(a) Best Integer Sobel result ($th = 0.1$)

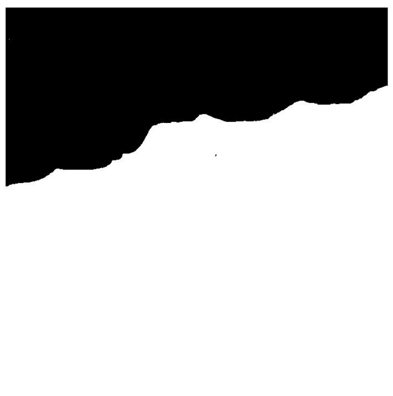

(b) Best Fract. Sobel result ($th = 0.9$ and $\alpha = 1.4$)

Figure 4.11: Best results for Figure 27 processed using Sobel algorithms.

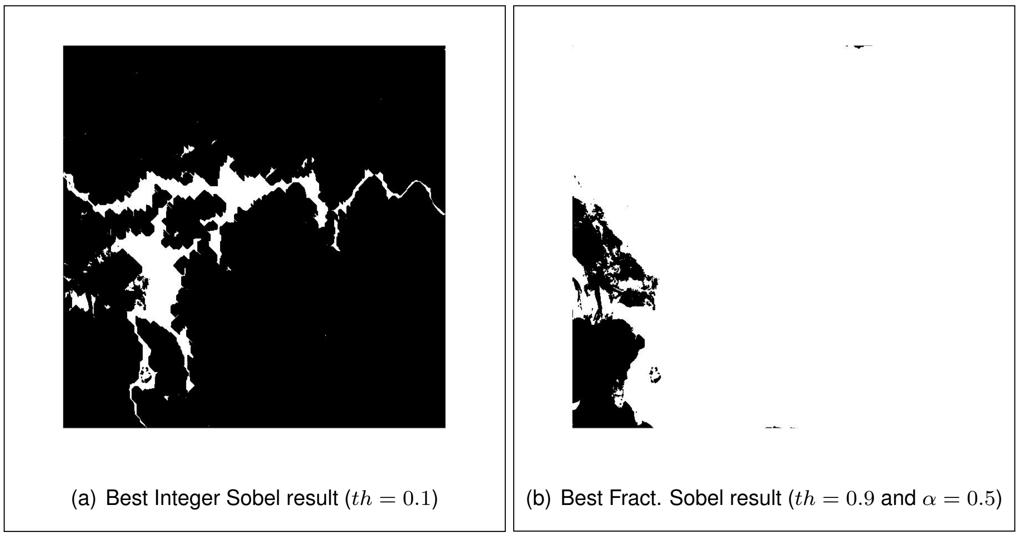

(a) Best Integer Sobel result ($th = 0.1$) (b) Best Fract. Sobel result ($th = 0.9$ and $\alpha = 0.5$)

Figure 4.12: Best results for Figure 38 processed using Sobel algorithms.

In figure 38, the integer algorithm does not have the sensitivity once more to identify all the gradients that constitute land. The fractional algorithm presents very good performance except for Specificity (Table 4.2). This low value is explained by the fact that the algorithm identifies a different tone of blue in the water as land.

4.2.3 Roberts vs Fractional Roberts

The Roberts Fractional mask formulated in subsection 2.4.2.1 works as the Sobel one. It depends on the order of the derivative (α) that one wants to use and the threshold used to select the strongest gradients.

The values of α and th were tested with the same values used before.

It is clear that the fractional Roberts algorithm enhances the performance of edge detection in figure 12. Integer Roberts has difficulties in closing the land due to lack of sensitivity. The fractional method increases sensitivity without compromising specificity (Figure 4.13).

It can be seen in Table 4.3 and Figure 4.14 that the fractional processing greatly improves the performance of the edge detection algorithm. The higher Jaccard coefficient improved 50%, the Dice coefficient 52 % and the sensitivity 51% without a great decrease in specificity. Note that the threshold parameter for the best performance is the same so only order is affecting the result. It can be also concluded that in this image, the algorithm finds it difficult once more to detect inner land gradients which is making it impossible to close the contours. The level of noise in the right bottom corner is the same in both integer and fractional methods. The global performance again does not climb over the 70% mark because of the blueish tone of the coast in some areas.

Regarding picture 27 (Figure 4.15), the fractional Roberts operator allows the algorithm to close the contours almost perfectly while the integer one struggles to do it. This presents again evidence that the fractional Roberts achieves greater sensitivity.

The integer algorithm struggles once more to find inner edges in figure 38 and the fractional continued to identify regions of water as land (Figure 4.16).

Image #		threshold	α	J	D	Sensitivity	Specificity
12	Best J	0.1	0.8	**0.9245**	0.9608	0.9595	0.9938
	Best D	0.1	0.8	0.9245	**0.9608**	0.9595	0.9938
	Best Sens	0.1	-1.9	0.5622	0.7197	**1.0000**	0.8718
	Best Spec	0.7	1.5	0.0002	0.0003	0.0002	**1.0000**
	Best Overall	**0.1**	**0.8**	0.9245	0.9608	0.9595	0.9938
	Best Integer	**0.1**	**-**	0.4440	0.6149	0.4486	0.9983
18	Best J	0.1	-0.3	**0.6588**	0.7943	0.6771	0.9907
	Best D	0.1	-0.3	0.6588	**0.7943**	0.6771	0.9907
	Best Sens	0.1	-1.9	0.6584	0.7940	**0.6774**	0.9903
	Best Spec	0.9	-0.3	0.0008	0.0015	0.0008	**1.0000**
	Best Overall	**0.1**	**-0.3**	0.6588	0.7943	0.6771	0.9907
	Best Integer	**0.1**	**-**	0.1589	0.2742	0.1619	0.9934
27	Best J	0.1	2.4	**0.9952**	0.9976	0.9981	0.9940
	Best D	0.1	2.4	0.9952	**0.9976**	0.9981	0.9940
	Best Sens	0.1	-1.9	0.9948	0.9974	**0.9986**	0.9920
	Best Spec	0.9	-1.6	0.0078	0.0155	0.0078	**1.0000**
	Best Overall	**0.1**	**2.4**	0.9952	0.9976	0.9981	0.9940
	Best Integer	**0.1**	**-**	0.8614	0.9255	0.8618	0.9990
38	Best J	0.1	-1.7	**0.9613**	0.9803	0.9990	0.6877
	Best D	0.1	-1.7	0.9613	**0.9803**	0.9990	0.6877
	Best Sens	0.1	-1.9	0.9610	0.9801	**0.9993**	0.6827
	Best Spec	0.9	2.8	0.0001	0.0002	0.0001	**1.0000**
	Best Overall	**0.1**	**-1.7**	0.9613	0.9803	0.9990	0.6877
	Best Integer	**0.1**	**-**	0.0618	0.1163	0.0622	0.9451

Table 4.3: Table with best performance results using Roberts edge detectors on selected images

(a) Best Integer Roberts result ($th = 0.1$)

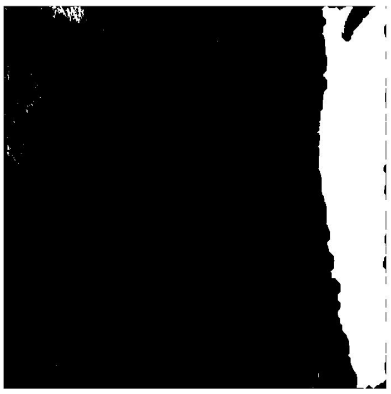
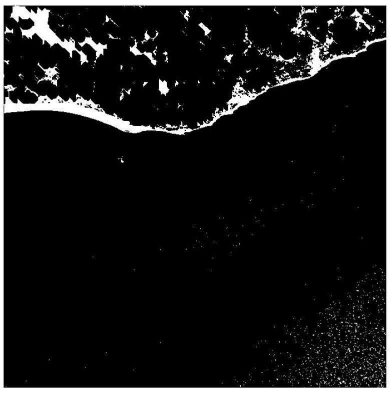

(b) Best Fract. Roberts result ($th = 0.1$ & $\alpha = 0.8$)

Figure 4.13: Best results for Figure 12 processed using Roberts algorithms.


(a) Best Integer Roberts result ($th = 0.1$)

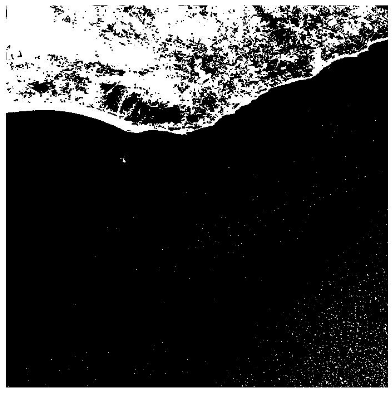
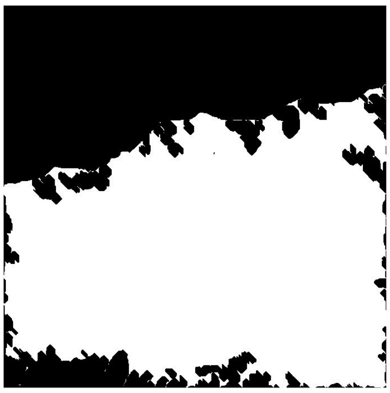
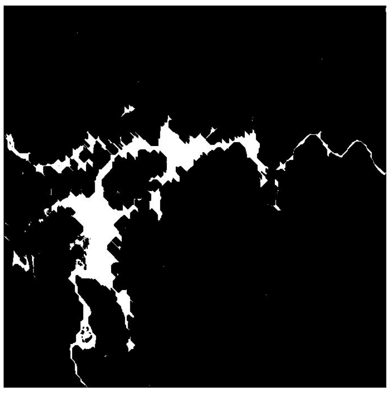

(b) Best Fract. Roberts result ($th = 0.1$ & $\alpha = -0.3$)

Figure 4.14: Best results for Figure 18 processed using Roberts algorithms.


(a) Best Integer Roberts result ($th = 0.1$)

(b) Best Fract. Roberts result ($th = 0.1$ & $\alpha = 2.4$)

Figure 4.15: Best results for Figure 27 processed using Roberts algorithms.


(a) Best Integer Roberts result ($th = 0.1$)

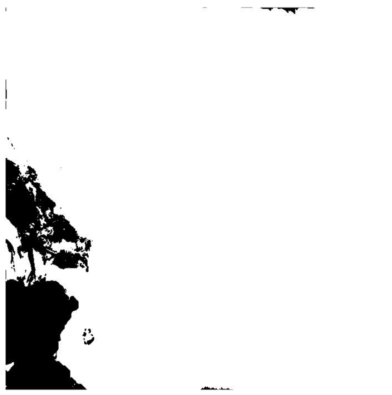

(b) Best Fract. Roberts result ($th = 0.1$ & $\alpha = -1.7$)

Figure 4.16: Best results for Figure 38 processed using Roberts algorithms.

4.2.4 LoG vs Fractional LoG

The Laplacian of Gaussian mask formulated in subsection 2.4.2.3 was also tested. It depends on the order (α) and the threshold used to select the strongest gradients.

The same values of α as in the Roberts and Sobel operators were tested. For the integer LoG the values from 0.1 to 0.9 did not work for any image so a th between 0.01 and 0.09 was adopted with a step of 0.01. The best results for the selected images are shown in Table 4.4:

Image #		threshold	α	J	D	Sensitivity	Specificity
12	Best J	0.3	-2.0	**0.9480**	0.9733	0.9718	0.9959
	Best D	0.3	-2.0	0.9480	**0.9733**	0.9718	0.9959
	Best Sens	0.1	3.0	0.5629	0.7203	**0.9999**	0.8722
	Best Spec	0.9	-1.1	0.0001	0.0002	0.0001	**1.0000**
	Best Overall	**0.3**	**-2.0**	0.9480	0.9733	0.9718	0.9959
	Best Integer	**0.01**	**-**	0.5737	0.7291	0.5820	0.9976
18	Best J	0.3	3.0	**0.6594**	0.7948	0.6771	0.9910
	Best D	0.3	3.0	0.6594	**0.7948**	0.6771	0.9910
	Best Sens	0.1	2.8	0.6591	0.7945	**0.6774**	0.9907
	Best Spec	0.9	-1.0	0.0010	0.0020	0.0010	**1.0000**
	Best Overall	**0.3**	**3.0**	0.6594	0.7948	0.6771	0.9910
	Best Integer	**0.01**	**-**	0.1051	0.1902	0.1065	0.9957
27	Best J	0.1	-1.2	**0.9957**	0.9978	0.9982	0.9950
	Best D	0.1	-1.2	0.9957	**0.9978**	0.9982	0.9950
	Best Sens	0.1	3.0	0.9947	0.9974	**0.9987**	0.9919
	Best Spec	0.9	-1.8	0.0274	0.0533	0.0274	**1.0000**
	Best Overall	**0.1**	**-1.2**	0.9957	0.9978	0.9982	0.9950
	Best Integer	**0.01**	**-**	0.8321	0.9084	0.8326	0.9988
38	Best J	0.1	1.8	**0.9637**	0.9815	0.9994	0.7046
	Best D	0.1	1.8	0.9637	**0.9815**	0.9994	0.7046
	Best Sens	0.1	3.0	0.9585	0.9788	**0.9999**	0.6562
	Best Spec	0.5	-0.7	0.0002	0.0004	0.0002	**1.0000**
	Best Overall	**0.1**	**1.8**	0.9637	0.9815	0.9994	0.7046
	Best Integer	0.01	**-**	0.0887	0.1630	0.0898	0.9076

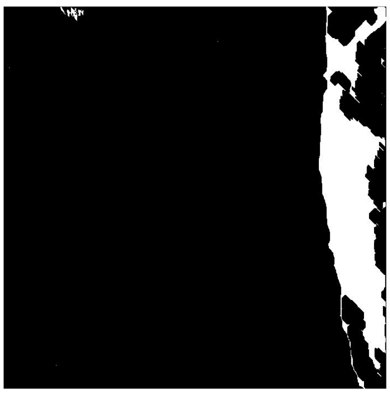
Table 4.4: Table with best performance results using LoG edge detectors on selected images

In figure 12, the fractional method presented better performance. One may note that despite the method being fractional, the best order was integer (-2). Nevertheless, it can be concluded that the fractional mask obtained using the GL definition of fractional derivative presents best sensitivity.

The fractional algorithm continues to perform better in the sensitivity of inner land gradients in picture 18. However, the best order was again an integer one (3) and the noise in the right bottom corner is more pronounced when the fractional LoG mask was used (Figure 4.18).

The integer processing of image 27 presents, once more, acceptable results. However, the fractional algorithm with an α of -1.2 and a th of 0.1 shows a major improvement of 16%. This can be seen in Figure 4.19 where the integer algorithm was not able to close the contour of the coast.

The fractional algorithm closes almost perfectly the coast and presents a Jaccard coefficient of 99.57%. Note that until now, this is the first selected figure in which the LoG fractional algorithm presents better performance with non integer order.


(a) Best Integer LoG result ($th = 0.01$)

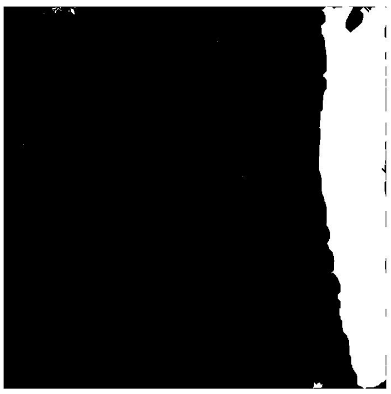
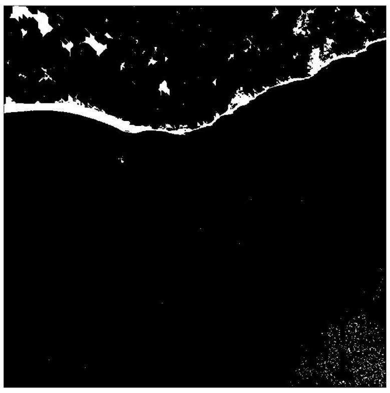

(b) Best Fract. LoG result ($th = 0.3$ and $\alpha = -2$)

Figure 4.17: Best results for Figure 12 processed using LoG algorithms.


(a) Best Integer LoG result ($th = 0.01$)

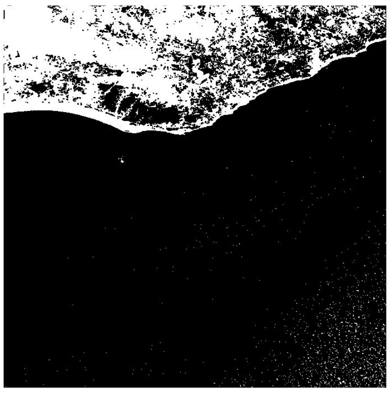
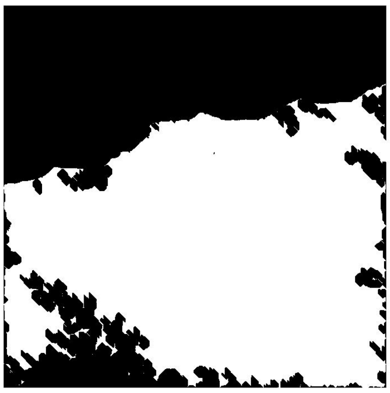

(b) Best Fract. LoG result ($th = 0.3$ and $\alpha = 3$)

Figure 4.18: Best results for Figure 18 processed using LoG algorithms.


(a) Best Integer LoG result ($th = 0.01$)

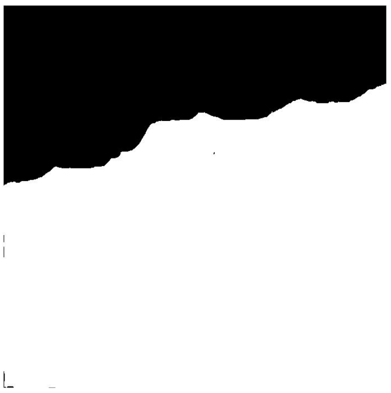

(b) Best Fract. LoG result ($th = 0.1$ and $\alpha = -1.2$)

Figure 4.19: Best results for Figure 27 processed using LoG algorithms.

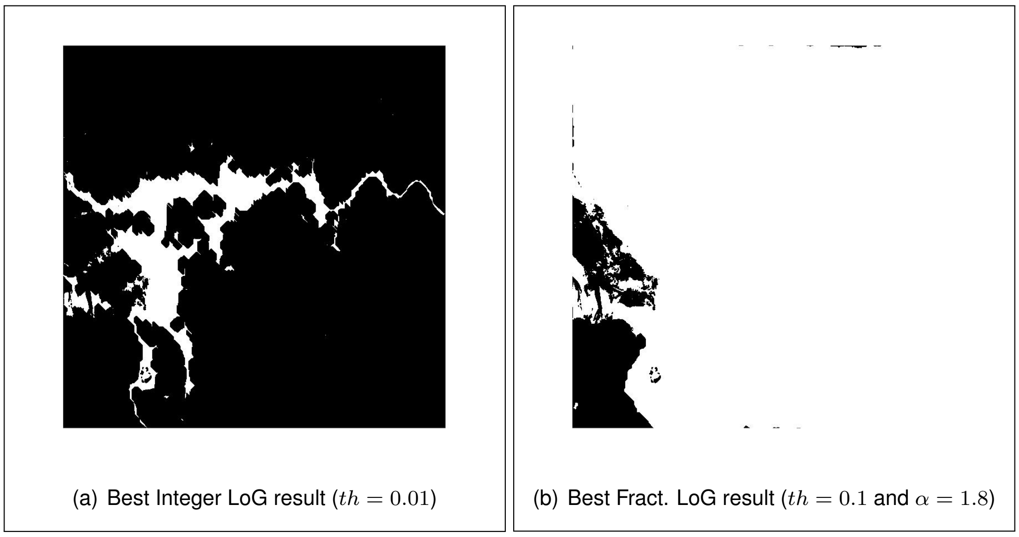

(a) Best Integer LoG result ($th = 0.01$) (b) Best Fract. LoG result ($th = 0.1$ and $\alpha = 1.8$)

Figure 4.20: Best results for Figure 38 processed using LoG algorithms.

As observed for previous detectors, the integer LoG algorithm does not have enough sensitivity to detect smooth gradients that figure 38 presents(Figure 4.20). However, the fractional algorithm reached an impressive Jaccard coefficient of 96.37% (Table 4.4). The fractional algorithm due to its high sensitivity is unable to present immunity to clear tones in the water so the specificity decreases.

4.3 Performance Analysis of other Non-integer Algorithms

Besides the algorithms already analysed above, a CRONE and a Fractional Derivative mask were implemented. These detectors do not present integer alternative, thus it is impossible to perform any performance comparison. Nevertheless, in this section, the results of these algorithms' testing will be displayed. The same figures were selected to perform the analysis.

4.3.1 CRONE Detector Performance Analysis

The CRONE mask (subsection 2.4.2.4.1) depends on the order (α) and size (k).

The same values of α were tested. The size of the Crone Mask was iterated between 1 and 5 pixels. The best testing results are presented in Table 4.5 and figures 4.21, 4.22, 4.23 and 4.24.

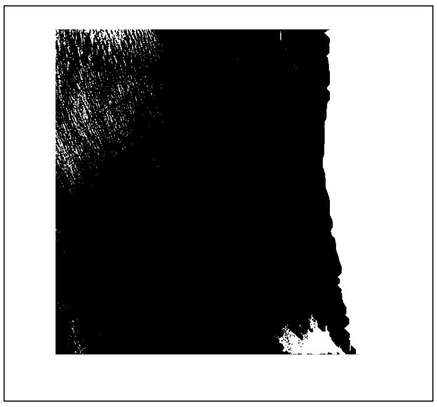


Figure 4.21: Best result for Figure 12 processed using CRONE algorithm ($k = 5$ and $\alpha = -1.2$).

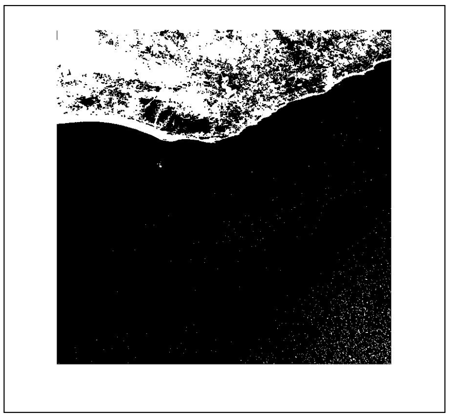


Figure 4.22: Best result for Figure 18 processed using CRONE algorithm ($k = 1$ and $\alpha = 1.1$).

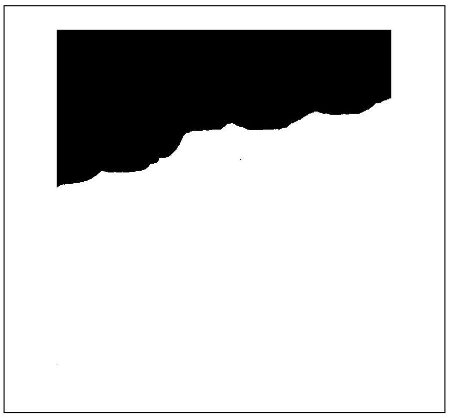


Figure 4.23: Best result for Figure 27 processed using CRONE algorithm ($k = 5$ and $\alpha = 3$).

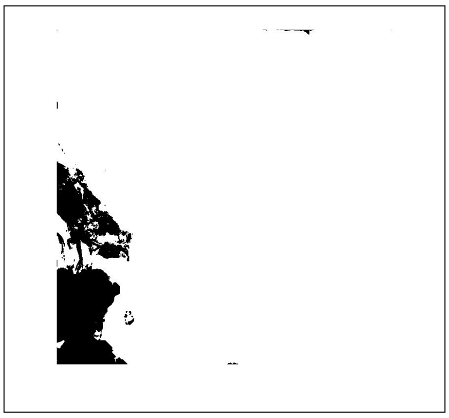


Figure 4.24: Best result for Figure 38 processed using CRONE algorithm ($k = 2$ and $\alpha = 3$).

Image #		k	α	J	D	Sensitivity	Specificity
12	Best J	5.0	-1.2	**0.8138**	0.8974	0.9977	0.9628
	Best D	5.0	-1.2	0.8138	**0.8974**	0.9977	0.9628
	Best Sens	2.0	2.7	0.5619	0.7195	**1.0000**	0.8716
	Best Spec	5.0	-1.2	0.8138	0.8974	0.9977	**0.9628**
	Best Overall	**5.0**	**-1.2**	0.8138	0.8974	0.9977	0.9628
18	Best J	1.0	1.1	**0.6585**	0.7941	0.6773	0.9905
	Best D	1.0	1.1	0.6585	**0.7941**	0.6773	0.9905
	Best Sens	5.0	1.6	0.6583	0.7939	**0.6774**	0.9903
	Best Spec	5.0	-1.6	0.1143	0.2051	0.1150	**0.9979**
	Best Overall	**1.0**	**1.1**	0.6585	0.7941	0.6773	0.9905
27	Best J	5.0	3.0	**0.9951**	0.9975	0.9985	0.9930
	Best D	5.0	3.0	0.9951	**0.9975**	0.9985	0.9930
	Best Sens	2.0	0.7	0.9948	0.9974	**0.9987**	0.9921
	Best Spec	5.0	3.0	0.9951	0.9975	0.9985	**0.9930**
	Best Overall	**5.0**	**3.0**	0.9951	0.9975	0.9985	0.9930
38	Best J	2.0	3.0	**0.9612**	0.9802	0.9995	0.6818
	Best D	2.0	3.0	0.9612	**0.9802**	0.9995	0.6818
	Best Sens	2.0	0.7	0.9588	0.9790	**0.9999**	0.6582
	Best Spec	5.0	3.0	0.3704	0.5405	0.3838	**0.7120**
	Best Overall	**2.0**	**3.0**	0.9612	0.9802	0.9995	0.6818

Table 4.5: Table with best performance results using CRONE edge detectors on selected images

4.3.2 Fractional Mask Performance Analysis

It is interesting now to evaluate the processing of the images in the database convolving them directly with the fractional derivative mask in Equation 2.35. Like the Roberts detector, the parameters here are the threshold th and the order α. The same values of α and th were used in the testing.

The following pages contain the results of the experimentation performed in the selected images.

From the results it is clear that Figure 12 presented great performance using this mask (Table 4.6). The top left corner high level of noise was almost completely vanished and the land perfectly identified (Figure 4.25).

All the other selected figures presented results similar to other detectors' results.

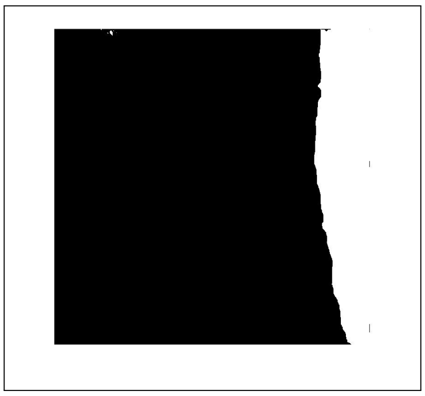


Figure 4.25: Best result for Figure 12 processed using a Fractional Derivative Mask ($th = 0.9$ and $\alpha = 0.8$).

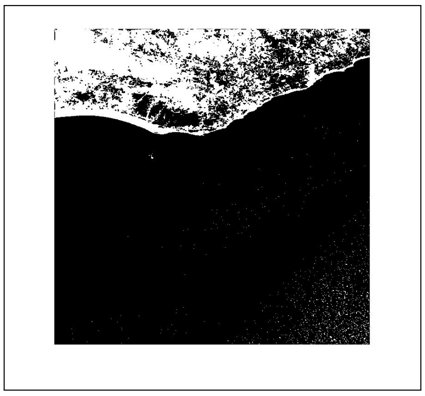


Figure 4.26: Best result for Figure 18 processed using a Fractional Derivative Mask ($th = 0.5$ and $\alpha = 1.7$).

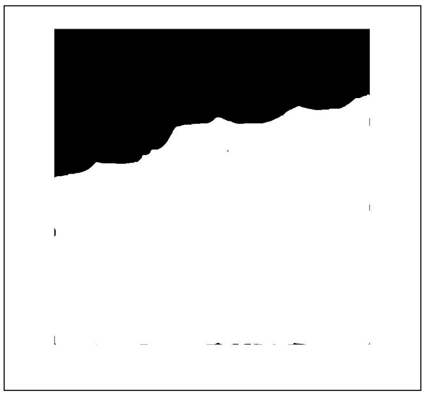


Figure 4.27: Best result for Figure 27 processed using a Fractional Derivative Mask ($th = 0.9$ and $\alpha = 0.8$).

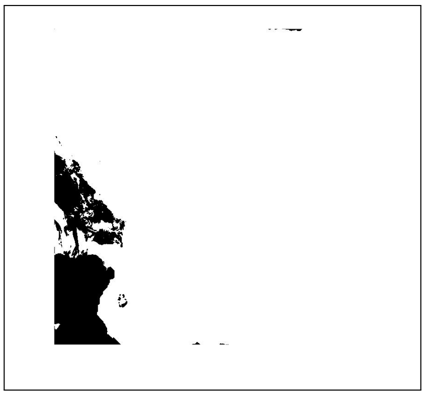


Figure 4.28: Best result for Figure 38 processed using a Fractional Derivative Mask ($th = 0.9$ and $\alpha = 2.3$).

Image #		threshold	α	J	D	Sensitivity	Specificity
12	Best J	0.9	0.8	**0.9828**	0.9913	0.9960	0.9978
	Best D	0.9	0.8	0.9828	**0.9913**	0.9960	0.9978
	Best Sens	0.3	2.0	0.5618	0.7195	**1.0000**	0.8716
	Best Spec	0.3	1.0	0.0002	0.0005	0.0002	**0.9999**
	Best Overall	**0.9**	**0.8**	0.9828	0.9913	0.9960	0.9978
18	Best J	0.5	1.7	**0.6594**	0.7947	0.6762	0.9915
	Best D	0.5	1.7	0.6594	**0.7947**	0.6762	0.9915
	Best Sens	0.1:0.9	-2:0.2	0.6581	0.7938	**0.6774**	0.9902
	Best Spec	0.3	1.0	0.0067	0.0133	0.0067	**0.9998**
	Best Overall	**0.5**	**1.7**	0.6594	0.7947	0.6762	0.9915
27	Best J	0.7	0.9	**0.9958**	0.9979	0.9977	0.9962
	Best D	0.7	0.9	0.9958	**0.9979**	0.9977	0.9962
	Best Sens	0.1:0.9	-2:0.5	0.9936	0.9968	**0.9989**	0.9891
	Best Spec	0.9	1.2	0.0573	0.1084	0.0573	**1.0000**
	Best Overall	**0.7**	**0.9**	0.9958	0.9979	0.9977	0.9962
38	Best J	0.9	2.3	**0.9625**	0.9809	0.9995	0.6936
	Best D	0.9	2.3	0.9625	**0.9809**	0.9995	0.6936
	Best Sens	0.1:0.9	-2:0.5	0.9582	0.9786	**0.9999**	0.6529
	Best Spec	0.9	1.3	0.0024	0.0047	0.0024	**0.9998**
	Best Overall	**0.9**	**2.3**	0.9625	0.9809	0.9995	0.6936

Table 4.6: Table with best performance results using Fractional Mask edge detectors on selected images

4.4 Grey-Scale Edge Detectors vs Color Based Edge Detectors

As explained before, color edge detection algorithms were also implemented and tested to the whole data set due to their good results in other applications. All the six methods were adapted and applied in the main algorithm in order to perform color-based edge detection. The best results of the performance assessment for this versions of the algorithms are presented above and are compared with the corresponding best grey-scale result. Appendix A also includes plots of the performance of the color versions for all the detectors studied before.

Note that until here all the metrics were used to evaluate performance. From the results it was clear that the Jaccard Coefficient is a good global indicator of performance since the maximum Jaccard Coefficient's processing corresponded always to the best overall result. Thus, from now on, only this coefficient will be used to conduct experimentation analysis.

Due to scale issues, many of the results that are close to zero in the following histograms are difficult to access if the difference in performance is positive or negative. In order to identify these easily, the y-axis values which corresponded to ΔJ were transformed to $10^{\Delta J}$. The resulting histograms are displayed in Appendix B.

4.4.1 Grey-Scale vs Color Canny Algorithm

The color based algorithm, after being implemented, was run for the same interval of parameters and the results treated in an Excel file. Excel allowed to compare the best color-based results with the best Grey-Scale Canny outcomes already analysed before.

Figure 4.29 presents the differences in performance between the best color and best grey-scale results for each of all images in the data set.

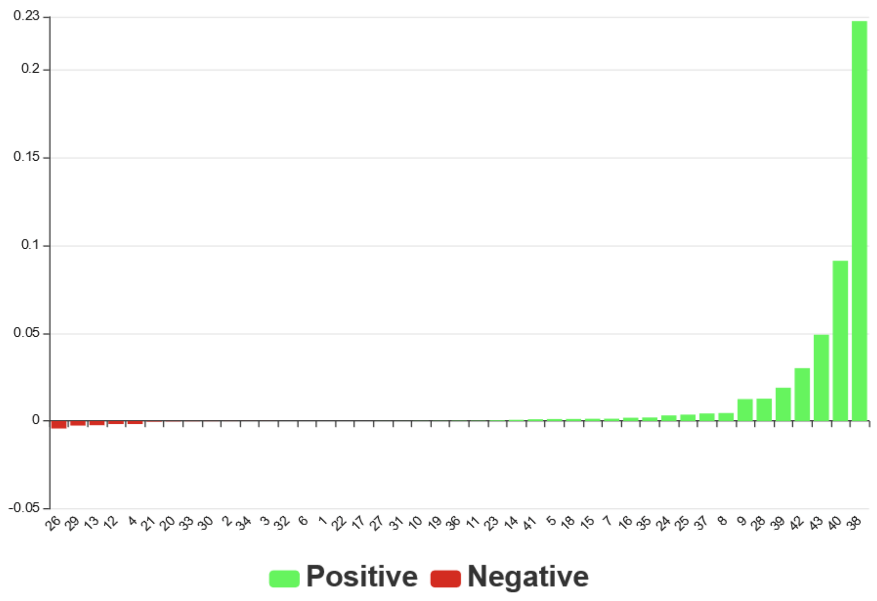


Figure 4.29: Performance comparison between grey-scale and color based Canny (Whole Data Set).

The greatest difference in performance between the two algorithms was observed in picture 38.

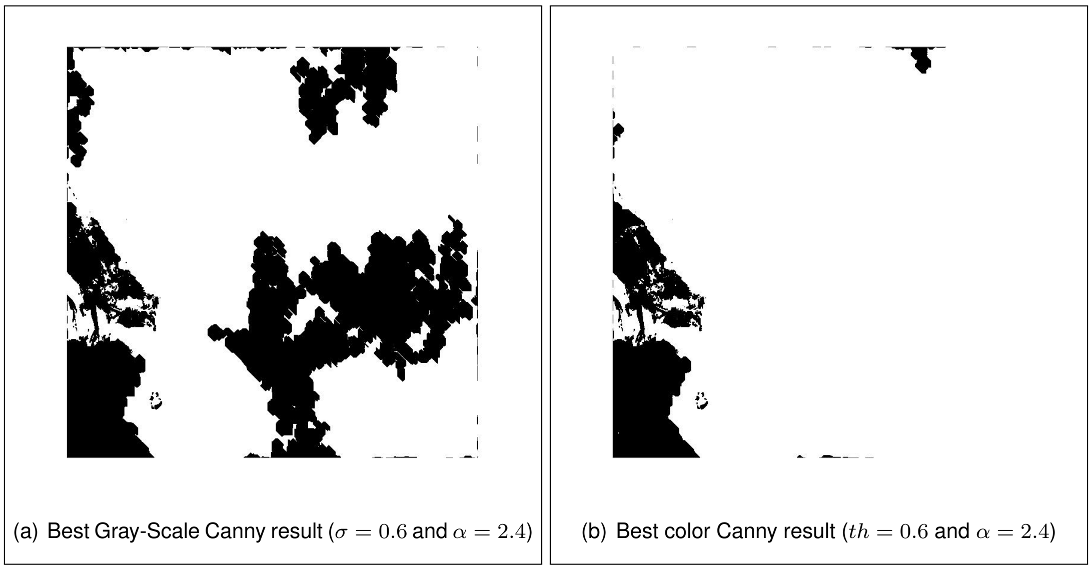

(a) Best Gray-Scale Canny result ($\sigma = 0.6$ and $\alpha = 2.4$)
(b) Best color Canny result ($th = 0.6$ and $\alpha = 2.4$)

Figure 4.30: Best results for Figure 38 processed using different types of Canny algorithms.

The Jaccard Coefficient rose from 73.53% to 96.32% which presents an enormous increase. In Figure 4.30 it can be observed that the grey-scale algorithm struggles to close inner-land areas. The fact that the color algorithm can identify these areas may be an evidence that color-based detection allows to find gradients that grey-scale detection cannot.

4.4.2 Grey-Scale vs Color Sobel Algorithm

The process was repeated using the Color adaptation of the Sobel algorithm. Once more, Excel allowed to compare the best results of the color algorithm against the grey-scale version ones.

Figure 4.31 presents the differences in performance between the best color-based and best grey-scale Sobel results for all images.

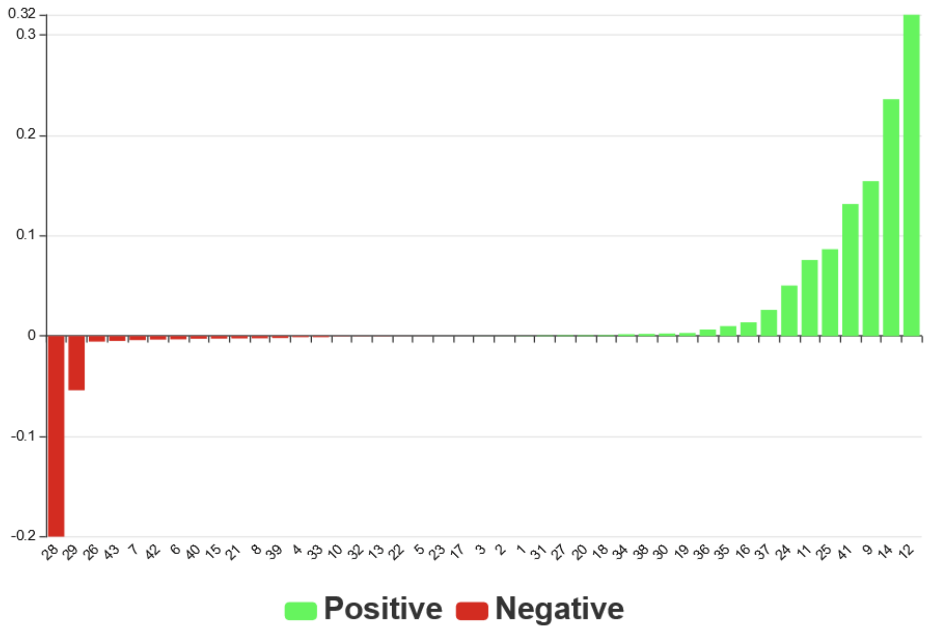


Figure 4.31: Performance comparison between grey-scale and color based Sobel (Whole Data Set).

In most images, the algorithms present similar performances. There are some relevant cases where the Jaccard coefficient rose significantly. However, there are two cases where the performance decreased more than 5%. The greatest differences in performance between the two algorithms were observed in images 12, 14 and 28.

In image number 12, the Jaccard Coefficient rose from 56.20% to 88.27% which presents an enormous increase of 32.07%. In Figure 4.32 it can be observed that the grey-scale algorithm cannot deal with the high level of noise in the image. The color-based algorithm is able to define the coast and presents high immunity to noise. The same conclusions can be drawn looking at the results obtained for image 14 (Figure 4.33).

In contrast, Figure 4.34 shows that in image 28 the result of the color-based algorithm presents higher level of noise (pixels incorrectly identified as land) than the grey-scale method's one. This results in a decrease of almost 20% in performance.

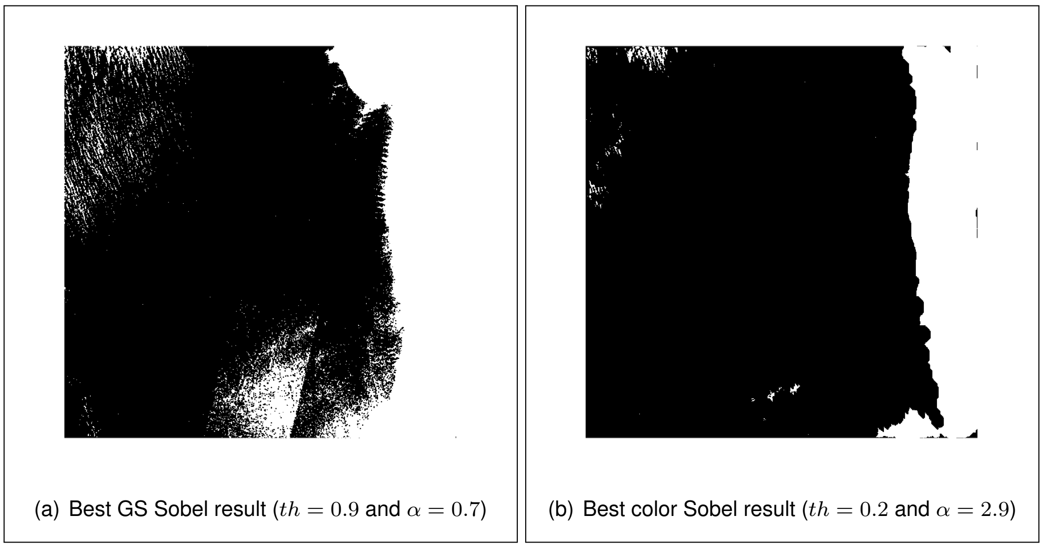

(a) Best GS Sobel result ($th = 0.9$ and $\alpha = 0.7$) (b) Best color Sobel result ($th = 0.2$ and $\alpha = 2.9$)

Figure 4.32: Best results for Figure 12 processed using different types of Sobel algorithms.

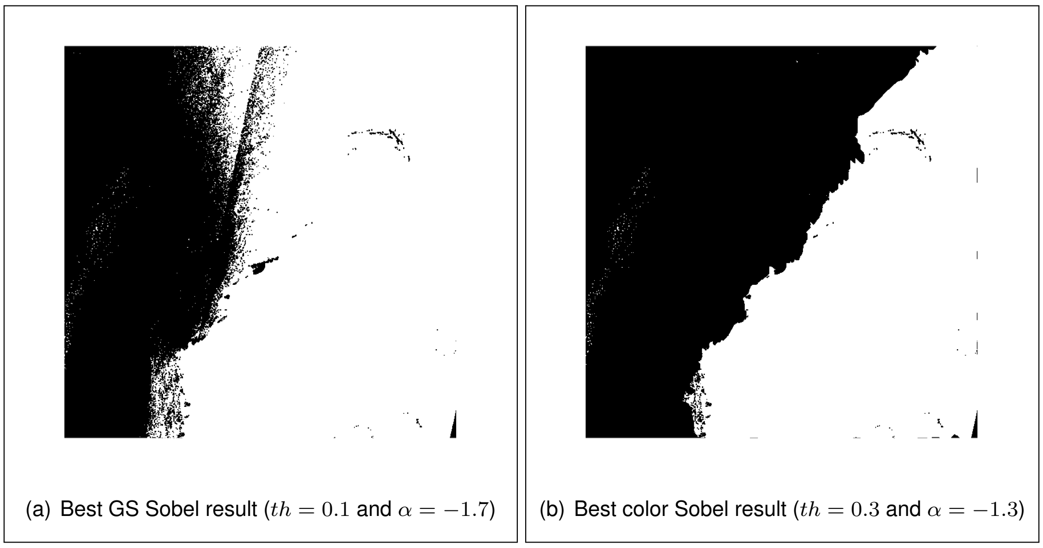

(a) Best GS Sobel result ($th = 0.1$ and $\alpha = -1.7$) (b) Best color Sobel result ($th = 0.3$ and $\alpha = -1.3$)

Figure 4.33: Best results for Figure 14 processed using different types of Sobel algorithms.

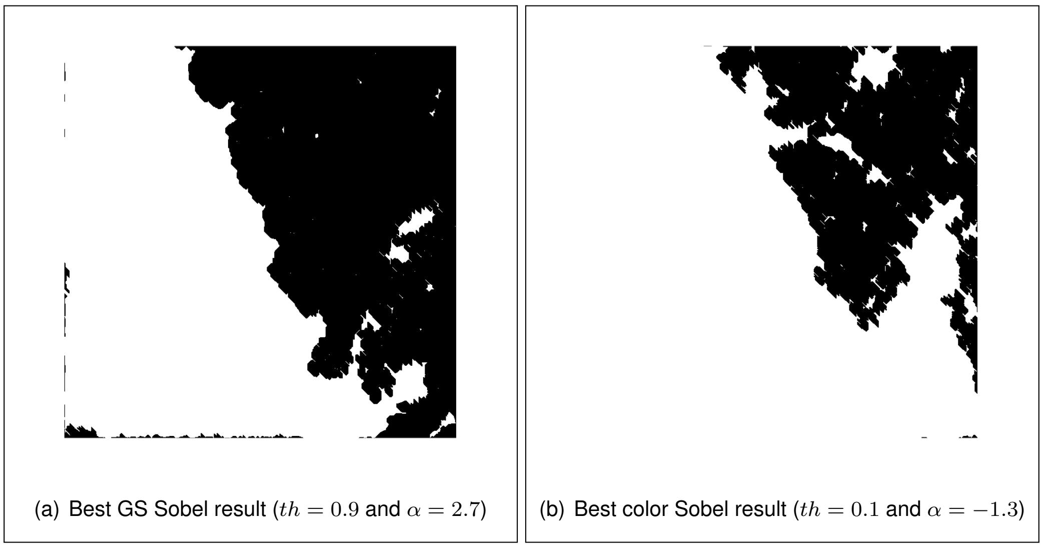

(a) Best GS Sobel result ($th = 0.9$ and $\alpha = 2.7$) (b) Best color Sobel result ($th = 0.1$ and $\alpha = -1.3$)

Figure 4.34: Best results for Figure 28 processed using different types of Sobel algorithms.

4.4.3 Grey-Scale vs Color Roberts Algorithm

The same comparison was performed for the versions of the Roberts detector. Figure 4.35 illustrates the differences in performance between Roberts' best color and best grey-scale methods' results .

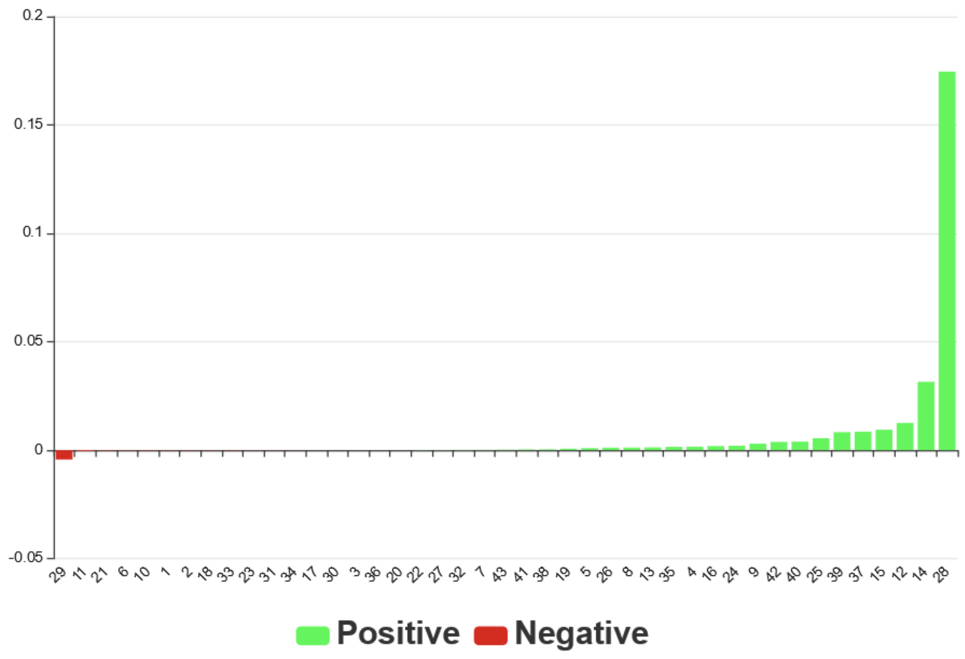
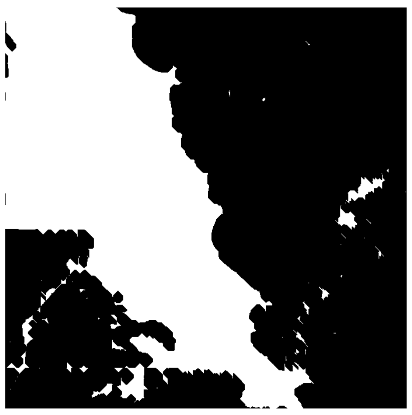


Figure 4.35: Performance comparison between grey-scale and color based Roberts (Whole Data Set).

In most images, the color-based algorithm presents similar or increased performance. The greatest variation was observed in image 28. In that figure, the Jaccard Coefficient rose from 68.80% to 86.25% which means an increase of 17.45%. In Figure 4.36 it can be observed that the grey-scale algorithm does not have the sensitivity to find all the gradients belonging to the inner land. The color-based algorithm defines the coast better in spite of the fact that it presents reduce immunity to noise in the water.


(a) Best GS Roberts result ($th = 0.1$ and $\alpha = -1.9$)

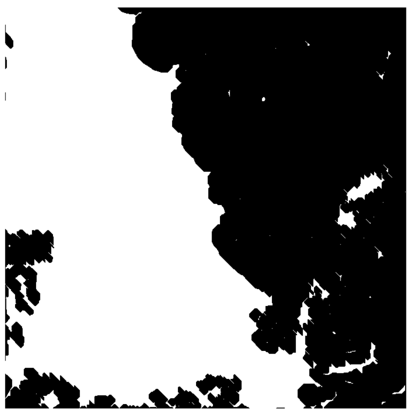

(b) Best color Roberts result ($th = 0.1$ and $\alpha = -1.8$)

Figure 4.36: Best results for Figure 28 processed using different types of Roberts algorithms.

4.4.4 Grey-Scale vs Color LoG Algorithm

After adapting the conventional Laplacian of Gaussian detector to find zero-crossings in at least one of the three channels of the RGB color space, the main algorithm was also tested using this method.

Figure 4.37 plots the comparison in performance between the Laplacian of Gaussian best color and best grey-scale methods' results for all images.

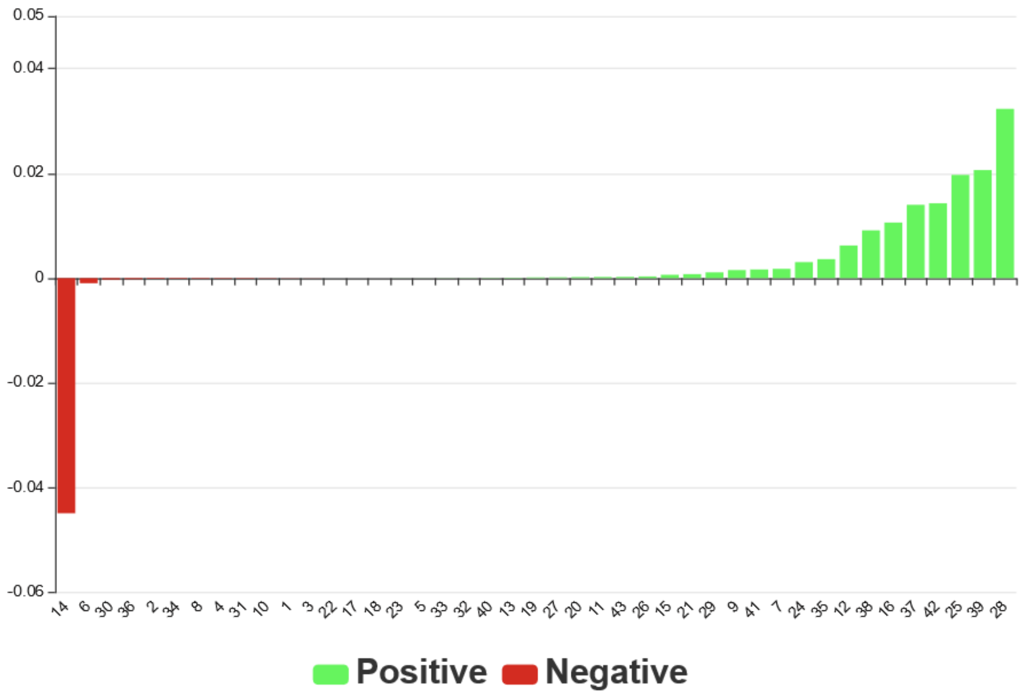


Figure 4.37: Performance comparison between grey-scale and color based LoG (Whole Data Set).

In most images, both types of algorithms perform identically. The variation does not exceed the 5% mark. Actually, the greatest difference in performance between the two algorithms was observed in image 14 with a decrease of 4.49%.

In conclusion, the color-based algorithm performs slightly better in the majority of the photos but in none of them the improvement is higher than 4%.

4.4.5 Grey-Scale vs Color CRONE Algorithm

The color based CRONE detector, after being adapted from the grey-scale one was tested for the same range of parameters and the results exported to an Excel file. Then the values were compared with the grey-scale CRONE already analysed before.

Figure 4.38 shows the differences in performance encountered between color-based Crone and grey-scale CRONE algorithms.

It is clear that the difference between the performance of the two algorithms is small in the majority of the cases. However, in most cases, the color based algorithm equaled or outperformed the grey-scale one. The greatest variation in performance happened in figure 12 with a 4% increase.

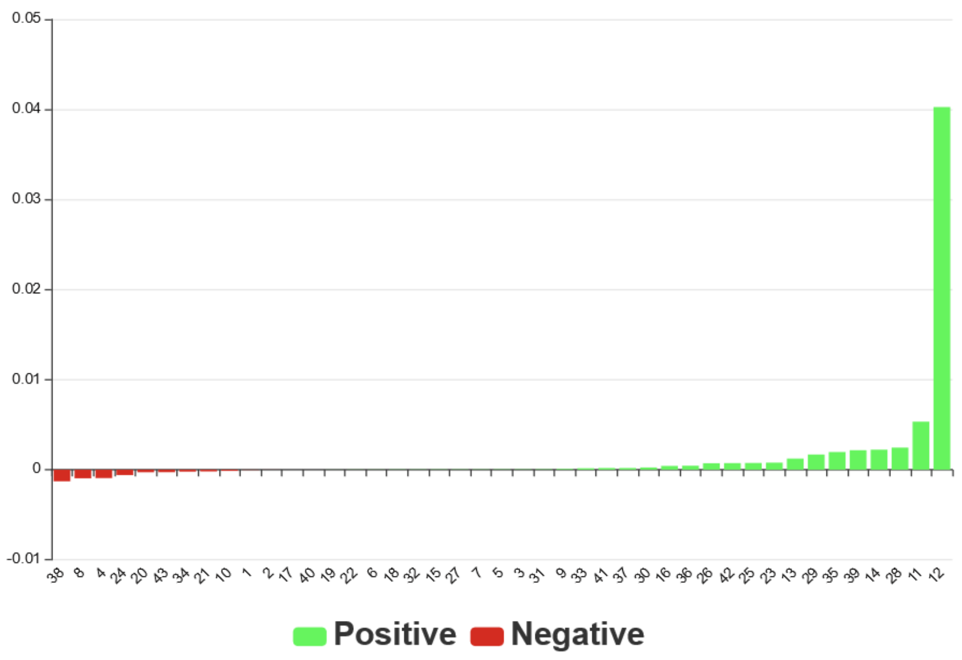


Figure 4.38: Performance comparison between grey-scale and color based CRONE (Whole Data Set).

4.4.6 Grey-Scale vs Color Fractional Derivative Mask Algorithm

The color-based Fractional Derivative operator was also tested for the same range of parameters and its results analysed against the grey-scale version.

Figure 4.39 shows the differences in performance encountered between color-based best Fractional Mask algorithm results and grey-scale algorithm ones.

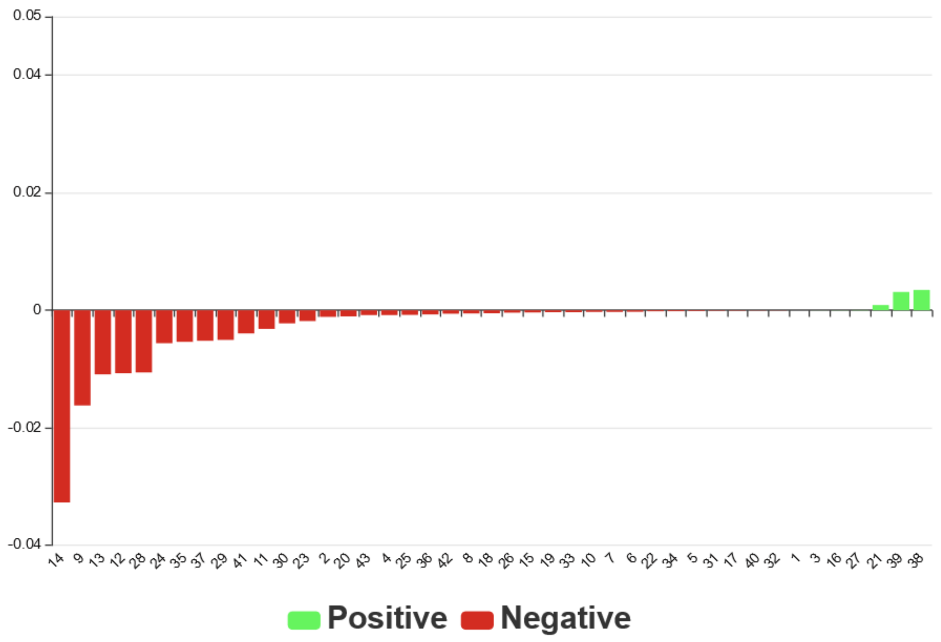


Figure 4.39: Performance comparison between grey-scale and color based Fractional Derivative Mask (Whole Data Set).

It is clear that the difference between the performance of the two algorithms is once more small in most cases. However, in this operator, the color based method performs worse in the majority of the images. The greatest variation in performance happened in figure 14 with a 3.5% decrease.

4.5 Overall Performance Analysis

After studying each of the algorithms independently, it is interesting to compare the different detectors and understand if there are substantial differences in performance between them.

First and for each algorithm, the best results for all images were obtained. Then, for each image the method with best Jaccard Coefficient was identified. The results of this analysis are shown in Figure 4.40.

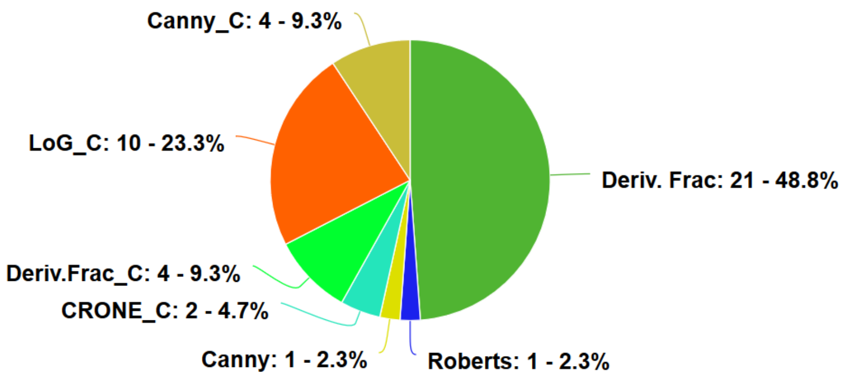


Figure 4.40: Dispersion of absolute best results by method for all images in the data set.

Note that no Sobel operator is not present in the pie chart. This happens because no processed figure showed best performance coefficients for this operator.

To further analyse the performance of the different detectors, the mean of the best results for each edge detection operator was computed. The results are shown in Table 4.7.

Detector	$\overline{J}$
Fractional Derivative Op.	0.9623
Color Fractional Derivative Op.	0.9596
Color Laplacian of Gaussian	0.9554
Color Roberts	0.9537
Laplacian of Gaussian	0.9531
Color Sobel	0.9478
Roberts	0.9474
Color CRONE	0.9461
Color Canny	0.9448
CRONE	0.9447
Canny	0.9342
Sobel	0.9282

Table 4.7: Ranking of the best results average for all images.

From Figure 4.40 and Table 4.7 some conclusions can be drawn:

- The two versions of the Fractional Derivative mask and the Laplacian of Gaussian operator are the most successful detectors. Together they represent more than 80% of the best results;
- Color versions of the algorithms performed generally better. The only detector that presents decreased average performance for its color version is the Fractional derivative operator;
- Sobel operator, despite its grey-scale version being the worst detector, presents higher performance for the color version exceeding the same versions of Canny and CRONE detectors.

The absolute best results using varying parameters for each image using the different operators are presented extensively in Appendix C.

4.6 Parameter Estimation for Automatic Image Processing

The above sections analysed from different perspectives the algorithm and the performance that it can reach in an edge detection identification problem.

However, it was noticed that the performance of the algorithm is deeply influenced by the chosen parameters, such as the order of the derivative used (α), the standard deviation (σ) of the Gaussian smoothing operation performed in some algorithms, the threshold or even the size of the mask of convolution (k).

One important objective of this work is to access if it is possible to find and easily tune parameters so that the fractional algorithm receives images as input and is able, in an automatic way, to process them with high performance and reliability. In the medical images works [2, 3] this was an issue.

Thus, in this section, a search for optimal parameters that allow a good detection performance for the whole data set will be conducted.

At first, it will be considered for this demand, the highest performance operator which is the grey-scale Fractional Derivative mask, as proven in section 4.5. The parameters that influence performance in this method are the threshold and α. The frequency histograms of the parameters in the best results for this operator are shown in Figure 4.41 and Figure 4.42.

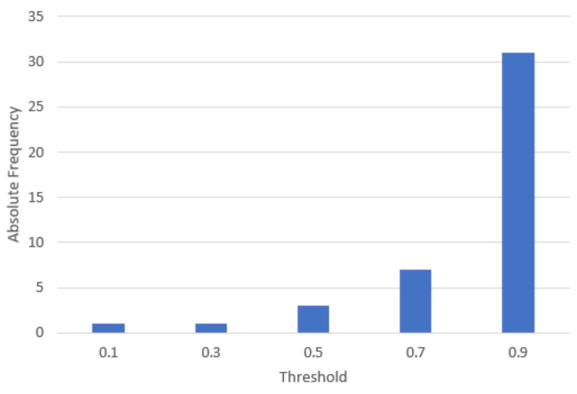


Figure 4.41: Absolute frequency of the threshold parameter for the best results using the fractional mask.

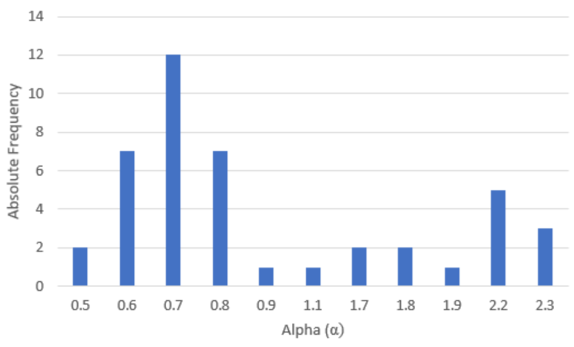


Figure 4.42: Absolute frequency of the α parameter for the best results using the fractional mask.

The most frequent parameters are $threshold = 0.9$ and $\alpha = 0.7$. These values revealed good performances in a high number of images. But can one use them together to process any image?

The results using this parameters were gathered. A comparison of these results with the best performances using the fractional mask detector might be an excellent way of understanding if the above mentioned parameters may be isolated. This comparison is shown in Figure 4.43.

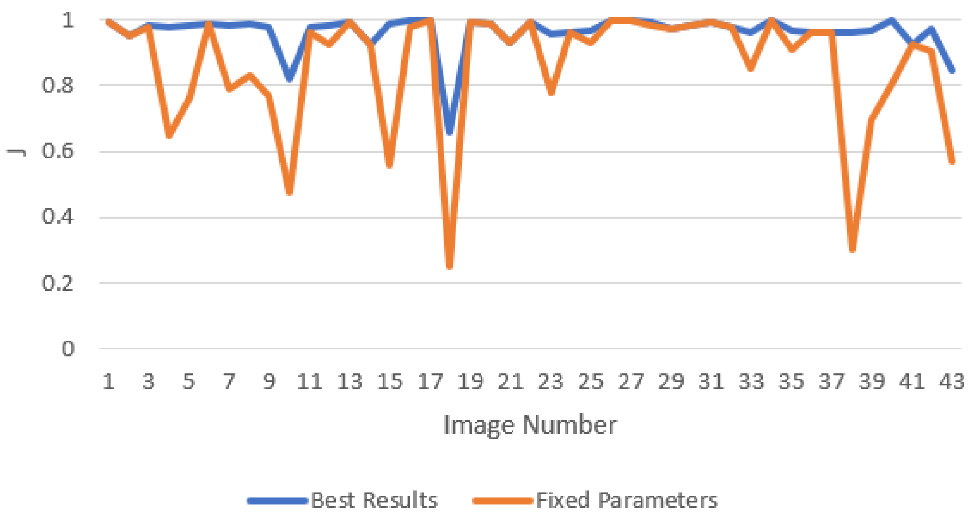


Figure 4.43: Best Results vs Best Results fixing parameters ($threshold = 0.9$ and $\alpha = 0.7$).

Figure 4.43 shows that despite having a few peaks that coincide with the best results, fixing the α parameter in 0.7 leads to a lot of performance flaws. Actually the mean performance decreased from 96.23% to 86.32%, almost 10%.

Observing again Figure 4.42, an interesting peak at $\alpha = 2.2$ was identified. This parameter performed optimally less times than 0.6 or 0.8, but it is much different than the value already tested which shows high potential of performing better. In order to seek a good overall response, the value of 2.2 for α was analysed.

Figure 4.44 shows that fixing the parameters with $threshold = 0.9$ and $\alpha = 2.2$ lead to overall better results despite having some images with low performance that correspond to a maximum of approximately 20% decrease. The mean performance of all images for these fixed parameters is 93.70% which corresponds to a decrease of less than 3% comparing with the variable parameters' best results.

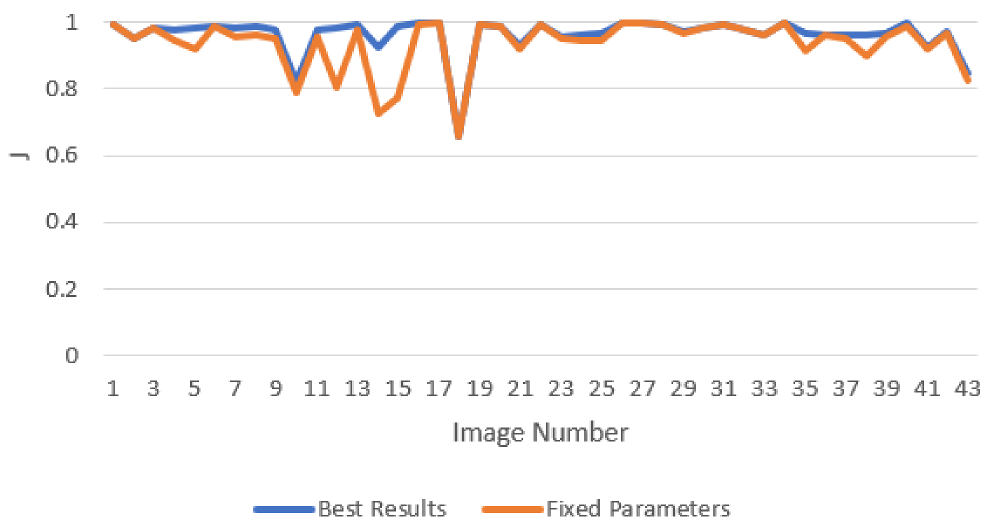


Figure 4.44: Best Results vs Best Results fixing parameters ($threshold = 0.9$ and $\alpha = 2.2$).

In order to further analyse the fixation of parameters in the main algorithm, an analysis using the whole image database (full-volume) was conducted in order to find the pair of parameters and detector that achieves the highest mean global performance. All the results from the full volume fixed-parameters analysis can be consulted in Appendix D under the form of global mean performance plots. Table 4.8 presents the best fixing parameters results for each detector.

Detector	$threshold$	k	σ	α	$\overline{J}$
Color Fractional Derivative Op.	0.9	-	-	0.8	0.9436
Fractional Derivative Op.	0.7	-	-	0.8	0.9428
Color CRONE	-	5	-	0.9	0.9328
CRONE	-	5	-	1.1	0.9261
Color Canny	-	-	0.7	1.7	0.9193
Color Roberts	0.1	-	-	1.4	0.9174
Color Sobel	0.3	-	-	-0.2	0.9153
Sobel	0.9	-	-	0.2	0.9111
Color Laplacian of Gaussian	0.1	-	-	-0.9	0.9108
Roberts	0.1	-	-	-1.3	0.9076
Laplacian of Gaussian	0.1	-	-	-1.4	0.9028
Canny	-	-	0.6	0	0.9011

Table 4.8: Ranking of the best results average using fixed parameters for all images.

Regarding the best results using fixed parameters presented above, one can conclude that:

- All detectors achieved an average score of at least 90% which means that any of the studied methods fits the purpose of this work which is identifying land and water in satellite images;
- All color-based versions of the detectors outperformed the respective gray-scale ones. This includes the color Fractional Derivative operator which had shown poorer results in comparison to the correspondent gray-scale detector with varying parameters.

- Both of the Laplacian of Gaussian versions which proved to be part of the best operators for the varying parameters results, present here a low score;
- The grey-scale Sobel presents better relative results with fixed parameters.

Chapter 5

Conclusions

In this section the analysis conducted in the previous chapter will be summarized and the final conclusions presented as well as future work that may be performed in the matter.

5.1 Performance of Fractional VS Integer methods

As explained before, the images withdrawn randomly from the Copernicus database were first used to obtain the ground truth for coast identification. This ground truth was then used to access the performance of the algorithm by comparing it to the processed images.

To have a global view of the performance for all of the eight detectors used, Appendix A presents the graphs of performance for the selected images in section 4.2 with varying parameters in the intervals defined in chapter 3. The extended results are available in [57].

The data presented in Appendix A and in section 4.2 allows to conclude that the use of fractional derivatives in edge detection for this application matched or improved, in most cases, the performance of the conventional integer methods.

The graphs presented in Appendix A show that it is difficult to find a single derivative order that outperforms all the others for different images. However, for this application, the performance graphics are well defined and therefore it is possible to use them to find ranges of values for the parameters that present optimal performance.

5.2 Performance of Grey-Scale VS Color based methods

In section 4.4 the results of fractional color edge detection are compared with the fractional ones using grey-scale images.

In one hand, the data in this section allows to conclude that, in the majority of cases, using the color based detector improves the performance of the algorithm. The color based Canny algorithm, for example, increased more than 20% the processing of image 38.

On the other hand, for most images in this database, the increase in performance that the color based algorithm provides is not higher than 1%. The high resolution of the satellite images may explain these results, since the conversion of pixels values in grey-scale may be better defined than a low quality figure. This means that the grey-scale identification already performs very well for this application which leaves a small margin for improvements in general.

In conclusion, the color based algorithms allowed to equal or improve the performance of grey-scale methods in most cases. However, often the increase in performance is low in percentage. Nevertheless, and since we are dealing with images that are composed of more than 120 million pixels, a percentage of 1% increase corresponds to more than 1 million pixels correctly identified. The color based detector is also heavier computationally since it usually requires more than one convolution (at least one for each color channel). The use of this type of operators in the context of this thesis may be useful if one is not satisfied with the results of grey-scale identification.

5.3 Overall Performance

In section 4.5, the results presented in Figure 4.40 and Table 4.7 allow to conclude that the direct use of the Fractional Derivative Mask in Equation 2.35 as a grey-scale detector, presents the highest mean Jaccard coefficient for all figures ($\overline{J}$ = 0.9623).

The analysis shows that the use of fractional derivatives presents promising results in edge detection. However, the means presented are computed through the best results for each image using diverse parameters. This wide range of optimal parameters may be hard to reach in an automatic solution.

5.4 Fixed Parameters for Automatic Detection

In section 4.6, optimal parameters for the grey-scale version of the Fractional Derivative mask were sought. Fixing the threshold in 0.7 and the order of the derivative (α) in 0.8 allowed to reach a maximum mean Jaccard coefficient for all images of 94.28%. This presents a solution with fixed parameters that performs very similarly as the best results with varying parameters.

In contrast with the varying parameters analysis, the fixation of parameters revealed that all the color versions of the operators performed better than the grey-scale ones. This said, the color version of the Fractional Mask Operator presents the best mean for the Jaccard coefficient in the results using fixed parameters for all images. This operator presents a mean performance coefficient of 94.36% with $\alpha = 0.8$ and $threshold = 0.9$. Thus, the color-based operator increased performance in 0.08%. Once more this may seem a reduced percentage but it means that the color-based algorithm detects correctly almost 100 thousand more pixels than the gray-scale's best fixing parameters result.

In conclusion, the solution using the color-based algorithm presented above achieved the best mean performance. Fixing the parameters in the given values meant a decrease in performance of less than 2% comparing it to the best result with varying constants. This implies that the proposed parameters work in the automatic identification of coasts in high definition satellite images.

5.5 Final Considerations

The initial goal of finding if the use of fractional derivatives in the processing of high quality satellite images enhances performance was tested in an identification problem where one desires to segment image in land and water.

It was proved that generalizing the conventional integer method adding fractional orders using Grünwald-Letnikov definition often increases the possibility of achieving a better detection performance.

In most cases, the implementation of color-based detectors matched or improved the performance of the algorithm. However, the improvements for this application are often small in percentage and the required computational power very high. This may happen due to the high resolution that satellite images possess which allows grey-scale detection to be already an optimal result. Thus, the color-based detector is seen as an exceptional alternative if the grey-scale processing does not achieve high performance standards.

The best detector tested in this research was the color-based version of the Fractional Derivative mask. Fixing parameters using this operator in the main algorithm provided great results. The chosen parameters are $threshold = 0.9$ and $\alpha = 0.8$ which present great potential in automatic shore identification in satellite images.

5.6 Future Work

The initial objective of this dissertation was to evaluate the usage of fractional derivatives in the processing of satellite images namely in the detection of coasts. This goal was achieved and it was possible to further investigate whether color detection improved results and if it is possible to fix parameters of detection in order to perform automatic detection.

Besides all this investigation, as always, it is possible to think of ways of further investigate and optimize the knowledge in this matter. Seeing as the scope of the project on which this thesis is inserted the fractional processing of satellite images, the following points represent some of the ideas on how to further investigate the matter and/or optimize the present one:

- Find ways to optimize the main algorithm present in this thesis. The presented method includes morphological operations that require the construction of structuring elements. These elements were chosen iteratively in order to maximize performance. Nevertheless, some images still have contours that the algorithm struggles to close so it may be possible to find structuring elements that operate better and make the algorithm achieve higher performance;

- Work on the optimization of detection parameters. As one can check in Appendix A, the graphs of the variation in parameters are well defined.Thus, optimization algorithms could be used to create an iterative algorithm that chooses the final parameters for an image based on performance. This could be a solution in which a different pair of parameters would be used in each image. However, the algorithm would be probably computationally massively heavy.

- Color-base edge detection methods showed once more excellent potential. The use of this detectors together with fractional derivative orders may be subject of further investigation. Other types of images may be used in a different study to corroborate the idea that color-based fractional processing of images work better than grey-scale conventional ones.
- In this work six detectors were implemented and analysed. Of course, other detectors exist that can equally detect edges efficiently and may be matter of further investigation. Some of them are Prewitt [39] and Deriche [58] operators or even Cumani's [59] color-based operator;
- Regarding INFANTE project, it was discussed with Prof. Paulo Gordo the possibility of adapting the current algorithm to detection of crude stains in satellite images which can be subject of further investigation and a new application to the developed method.

Bibliography

[1] ESA. Sentinel-2. https://sentinel.esa.int/web/sentinel/missions/sentinel-2/mission-objectives, . (Visited on 12/06/2020).

[2] T. F. C. Bento. Tratamento de Imagem Aplicado à Imagiologia Médica Usando Derivadas Fraccionárias. Master's thesis, Instituto Superior Técnico, 2015.

[3] J. F. M. Gonçalves. 3D Fractional Order Image Processing for Cancer Detection. Master's thesis, Instituto Superior Técnico, 2018.

[4] ESA. Copernicus Open Access Hub. https://scihub.copernicus.eu/, . (Visited on 31/03/2020).

[5] Infante project. http://www.infante.space/. (Visited on 31/04/2020).

[6] GeeksforGeeks. Satellite Image Processing. https://www.geeksforgeeks.org/satellite-image-processing/. (Visited on 10/06/2020).

[7] H. S. University. Resolution. http://gsp.humboldt.edu/OLM/Courses/GSP_216_Online/lesson3-1/resolution.html. (Visited on 10/06/2020).

[8] T. Reichhardt. First Photo From Space. *Air and Space Magazine*, 2006.

[9] ESA. History of Earth observation. http://www.esa.int/SPECIALS/Eduspace_EN/SEM1NP3Z2OF_0.html, . (Visited on 12/06/2020).

[10] NASA. A Landsat Timeline. https://landsat.gsfc.nasa.gov/about/landsat-timeline. (Visited on 25/11/2020).

[11] M. I. Skolnik. Radar. https://www.britannica.com/technology/radar, 2020. (Visited on 25/11/2020).

[12] M. P. Lazarević, M. R. Rapaic, and T. B. Sekara. *Advanced Topics on Applications of Fractional Calculus on Control Problems, System Stability and Modeling*, chapter 1-Introduction to Fractional Calculus with Brief Historical Background, pages 3–16. WSEAS Press, 2014.

[13] S. F. Lacroix. *Traité Du Calcul Différential et du Calcul Intégral*, volume 3. Paris Courcier, 2nd edition, 1819.

[14] B. Ross. A Brief History and Exposition of The Fundamental Theory of Fractional Calculus. In *Fractional Calculus and Its Applications*, volume 457. Springer, Berlin, Heidelberg, 1975. doi: https://doi.org/10.1007/BFb0067096.

[15] B. Riemann. *Versuch einer allgemeinen Auffassung der Integration und Differentiation. (1847.)*, page 331–344. Cambridge Library Collection - Mathematics. Cambridge University Press, 2013. doi: 10.1017/CBO9781139568050.020.

[16] H. Laurent. Sur le calcul des dérivées à indices quelconques. *Nouvelles annales de mathématiques 3esérie, tome 3*, pages 240–252, 1884.

[17] A. K. Grünwald. Über "begrenzte" Derivationen und deren Anwendung. *Zeitschrift für Mathematik und Physik, 12*, pages 441–480, 1867.

[18] M. Caputo. Linear Model of Dissipation whose Q is almost Frequency Independent-II. *Geophysical Journal International*, 13(5):529–539, November 1967. doi: 10.1111/j.1365-246X.1967.tb02303.x.

[19] S. W. Wheatcraft and M. M. Meerschaert. Fractional Conservation of Mass. *Advances in Water Resources*, 31(10):1377 – 1381, 2008. doi: 10.1016/j.advwatres.2008.07.004.

[20] J. A. T. Machado. Some Applications of Fractional Calculus in Engineering. `https://www.hindawi.com/journals/mpe/2010/639801/`, 2010. (Visited on 15/07/2020).

[21] A. Atangana and N. Bildik. The Use of Fractional Order Derivative to Predict the Groundwater Flow. *Mathematical Problems in Engineering*, pages 1–9, 2013. doi: 10.1155/2013/543026.

[22] N. Sebaa, Z. E. A. Fellah, and C. Depollier. Application of Fractional Calculus to Ultrasonic Wave Propagation in Human Cancellous Bone. *Signal Processing*, 86:2668–2677, 2006. doi: 10.1016/j.sigpro.2006.02.015.

[23] D. Benson, S. Wheatcraft, and M. Meerschaert. Application of a Fractional Advection-Dispersion Equation. *Water Resources Research*, 36, September 2004. doi: 10.1029/2000WR900031.

[24] R. Metzler and J. Klafter. The Random Walk's Guide to Anomalous Diffusion: a Fractional Dynamics Approach. *Physics Reports*, 339(1):1–77, 2000. doi: 10.1016/S0370-1573(00)00070-3.

[25] F. Mainardi. *Fractional Calculus and Waves in Linear Viscoelasticity: An Introduction to Mathematical Models*. May 2010. doi: 10.1142/P614.

[26] A. Bhrawy and M. Zaky. An Improved Collocation Method for Multi-dimensional Space–time Variable-order Fractional Schrödinger Equations. *Applied Numerical Mathematics*, 111:197–218, January 2017. doi: 10.1016/j.apnum.2016.09.009.

[27] A. Letnikov. Theory of Differentiation with an Arbitrary Index (Russian). *Moscow Matem. Sbornik*, 6:413–445, 1872.

[28] D. Valério and J. S. da Costa. Introduction to single-input, single-output fractional control. *IET Control Theory and Applications Vol. 5, no. 8*, page 1033–1057, May 2011. doi: 10.1049/iet-cta.2010.0332.

[29] M. Rensen. The Bartlane System (Coded System). `http://www.hffax.de/history/html/bartlane.html`. (Visited on 23/07/2020).

[30] B. Mac Namee. Digital Image Processing: Introduction. `http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/Image%20Processing-Introduction-Bryan-Mac-Namee.pdf`. (Visited on 23/07/2020).

[31] A. Arora. Fundamental Steps of Digital Image Processing. `https://medium.com/futframe-ai/fundamental-steps-of-digital-image-processing-d7518d6bb23c`. (Visited on 23/07/2020).

[32] T. Chakrabarty. 11 fundamental steps in digital image processing with a neat block diagram. `https://onlineclassnotes.com/2011/10/describe-fundamental-steps-of-digital.html#.Xw8ZGuf0VPY`. (Visited on 23/07/2020).

[33] MC.AI. Convolution Operation: Comprehensive Guide. `https://mc.ai/convolution-operation-comprehensive-guide/`. (Visited on 23/07/2020).

[34] Y. Tan. *GPU-based Parallel Implementation of Swarm Intelligence Algorithms*. Morgan Kaufmann, 2016.

[35] P. Sharma. Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques (Part 1). `https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/`. (Visited on 27/07/2020).

[36] Mathworks. Global image threshold using Otsu's method. `https://www.mathworks.com/help/images/ref/graythresh.html`. (Visited on 15/08/2020).

[37] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Image Processing Learning Resources. `http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm`. (Visited on 30/07/2020).

[38] L. Roberts. *Machine Perception of 3-D Solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[39] M. C. E. Laboratory. Edge Detector. `http://lab.must.or.kr/(S(2iatfs55hncsa455z1rxqt55))/History.aspx?Page=Edge-Detector&Revision=12&AspxAutoDetectCookieSupport=1`. (Visited on 31/07/2020).

[40] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. doi: 10.1109/TPAMI.1986.4767851.

[41] A. Mordvintsev and A. K. Canny Edge Detection. `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html`. (Visited on 31/07/2020).

[42] A. Sachan. Deep Learning based Edge Detection in OpenCV. `https://cv-tricks.com/opencv-dnn/edge-detection-hed/`. (Visited on 31/07/2020).

[43] CVonline. Edges: The Canny Edge Detector. `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm`. (Visited on 31/07/2020).

[44] X. M. Zhao, W. X. Wangand, and L. P. Wang. Parameter Optimal Determination for Canny Edge Detection. *The Imaging Science Journal Vol 59*, pages 332–341, 2011. doi: 10.1179/136821910X12867873897517.

[45] W. K. Pratt. Second-Order Derivative Edge Detection. In *Digital Image Processing*. John Wiley Sons, Inc., 2007. doi: 10.1002/0470097434.

[46] X. Chen and X. Fei. Improving Edge-detection Algorithm based on Fractional Differential Approach. In *IPCSIT*, volume 50, 2012. doi: 10.7763/IPCSIT.2012.V50.48.

[47] C. Yaacoub and R. A. Zeid Daou. Fractional Order Sobel Edge Detector. In *2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–5, 2019. doi: 10.1109/IPTA.2019.8936101.

[48] D. Tian, J. Wu, and Y. Yang. A Fractional-Order Laplacian Operator for Image Edge Detection. *Applied Mechanics and Materials*, 536-537:55–58, 04 2014. doi: 10.4028/www.scientific.net/AMM.536-537.55.

[49] B. Mathieu, P. Melchior, A. Oustaloup, and C. Ceyral. Fractional differentiation for edge detection. *Signal Processing*, 83(11):2421–2432, 2003. doi: 10.1016/S0165-1684(03)00194-4.

[50] S. Subhasini and M. Singh. Color Image Edge Detection: A Survey. *Int. J. Innov. Eng. Technol.*, 8: 235–247, 2017. doi: 10.21172/ijiet.81.032.

[51] A. Koschan and M. Abidi. Detection and Classification of Edges in Color Images. *IEEE Signal Processing Magazine*, 22(1):64–73, February 2005. doi: 10.1109/MSP.2005.1407716.

[52] G. S. Robinson. Color Edge Detection. *Optical Engineering*, 16(5):479 – 484, 1977. doi: 10.1117/12.7972120.

[53] T. Kanade and S. Shafer. Image Understanding Research at Carnegie Mellon. In *Proceedings of a Workshop on Image Understanding Workshop*, page 32–48, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1558600701.

[54] GIMP. GNU Image Manipulation Program. `https://www.gimp.org/`. (Visited on 4/4/2020).

[55] S.-Y. Zhu, K. N. Plataniotis, and A. N. Venetsanopoulos. Comprehensive Analysis of Edge Detection in Color Image Processing. *Optical Engineering*, 38(4):612–625, April 1999. doi: 10.1117/1.602105.

[56] P. Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[57] M. Henriques˙ Results Repository. `https://drive.google.com/drive/folders/1GMeKvc3oqNWfzd4h-GyRwHT9yFJ_UDLP?usp=sharing`.

[58] R. Deriche. Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector. *International Journal of Computer Vision*, 1(2):167–187, 1987. doi: 10.1007/bf00123164.

[59] A. Cumani. Edge detection in multispectral images. *CVGIP: Graphical models and image processing*, 53(1):40–51, 1991. doi: 10.1016/1049-9652(91)90018-F.

Appendix A

Parameter Variation Illustration for all detection methods

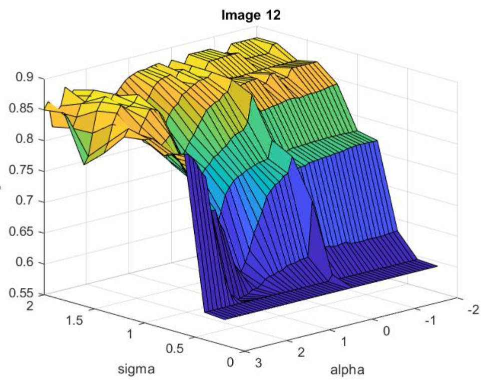
In this appendix, all the results of the variation of parameters are shown in the form of 3D plots.

Parameters which happen to cause computational problems and yield zero performance were ignored in order not to have this results influencing the plots' scale. The orders ignored are $\alpha = 1$ for the Sobel and LoG, $\alpha = 0$ for CRONE and $\alpha = \{-2, -1\}$ for the Fractional Derivative operator and the Roberts Operator.

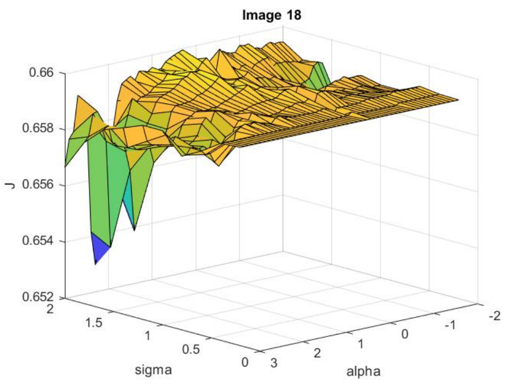
Since it would be impossible to present graphs for all methods and images, the four selected images in section 4.2 were also chosen to have their graphs of parameter variation represented here.

A.1 Grey-Scale Methods

A.1.1 Grey-scale Canny

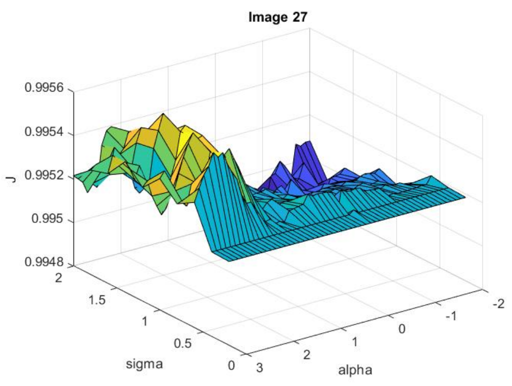


(a) Coast(12)

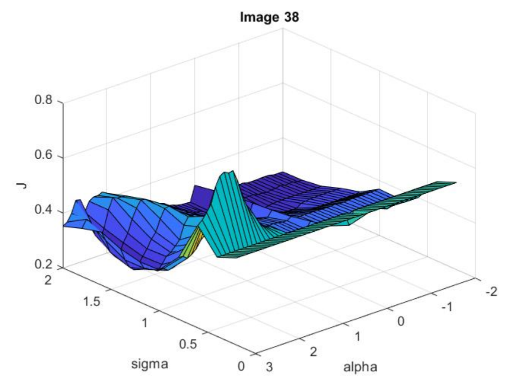


(b) Coast(18)

Figure A.1: Algorithm Performance using grey-scale Canny detector (1).



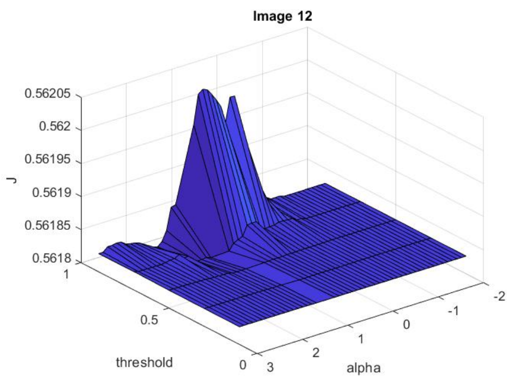
(a) Coast(27)



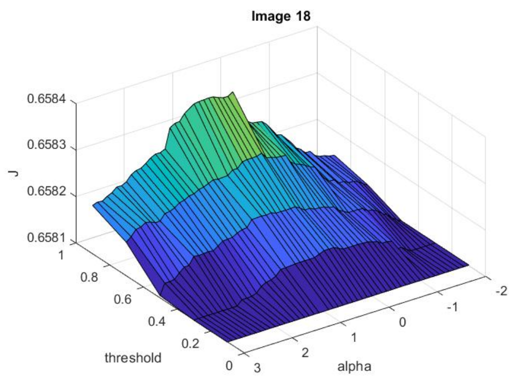
(b) Coast(38)

Figure A.2: Algorithm Performance using grey-scale Canny detector (2).

A.1.2 Grey-scale Sobel

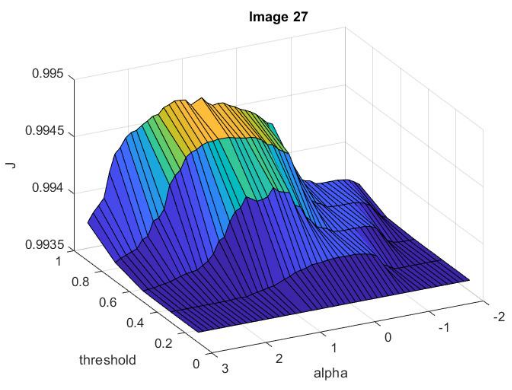


(a) Coast(12)

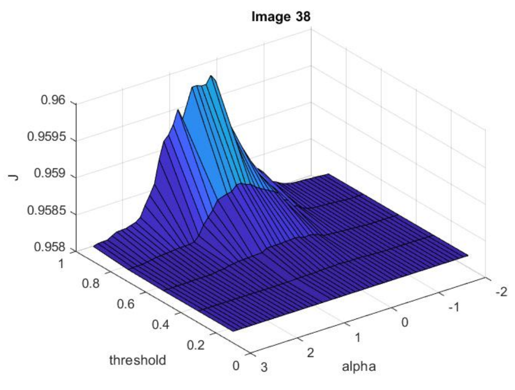


(b) Coast(18)

Figure A.3: Algorithm Performance using grey-scale Sobel detector (1).



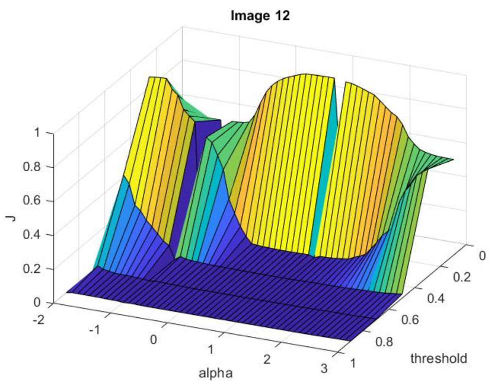
(a) Coast(27)



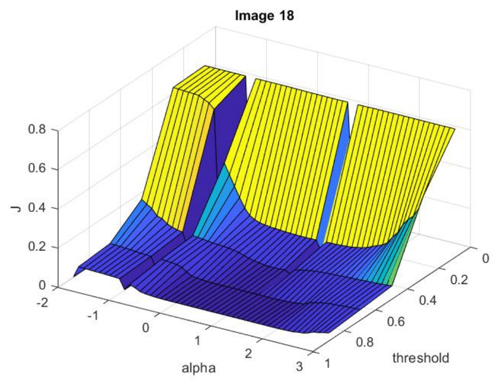
(b) Coast(38)

Figure A.4: Algorithm Performance using grey-scale Sobel detector (2).

A.1.3 Grey-scale Roberts

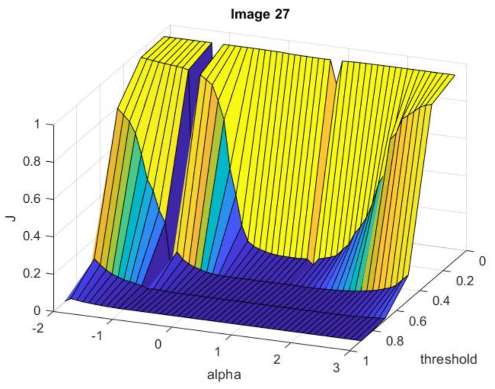


(a) Coast(12)

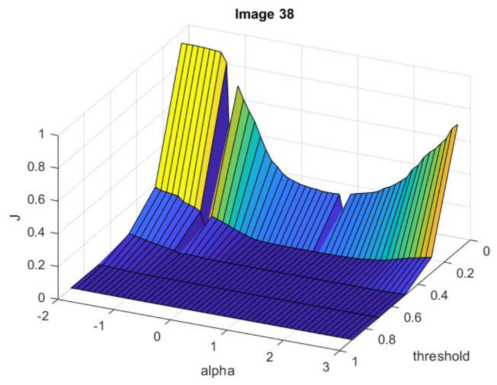


(b) Coast(18)

Figure A.5: Algorithm Performance using grey-scale Roberts detector (1).



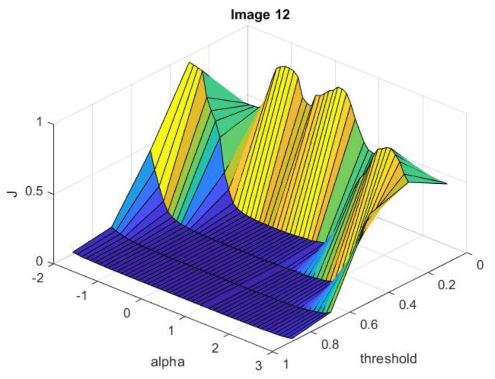
(a) Coast(27)



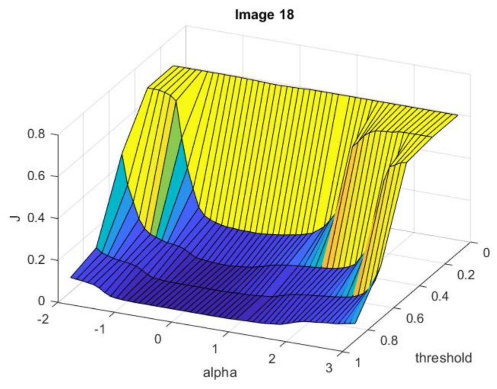
(b) Coast(38)

Figure A.6: Algorithm Performance using grey-scale Roberts detector (2).

A.1.4 Grey-scale Laplacian of Gaussian

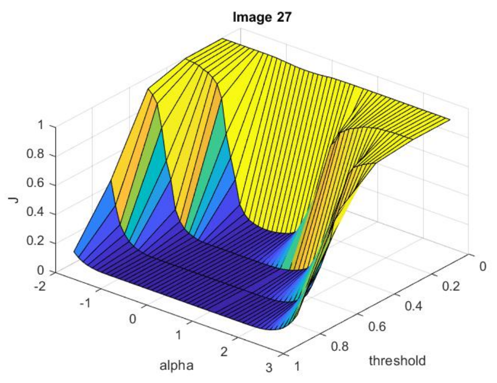


(a) Coast(12)

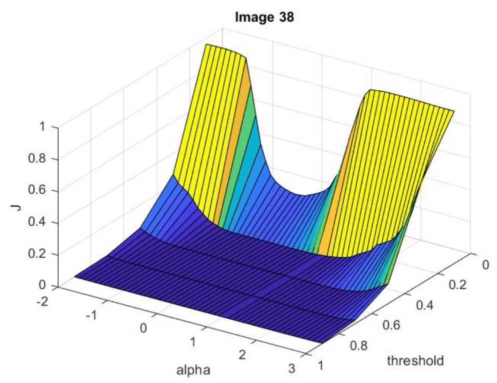


(b) Coast(18)

Figure A.7: Algorithm Performance using grey-scale LoG detector (1).



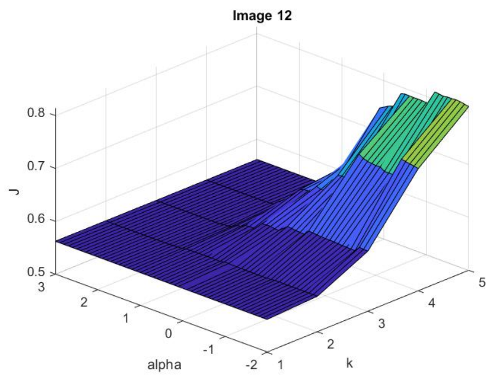
(a) Coast(27)



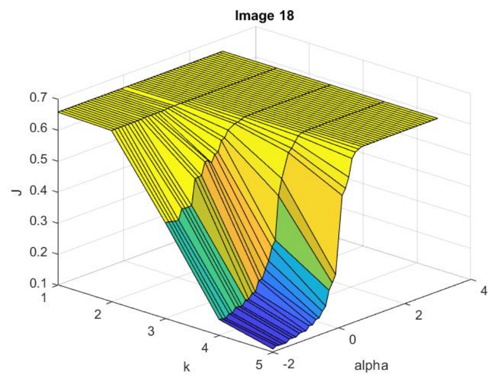
(b) Coast(38)

Figure A.8: Algorithm Performance using grey-scale LoG detector (2).

A.1.5 Grey-scale CRONE

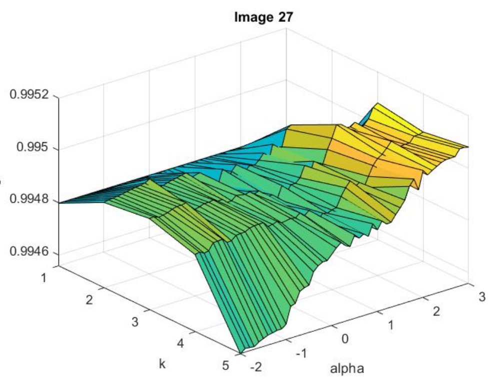


(a) Coast(12)

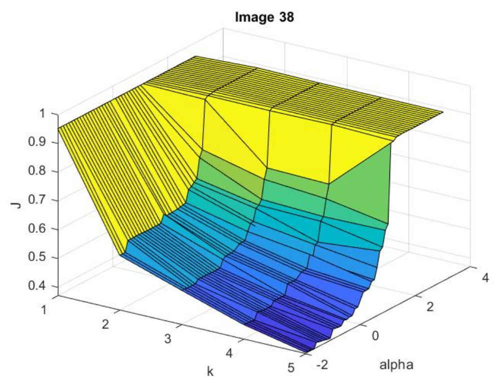


(b) Coast(18)

Figure A.9: Algorithm Performance using grey-scale CRONE detector (1).



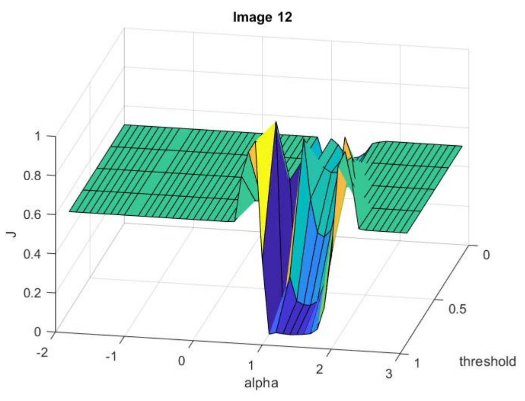
(a) Coast(27)



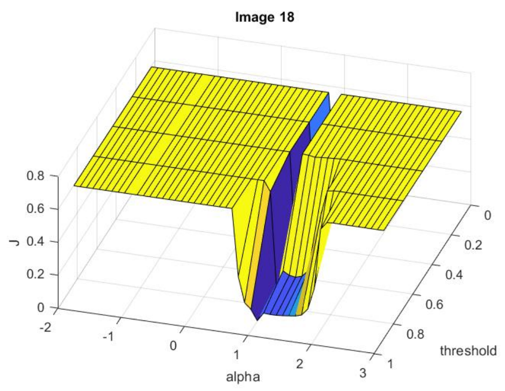
(b) Coast(38)

Figure A.10: Algorithm Performance using grey-scale CRONE detector (2).

A.1.6 Grey-scale Fractional Derivative Operator

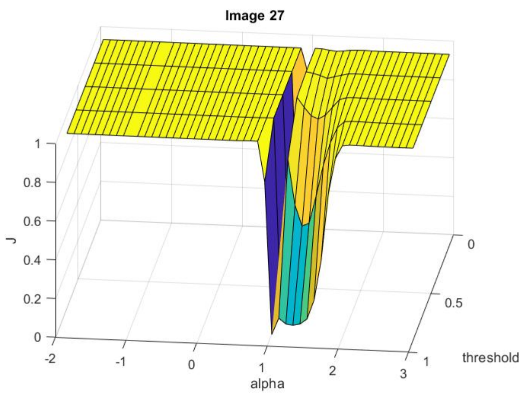


(a) Coast(12)

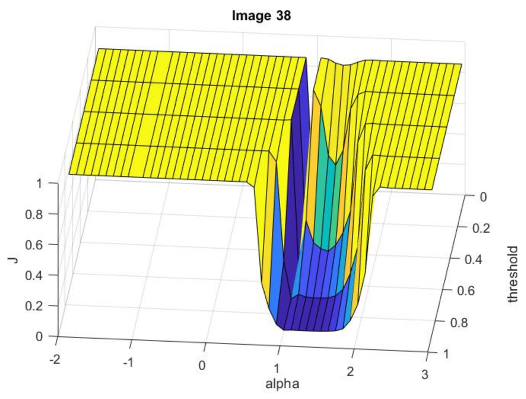


(b) Coast(18)

Figure A.11: Algorithm Performance using grey-scale Fractional Derivative detector (1).



(a) Coast(27)

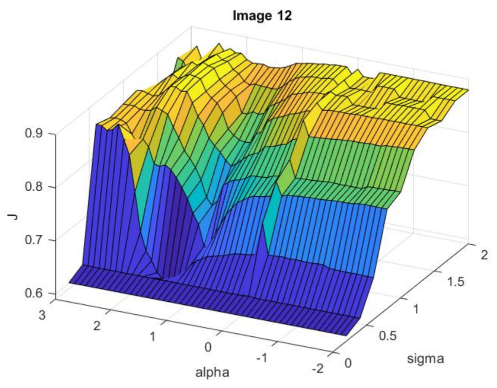


(b) Coast(38)

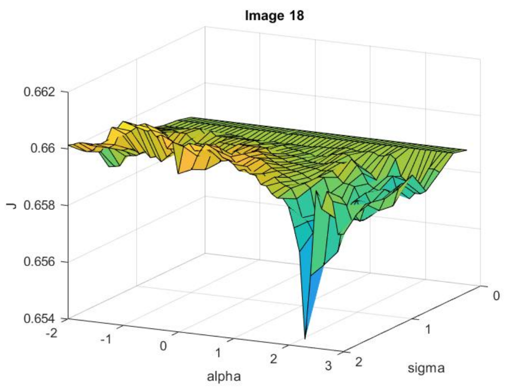
Figure A.12: Algorithm Performance using grey-scale Fractional Derivative detector (2).

A.2 Color-based Methods

A.2.1 Color-based Canny

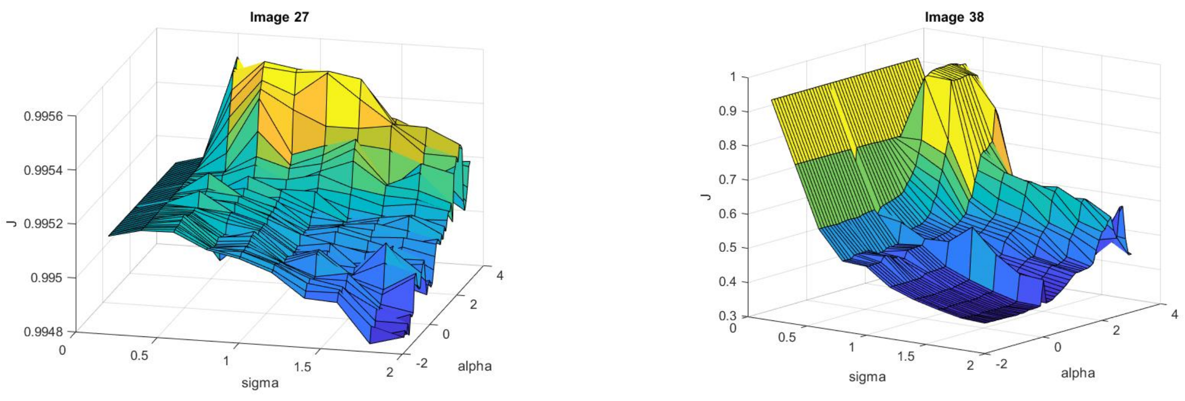


(a) Coast(12)



(b) Coast(18)

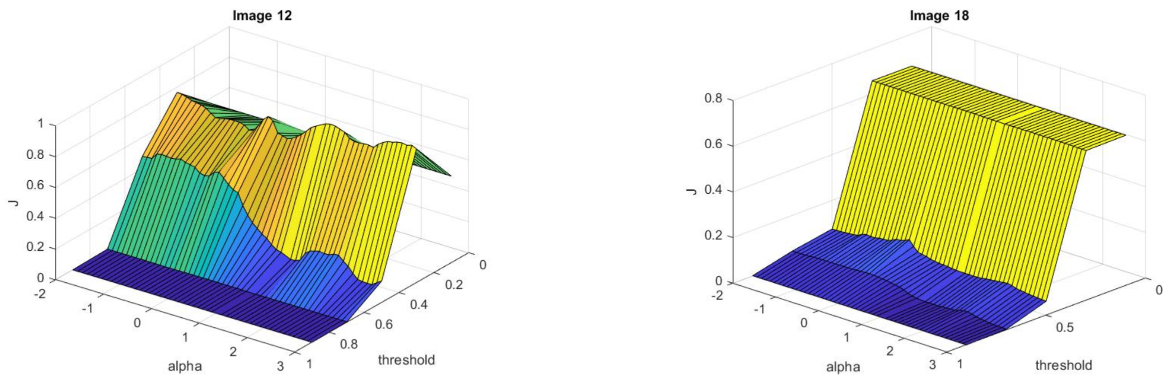
Figure A.13: Algorithm Performance using color-based Canny detector (1).



(a) Coast(27) (b) Coast(38)

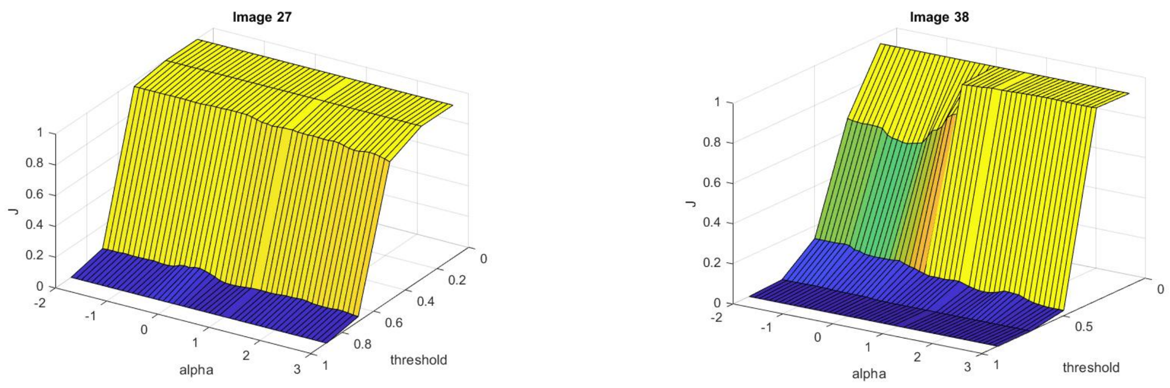
Figure A.14: Algorithm Performance using color-based Canny detector (2).

A.2.2 Color-based Sobel



(a) Coast(12) (b) Coast(18)

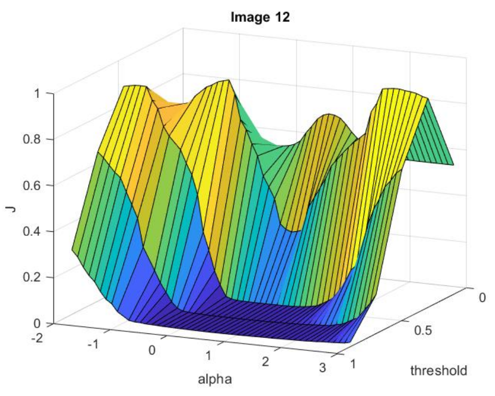
Figure A.15: Algorithm Performance using color-based Sobel detector (1).



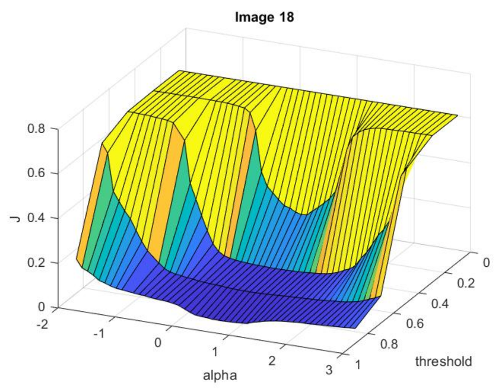
(a) Coast(27) (b) Coast(38)

Figure A.16: Algorithm Performance using color-based Sobel detector (2).

A.2.3 Color-based Roberts

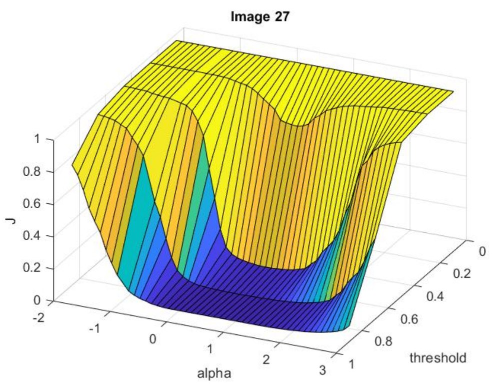


(a) Coast(12)

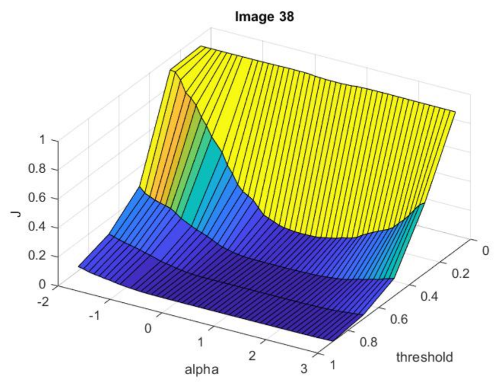


(b) Coast(18)

Figure A.17: Algorithm Performance using color-based Roberts detector (1).



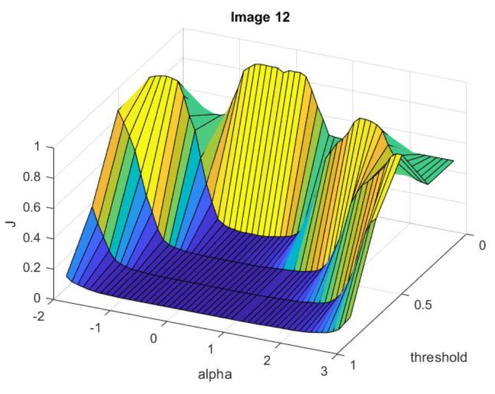
(a) Coast(27)



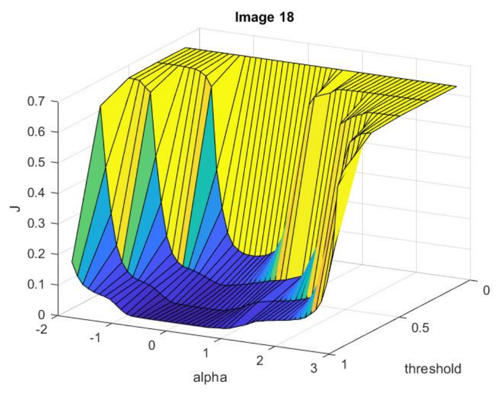
(b) Coast(38)

Figure A.18: Algorithm Performance using color-based Roberts detector (2).

A.2.4 Color-based Laplacian of Gaussian

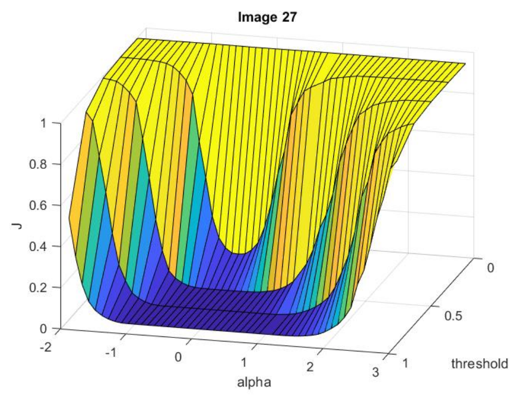


(a) Coast(12)

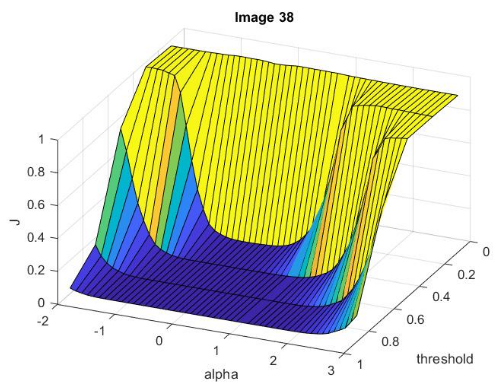


(b) Coast(18)

Figure A.19: Algorithm Performance using color-based LoG detector (1).



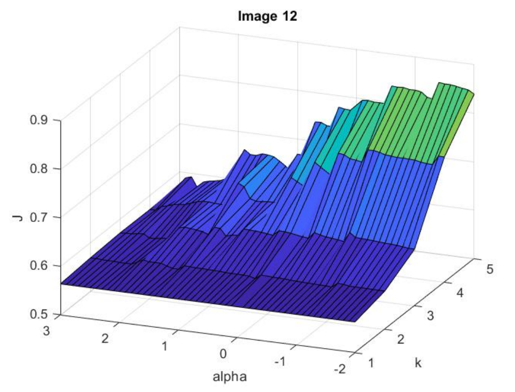
(a) Coast(27)



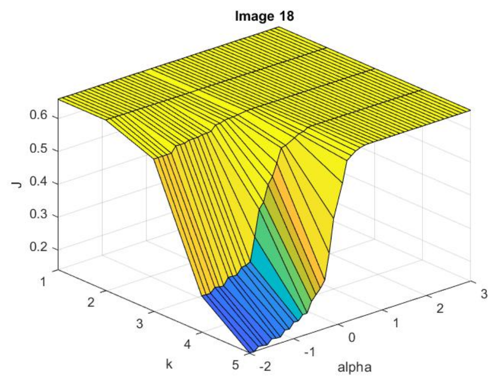
(b) Coast(38)

Figure A.20: Algorithm Performance using color-based LoG detector (2).

A.2.5 Color-based CRONE

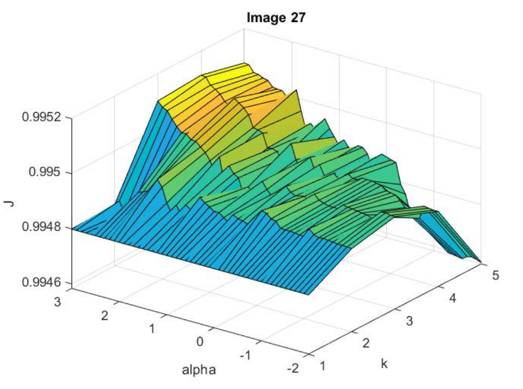


(a) Coast(12)

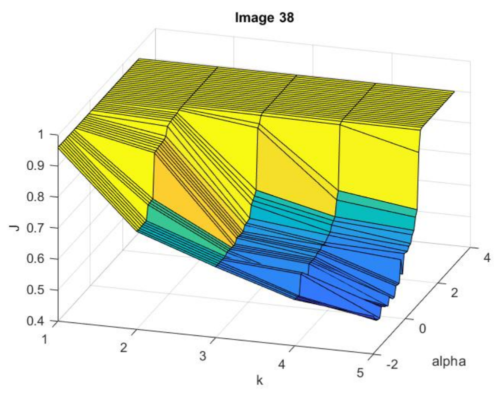


(b) Coast(18)

Figure A.21: Algorithm Performance using color-based CRONE detector (1).



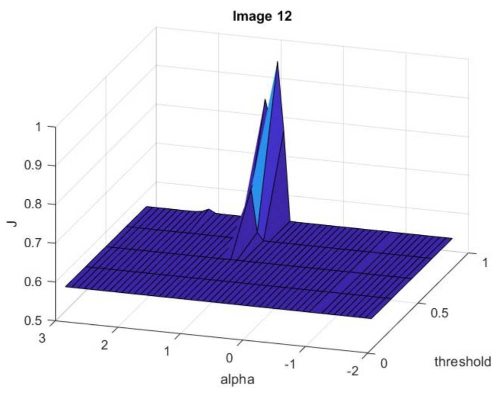
(a) Coast(27)



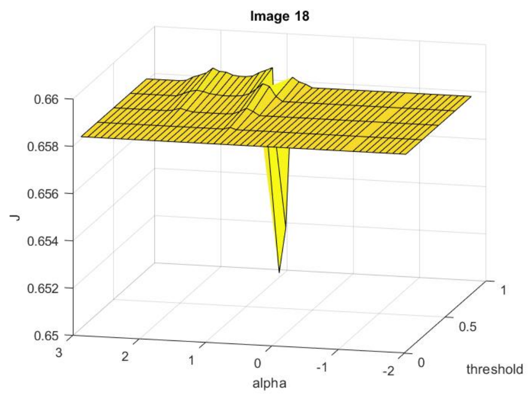
(b) Coast(38)

Figure A.22: Algorithm Performance using color-based CRONE detector (2).

A.2.6 Color-based Fractional Derivative Operator

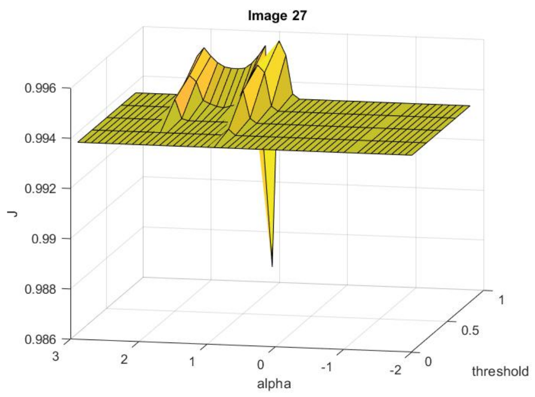


(a) Coast(12)

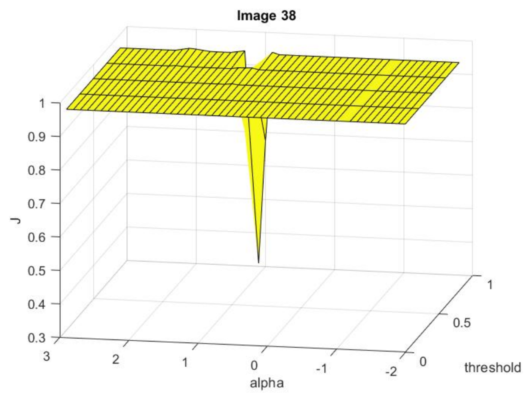


(b) Coast(18)

Figure A.23: Algorithm Performance using color-based Fractional Derivative detector (1).



(a) Coast(27)



(b) Coast(18)

Figure A.24: Algorithm Performance using color-based Fractional Derivative detector (2).

Appendix B

Grey-scale VS Color-based Best Results' Exponential Graphs

The graphs in section 4.4 are essential to know the exact values of the greatest variations in performance between different versions of edge detectors. However, this graphs' scale does not let one understand if the difference in images that present reduced variation is in fact positive or negative. In order to further analyse this cases, the variation in Jaccard coefficient (ΔJ) was transformed according to Equation B.1

$$y = 10^{\Delta J} \tag{B.1}$$

In this appendix, the histograms that compare best grey-scale against color-based results were converted to exponential graphs. Note that are considered even, variations in performance lower than 1×10^{-4}.

B.1 Grey-Scale vs Color Canny Algorithm

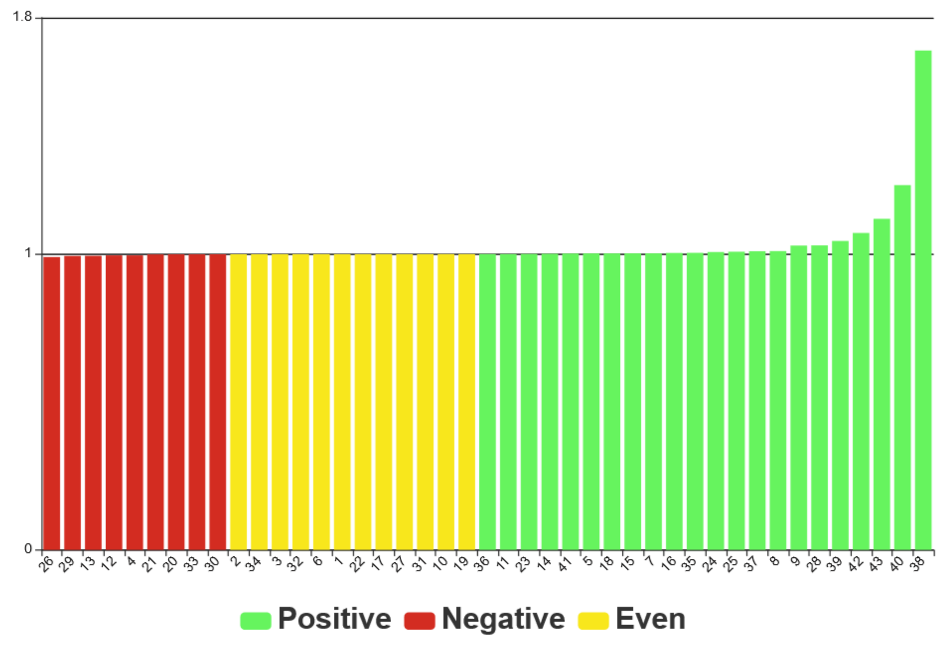


Figure B.1: Analysis of performance comparison between grey-scale and color based Canny (Exponential Version).

B.2 Grey-Scale vs Color Sobel Algorithm

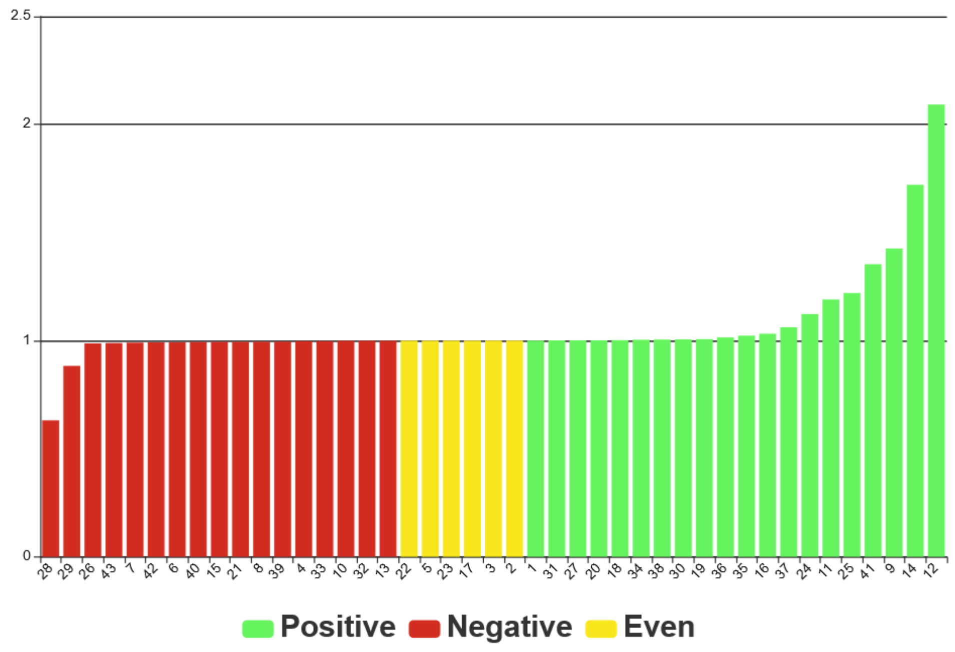


Figure B.2: Analysis of performance comparison between grey-scale and color based Sobel (Exponential Version).

B.3 Grey-Scale vs Color Roberts Algorithm

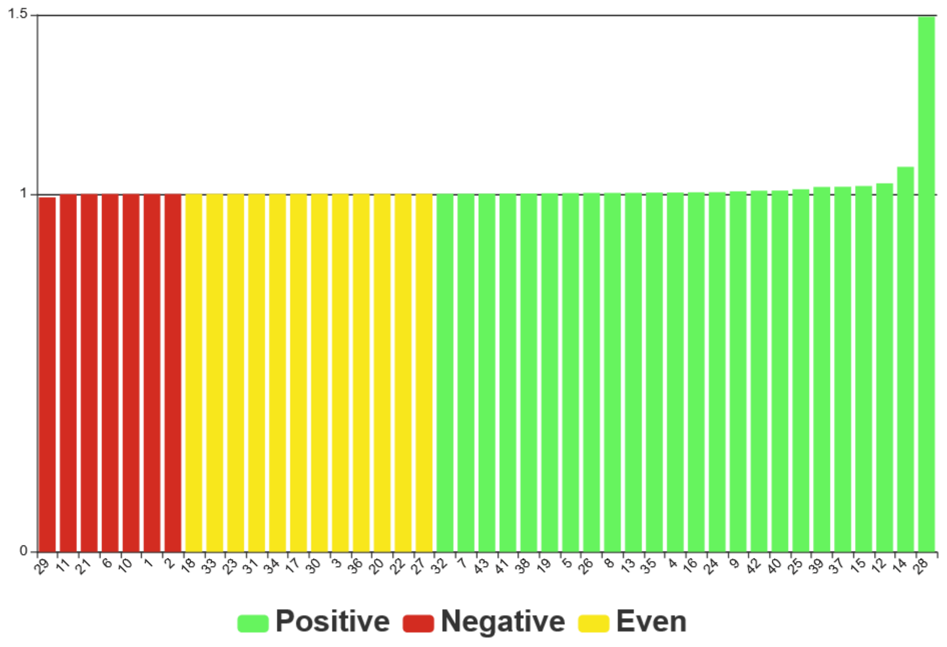


Figure B.3: Analysis of performance comparison between grey-scale and color based Roberts (Exponential Version).

B.4 Grey-Scale vs Color Laplacian of Gaussian Algorithm

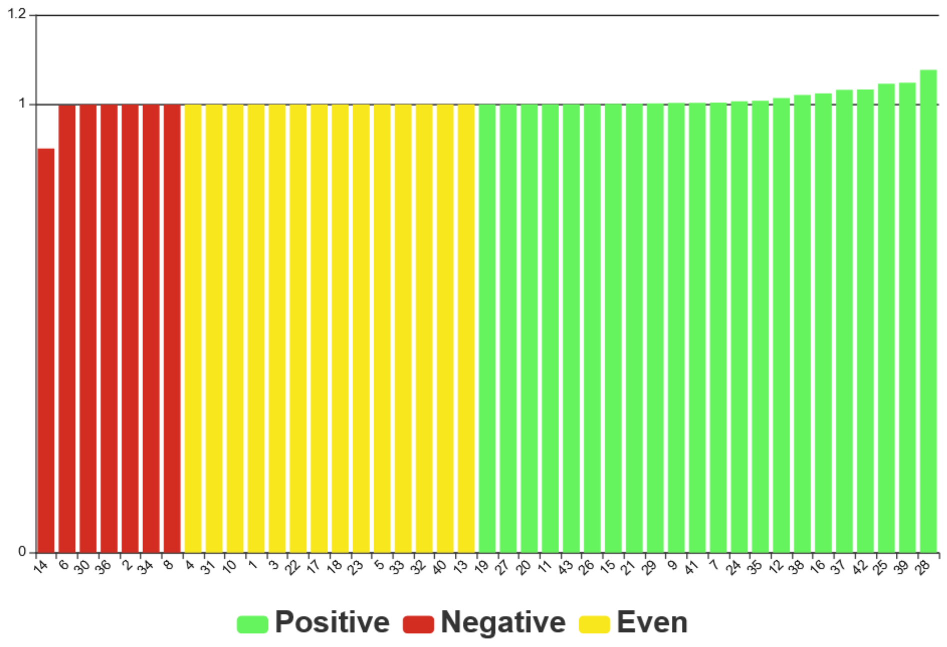


Figure B.4: Analysis of performance comparison between grey-scale and color based LoG (Exponential Version).

B.5 Grey-Scale vs Color CRONE Algorithm

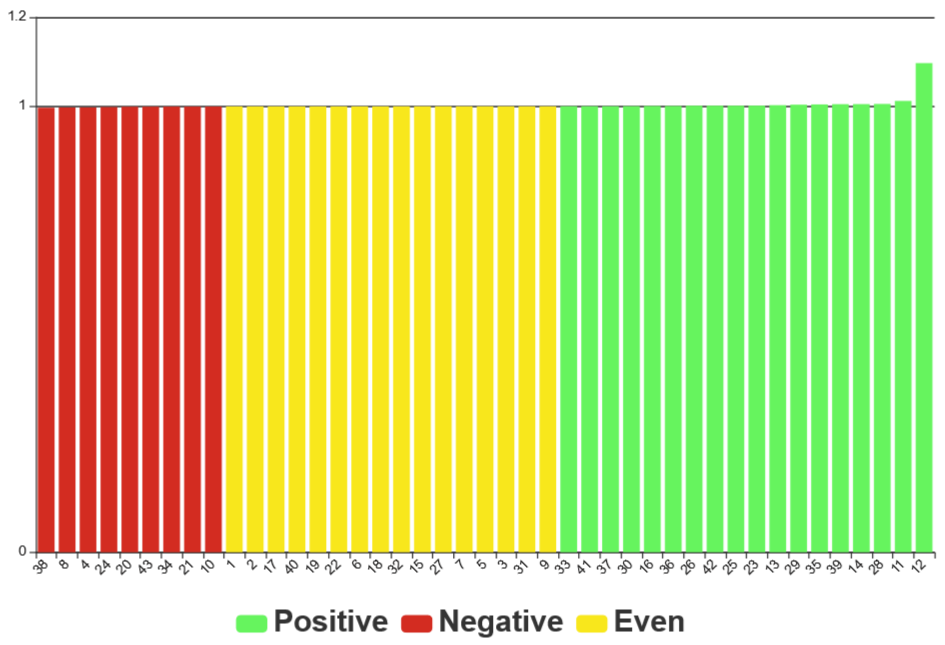


Figure B.5: Analysis of performance comparison between grey-scale and color based CRONE (Exponential Version).

B.6 Grey-Scale vs Color Fractional Derivative Operator Algorithm

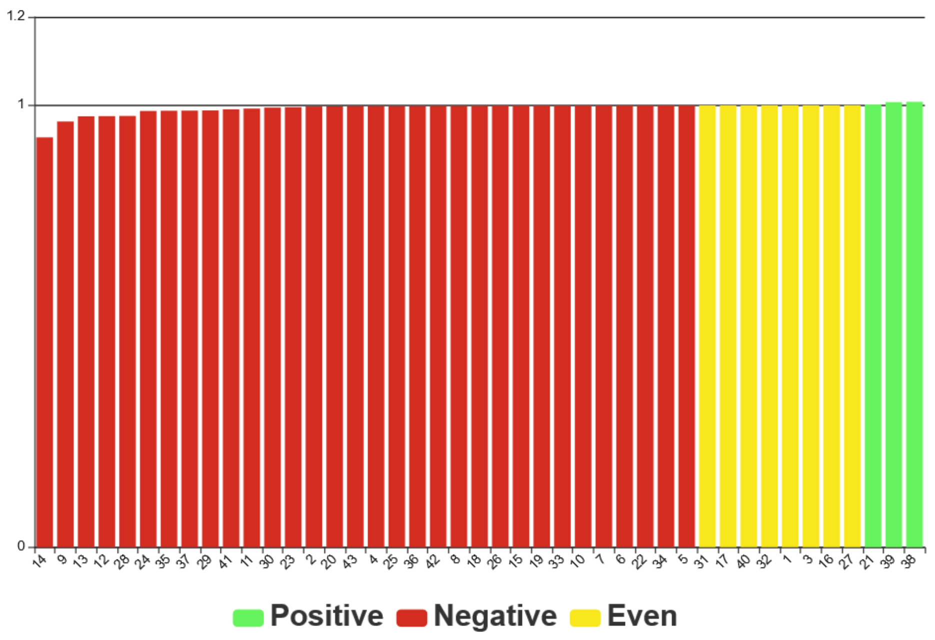


Figure B.6: Analysis of performance comparison between grey-scale and color based Fractional Derivative Operator (Exponential Version).

Appendix C

Absolute Best Results with Varying Parameters

In this appendix, the absolute best results are presented. For each operator, Table C.1 and Table C.2 show the best performances and used parameters in all detectors. It is also presented the mean varying parameters performance value for each detector.

Best Results			Canny			C Canny			CRONE			C CRONE			Oper. Deriv. Frac			C Oper.Deriv.Frac		
J	**Method**	**N**	σ	α	**J**	σ	α	**J**	**k**	α	**J**	**th**	α	**J**	**th**	α	**J**	**th**	α	**J**
0.9933	Roberts	1	1	2.5	0.992954	2.6	1	0.99295	5	2.1	0.9933	5	2	0.99326	0.5	1.1	0.99309	0.9	0.9	0.993094
0.95103	DERIV	2	0.6	2.5	0.949059	2.5	0.6	0.94898	4	3	0.94939	4	3	0.94938	0.9	0.8	0.95103	0.9	1.2	0.949919
0.98183	C DERIV	3	1.4	1.7	0.97949	2.1	1	0.97944	5	1.3	0.97894	5	1.3	0.97897	0.9	2.2	0.98181	0.9	1.2	0.981831
0.97917	DERIV	4	1	1.1	0.928494	1.1	0.8	0.9268	4	2.8	0.97448	3	3	0.97354	0.9	0.6	0.97917	0.9	1.3	0.978363
0.98494	C LoG	5	1	2	0.92196	2	1	0.92311	5	2.6	0.98458	5	3	0.98461	0.9	0.5	0.98476	0.7	1.9	0.984647
0.98871	DERIV	6	0.8	1.4	0.983291	1.5	1	0.98327	3	0.6	0.9872	4	1	0.98721	0.9	0.6	0.98871	0.9	0.8	0.988454
0.98139	DERIV	7	0.6	1.9	0.97334	1.8	0.6	0.97464	5	1.6	0.98076	5	1.8	0.98079	0.9	0.6	0.98139	0.7	0.9	0.981116
0.98792	DERIV	8	0.6	1.5	0.931001	1.6	0.8	0.93554	3	2.9	0.97861	2	2.3	0.97766	0.9	0.6	0.98792	0.9	0.8	0.987387
0.9795	DERIV	9	1.2	-2	0.955009	-1.2	1.4	0.96751	5	-1.6	0.96941	5	-1.6	0.96949	0.9	0.6	0.9795	0.9	0.8	0.963302
0.81755	DERIV	10	1.2	0	0.816913	1.6	1	0.81698	1	-2	0.81717	2	0.4	0.81703	0.7	0.7	0.81755	0.9	2.2	0.817272
0.97653	C LoG	11	0.6	2.6	0.975759	2.6	0.6	0.97592	5	2.7	0.90572	4	3	0.91105	0.5	1.8	0.97436	0.9	1.1	0.971197
0.98282	DERIV	12	1.4	2.2	0.888505	2.3	1.6	0.88679	5	-1.2	0.81383	5	-1.4	0.85408	0.9	0.8	0.98282	0.9	0.9	0.97211
0.99066	DERIV	13	2	0	0.986739	0	2	0.98437	5	-0.1	0.97115	5	-1.9	0.97238	0.9	0.7	0.99066	0.9	0.8	0.979754
0.98304	C CANNY	14	1.8	-1.6	0.982362	-1.1	2	0.98304	5	-1.3	0.96604	5	-1.5	0.96826	0.9	0.7	0.92424	0.9	0.9	0.891528
0.98799	C LoG	15	2	0.8	0.98475	0.6	1.8	0.98603	4	1.3	0.98621	5	3	0.98624	0.9	2.3	0.98772	0.7	0.9	0.987347
0.99616	C DERIV	16	1.4	1.6	0.99183	2.7	1.8	0.99369	5	0.6	0.99476	5	0.8	0.99517	0.7	0.7	0.99614	0.9	0.7	0.996163
0.99622	DERIV	17	2	0	0.993945	0	1.8	0.99395	5	-0.7	0.99438	5	-0.3	0.99438	0.9	0.8	0.99622	0.9	0.8	0.996158
0.66121	C CANNY	18	1.8	-0.1	0.660028	-0.6	2	0.66121	2	1.1	0.65865	3	0.4	0.65867	0.5	1.7	0.65937	0.9	1.1	0.658868
0.99345	DERIV	19	1	2.6	0.988987	2.6	1	0.98908	5	1.7	0.98725	5	0.5	0.98725	0.9	0.7	0.99345	0.9	0.9	0.993125
0.98897	DERIV	20	0.6	2.5	0.987272	2.6	0.6	0.98705	5	2.7	0.98663	4	2.5	0.98635	0.9	2.2	0.98897	0.9	0.8	0.987928
0.93155	C DERIV	21	0.6	2.5	0.89447	2.5	0.6	0.89412	4	1.8	0.90237	5	2.5	0.90217	0.9	0.7	0.93066	0.9	0.9	0.931554
0.99519	DERIV	22	2	2.9	0.99464	3	2	0.99465	3	3	0.99454	5	3	0.99455	0.7	0.8	0.99519	0.9	0.8	0.995029
0.95535	DERIV	23	0.6	2	0.952813	1.6	0.8	0.95322	5	-0.2	0.9535	5	-0.3	0.95428	0.9	0.6	0.95535	0.9	1.2	0.95352
0.96229	DERIV	24	1	2.4	0.943228	2.1	0.8	0.94643	5	1.3	0.93399	5	1.3	0.93339	0.9	0.7	0.96229	0.9	0.9	0.956685
0.9666	DERIV	25	0.8	0.9	0.945886	1.5	1	0.94953	4	-0.2	0.94889	3	-0.8	0.94963	0.7	0.8	0.9666	0.9	0.8	0.96582
0.99838	DERIV	26	1.2	2.4	0.962393	2.2	0.8	0.95814	4	3	0.99647	5	1.7	0.99718	0.9	2.2	0.99838	0.7	0.9	0.997984
0.99589	C DERIV	27	0.6	2.6	0.99558	2.3	0.8	0.9956	5	2.8	0.99508	4	3	0.99511	0.7	0.9	0.99585	0.9	0.9	0.995889
0.993	DERIV	28	0.4	0	0.67918	-0.5	0.4	0.6919	2	1.2	0.70048	3	0.7	0.70293	0.9	0.8	0.993	0.9	1.2	0.982422
0.96974	DERIV	29	0.6	1.9	0.924492	1.8	0.6	0.92189	2	0.8	0.88711	1	-2	0.8888	0.9	0.7	0.96974	0.9	0.8	0.964703
0.98113	DERIV	30	0.6	1.7	0.979597	1.5	0.8	0.97949	5	3	0.97715	5	2.7	0.97739	0.9	0.7	0.98113	0.9	0.8	0.978903
0.99184	C CANNY	31	1	2.4	0.991812	2.4	1	0.99184	3	3	0.99147	4	3	0.99151	0.9	2.2	0.99141	0.7	1	0.991331
0.9768	C LoG	32	1.8	0	0.976668	0	2	0.97665	5	-1.9	0.97606	5	-2	0.97608	0.9	0.5	0.97625	0.7	0.8	0.976223
0.96218	C CRONE	33	2	0	0.960637	1.6	1.6	0.96049	3	0.5	0.96203	3	0.7	0.96218	0.1	1.8	0.96177	0.7	2	0.961454
0.99622	CANNY	34	1.8	2.8	0.996216	2.2	1.2	0.99616	4	3	0.99577	2	2	0.99556	0.9	2.2	0.99594	0.9	2	0.995784
0.97056	C LoG	35	0.6	2.6	0.964984	2.6	0.6	0.96698	5	3	0.95687	5	2.8	0.95883	0.7	1.7	0.96755	0.9	1	0.962181
0.96267	C CANNY	36	1.8	-0.3	0.962514	-0.6	2	0.96267	3	0.7	0.96052	2	-0.2	0.96096	0.9	0.7	0.96221	0.9	1.1	0.961512
0.96583	C LoG	37	0.8	3	0.951617	2.4	1	0.95592	5	-0.4	0.94091	5	-0.5	0.9411	0.9	0.7	0.9613	0.9	0.9	0.956112
0.97279	C LoG	38	0.6	2.4	0.735648	2.4	0.6	0.96322	2	3	0.96115	5	3	0.95986	0.9	2.3	0.9625	0.9	2	0.965964
0.97658	C LoG	39	0.6	1.5	0.949678	1.6	0.6	0.96864	1	-2	0.96934	1	-2	0.9715	0.9	0.6	0.96732	0.9	2	0.970437
0.99797	C LoG	40	0.6	1	0.885544	-2	0.2	0.97672	3	2.5	0.99765	2	2	0.99765	0.3	1.9	0.9977	0.5	1	0.997653
0.92839	C CRONE	41	1.4	0.1	0.926327	0.2	1.6	0.92733	5	-1	0.9282	5	-1.6	0.92839	0.9	0.7	0.92501	0.9	0.8	0.921079
0.98291	C LoG	42	0.6	2.9	0.934707	2.9	0.6	0.96478	3	2.9	0.96916	3	2.9	0.96989	0.7	0.8	0.97273	0.7	0.9	0.972147
0.8481	DERIV	43	1	2.6	0.78843	2.6	1	0.83755	3	2.5	0.84592	5	1.3	0.84565	0.9	2.3	0.8481	0.9	1.4	0.847281
			Mean:		**0.9342**			**0.9448**			**0.9447**			**0.9461**			**0.9623**			**0.9596**

Table C.1: Best absolute performances for all images using each detector(1)

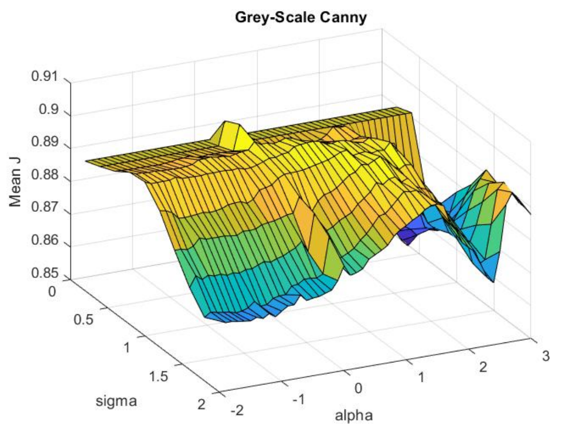
Best Results			LoG			C LoG			Roberts			C Roberts			Sobel			C Sobel		
J	**Method**	**N**	**th**	α	**J**	**th**	α	**J**	**th**	α	**J**	**th**	α	**J**	**th**	α	**J**	**th**	α	**J**
0.9933	Roberts	1	0.1	-0.3	0.992848	0.3	-1.4	0.99281	0.3	2.9	0.9933	0.3	1.8	0.99319	0.1	1	0.99298	0.3	2.7	0.993167
0.94975	DERIV	2	0.7	-2	0.949753	0.3	-1	0.94962	0.3	1.9	0.94959	0.5	1.7	0.94948	0.1	1	0.9491	0.5	0.2	0.949177
0.97944	C DERIV	3	0.1	-1.2	0.979436	0.1	-1.2	0.97941	0.1	-0.1	0.97938	0.3	-1.1	0.97941	0.9	0.6	0.97925	0.3	-1.9	0.979315
0.97678	DERIV	4	0.1	-2	0.97678	0.1	1.9	0.97668	0.1	-1.8	0.97449	0.1	-0.9	0.97612	0.9	0.5	0.97648	0.1	-1.2	0.975353
0.98494	C LoG	5	0.1	2.7	0.984926	0.1	2.1	0.98494	0.1	-1.9	0.98369	0.1	-1.7	0.98463	0.7	0.3	0.98465	0.1	-0.7	0.98458
0.98666	DERIV	6	0.1	-1.6	0.986379	0.1	-1.3	0.98541	0.1	-1.6	0.98666	0.1	-0.4	0.98652	0.9	0.9	0.98589	0.3	-0.9	0.982622
0.98135	DERIV	7	0.1	-1.5	0.97956	0.1	-1.2	0.98135	0.1	-1.1	0.97979	0.1	-0.2	0.97994	0.9	0.6	0.97971	0.1	-0.7	0.975656
0.98761	DERIV	8	0.1	-1.9	0.986339	0.1	-1.6	0.98624	0.1	-1.9	0.98575	0.1	-0.9	0.98694	0.9	0	0.98761	0.1	0.8	0.985377
0.93759	DERIV	9	0.1	-1	0.935719	0.3	-2	0.93727	0.1	-0.3	0.93178	0.3	-1.3	0.93482	0.9	0.7	0.78329	0.3	-0.6	0.937591
0.81728	DERIV	10	0.1	2.2	0.817285	0.1	-1.6	0.81722	0.1	-1.5	0.81718	0.1	-0.4	0.81703	0.9	0.6	0.81715	0.1	0.9	0.816872
0.97653	C LoG	11	0.1	-0.8	0.976293	0.3	-1.5	0.97653	0.1	0.6	0.97404	0.3	2.6	0.97363	0.9	0.9	0.90026	0.3	3	0.976014
0.9542	DERIV	12	0.3	-2	0.947963	0.1	-0.4	0.9542	0.1	0.8	0.92449	0.3	2.5	0.93709	0.9	0.7	0.56203	0.3	2.9	0.882684
0.97181	DERIV	13	0.1	3	0.971732	0.1	-2	0.97181	0.1	-1.9	0.97014	0.1	-1.2	0.97141	0.5	0.1	0.97145	0.1	-2	0.971239
0.9604	C CANNY	14	0.1	0.1	0.960397	0.5	-1.9	0.91555	0.3	-1.7	0.91948	0.3	2.3	0.95105	0.1	-1.7	0.72217	0.3	-1.3	0.958023
0.98799	C LoG	15	0.1	2.5	0.98734	0.1	1.6	0.98799	0.1	-1.9	0.97729	0.1	-0.8	0.98673	0.9	1.5	0.98644	0.1	0.3	0.983904
0.99343	C DERIV	16	0.1	3	0.9794	0.1	0.3	0.99001	0.1	2.2	0.98618	0.1	1	0.98807	0.7	0.5	0.97972	0.3	-1.3	0.99343
0.99425	DERIV	17	0.1	2.9	0.993883	0.1	2.9	0.99389	0.3	2.4	0.99421	0.5	2.2	0.99421	0.1	-2	0.99425	0.1	-0.7	0.994237
0.65942	C CANNY	18	0.3	3	0.659412	0.1	0.1	0.65942	0.1	-0.3	0.65882	0.3	-1.3	0.65873	0.9	0.6	0.65836	0.3	0.3	0.659263
0.99041	DERIV	19	0.3	-1.5	0.989988	0.3	-1.3	0.99012	0.3	-1.2	0.98971	0.3	2.2	0.99041	0.1	1	0.98587	0.3	1.2	0.988955
0.98787	DERIV	20	0.1	1.6	0.987663	0.1	0.8	0.98787	0.1	2.8	0.98659	0.1	1.4	0.98664	0.9	0.5	0.98641	0.3	1.8	0.987152
0.90805	C DERIV	21	0.1	1.8	0.907288	0.1	1.1	0.90805	0.1	-1.9	0.9021	0.1	2.8	0.90192	0.9	0.5	0.90387	0.1	-0.3	0.901552
0.99484	DERIV	22	0.1	-1.9	0.994835	0.1	-1.5	0.99484	0.1	-1.4	0.99444	0.1	-1.9	0.99451	0.9	2	0.99452	0.1	-0.5	0.994439
0.95408	DERIV	23	0.1	-1.2	0.954069	0.1	-0.9	0.95408	0.1	-0.9	0.95332	0.1	1.1	0.95327	0.9	0.6	0.95254	0.3	0.1	0.952506
0.94974	DERIV	24	0.3	-1.6	0.942415	0.3	-1.4	0.94547	0.1	0.8	0.94235	0.3	2.3	0.94442	0.1	1	0.8994	0.3	2.3	0.949742
0.9465	DERIV	25	0.1	-1.1	0.853231	0.1	0.3	0.87293	0.1	2.8	0.92975	0.1	1.7	0.93529	0.9	0	0.86	0.3	-0.6	0.946502
0.99708	DERIV	26	0.1	2.6	0.996542	0.1	-1.7	0.99687	0.1	-1.9	0.99557	0.1	-1.2	0.99674	0.7	0.2	0.99708	0.1	-2	0.99155
0.99587	C DERIV	27	0.1	-1.2	0.995702	0.3	2.8	0.99587	0.1	2.4	0.99517	0.1	0.6	0.99525	0.9	1.4	0.99476	0.3	3	0.995491
0.94516	DERIV	28	0.1	3	0.912864	0.1	2.3	0.94516	0.1	-1.9	0.68796	0.1	-1.8	0.86253	0.9	2.7	0.90104	0.1	-1.3	0.701257
0.95314	DERIV	29	0.1	2.4	0.94563	0.1	-1.6	0.94674	0.1	-1.4	0.95103	0.1	2.5	0.94678	0.9	0.6	0.95314	0.3	0.1	0.899154
0.97941	DERIV	30	0.1	-1.4	0.979405	0.1	-0.9	0.97915	0.1	-0.7	0.97827	0.1	1.1	0.97828	0.9	0.5	0.9761	0.3	0.3	0.978636
0.99179	C CANNY	31	0.1	-0.9	0.991785	0.3	-2	0.99169	0.1	1.7	0.99145	0.3	3	0.99141	0.9	1.4	0.99095	0.3	1.2	0.991521
0.9768	C LoG	32	0.1	3	0.97675	0.1	2.8	0.9768	0.1	-1.8	0.97589	0.1	-1.1	0.976	0.9	-0.8	0.97615	0.1	-0.8	0.975918
0.96189	C CRONE	33	0.1	-1.8	0.961738	0.1	-1.4	0.96176	0.1	-1.8	0.96189	0.1	2.2	0.9618	0.9	0.8	0.96093	0.3	-0.8	0.959842
0.99609	CANNY	34	0.1	-1.3	0.996091	0.1	-0.9	0.99597	0.1	2.3	0.99585	0.1	0.5	0.99583	0.9	0.2	0.99401	0.3	0.2	0.995846
0.97056	C LoG	35	0.3	1.8	0.96694	0.3	1.7	0.97056	0.3	2.8	0.96088	0.3	1.5	0.96244	0.1	1	0.95594	0.3	3	0.965878
0.9623	C CANNY	36	0.1	-0.5	0.962125	0.3	-1.7	0.96194	0.3	-1.7	0.9621	0.3	2.2	0.96213	0.9	0	0.95581	0.1	1	0.9623
0.96583	C LoG	37	0.1	0.6	0.951833	0.3	-1.6	0.96583	0.1	-0.1	0.94662	0.3	2.7	0.95518	0.9	0.7	0.92989	0.3	3	0.955992
0.97279	C LoG	38	0.1	1.8	0.96367	0.3	2.1	0.97279	0.1	-1.7	0.96134	0.1	1.7	0.96172	0.9	0.5	0.96002	0.3	2	0.962201
0.97658	C LoG	39	0.1	3	0.955954	0.3	2.2	0.97658	0.1	-1.3	0.96304	0.1	1.9	0.97136	0.9	-0.2	0.96656	0.3	0	0.964573
0.99797	C LoG	40	0.1	2.4	0.997904	0.1	-1.5	0.99797	0.1	-1.9	0.99365	0.1	2.4	0.99761	0.9	1.7	0.99766	0.1	-0.3	0.994988
0.92331	C CRONE	41	0.1	0.4	0.920258	0.1	-0.1	0.92192	0.1	1.3	0.92195	0.3	2.7	0.92232	0.9	0.3	0.79179	0.3	-0.9	0.923309
0.98291	C LoG	42	0.1	2.5	0.96862	0.3	2.7	0.98291	0.1	-1.8	0.96548	0.1	-0.7	0.96933	0.7	1.1	0.96914	0.3	2.4	0.96563
0.84615	DERIV	43	0.1	2	0.845904	0.3	2.9	0.84615	0.1	-1.9	0.845	0.1	-0.9	0.84523	0.9	0.5	0.84594	0.3	3	0.841037
					0.9531			**0.9554**			**0.9474**			**0.9537**			**0.9281**			**0.9475**

Table C.2: Best absolute performances for all images using each detector(2)

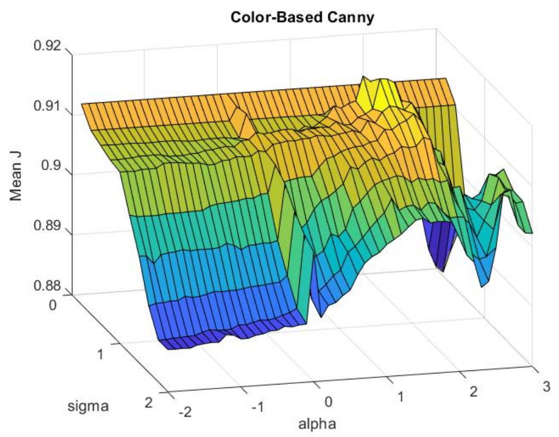
Appendix D

Parameter Fixation Mean Performance graphs

In this appendix, the mean performance of all detectors obtained by fixing all combinations of parameters are extensively presented under the form of 3D plots.

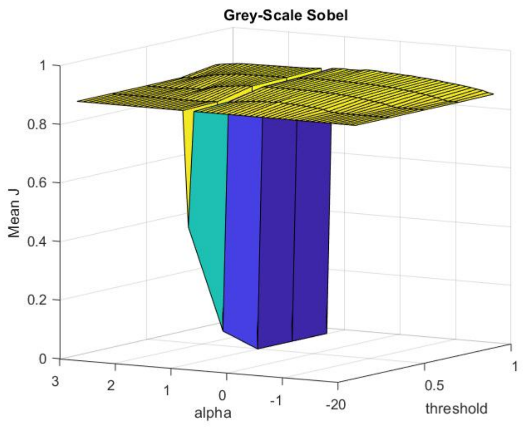


(a) Grey-Scale Canny

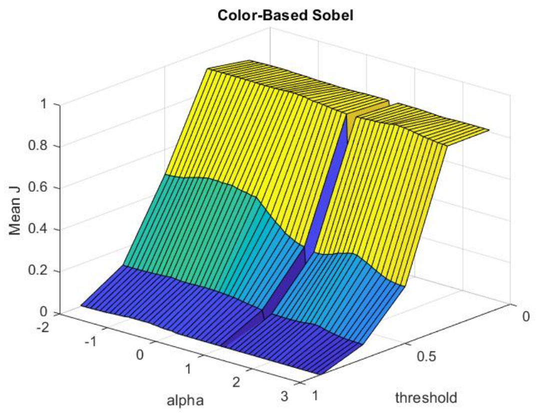


(b) Color-Based Canny

Figure D.1: Mean Performance of Canny algorithms graphs using same parameters for all images.

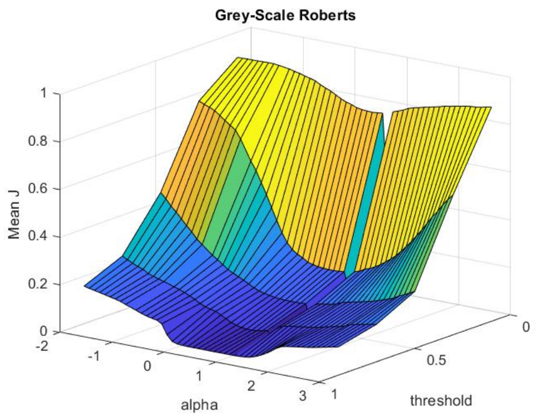


(a) Grey-Scale Sobel

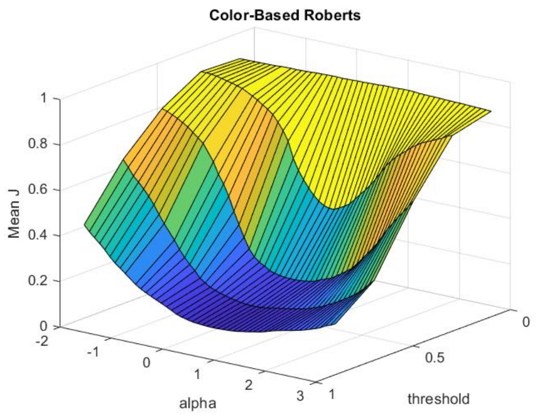


(b) Color-Based Sobel

Figure D.2: Mean Performance of Sobel algorithms graphs using same parameters for all images.

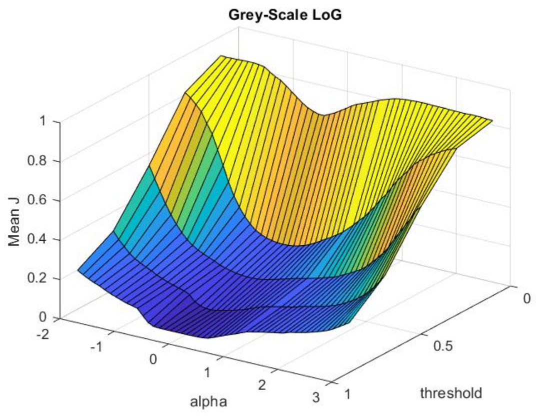


(a) Grey-Scale Roberts

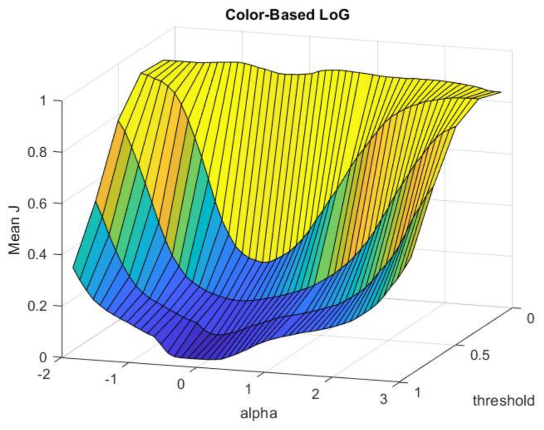


(b) Color-Based Roberts

Figure D.3: Mean Performance of Roberts algorithms graphs using same parameters for all images.

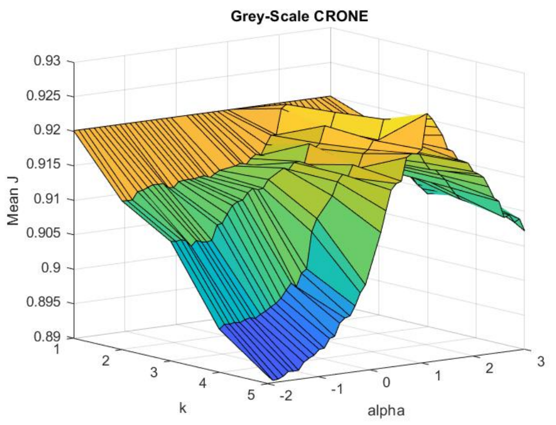


(a) Grey-Scale LoG

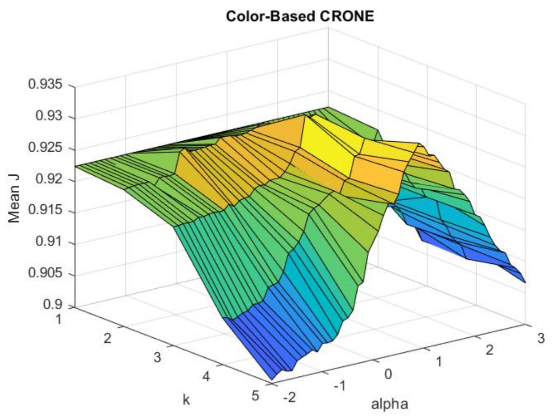


(b) Color-Based LoG

Figure D.4: Mean Performance of LoG algorithms graphs using same parameters for all images.

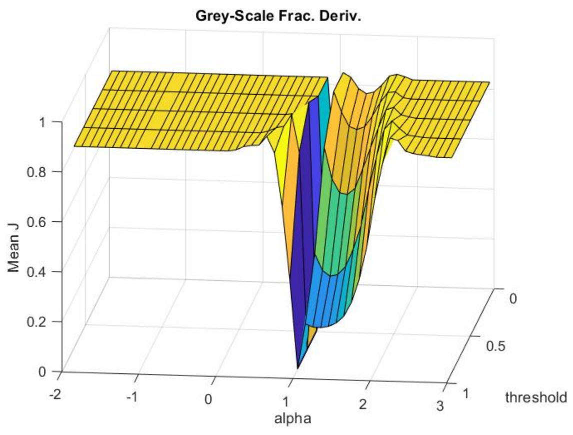


(a) Grey-Scale CRONE

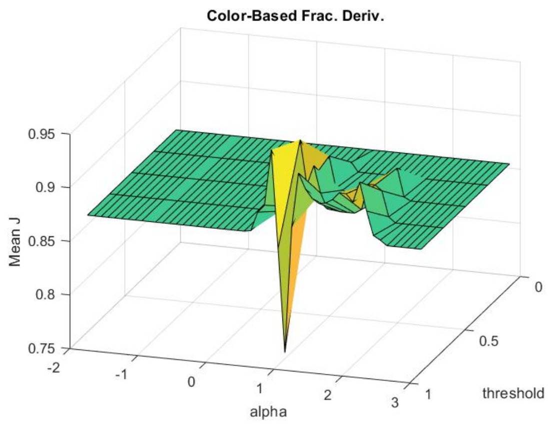


(b) Color-Based CRONE

Figure D.5: Mean Performance of CRONE algorithms graphs using same parameters for all images.



(a) Grey-Scale Frac. Deriv. Op.



(b) Color-Based Frac.Deriv. Op.

Figure D.6: Mean Performance of Fractional Derivative Mask algorithms graphs using same parameters for all images.