Pontificia Universidad Javeriana Cali
Faculty of Engineering
Department of Electronics and Computer Sciences
Electronic Engineering
Undergraduate research proposal

GRAPH LEARNING APPROACH BASED ON SIGNAL REPRESENTATION METHODS FOR MULTISPECTRAL IMAGE FUSION AND CHANGE DETECTION

Juan Carlos Alvear Muñoz
David Alfredo Quintero Olaya

Supervised by: Ph.D. Hernán Darío Benítez Restrepo
M.Sc. David Alejandro Jimenez Sierra

January 29th, 2021

Santiago de Cali, Junio 22, 2020.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Eugenio Tamura Morimitsu
Director Carrera de Ingeniería Electrónica.
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que los estudiantes de Ingeniería Electrónica Juan Carlos Alvear Muñoz (cod: 8912240) y David Alfredo Quintero Olaya (cod: 8912509) trabajan bajo mi dirección en el proyecto de grado titulado "GRAPH LEARNING APPROACH BASED ON SIGNAL REPRESENTATION METHODS FOR MULTISPECTRAL IMAGE FUSION AND CHANGE DETECTION".

Atentamente,

Hernán Benítez

_____
Dr. Hernán Darío Benítez Restrepo

Santiago de Cali, Junio 22, 2020.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Eugenio Tamura Morimitsu
Director Carrera de Ingeniería Electrónica.
Cali.

Cordial Saludo.

Nos permitimos presentar a su consideración el anteproyecto de grado titulado "GRAPH LEARNING APPROACH BASED ON SIGNAL REPRESENTATION METHODS FOR MULTISPECTRAL IMAGE FUSION AND CHANGE DETECTION" con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero Electrónico.

Al firmar aquí, damos fe que entendemos y conocemos las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,

Juan Carlos Alvear M.

Juan Carlos Alvear Muñoz
Código: 8912240

David Alfredo Quintero O.

David Alfredo Quintero Olaya
Código: 8912509

Abstract

Graphs are a mathematical tool widely used to represent several types of data involved in a determined process and the existent relations between them capturing the structure and behavior of the process. They are used in applications such as the modeling of disease spreading in a population, financial transactions, sensor networks, and remote sensing. One of the novel applications that are in continuous development in remote sensing, specifically in change detection is the state-of-the-art graph-based fusion for change detection (**GBF-CD**) model, in which the authors use graphs to represent the pre- and post-event images. This model uses a Gaussian kernel in the graph learning stage, which overlooks specific underlying data structure. Graph learning models are based on statistics, physics, or signal representation. In this research, we use models based on signal representation, namely signal smoothness and spectral filtering, for the **GBF-CD** graph learning stage in order to improve the model performance in metrics such as: missed alarms, false alarms, precision, recall, Cohen's kappa, overall error, and execution time. In addition to modifying the graph learning stage, in this new model, we apply graph cut segmentation instead of Nyström extension to reduce computational complexity. We carry out tests on 14 real cases datasets of multispectral images including some multimodal acquisitions of natural disasters. Our proposed model outperforms the **GBF-CD** model performance in 9 out of 14 datasets. An implementation of our model can be found at https://github.com/quintero98-hub/Smoothness-Signal-and-Spectral-Filtering-Graph-Learning-Approaches-for-Change-Detection-Algorithm.git.

Contents

Abstract v

Abbreviations xiii

Acknowledgment xv

1 Introduction 1

2 State of the art approaches 3

3 Theoretical background 5
- 3.1 Graph Theory . . . 5
  - 3.1.1 Graph Definition . . . 5
  - 3.1.2 Graph Spectrum . . . 6
- 3.2 Graph Learning Approaches . . . 6
  - 3.2.1 Statistical Models . . . 6
  - 3.2.2 Physically Motivated Models . . . 8
  - 3.2.3 Signal Representation Perspective . . . 8
- 3.3 Confusion Matrix . . . 10
  - 3.3.1 Definition . . . 10
  - 3.3.2 Metrics . . . 11
- 3.4 Hypotheses . . . 11
- 3.5 Test datasets description . . . 12
- 3.6 GBF-CD Algorithm . . . 18

4 Signal smoothness 21
- 4.1 Introduction . . . 21
- 4.2 Graph Laplacian Learning Algorithm . . . 21
- 4.3 Large Scale Graph Learning from Smooth Signals . . . 23
- 4.4 Experiments . . . 24
  - 4.4.1 Experimental settings . . . 25
  - 4.4.2 Results on synthetic data . . . 25
  - 4.4.3 Smoothness Approach in Change Detection . . . 26

5 Spectral filtering 31
- 5.1 Introduction . . . 31
- 5.2 Models based on spectral filtering of graph signals . . . 31
  - 5.2.1 Network topology inference from spectral templates . . . 31

5.2.2 Inference of graph diffusion processes from observations of stationary signals . 33
5.2.3 Bilateral Filter 33
5.2.4 Non-negative kernel regression (NNK) graph 35
5.3 Spectral Filtering in Change Detection 37

6 Results and discussion **39**

7 Conclusions **51**

Bibliography **53**

List of Figures

3.1 Undirected graph example 5
3.2 Sardania. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 12
3.3 Omodeo. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 12
3.4 Alaska. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 13
3.5 Madeirinha. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 13
3.6 Katios. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 13
3.7 Dique. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 13
3.8 San Francisco. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 14
3.9 Wenchuan. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 14
3.10 Canada. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 14
3.11 California. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 15
3.12 Contest. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 15
3.13 Toulouse. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 16
3.14 Bastrop. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 16
3.15 Gloucester. From left to right to the pre-event image, the post-event image and the change map (groundtruth) 16

4.1 The learned graph Laplacian matrices. The columns from the left to the right are the groundtruth Laplacian, the Laplacian learned by GL-SigRep (Dong's algorithm [1]) and the Laplacian learned by GSP toolbox (Kalofolias's algorithm [6]). 26
4.2 Graph-cut applied to Sardania dataset: From left to right, pre-event image, post-event image and Graph-cut applied with 500 classes. 29

6.1 Sardania Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. 39

6.2 Omodeo Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 39
6.3 Alaska Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 40
6.4 Madeirinha Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 40
6.5 Katios Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 40
6.6 Dique Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 40
6.7 San Francisco Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 41
6.8 Wenchuan Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 41
6.9 Canada Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 41
6.10 California Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 42
6.11 Contest Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 42
6.12 Toulouse Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 43
6.13 Bastrop Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 43
6.14 Gloucester Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods. . . . 43
6.15 Metrics of the three methods for the 14 datasets 48
6.16 Model's performance - Bar charts that evaluate the performance of each method over all metrics and datasets. The count for each method in one of the six possible metrics means that in one dataset, the model outperformed all the competing methods in that metric. . . . 49

List of Tables

3.1 Confusion Matrix 11
3.2 Databases used to evaluate the performance of the proposed method. 17

4.1 Graph learninng performance for **GL-SigRep** and **GL-LogDeg**. 26

6.1 Server specifications 44
6.2 Models performance for dataset Sardania 44
6.3 Models performance for dataset Omodeo 45
6.4 Models performance for dataset Alaska 45
6.5 Models performance for dataset Madeirinha 45
6.6 Models performance for dataset Katios 45
6.7 Models performance for dataset Dique 45
6.8 Models performance for dataset San Francisco 45
6.9 Models performance for dataset Wenchuan 46
6.10 Models performance for dataset Canada 46
6.11 Models performance for dataset California 46
6.12 Models performance for dataset Contest 46
6.13 Models performance for dataset Toulouse 46
6.14 Models performance for dataset Bastrop 46
6.15 Models performance for dataset Gloucester 47

Abbreviations

CD	Change Detection
CGP	Causal Graph Processes
DSPG	Discrete Signal Processing on Graphs
ET	Execution Time
FA	False Alarms
GBF	Graph Based Fusion
GL	Graph Learning
GSP	Graph Signal Processing Toolbox
K	Cohen's Kappa
KI	Kittler-Illingsworth min. error thresholding
LogDeg	Logarithmic Degree
MA	Missed Alarms
MRF	Markov Random Field
MS	Multi-Spectral
NNK	Non-Negative Kernel Regression
OE	Overall Error
P	Precision
PCA	Principal Component Analysis
R	Recall
rR	Rayleigh-Rice method for CD
rrR	Rayleigh-Rayleigh-Rice method for CD
SEG	Segmentation
SigRep	Signal Representation
U-CD-HPT	Unsupervised Image Regression for Heterogeneous Change Detection

Acknowledgment

We would like to express our gratitude to our families for their continuous support, and also to Hernán Benítez and David Jimenez, our advisors, whose observations helped us to improve the quality of our work. Moreover, we would like to thank Xiaowen Dong [1], Bastien Pasdeloup [2], Santiago Segarra [3], Sarath Shekkizhar [4], and Shuiping Gou [5] for kindly providing their codes and datasets to recreate their results and incorporate them to our algorithm, as well as Kalofolias [6] for sharing the GSP toolbox [7] to the public.

This work was funded by the OMICAS program: "Optimización Multiescala In-silico de Cultivos Agrícolas Sostenibles (Infraestructura y validación en Arroz y Caña de Azúcar)", anchored at the Pontificia Universidad Javeriana in Cali and funded within the Colombian Scientific Ecosystem by The World Bank, the Colombian Ministry of Science, Technology and Innovation, the Colombian Ministry of Education, the Colombian Ministry of Industry and Tourism, and ICETEX, under grant ID: FP44842-217-2018 and OMICAS Award ID: 792-61187.

Chapter 1

Introduction

As a consequence of population growth and efforts to supply its needs, environmental disasters have increased exponentially. It is well known that natural catastrophes have been occurring in response to the planet's excessive exploitation. Such catastrophes have been seen more frequently in the last decade, such as the earthquakes in Haiti, Japan and Chile, the tropical storm "Washi" and "Typhoon Bopha" in the Philippines, the monsoon rains in India, the avalanche of mud in Afghanistan, earthquakes in Nepal and Ecuador, drought in Somalia, the United States and Mexico, Hurricane Dorian in the Bahamas, the Australian bushfires and Amazon rainforest fires. Such natural disasters have caused the death of over 600 thousand people around the world in the past 10 years, as well as irreversible damage on the planet's vegetation [8]. The damage could have been less if the response capacity of the risk management entities had been higher and more effective.

In order to tackle that, change detection (CD) can be applied. Change detection on the context of Remote Sensing embraces the task of analyzing two images of a same land area, typically a pre- and a post-event images captured through a satellite, i.e. in two different time instances (multitemporal), with the aim of detecting patches where the land-cover type has changed between the acquisitions. The modern satellites have the capacity of capturing multispectral images, that is, images in different frequency bands, from which normally the Red and the Near-Infrarred (NIR) bands are the most successful for identifying events such as fires and floods. However, satellite images are affected by factors such as noise, misalignments between samples, variations in luminosity, among others. Therefore, the images need to be coregistered before carrying out a comparison between them. Furthermore, the tremendous computational capacity that multispectral image processing demands (due to the image size), makes developing new tools, that prevent and determine the impact of catastrophes, a real challenge.

There are different state of the art methods for applying change detection such as Rayleigh-Rice (**rR**) [9], Rayleigh-Rayleigh-Rice (**rrR**) [10], Unsupervised Image Regression for Heterogeneous Change Detection (**U-CD-HPT**) [11] and the classic KittlerIllingsworth (**KI**) [12]. The graph-based fusion for change detection (**GBF-CD**) method [13] is currently under development and aims to outperform all of the above approaches. One of the main advantages of using graphs is that it provides spatial awareness, which allows structuring complex data. Furthermore, the use of graph-based methods and other mathematical techniques may enable dimensionality reduction of the problem and thereby decreases the computational complexity required for this type of analysis with techniques such as Nyström's extension (used in the **GBF-CD** method).

Jimenez et al. [13] use graphs to represent the pre- and post-event images. This model uses a Gaussian kernel in the graph learning stage. Nevertheless, this overlooks explicit priors or models of the data. Hence, the results can be sensitive to noise and its hyperparameters can be difficult to

tune. Thereby, it is necessary to explore graph learning methods that embeds explicit priors or data models into learning graph models. There are three different approaches or meaningful data models to learn graphs, such as statistical models, physically motivated models, and signal representation perspectives [14]. The objective of this work is to apply and evaluate graph learning models based on signal representation applied to the **GBF-CD** method.

Models based on signal representation perspectives enable to generalize classical signal processing concepts, tools, and methods, such as time-frequency analysis and filtering, on graphs. These models include signal smoothness [1] and spectral filtering [2]. Therefore, the application of these models is expected to improve the performance of the **GBF-CD** method in terms of missed alarms, false alarms, precision, recall, Cohen's kappa, and overall error. The research question which motivates this work is: How to apply graph learning approaches based on signal representation to the **GBF-CD** method?

This document is organized as follows. We provide a brief overview of some state of the art approaches in chapter 2 and then, in chapter 3, we introduce some basic concepts concerning the basis of graph learning theory and present a general explanation of the **GBF-CD** method. Next, we present the Smoothness and Spectral Filtering Graph learning approaches in chapter 4 and chapter 5, respectively. Then, we expound the results and discussion in chapter 6. Finally, we set out the conclusions in chapter 7.

Chapter 2

State of the art approaches

Many challenges in the research field of Remote Sensing area have risen in recent years. Some of them are presented in [15]. This paper explains "perspectives for data fusion in remote sensing leveraging results from Data Fusion Contests issued from 2006 to 2014 organized by the IEEE Geoscience and Remote Sensing Society".

Remote sensing is used to extract information about the earth's surface, such as the spatial distribution of objects in a region, their height and size, identification of materials, etc. There are different types of sensors for each task: optical sensors provide information about the structure, LiDAR about elevation, and multispectral about material content. These acquisitions can be coupled and analyzed by data fusion to obtain a better representation of the scenario.

Data Fusion approaches are normally classified into three groups according to the processing level where it takes place: raw data level, feature level, and decision level. Some data fusion problems in remote sensing are pansharpening, change detection, classification, and some miscellaneous applications.

In [13], Jimenez et al. present state-of-art techniques used to detect changes in Multi-Spectral (MS) images such as probabilistic thresholding and machine learning. Probabilistic thresholding is sensitive to MS image noise. For that reason, it requires a high precision to compute the difference image probabilistic distribution. Similarly, machine learning techniques are divided into classification and clustering, where the former requires a multi-temporal reference from the raw-data, which is not practical. Clustering needs a parameter initialization which can lead to local-minima in the learning stage, which is also undesirable. Jimenez et al's method, the Graph-Based Fusion for Change Detection method (**GBF-CD**), plays an important role because it reduces the effect of the small variability and artifacts present in MS images, where artifacts should be understood as errors that occur due to sensor's imperfections, such as brightness in certain zones of the image and presence of electronic noise in a completely dark scene.

Our project proposes a graph-based approach to change detection. Stankovic et. al [16] refers to graphs as irregular structures that represent multifaceted data attributes, and highlight that the main difference between classical signal processing and graph signal processing is that the last incorporates spatial sensing, awareness, physical intuition, and sensor importance. These attributes are important for our project, taking into consideration that spatial awareness is a fundamental factor in change detection using multispectral images.

Applications that include data analysis are growing exponentially together with the development of new structures that increase the complexity of the information, it has converged on the need for a shift in thinking and new analytical frameworks, for example, the inclusion of graph theory for data analysis and representation [16]. To highlight the importance of spatial awareness that these

graph models include, the authors of [16] show a meaningful and intuitive real-world example of geographically distributed estimation of multi-sensor temperature measurements, proving that graph application can reduce noise effect caused by sensor readings or faulty sensor activity, obtaining an improvement of 6 dB over the original signal to noise ratio.

Jimenez et al. [13] present a data-driven approach for change detection, more concretely a data-driven framework of graph-based data fusion. The author employs the Landsat-8 satellite images [17] as a data source for the algorithm's validation.

The steps of the method pointed out on Jimenez's article are: (i) The generation of a multi-temporal pixel-based graph, by the fusion of intra-graphs (normalized graph Laplacian) of each temporal data; (ii) the use of Nyström extension to extract the eigenvectors and eigenvalues of the fused graph, computing the mutual information (MI) between the given and the prior image and the selection of the final change map by the identification of the relevant eigenvector and its respective eigenvalue that captures the global change (maximize MI).

Jimenez's Method outperforms the state-of-art methods such as Rayleigh-Rice (rR) [9], Rayleigh-Rayleigh-Rice (rrR) [10], and the classic KittlerIllingsworth (KI) [12]. The indicators used in the evaluation were: missed alarms (MA), false alarms (FA), precision (P), recall (R), Cohen's kappa (K), and overall error (OE).

The GBF-CD method is the reference technique for the research problem addressed in this document. Therefore, it was taken as a reference to do the Systematic Literature Research (SLR) [18]. The aim of this research is to apply different signal representation models on this method (modifying the estimation of the Laplacian matrix) and to evaluate its performance using the same indicators which have been used for the current algorithm's version, and also considering the execution time and the computational resources usage.

There are two different ways to represent a signal as a graph. One of them is a natural choice of the graph structure. However, sometimes it is not possible due to data complexity or other factors that do not allow a clear and simple structuring. In this case, it is necessary to infer or learn a graph topology from the data. In [14] Dong et al. present solutions to the problem of graph learning (including classical viewpoints such as statistics and physics). The authors also present a signal representation perspective, including signal smoothness, spectral filtering of graph signals, and causal dependencies on graphs.

Chapter 3

Theoretical background

3.1 Graph Theory

3.1.1 Graph Definition

A graph is a structure built by a set of points, called vertices ($\boldsymbol{\mathcal{V}}$), and lines between those points, called edges ($\boldsymbol{\mathcal{E}}$).Thus, it is fully specified by the set of its vertices and their connectivity scheme (designated by edges). The edges may be defined by the adjacency matrix, $\boldsymbol{A}$, with $\boldsymbol{A_{mn}} \in [0, 1]$ (set not interval) for unweighted graphs.

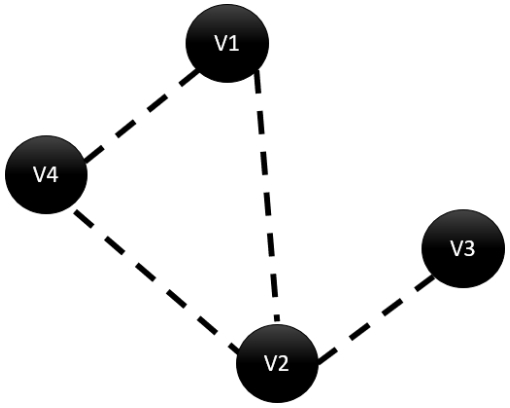


Figure 3.1: Undirected graph example

The undirected and unweighted graph shown in Figure 3.1 can be represented as an Adjacency Matrix ($\boldsymbol{A}$).

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \tag{3.1}$$

Where the value of "1" establishes that there is a connection between the two analyzed vertices and the value of "0" that there is not. For example, the first row of the matrix $\boldsymbol{A}$, shows that (V1,V2) and (V1,V4) are connected. The graph can also be defined by the "connectivity strength" matrix, referred to as the weight matrix, $\boldsymbol{W}$, with $w_{i,j} \in \mathbb{R}$ normally between 0 and 1, for weighted graphs.

The difference between weighted and unweighted graphs is that the first has a weighting in each edge. Thus, a weighted matrix can contain values different from the binary set. In our project each multispectral image must be represented as a graph. It must be specified by a weight matrix, knowing that each node represents a pixel, and each edge the distance between pixels. In images, this distance could be geometric distance, radiometric distance, or their combination [19]. It is defined by:

$$W_{i,j} = \exp(-\frac{||l_i - l_j||_2^2}{\sigma_l^2}) \exp(-\frac{||x_i - x_j||_2^2}{\sigma_x^2}) \tag{3.2}$$

where l_i is the location of pixel i on the 2-D image grid, x_i is the intensity of pixel i, and σ_l^2 and σ_x^2 are two parameters, which represent the variance of location data and intensity data, respectively. Hence, $0 \leq w_{i,j} \leq 1$. Larger geometric and/or radiometric distances between pixels i and j would mean a smaller weight $w_{i,j}$.

3.1.2 Graph Spectrum

The Laplacian Matrix ($\boldsymbol{L}$) is defined as the difference between the degree matrix that contains the degrees of the vertices along the diagonal ($\boldsymbol{D}$) and the weighted adjacency matrix of the graph ($\boldsymbol{W}$). Thereby, the resulting matrix $\boldsymbol{L}$ is symmetric, and therefore it can be eigendecomposed into

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \tag{3.3}$$

Where $\boldsymbol{\Lambda}$ is a diagonal matrix containing real eigenvalues λ_k along the diagonal, and $\boldsymbol{U}$ is an eigenmatrix composed of orthogonal eigenvectors u_i as columns. The eigenvalues can be interpreted as graph frequencies, and eigenvectors interpreted as corresponding graph frequency components of the Fourier representation of the graph [19]. The frequency analysis is fundamental in the development of different graph learning models.

3.2 Graph Learning Approaches

There are some intuitive methods to infer a graph topology purely from the observed data samples such as sample correlation and similarity function (e.g. Gaussian radius basis function kernel). These approaches are sensitive to noise and it is laborious to tune their parameters. As previously stated, there are three different approaches or meaningful data models to learn from graphs, such as statistical models, physically motivated models, and signal representation perspectives whose outcome, a graph topology, reveals better the inherent relationship among the data entities.

We observe a data matrix $\boldsymbol{X}$ of observations of each variable and we want to learn the graph topology $\boldsymbol{\mathcal{G}}$ which is depicted by $\boldsymbol{X} \sim \mathcal{F}\{\boldsymbol{\mathcal{G}}\}$, where $\mathcal{F}$ is a model for the generative process.

3.2.1 Statistical Models

In this class of models, $\mathcal{F}$ represents a distribution determined by $\boldsymbol{\mathcal{G}}$ (e.g. probabilistic graphical models). On this approach, we learned a factorization of a joint probability distribution of the

random variables in which we are interested.

One approach is learning probabilistic graphical models such as:

- Markov random fields (MRF): Undirected graphical models. The graph topology captures a conditionally independent structure of the random variables the nodes represent [14].
- Bayesian Networks (BN): Directed graphical models. Allow the bidirectionality of the edges and it brings a more complex notion of conditional independence [14].

MRF are of great interest because its similarity with GSP models, which are explained below.

They represent a collection of random variables that satisfies Markov property. The pairwise Markov property means that if two nodes v_i and v_j in the graph do not have an edge between them, the random variables, x_i and x_j, which they represent, are conditionally independent given the rest. The pairwise Markov property is shown in Equation 3.4.

$$(v_i, v_j) \notin \mathcal{E} \Leftrightarrow p(x_i|x_j, \mathbf{x}\backslash\{x_i, x_j\}) = p(x_i|\mathbf{x}\backslash\{x_i, x_j\}) \tag{3.4}$$

This model can be generally represented by a family of exponential distributions which is written over a parameter matrix $\mathbf{\Theta}$, as shown in Equation 3.5.

$$p(x|\Theta) = \frac{1}{Z(\Theta)} exp\left(\sum_{v_i \in \mathcal{V}} \theta_{ii} x_i^2 + \sum_{(v_i, v_j) \in \mathcal{E}} \theta_{ij} x_i x_j\right) \tag{3.5}$$

Where $p(x|\mathbf{\Theta})$ represents the joint distribution of all the random variables and $Z(\mathbf{\Theta})$ is a normalization constant.

There are two types of MRF, one is for continuous variables which is called the Gaussian Markov random field. If it is discrete it is called a discrete MRF. For the Gaussian MRF the prior equation can be simplified as shown in Equation 3.6.

$$p(x|\Theta) = \frac{|\Theta|^{1/2}}{(2\pi)^{N/2}} exp\left(-\frac{1}{2} x^T \Theta x\right) \tag{3.6}$$

where the parameter matrix $\mathbf{\Theta}$ is the called precision matrix or inverse covariance matrix of the multi-variable Gaussian distribution.

In both cases (continuous and discrete cases), it is necessary to learn the parameter matrix $\mathbf{\Theta}$ that captures the conditional independence structure of the random variables. Usually we consider a sparse matrix $\mathbf{\Theta}$ for two reasons:

- Interactions in the real world are mostly local.
- The number of variables of matrix $\mathbf{\Theta}$ is N^2 where N is the number of nodes and sparse data will ease the computation.

Interested readers may refer to [14] for more information about the computation of matrix $\mathbf{\Theta}$.

3.2.2 Physically Motivated Models

In the physically motivated models the observations are considered outcomes of some physical phenomena on the graph specified by the function $\mathcal{F}(\mathcal{G})$. Therefore, the objective of these models is to infer the graph (structure inherent to the physics of the observed data) from the consequences of the phenomenon (samples). "One example of this kind of model is epidemic or information propagation models, where the physical process represents a disease spreading over a contact network or a meme spreading over social media" [14]. These models are most commonly used in telecommunications by "sending a sequence of packets from one source to many destinations, and sequences of received packets are used to infer the internal network topology". In [14] the authors show an example of the application of these models in the spread of an epidemic, where a matrix that represents the graph is inferred from different samples. In the learned graph the vertex represent the contagious ones and the direction of the edges represents a possible direction of infection of a vertex due to another one.

Methods for inferring networks from information cascades (vertices that are influenced by others) can be divided into two main categories depending on whether they are based on homogeneous or heterogeneous models. Methods based on homogeneous models assume that cascades propagate in a statistically identical manner across all edges while methods based on heterogeneous allow for cascades to propagate at different rates across different edges. These methods are explained in more detail in [14].

3.2.3 Signal Representation Perspective

This approach has emerged due to the need to generalize classical signal processing concepts, tools, and methods, such as time-frequency analysis and filtering, on graphs [14]. There are three different models of learning graphs with this approach: signal smoothness, spectral filtering, and causal dependencies.

The first model considered is a smoothness model, under which "the signal takes similar values at neighboring vertices" [14]. Practical examples of this model could be temperature sensed in different locations in a geographical region. The second graph signal model (Spectral filtering of graph signals) "goes beyond the global smoothness of the signal on the graph and focuses more on the general family of graph signals that are generated by applying a filtering operation to a latent (input) signal" [14]. The two models named above are designed for undirected graphs, while causal dependencies on graphs is a technique that has been designed for learning directed graphs. It consists of a "causal graph process based on the idea of sparse vector autoregressive (SVAR) estimation" [14]. Next we will explain in more detail each one of the methods.

3.2.3.1 Models based on signal smoothness

In [1] the authors explain in more detail the smoothness model. It casts the problem of learning graph topology as a problem of learning the previously called graph Laplacian matrix as it uniquely characterizes the graph. This matrix is defined as the difference between the degree matrix that

contains the degrees of the vertices along the diagonal ($\boldsymbol{D}$) and the weighted adjacency matrix of the graph ($\boldsymbol{W}$). Thus, $\boldsymbol{L}$ is a real and symmetric matrix, which has a complete set of orthonormal eigenvectors and associated eigenvalues. These eigenvalues can be interpreted as graph frequencies, and eigenvectors interpreted as corresponding graph frequency components of the Fourier representation of the graph. This is the first approach with the frequency domain of the graph and it is a key factor in the development of methods based on graph signal processing.

The measure of smoothness of a signal x on the graph $\boldsymbol{\mathcal{G}}$ is usually defined by the so-called Laplacian quadratic form which generates a scalar value that represents the smoothness of the signal. Equation 3.7 depicts it:

$$\mathcal{Q}(\boldsymbol{L}) = x^T \boldsymbol{L} x = \frac{1}{2} \sum_{i,j} w_{ij}(x(i) - x(j))^2, \tag{3.7}$$

where w_{ij} is the $ijth$ entry of the adjacency matrix $\boldsymbol{W}$ and $\boldsymbol{L}$ is the Laplacian matrix. Thus, the smaller the quadratic form, the smoother the signal on the graph. This is a key point in which one natural criterion is therefore to learn a graph (or, equivalently, its Laplacian matrix $\boldsymbol{L}$) such that the signal variation on the resulting graph (Laplacian quadratic), is small, taking into consideration that, as said above, in this technique, "the signal takes similar values at neighboring vertices" [14].

3.2.3.2 Models based on spectral filtering of graph signals

Some recently proposed methods infer directly the graph topology making assumptions on the graph properties such as sparsity of the graph and/or smoothness of the signals. In contrast, Pasdeloup et. al [2] gives a deeper description of a graph-based spectral filtering (diffusion by adjacency) approach to model relationships among the entries of the signals. The authors divide their approach into two steps: first, they characterize the matrices that may explain the relationships between the signal entries and then select one point of that convex matrices set based on two criteria of sparsity (measurement of the density of graph edges) and simplicity (a graph is simple when it has no self-connected vertex).

As mentioned above, the notion of frequency in classical signal processing and the eigenvalues of the Laplacian is tightly related. Moreover, a desired property for signals on graphs is smoothness. For this reason, the authors leverage the property that diffused signals become low-frequency after some diffusion steps, and thus smooth on the graph.

The stationary signals in Graph Signal Processing (GSP) ease the signal analysis as in the classical signal processing framework. And, as a matter of fact, diffusion of signals is a particular case of stationary processing. Because of that, Pasdeloup et. al [2] assume that each observation may be modeled as the result of passing a stationary signal, namely white noise, through a graph filter whose eigenvectors are the same as those of the normalized Laplacian. The method proposed in [2] may be used in our model implementation to learn the graph of a MS-image.

3.2.3.3 Models based on causal dependencies on graphs

Comparatively, some other GSP methods do not try to estimate the normalized or non-normalized graph Laplacian, but the adjacency matrix itself [20]. Recall that the graph Laplacian can be computed from the adjacency matrix, so both methods are valid. The work in [20] associates the graph with causal network effects using Discrete Signal Processing on Graphs DSP_G.

Nowadays, it is of significant importance to process and analyze the large amount of data coming from all kinds of contexts (financial, social media, health, geology, etc.) collected in the form of time series. Because of that, Mei contribution plays a key role here. His work concentrates on estimating the network structure capturing the dependencies among time series in the form of a possibly directed, weighted adjacency matrix $\boldsymbol{A}$.

Causal Graph Processes (CGP) model can be considered as a discrete-time series $x[k]$ on a graph in terms of a matrix polynomial in $\boldsymbol{A}$, statistical noise, and scalar polynomial coefficients. This method obeys the intuition that activity on the network travels at a fixed rate (one graph shift per sampling period), and network activity cannot be affected by network effects of a higher order than the limit set by this speed. In [20] the author considers that the data follows the CGP model and makes some assumptions on the polynomials of adjacency matrix [14].

The main objective of applying the methods described above is to design an appropriate underlying graph, connecting pixels with weights that reflect the image structure, then one can interpret the image as a signal on a graph. Defining a good graph from data observations is so important in many applications, including image compression, image restoration, image filtering, and image segmentation. In [19] the authors define compression as "the process of encoding an image x onto a codeword c(x), minimizing distortion in the reconstructed image for a given target bit rate". In this process [19] highlights that due to the technique based on smoothness prior, it does not take into account the actual cost of representing, and thus coding, it could be a major problem in compression. As we can see the smoothness signal method, does not have a good performance in compression. Nevertheless, this method had a high performance in temperature sensed in different locations in a geographical region as mentioned above.

Thus, there is not a universal method that has the best performance in all applications since each application has a specific requirement. That is why it is important to evaluate the performance of different learning graphs techniques in the GBF-CD method, which in turn is a new application, and precisely because of this, there are not many previous works with this approach.

3.3 Confusion Matrix

3.3.1 Definition

A confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the ground truth is known. It allows the visualization of the performance of an algorithm. The Positive/Negative label refers to the predicted outcome of an experiment, while the True/False refers to the actual outcome.

	Actual – True/False	
Predicted – Positive/Negative	True Positive	False Positive
	False Negative	True Negative

Table 3.1: Confusion Matrix

- True positive: A predicted positive outcome is indeed True.
- False positive (False Alarm): A predicted positive outcome is in reality false.
- False negative (Missed Alarm): A predicted negative outcome is in reality true.
- True negative : A predicted negative outcome is indeed false.

3.3.2 Metrics

- Accuracy (A): (all correct / all) = $(TP + TN)/(TP + TN + FP + FN)$
- Precision (P): (true positives / predicted positives) = $TP/(TP + FP)$
  High Precision indicates an example labelled as positive is indeed positive (a small number of FP).
- Recall aka Sensitivity (R): (true positives / all actual positives) = $TP/(TP + FN)$
  High Recall indicates the class is correctly recognized (a small number of FN).
- Cohen's kappa (K): $(A - P_e)/(1 - P_e)$, where P_e is the hypothetical probability of chance agreement. $P_e = P_{correct} + P_{incorrect}$ with $P_{correct} = (TP+FP/TP+FP+FN+TN)/(TP+FN/TP+FP+FN+TN)$ and $P_{incorrect} = (FN + TN/TP + FP + FN + TN)/(FP + TN/TP+FP+FN+TN)$. Cohen's Kappa represents the degree of accuracy and reliability in a statistical classification. It measures the agreement between two raters (judges) who each classify items into mutually exclusive categories.
- Overall error (OE): (all incorrect / all) = $(FP + FN)/(TP + TN + FP + FN)$

3.4 Hypotheses

The implementation of graph learning models based on a signal representation perspective such as signal smoothness or spectral filtering instead of using a Gaussian kernel, improves the performance of the **GBF-CD** method in terms of missed alarms (MA), false alarms (FA), precision (P), recall (R), Cohen's kappa (K) and overall error (OE), considering the execution time and the usage of the computational resources.

However, due to the uncertainty of the method's suitability on this application, it could be also conceived as a null hypothesis that:

The implementation of graph learning models based on a signal representation perspective such as signal smoothness and spectral filtering instead of using a Gaussian kernel, reduce the performance of the GBF-CD method in terms of missed alarms (MA), false alarms (FA), precision (P), recall (R), Cohen's kappa (K) and overall error (OE), considering the execution time and the usage of the computational resources.

3.5 Test datasets description

The following are the 14 data sets used to test the proposed methodology for change detection. First column are pre-event images, second column are post-event images and third column are the reference change map images known as "Groundtruth".

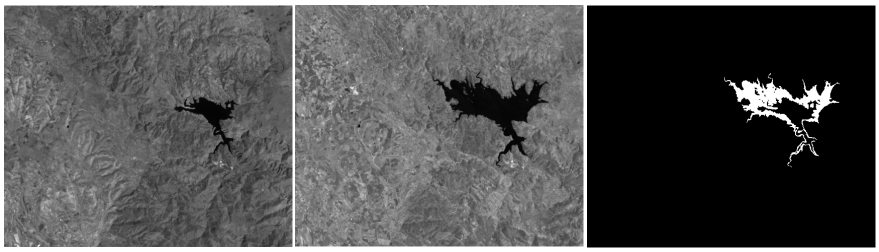


Figure 3.2: Sardania. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

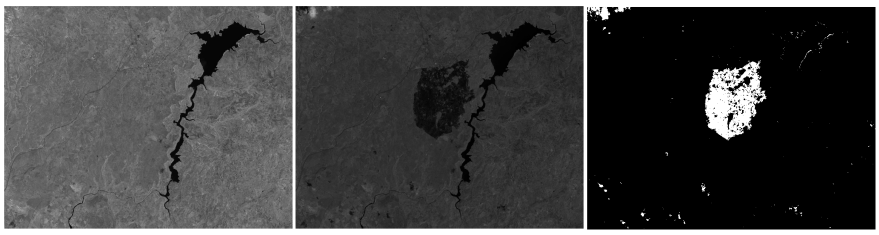


Figure 3.3: Omodeo. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

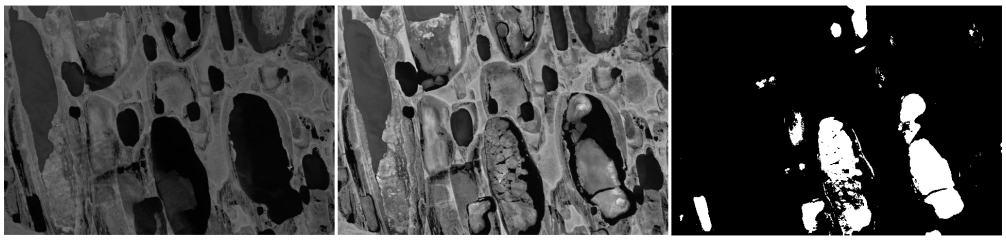


Figure 3.4: Alaska. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

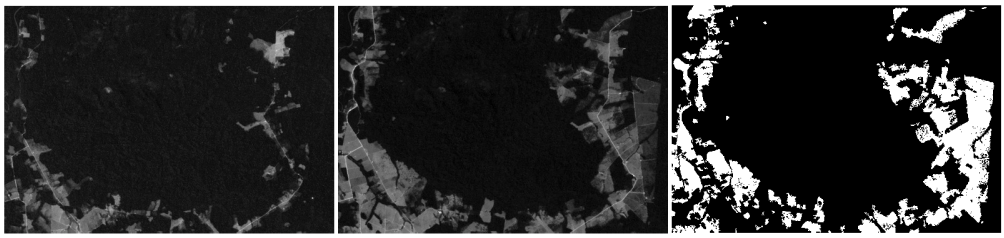


Figure 3.5: Madeirinha. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

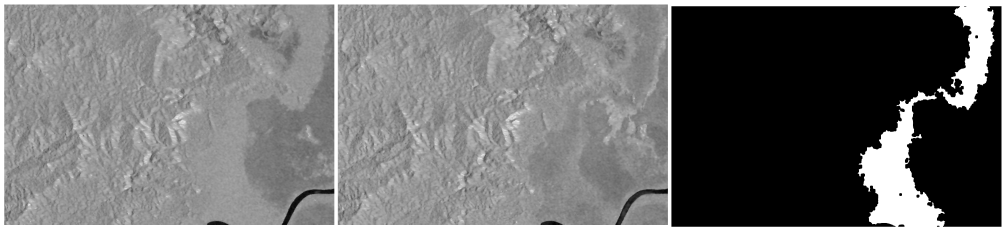


Figure 3.6: Katios. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

Figure 3.7: Dique. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

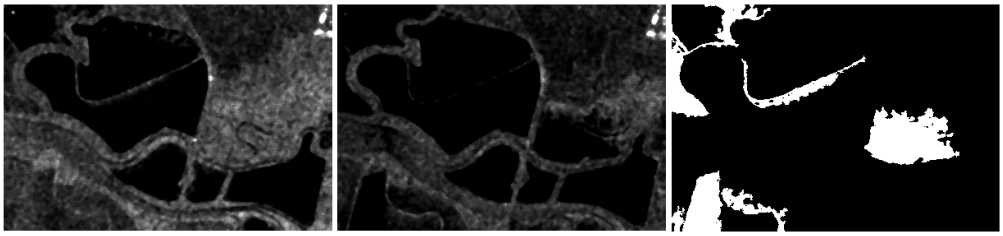


Figure 3.8: San Francisco. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

Figure 3.9: Wenchuan. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

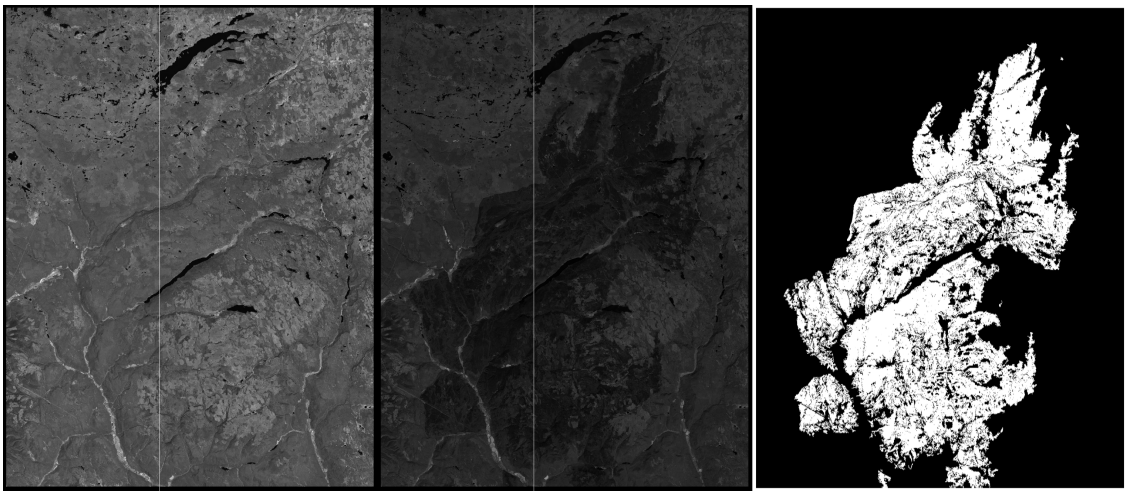


Figure 3.10: Canada. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

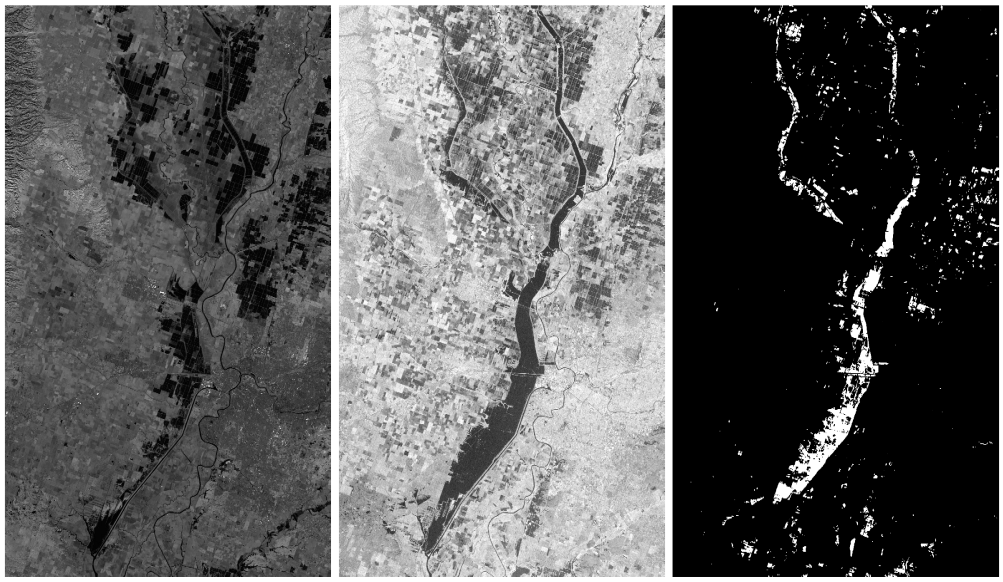


Figure 3.11: California. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

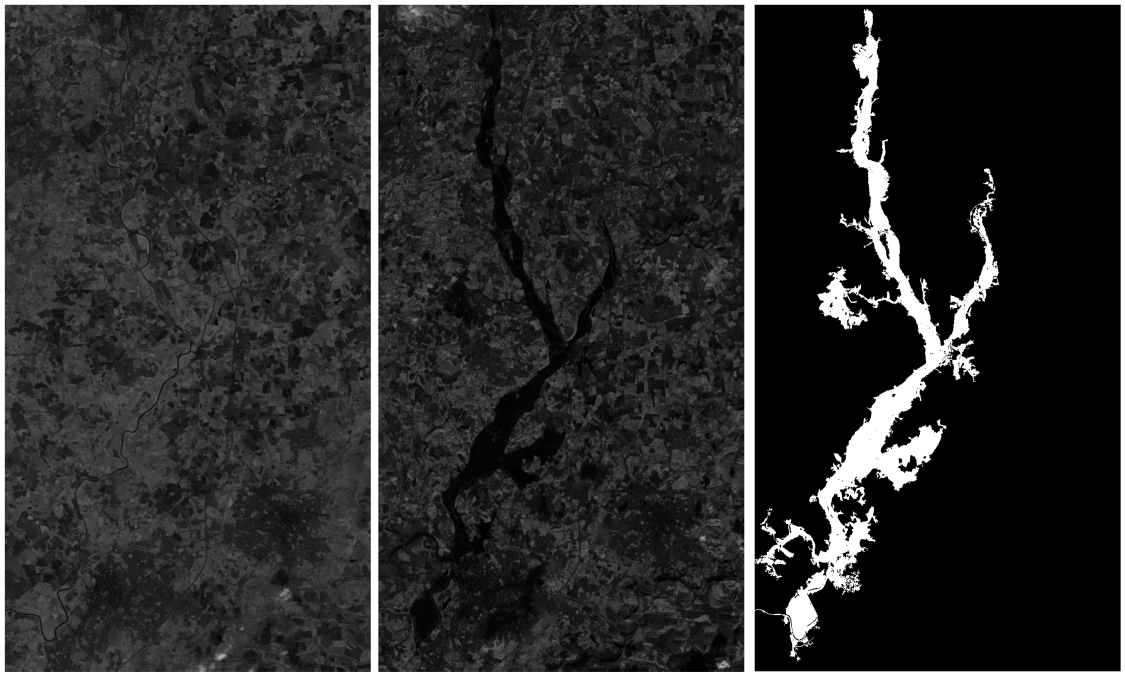


Figure 3.12: Contest. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

Figure 3.13: Toulouse. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

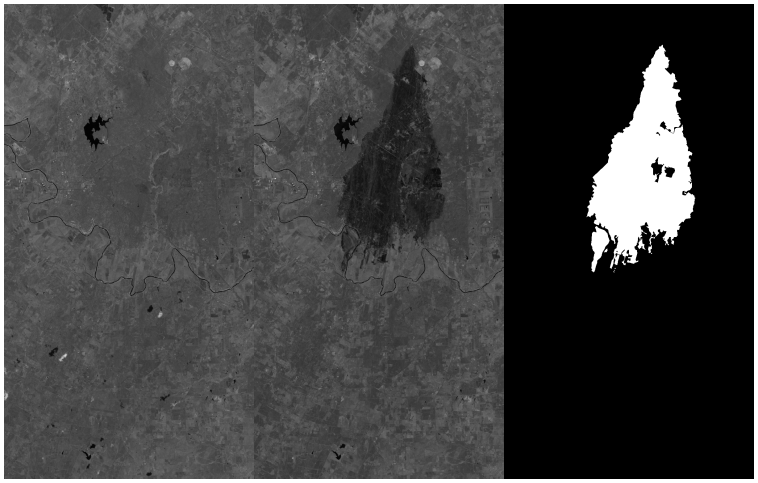


Figure 3.14: Bastrop. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

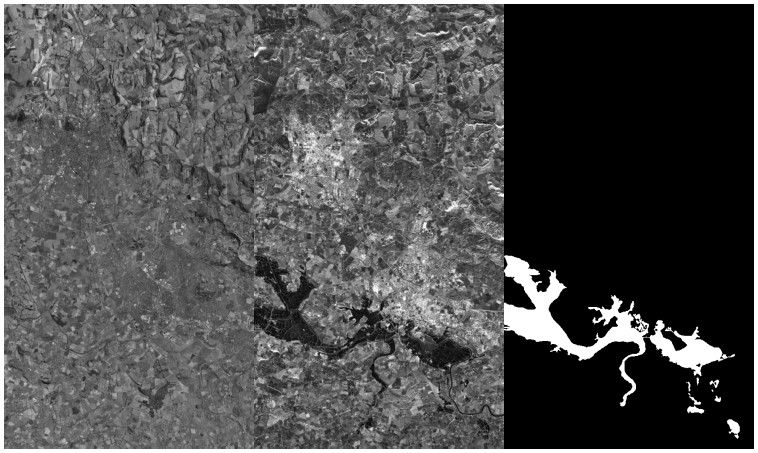


Figure 3.15: Gloucester. From left to right to the pre-event image, the post-event image and the change map (groundtruth)

Table 3.2: Databases used to evaluate the performance of the proposed method.

Place	Event	Pre-Date	Post-Date	Lat	Lon	Size	Band	Sensor
Sardinia Island	Flood	03 September 1995	03 July 1996	39.68, 39.55	9.10, 9.30	479 × 573	NIR	Landsat-5 TM
Omodeo lake	Fire	25 June 2013	10 August 2013	40.17, 39.97	8.66, 9.00	742 × 965	RED	Landsat-5 TM
Alaska	Melt Ice	24 June 1985	13 June 2005	70.761, 70.641	−153.074, −152.553	443 × 642	NIR	Landsat-5 TM
Brasil, Madeirinha	Farming building	15 July 2000	16 July 2006	−9.335, −9.433	−61.942, −61.798	364 × 527	RED	Landsat-5 TM
Colombia, Katios National Park	Fire	10 March 2019	27 April 2019	7.943, 7.832	−77.23, −77.063	879 × 1319	SAR	Sentinel 1 A
Colombia, Atlantico	Flood (dam)	28 April 2010	16 March 2011	10.439, 10.288	−75.14, −74.921	729 × 1056	SAR	ALOS/PALSAR
San Francisco	Flood	10 August 2003	16 May 2004	38.11, 38.00	121.41, 122.46	275 × 400	SAR	ERS-2 SAR
China, Wenchuan	Earthquake	03 March 2008	16 June 2008	31.049, 31.011	103.525, 103.581	301 × 442	SAR	ESA/ASAR
France, Toulouse	Building	10 February 2009	15 July 2013	43.5835, 43.5702	1.4318, 1.4817	2604 × 4404	SAR/NIR	TerraSAR-X Pleiades
Canada, Prince George	Fire	06 July 2017	22 August 2017	51.48, 50.80	−121.626, −120.863	2479 × 1905	NIR	Landsat-8
California	Flood	11 January 2017	26 February 2017	39.346, 39.348	−121.161, −121.924	3500 × 2000	NIR/SAR	Landsat-8 Sentinel 1 A
U.K., Gloucester-1	Flood	05 September 1999	17 November 2000	52.126, 52.134	−2.113, −2.280	4220 × 2320	NIR	SPOT
Bastrop	Fire	08 September 2011	22 October 2011	30.1316, 30.1321	−97.2898, −97.3182	1534 × 808	NIR/NIR	Landsat-5 TM EO-1 ALI
U.K., Gloucester-2, UK	Flood	14 June 2006	25 July 2007	51.8552, 51.8512	−2.2174, −2.1910	4220 × 2320	NIR/SAR	Quickbird 02 TerraSAR-X

3.6 GBF-CD Algorithm

This GBF-CD method [13] has three fundamental stages. The first is the graph learning stage, in which two graphs are obtained. One that represents the pre-event image and another that represents the post-event image. For this step, Jimenez et. al [13] use a Gaussian kernel, where the weight of the edges is determined by the radiometric distance of the pixels, as shown in Equation 3.8.

$$W_{i,j} = \exp(-\frac{||x_i - x_j||_2^2}{\sigma_x^2}) \tag{3.8}$$

where x_i is the intensity of pixel i, and σ_x^2 is a parameter, which is equal to the standard deviation of all the radiometric distances. It is important to note that the authors in [13] use the Nyström extension to reduce the computational cost required for the construction of the graphs and the following steps as shown in Algorithm 1. This can be seen in more detail in [13].

The second step is the fusion stage which is also shown in Algorithm 1, in which Jimenez et. al [13] maximize the radiometric distance (or minimize the similarity) among pixels. In this way, they obtain a fusion graph ($\boldsymbol{W_F}$), which is finally decomposed into eigenvalues ($\boldsymbol{D}$) and eigenvectors ($\boldsymbol{U}$) to obtain different change maps from the multiplying each eigenvector by the square root of its corresponding eigenvalue ($I_{u_i} = u_i\sqrt{d_i}$).

Finally, the change map that has the highest mutual information with a binarized prior signal ($\boldsymbol{I_{prior}}$), which is the first approximation to the final change map, is chosen. This prior comes from the normalized differences between pre-event and post-event images as shown in Equation 3.9.

$$\boldsymbol{I_{prior}} = bin(\frac{I_{bf} - I_{af}}{I_{bf} + I_{af}}) + bin(\frac{I_{af} - I_{bf}}{I_{bf} + I_{af}}) \tag{3.9}$$

Where I_{bf} and I_{af} correspond to the pre and post-event images respectively.

Algorithm 1 GBF for temporal data.

Require: Temporal images from band b or set of bands $\mathbf{X}^{b,k} \in \mathbb{R}^{m \times n}$, number of samples n_s
Ensure: Fused graph $\mathbf{W_F} \in \mathbb{R}^{(n_s+c)\times n_s}$

Initialize: $k = 1$, $N = m \times n$
while $k \leq 2$ **do**
  Scale the data by $\mathbf{X}^{b,k} = \frac{\mathbf{X}^{b,k}}{max(\mathbf{X}^{b,k})}$
  Take n_s samples uniformly distributed across $\mathbf{X}^{b,k}$ by spatial grid sampling.
    $X_{AA}^{b,k} = \text{sampler}(\mathbf{X}^{b,k}, n_s), \in \mathbb{R}^{n_s}$
  Find the complement $\overline{X}^{b,k} \in \mathbb{R}^{c}$ of $X_{AA}^{b,k}$ in $\mathbf{X}^{b,k}$.
  For each set $X_{AA}^{b,k}$ and $\overline{X}^{b,k}$, perform the pairwise distance between samples-samples ($\mathbf{d_{AA}}^{b,k} \in \mathbb{R}^{n_s \times n_s}$) and samples-complement ($\mathbf{d_{AB}}^{b,k} \in \mathbb{R}^{c \times n_s}$).
    $$\mathbf{d_{AA}}^{b,k} = \left\{ \left\| x_{AA_i}^{b,k} - x_{AA_j}^{b,k} \right\|_2 \right\}_{i \;\; j}^{n_s n_s}, \forall i \neq j$$
    $$\mathbf{d_{AB}}^{b,k} = \left\{ \left\| \overline{x}_{i}^{b,k} - x_{AA_j}^{b,k} \right\|_2^3 \right\}_{i \;\; j}^{c \;\; n_s}, \forall i \neq j$$
  Apply the normalized graph Laplacian ($\hat{\mathbf{D}}^{-\frac{1}{2}} \widehat{\mathbf{W}} \hat{\mathbf{D}}^{-\frac{1}{2}}$) by using the code in [21].
  Apply a Gaussian kernel ($\kappa_G(.)$) with $\sigma = mean(\mathbf{d_{AB}})$ on the normalized distances, and build the approximated normalized Laplacian matrix based on the Nyström approximation.
    $$\widehat{\mathbf{W}}_N^{b,k} = \left[\kappa_G(\mathbf{d_{AA}}^{b,k}); \kappa_G(\mathbf{d_{AB}}^{b,k}) \right]^\top$$
  $k = k + 1$
end while
$\mathbf{W_F} = min(\widehat{w}_{N_{ij}}^{b,k})$, with $i = 1, .., c; j = 1, .., n_s$.

Chapter 4

Signal smoothness

4.1 Introduction

Smoothness feature as GSP approach intends to learn a graph which gives a structure to unstructured data giving more weight to those edges that connect vertex with similar values. This approach has an issue related with its computational efficiency when the number of vertices grows. This chapter presents the results of applying this approach in the graph learning stage of a change detection application, namely Graph based Fusion - Change Detection method (**GBF-CD**) [13].

4.2 Graph Laplacian Learning Algorithm

In [1] Dong et al. consider a factor analysis model [22] [23] applied to graph signals and assume that the observed signals are controlled by a set of unobserved latent variables with some given probabilistic prior.

By imposing a Gaussian prior on the latent variables in their generalized factor analysis model, they obtain a Principal Component Analysis (PCA)-like representation for the graph signals, which come to be a smooth signal representation on graphs. The graph Laplacian matrix $\boldsymbol{L}$ supports the relationship between the Gaussian latent variables and the signal observations $\boldsymbol{X}$.

They designed a new algorithm for learning a valid graph Laplacian operator from data samples, such that the graph signal representation is smooth and consistent with the Gaussian prior on the latent variables. Specifically, given potentially noisy signal observations, their algorithm iterates between the learning of a smoothed version of the original signal measurements and the learning of a graph topology such that the variations of the smooth signals on the learned graph are minimized upon convergence.

We keep the constraints and considerations taken into account in the learning graph process that Dong exposed in his **GL-SigRep** algorithm [1], these are:

- **GL-SigRep**, stops when the maximum number of iterations is reached or the absolute change in the objective is smaller than 10^{-4}.

- The experiments are carried out on different sets of parameters, namely, for different values

of α and β in Equation 4.1.

$$\begin{aligned} \min_{L\in\mathbb{R}^{n\times n}, Y\in\mathbb{R}^{n\times p}} \quad & \|X-Y\|_F^2 + \alpha\, tr(Y^T L Y) + \beta\, \|L\|_F^2 \\ \text{s.t.} \quad & tr(L) = n, \\ & L_{ij} = L_{ji} \leq 0, i \neq j, \\ & L \cdot \mathbf{1} = \mathbf{0} \end{aligned} \tag{4.1}$$

where $X \in \mathbb{R}^{n\times p}$ contains the p input data samples $\{x_i\}_{i=1}^p$ as columns, α and β are two positive regularization parameters, and $\mathbf{1}$ and $\mathbf{0}$ denote the constant one and zero vectors. In addition, tr(.) and $\|.\|_F$ denote the trace and Frobenius norm of a matrix, respectively. Notice that the quadratic Laplacian Equation 3.7, which express the smoothness degree of a graph signal on a graph, is a term of the minimization problem. As a result, it enforces the smoothness property of the observed signals on the learned graph.

- Finally, they prune (equate to 0) insignificant edges that have a weight smaller than 10^{-4} in the learned graph.

The optimization problem of Equation 4.1 is not jointly convex in $\boldsymbol{L}$ and $\boldsymbol{Y}$. They transform it into a convex problem making some changes in its formulation and solving it in a iterative form, fixing one variable and solving the other. Specifically, they first initialize $\boldsymbol{Y}$ as the signal observations $\boldsymbol{X}$. Then, at the first step, for a given $\boldsymbol{Y}$, they solve the following optimization problem with respect to $\boldsymbol{L}$:

$$\begin{aligned} \min_{L} \quad & \alpha\, tr(Y^T L Y) + \beta\, \|L\|_F^2 \\ \text{s.t.} \quad & tr(L) = n, \\ & L_{ij} = L_{ji} \leq 0, i \neq j, \\ & L \cdot \mathbf{1} = \mathbf{0} \end{aligned} \tag{4.2}$$

At the second step, $\boldsymbol{L}$ is fixed and they solve the following optimization problem with respect to $\boldsymbol{Y}$:

$$\min_{Y} \quad \|X-Y\|_F^2 + \alpha\, tr(Y^T L Y) \tag{4.3}$$

The summarized algorithm is presented in Algorithm 2 as follows:

Algorithm 2 Graph Learning for Smooth Signal Representation (**GL-SigRep**) [1].

Require: Input signal X, number of iterations $iter$, α, β
Ensure: Output signal Y, graph Laplacian L
1: **Initialization:** $Y{=}X$
2: **for** $t = 1, 2, ..., iter$ **do**
3: **Step to update Graph Laplacian L:**
4: Solve the optimization problem of Equation 4.2 to update L.
5: **Step to update Y:**
6: Solve the optimization problem of Equation 4.3 to update Y.
7: **end for**
8: $L = L^{iter}, Y = Y^{iter}$.

One disadvantage of this algorithm is that its computational cost is considerable high, namely, $\mathcal{O} = n^2$ per iteration for n nodes. Furthermore, this algorithm requires the tuning of parameters α and β to retrieve a solution.

4.3 Large Scale Graph Learning from Smooth Signals

In [6], Kalofolias et al. propose the first scalable graph learning method, we will call it **GL-LogDeg** algorithm hereafter, with the same leading cost as approximate nearest neighbors (A-NN) algorithms, and with quality that approaches state-of-the-art graph learning method proposed for the same author [24]. The authors in this new approach [6] leverage the desired graph sparsity to reduce computation and automatically select the parameters of the model in [24] given a desired graph sparsity level. As in k-NN, the user chooses the number of neighbors k (edges per node), without performing grid search over two parameters α and β.

The minimization problem of [24] is presented in Equation 4.4

$$\begin{aligned} \min_{W} \quad & \sum_i \sum_j W_{ij} Z_{ij} - \alpha \sum_i \log(\sum_j W_{ij}) + \frac{\beta}{2} \|W\|_F^2 + \frac{c}{2} \|W - W0\|_F^2 \\ \text{s.t.} \quad & W_{ij} = W_{ji} \geq 0, i \neq j, \\ & diag(W) = \mathbf{0}, \end{aligned} \tag{4.4}$$

Where $\boldsymbol{Z}$ is a matrix with (squared) pairwise distances of nodes, α is a log prior constant (bigger $\alpha \rightarrow$ bigger weights in $\boldsymbol{W}$), β is a $\|W\|_F^2$ prior constant (bigger $\beta \rightarrow$ more dense $\boldsymbol{W}$, that is less sparsity). In [6] Kalofolias et. al show how to reduce the complexity of Equation 4.4 fixing parameters $\alpha = \beta = 1$ and multiplying the pairwise distance matrix $\boldsymbol{Z}$ by a parameter θ used to make the sparsity analysis simpler. This parameter is computed by each column of $\boldsymbol{Z}$ and is bounded between upper and lower values. Equation 4.5 shows the computation of this bound: for

the full $\boldsymbol{Z}$ matrix.

$$\theta_k \in \left(\frac{1}{n} \sum_{j=1}^{n} \theta_{k,j}^{lower}, \frac{1}{n} \sum_{j=1}^{n} \theta_{k,j}^{upper} \right]$$
$$\theta_k \in \left(\sum_{j=1}^{n} \frac{1}{n\sqrt{k\widehat{Z}_{k+1,j}^2 - B_{k,j}\widehat{Z}_{k+1,j}}}, \sum_{j=1}^{n} \frac{1}{n\sqrt{k\widehat{Z}_{k,j}^2 - B_{k,j}\widehat{Z}_{k,j}}} \right] \tag{4.5}$$

The summarized algorithm is presented in Algorithm 3 as follows:

Algorithm 3 Large Scale Graph Learning from Smooth Signals (**GL-LogDeg**) [6]

Require: Input signal X, K edges per node (sparsity level)
Ensure: Weighted Adjacency matrix W
1: **Initialization:** Z=gsp_distanz(X^T) %Z:Matrix with (squared) pairwise distances of nodes
2: **Step to compute $\boldsymbol{\theta}$**
3: Compute bounds of θ with Equation 4.5
4: Compute θ as a geometric mean between theta bounds
5: **Step to compute adjacency matrix $\boldsymbol{W}$**
6: Compute W with Equation 4.4

The advantage of this algorithm [6] is its computational cost, since the cost of learning a kr-A-NN graph (graph built according kr approximate nearest neighbors) is $\mathcal{O}$(n log(n)d) for n nodes and data in $\mathbb{R}^d$ where r is a small multiplicative factor to provide to the algorithm to select the right edges, while additionally learning the edge weights costs $\mathcal{O}$(krn) per iteration. The overall complexity is therefore $\mathcal{O}(nlog(n)d) + \mathcal{O}(nkrI)$ for I iterations. For large n, the dominating cost is asymptotically the one of computing the A-NN and not the cost of learning the weights on the reduced set. However, as **GL-SigRep** does, this algorithm depends as well on α and β parameters but, if the user wants, applying the approach [6],it depends only on k parameter. Thus, this algorithm still needs a grid-search to find the best parameter where the graph signal is smoothest on. We will use the **GSPBOX toolbox** [7] which integrates this approach and more functions to learn a graph under the signal smoothness assumption. This toolbox [7] was developed at the Signal Processing Laboratory LTS2 of the Ecole Polytechnique Fédérale de Lausanne EPFL.

4.4 Experiments

In this section, we assess the performance of the aforementioned graph learning algorithms. We first describe the general experimental setting, and then we present experimental results on synthetic data for different numbers of vertex to test computational efficiency on both algorithms. Moreover, we integrate the most efficient algorithm in the GBF-CD method and evaluate its performance on 14 different datasets.

4.4.1 Experimental settings

In our experiments, we use the convex optimization package CVX [25] to solve the convex optimization problems presented in both previous algorithms [1] [6].

Then, we compare visually as a scalar image the graph Laplacian learned by **GL-SigRep** and the graph Laplacian learned by **GL-LogDeg GSP toolbox**. In addition, we compare quantitatively through the following evaluation metrics commonly used in information retrieval: Precision, Recall, F-measure and Normalized Mutual Information (NMI) [26], to test the performance of both algorithms.

4.4.2 Results on synthetic data

We carry out this experiment on a synthetic graph of 20 vertices whose edges are determined based on Euclidean distances between vertices. To build the graph, we generate the coordinates of the vertices uniformly at random in the unit square, and compute the edge weights with the next function.

$$W(i,j) = \exp(\frac{-d(i,j)^2}{\sigma^2}) \tag{4.6}$$

Where $d(i,j)$ is the distance between vertices and $\sigma = 0.5$ is a kernel width parameter. We then remove all the edges whose weights are smaller than 0.75. Given the groundtruth synthetic graph, we compute the graph Laplacian and normalize the trace according to Equation 4.1. Then, we generate 100 signals ($X = \{x_i\}_{i=1}^{100}$) that follow the distribution Equation 4.7.

$$x \sim N(\mu_x, L^\dagger + \sigma_\varepsilon^2 I^2) \tag{4.7}$$

Where $L^\dagger$ is the pseudoinverse of L, $\mu_x = 0$ and $\sigma_\varepsilon = 0.5$. Then, having the signals X, we apply **GL-SigRep** and **GL-LogDeg** to learn the graph Laplacian matrices. It is important to highlight that the performance of both algorithms depends on some parameters, such as α and β for **GL-SigRep** and K for **GL-LogDeg**. Taking this fact into consideration, the optimal values for α and β were taken from [1], in which Dong et al. carry out the same experiment. On the other hand, we choose the parameter k through a grid search. The optimal values for α, β and k are 0.012, 0.79 and 6 respectively. To evaluate the performance, we first provide visual comparisons in Figure 4.1, where we show from the left to the right columns the Laplacian matrices of the groundtruth graph, the graph Laplacian learned by **GL-SigRep** and the graph Laplacian learned by **GL-LogDeg**. It is important to highlight that the output of **GL-LogDeg** is an adjacency matrix. Therefore, it is necessary to transform the adjacency matrix into a Laplacian matrix using Equation 3.3 and normalize this Laplacian matrix with the trace, taking into account that the output of **GL-SigRep** is a Laplacian matrix normalized by the trace, following the constraints in Equation 4.1.

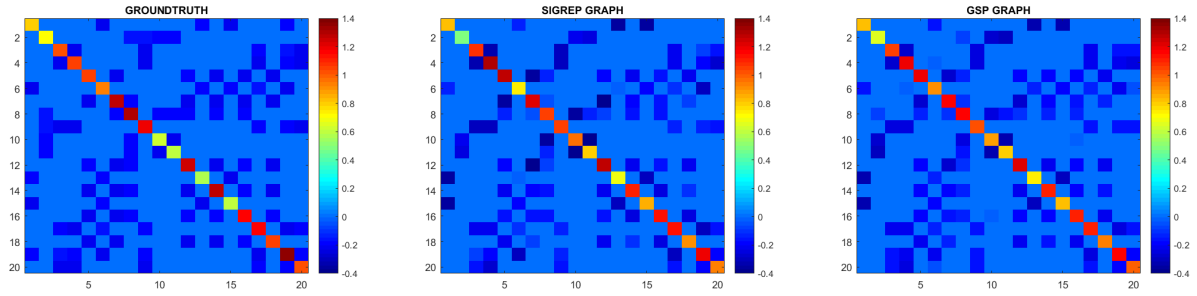


Figure 4.1: The learned graph Laplacian matrices. The columns from the left to the right are the groundtruth Laplacian, the Laplacian learned by GL-SigRep (Dong's algorithm [1]) and the Laplacian learned by GSP toolbox (Kalofolias's algorithm [6]).

As we can see in Figure 4.1, the Laplacian matrices learned by both algorithms are consistent with the groundtruth. Nevertheless, it is necessary to quantitatively evaluate the performance of both algorithms. As stated above, the evaluation metrics used for this process are Precision, Recall, F-measure and Normalized Mutual Information (NMI) [26]. We averaged the scores over ten random instances of the graph with the associated signals X and the results are shown in Table 4.1.

	F-measure	**Precision**	**Recall**	**NMI**
GL-SigRep	**0,8879**	**0,8696**	0,9092	**0,6166**
GL-GSP	0,8813	0,8537	**0,9158**	0,6095

Table 4.1: Graph learninng performance for **GL-SigRep** and **GL-LogDeg**.

The algorithm **GL-SigRep** provides competitive or superior performance compared to **GL-LogDeg** in most of the metrics as shown in Table 4.1. Nevertheless, the metric values of both algorithms are very similar. Taking this fact into consideration and knowing the advantage of **GL-LogDeg** in terms of its computational cost $\mathcal{O}$(n log(n)d) + $\mathcal{O}$(nkrI) over **GL-SigRep** $\mathcal{O}(n^2)$, we implemented the **GL-LogDeg** algorithm using the GSP toolbox [7] for the graph learning stage of the **GBF-CD** model.

4.4.3 Smoothness Approach in Change Detection

We now test the **GL-LogDeg** algorithm using the GSP-toolbox in the graph learning stage of the Graph based Fusion - Change Detection method (**GBF-CD**) [13]. Taking into account that each pixel of the image is represented as a node in the graph, one of the challenging factors of this method is that given the high number of pixels in a Multispectral (MS) image, the computational cost of calculating the full matrix $\boldsymbol{W} \in \mathbb{R}^{N \times N}$, where N is the number of pixels, is extremely high. To tackle this problem, Jimenez et al. [13] used the Nyström extension to find an approximation of $\boldsymbol{W}$ by using less computational resources. Applying Nyström extension it is possible to calculate $\boldsymbol{W}$ as:

$$\mathbf{W} \approx \widehat{\mathbf{W}} = \kappa_G \left(\begin{bmatrix} \boldsymbol{d_{AA}} \\ \boldsymbol{d_{AB}} \end{bmatrix} \right) \tag{4.8}$$

Where κ_G is a Gaussian kernel, $\boldsymbol{d_{AA}} \in \mathbb{R}^{n_s \times n_s}$ gives the graph distances within the n_s sample nodes parameter, $\boldsymbol{d_{AB}} \in \mathbb{R}^{n_s \times (N-n_s)}$ represents the distances between the n_s sample nodes and the remaining $N - n_s$ nodes (complement). Therefore it is possible to reduce the computational cost from calculating $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ to calculating $\boldsymbol{d_{AA}} \in \mathbb{R}^{n_s \times n_s}$ and $\boldsymbol{d_{AB}} \in \mathbb{R}^{n_s \times (N-n_s)}$. As we can see, the smaller n_s, the lower the computational cost required. Nevertheless, unless $n_s = N/2$, $\boldsymbol{d_{AB}}$ is a rectangular matrix, so it would not be possible to use the smoothness approach with GSP-Toolbox, considering that this algorithm only works with square matrices. Thereby, we set $n_s = N/2$ to obtain a square matrix for $\boldsymbol{d_{AB}}$. However, the computational cost only could be reduced from calculating $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ to calculating $\boldsymbol{d_{AA}} \in \mathbb{R}^{\frac{N}{2} \times \frac{N}{2}}$ and $\boldsymbol{d_{AB}} \in \mathbb{R}^{\frac{N}{2} \times \frac{N}{2}}$, which still demands a extremely high computational cost given a high number of pixels in a MS image. For this reason, for the signal smoothness based approach, Nyström is not used.

In order to reduce computational cost with smoothness's approach, we applied the Graph-Cut method [5], which segments the pre- and post-event original images into a reduced number of local homogeneous regions/classes. Since we must create a graph for each image (pre- and post-event) to later perform the fusion step, it is necessary that when applying Graph-cut to both images, they have the same regions (patches). To do so, we first over-segment the prior ($\boldsymbol{I}_{prior}$) explained in Equation 3.9, with graph-Cut [5] into $\boldsymbol{q}$ classes, and obtain the label matrix $\boldsymbol{L_M}$ of size $\mathbf{M} \times \mathbf{N}$, which presents the label for each pixel in $\boldsymbol{I}$ after graph cut process. This process can be visually observed in Figure 4.2. Then according to the label matrix $\boldsymbol{L_M}$, we segment the pre and post-event images into $\boldsymbol{q}$ patches, denoted as:

$$\begin{aligned} \boldsymbol{D_{pre}} &= \{p_{pre}(i) | 1 \leq i \leq q\} \\ p_{pre}(i) &= \frac{1}{N_i} \sum_{L_M(m,n)=i} X_{pre}(m,n) \\ \boldsymbol{D_{post}} &= \{p_{post}(i) | 1 \leq i \leq q\} \\ p_{post}(i) &= \frac{1}{N_i} \sum_{L_M(m,n)=i} X_{post}(m,n) \end{aligned} \tag{4.9}$$

Where $\boldsymbol{p_{pre}}(i)$ and $\boldsymbol{p_{post}}(i)$ represent the mean values of the $i-th$ patches of the pre- and post-event images respectively, and N_i is the number of pixels belonging to $i-th$ class (patch or region). $\boldsymbol{D_{pre}}$ and $\boldsymbol{D_{post}}$ present the over-segmented pre- and post-event images after the graph cut process, which consists in q patches and each patch has its own label i, where $1 \leq i \leq q$.
After the graph cut processing, the graph learning stage is done applying smoothness approach with $\boldsymbol{D_{pre}}$ and $\boldsymbol{D_{post}}$ as inputs. In this way, we get $\boldsymbol{W_{pre}}$ and $\boldsymbol{W_{post}}$ as graphs that represent the pre and post-event images respectively sized $(q \times q)$. With these graphs we perform the fusion stage minimizing the similarity, that is taking the smaller value per position and $\boldsymbol{W_F}$ is decomposed into

its eigenvectors and eigenvalues, i.e. $\boldsymbol{u_i}$ and d_i, respectively. Nonetheless, the vectors that we obtain by multiplying each eigenvalue and eigenvector of $\boldsymbol{W_F}$ ($\boldsymbol{u_i}\sqrt{d_i}$) are $q \times 1$ in size, so it is necessary to segment the prior ($\boldsymbol{I}_{prior}$) into classes (with the same regions than the pre and post-event images) for the comparison stage between each vector $\boldsymbol{u_i}\sqrt{d_i}$ and the prior in order to obtain the highest mutual information.Therefore, we get $\boldsymbol{D_{prior}}$ as follows:

$$\begin{aligned} \boldsymbol{D_{prior}} &= \{p_{prior}(i) | 1 \leq i \leq q\} \\ p_{prior}(i) &= \frac{1}{N_i} \sum_{L_M(m,n)=i} I_{prior}(m,n) \end{aligned} \tag{4.10}$$

Finally, after obtaining the change map vector $\tilde{\boldsymbol{C}}_{\boldsymbol{M}} \in \mathbb{R}^{q \times 1}$ from the set of vectors $\boldsymbol{u_i}\sqrt{d_i}$ with the highest mutual information with the reduced prior ($\boldsymbol{D_{prior}}$), it is necessary to obtain the original size change map $\tilde{\boldsymbol{C}}_{\boldsymbol{M}} \in \mathbb{R}^{m \times n}$. This is achieved by assuming that if a class (i.e segmented region) changes, all pixels belonging to that class change as well. To do this, we used the $\boldsymbol{L_M}$ matrix, which contains the pixels belonging to each class.

Algorithm 4 Graph Based Fusion to Change Detection using Smoothness Approach combined with Graph Cut segmentation for graph learning stage (**GBF-SEG-GSP**).

Require: Temporal images pre- and post-event from band b or set of bands $\boldsymbol{X}^{b,c} \in \mathbb{R}^{m \times n}$, number of non-zero edges per node on average $\boldsymbol{k}$, stop used for the Graph Cut Segmentation $\boldsymbol{stop}$.
Ensure: Change Map $\boldsymbol{C_M} \in \mathbb{R}^{m \times n}$

1: **Find the Change Map prior** $\Rightarrow \boldsymbol{I_{prior}} = bin(\frac{I_{bf}-I_{af}}{I_{bf}+I_{af}}) + bin(\frac{I_{af}-I_{bf}}{I_{bf}+I_{af}})$
2: **Graph Cut applied to prior, Pre- and Post-image [5]** $\Rightarrow \boldsymbol{D_{pre}}, \boldsymbol{D_{post}}, \boldsymbol{D_{prior}}$
3: Fixing $\boldsymbol{stop}$ to group $\boldsymbol{I_{prior}}$ in $\boldsymbol{q}$ classes (each pixel value has now a class number) $\Rightarrow \boldsymbol{L_M}$
4: Pre-image $\boldsymbol{X}^{b,1}$ is reduced to $\boldsymbol{q}$ patches. $\Rightarrow \boldsymbol{D_{pre}} = \tilde{\boldsymbol{X}}^{b,1}$. See Equation 4.9
5: Post-image $\boldsymbol{X}^{b,2}$ is reduced to $\boldsymbol{q}$ patches. $\Rightarrow \boldsymbol{D_{post}} = \tilde{\boldsymbol{X}}^{b,2}$. See Equation 4.9
6: $\boldsymbol{I_{prior}}$ is reduced to $\boldsymbol{q}$ patches. $\Rightarrow \boldsymbol{D_{prior}} = \tilde{\boldsymbol{I}}_{\boldsymbol{prior}}$. See Equation 4.10
7: **Computations in small scale**
8: **Learn graphs from data**$\Rightarrow$ **GL-LogDeg**$(\tilde{\boldsymbol{X}}^{b,c}, k) \rightarrow \tilde{\boldsymbol{W}}_{\boldsymbol{q}}^{\boldsymbol{b,c}}; c = [1,2]$
9: **Fusion step** $\Rightarrow \tilde{W}_F = min(\tilde{\boldsymbol{W}}_{\boldsymbol{q_{i,j}}}^{\boldsymbol{b,c}})$, with $i = 1, .., q$ and $j = 1, ..., q$
10: **Eigendecomposition of** $\tilde{W}_F \Rightarrow [U, D] = eig(\tilde{W}_F)$
11: **Compute the new image from each column** u_i **weighted by this square root of his eigenvalue** $(d_i) \Rightarrow I_{u_i} = u_i\sqrt{d_i}, I_{u_i} \in \mathbb{R}^{(q \times 1)}$
12: **Calculate the mutual information from the given image** (I_{u_i}) **and the reduced prior**$(\tilde{I}_{prior}) \Rightarrow MI(I_{u_i}, \tilde{I}_{prior}) = E_{P_{I_{u_i}\tilde{I}_{prior}}} log \frac{P_{I_{u_i}\tilde{I}_{prior}}}{P_{I_{u_i}} P_{\tilde{I}_{prior}}}$
13: **Select the eigenvector** (u_i) **and eigenvalue** (d_i) **that maximize MI** $(\tilde{\boldsymbol{C}}_{\boldsymbol{M}} \in \mathbb{R}^{q \times 1})$
14: **Up-sampling the reduced Change Map** $(\tilde{\boldsymbol{C}}_{\boldsymbol{M}})$: Considering $\boldsymbol{L_M}$, change the value of all pixels belonging to the same class to the value of the representative pixel in $\tilde{C}_M$. $\Rightarrow \tilde{\boldsymbol{C}}_{\boldsymbol{M}} \in \mathbb{R}^{q \times 1} \rightarrow \boldsymbol{C_M} \in \mathbb{R}^{m \times n}$

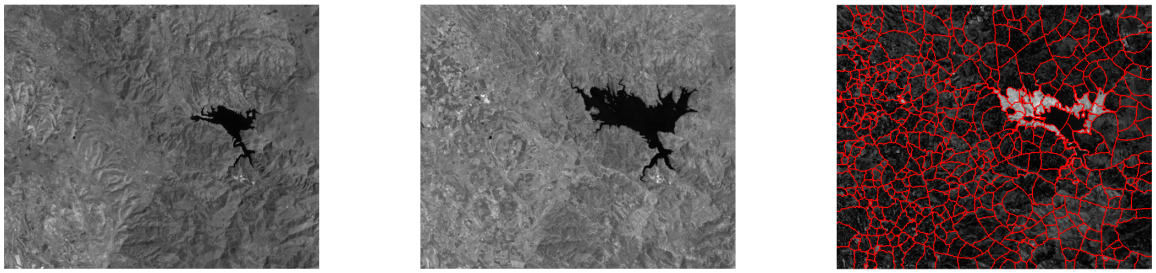


Figure 4.2: Graph-cut applied to Sardania dataset: From left to right, pre-event image, post-event image and Graph-cut applied with 500 classes.

Using the Algorithm 4, considering the datasets shown in section 3.5, and fixing **stop**, which is the parameter that modify the number of classes, so that we get sufficient number of classes in order to delineate the prior image capturing the image edges with detail, since the prior is the first approximation to the change map. The results of this approach, the **GBF-SEG-GSP** method and the **GBF-SEG-NNK** method, explained in detail in the following chapter, are shown in chapter 6.

Chapter 5

Spectral filtering

5.1 Introduction

Models based on spectral filtering of graph signals represent one of the most elaborated approaches of graph learning through signal representations. That is because these models consider the case where graph signals are generated through a filtering process to the input signal, e.g. a diffusion process [14]. Its importance resides in that filtering allows us to discriminate some frequency components of the input signal and consequently modify its spectrum. As in classical filtering, a graph filter can smooth the signal.

In this chapter, we introduce first some spectral filtering algorithms which model the graph signals as stationary processes on graphs [2] [3] and others as a spectral-domain transform, e.g. bilateral filter [27] and non-negative kernel (NNK) regression graph [4]. Simultaneously, we discuss the assumptions that the authors of these techniques make as well as their characteristics. Then, we choose the best algorithm to integrate with the **GBF-CD** method [13]. Next, we present the experiments and their corresponding results, and finally, we draw some conclusions.

5.2 Models based on spectral filtering of graph signals

Spectral filtering approaches are typically used to depict the propagation of a diffusion process. Hence, in comparison to the smoothness approach, spectral filtering suits better to spreading phenomena such as propagation of a heatwave in a region, virus in a population, trends in social media, among others circumstances.

Following we explain thoroughly the concepts behind some models based on spectral filtering. We will denote its assumptions, characteristics, advantages, and disadvantages. We will see that, as occurred in the smoothness approaches, these algorithms still are very limited in vertex quantity due to its computational complexity. Thus, we will still need to segment the images before the graph learning stage.

5.2.1 Network topology inference from spectral templates

In [3], Segarra et al. address the problem of identifying the graph structure $\mathcal{G}$ given the observations on its nodes. This graph $\mathcal{G}$ is undirected and aims to discover the hidden relationships between N nodes (vertex) produced by a diffusion process. Here, the observed signals are generated after an input signal $\boldsymbol{w}$ — normally a zero-mean white signal with covariance matrix $\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T] = \boldsymbol{I}$ — is diffused on the graph. That is to see in Equation 5.1. Their goal is to find a graph shift operator

(GSO) $\boldsymbol{S}$ —matrix which represents the graph structure— where each observation $\boldsymbol{x}$ is stationary. They used the (weighted) adjacency matrix or normalized Laplacian as GSO $\boldsymbol{S}$.

$$\boldsymbol{x} = \alpha_0 \prod_{l=1}^{\infty} (\boldsymbol{I} - \alpha_l \boldsymbol{S})\boldsymbol{w} = \sum_{l=0}^{\infty} \beta_l \boldsymbol{S}^l \boldsymbol{w} \tag{5.1}$$

Where $\boldsymbol{S}$ can be supposed to be symmetric and thus, diagonalizable. In Equation 5.2, orthogonal eigenvector matrix $\boldsymbol{V} := [\boldsymbol{v_1}, ..., \boldsymbol{v_N}]$ and the eigenvalue matrix $\boldsymbol{\Lambda} := \boldsymbol{diag(\lambda)}$ with $\boldsymbol{\lambda} := [\lambda_1, ...; \lambda_N]^T$, it holds that:

$$\boldsymbol{S} = \boldsymbol{V\Lambda V^T} = \boldsymbol{V diag(\lambda) V^T} \tag{5.2}$$

Stationarity [3]: Each observation $\boldsymbol{x}$ is stationary in GSO $\boldsymbol{S}$, if either it is generated by passing white inputs through a graph filter $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ as $\boldsymbol{H} := \sum_{l=0}^{L-1} h_l \boldsymbol{S^l}$ which is a polynomial of $\boldsymbol{S}$ where $\boldsymbol{h} := [h_0, ..., h_{L-1}]^T$ is the filter coefficients vector, or if its covariance matrix $\boldsymbol{C_x}$ is simultaneously diagonalizable with $\boldsymbol{S}$. This property is important to estimate the eigenvectors of GSO $\boldsymbol{S}$. Granted the first condition, the generative model of Equation 5.1 can be rewritten as Equation 5.3:

$$\boldsymbol{x} = \left(\sum_{l=0}^{L-1} h_l \boldsymbol{S}^l \right) \boldsymbol{w} = \boldsymbol{H}\boldsymbol{w} \tag{5.3}$$

Since the filter $\boldsymbol{H}$ can be expressed as a polynomial of $\boldsymbol{S}$ (Stationarity def.) and in turn the observation $\boldsymbol{x}$ in terms of $\boldsymbol{H}$ (Equation 5.3), after doing the calculations shown in Equation 5.4 we can conclude that the GSO $\boldsymbol{S}$ and the covariance matrix $\boldsymbol{C_x}$ of the observation vector $\boldsymbol{x}$ **share the same eigenvectors**.

$$\boldsymbol{C_x} := \mathbb{E}[\boldsymbol{x x^T}] = \mathbb{E}[\boldsymbol{Hw(Hw)^T}] = \boldsymbol{HH^T} = \boldsymbol{V} \left(\sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l \right)^2 \boldsymbol{V}^T \tag{5.4}$$

Then, having the eigenvectors of the sample covariance matrix, we need to find the eigenvalues to estimate GSO $\boldsymbol{S}$ and this process can be done encouraging solutions with some desired properties such as sparsity. i.e. solutions with few number of non-zero entries.

Next, we can extrapolate the theory aforementioned to the whole set of observations $\boldsymbol{X}$ (graph signals). Recall that the main goal of this algorithm is to find the sparsest GSO S which better describes the structure of the graph signal $\boldsymbol{x}$, that is the relation between nodes under a diffusion process, taking into account all samples of $\boldsymbol{x}$.

It is important to observe that, since this algorithm relies on the sample covariance matrix, it has better performance, namely less recovery error, only when we have a large number of observations, about more than one thousand [3] [14].

Computational complexity: The algorithm proposed by Segarra in [3] has a polynomial complexity of $\mathcal{O}(N^3)$ when computes the eigenvectors and also per each iteration when solves the minimization problems seeking sparse solutions.

Therefore, since we only have one observation per event (pre- or post-event) and a huge number of nodes i.e. pixels, this algorithm does not suit the change detection application in the way that it is conceived in this thesis.

5.2.2 Inference of graph diffusion processes from observations of stationary signals

In [2], Pasdeloup et al. continue using the useful property of stationary signals pointed out in the Segarra's algorithm but here the graph shift operator is a diffusion matrix instead of the (weighted) adjacency matrix or normalized Laplacian matrix.

Recall that the first step is to estimate the covariance matrix eigenvectors of the observations (signals diffused on the graph). But, since the covariance matrix is not obtainable in practical cases, they compute the sample covariance matrix $\tilde{\boldsymbol{\Sigma}}$ as follow:

$$\tilde{\boldsymbol{\Sigma}} \triangleq \frac{1}{M-1}(\boldsymbol{X}-\boldsymbol{M})(\boldsymbol{X}-\boldsymbol{M})^T, \tag{5.5}$$

Where $M(i,j) \triangleq \frac{1}{M}\sum_{k=1}^{M} X(i,k)$ is an $N \times M$ matrix with each row containing the mean signal value for the associated vertex.

After that, they consider a polytope or set of possible solutions that contains the estimated covariance matrix and all its powers. Then, they select one "point" of the polytope favoring solutions with certain characteristics such as sparsity or simplicity, the latter means solutions where its nodes do not have self-loops i.e. the diagonal entries of the associated adjacency matrix are null.

Like in Segarra's algorithm, Pasdeloup's approach is based on the sample covariance matrix, which requires a large number of signals so that its eigenvectors converge to the ones of the covariance matrix and thus assuage the recovery error [2]. Additionally, Pasdeloup's algorithm has a polynomial complexity of $\mathcal{O}(N^3)$ in time and space. For both reasons, we conclude that stationarity-based spectral filtering models that leverage the covariance matrix do not qualify to do the integration with the GBF-CD method.

5.2.3 Bilateral Filter

In [27], Gadde, Narang, and Ortega revisit the original Bilateral Filter (BF) algorithm proposed by Tomasi and Manduchi to give a graph spectral interpretation of it and as well to provide some extensions of it. BF is a precursory graph-based technique born merely from the intuition that the relation between two nodes is molded not only because of the geometric distance between them but also the photometric distance they differ from each other, which provides local adaptability to the

given data. Considering an input image $\boldsymbol{x_{in}}$ to the BF the value at each position in the output image $\boldsymbol{x_{out}}$ is given by the weighted average of the pixels in $\boldsymbol{x_{in}}$:

$$\boldsymbol{x}_{out}(j) = \sum_i \frac{w_{ij}}{\sum_i w_{ij}} \boldsymbol{x}_{in}(i) \tag{5.6}$$

As mentioned above, the weights w_{ij} depend on both the Euclidean and photometric distance between the pixels $\boldsymbol{x_{in}}(i)$ and $\boldsymbol{x_{in}}(j)$. Let p_i denote the position of the pixel i. The weights are given by

$$w_{ij} = \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_d^2}\right).\exp\left(-\frac{(\boldsymbol{x}_{in}(i) - \boldsymbol{x}_{in}(j))^2}{2\sigma_r^2}\right) \tag{5.7}$$

Where σ_d and σ_r are the filter parameters [27]. As we can see, the closer in position or the more similar in intensity two pixels are, the greater the weight. Considering $\boldsymbol{W}$ as the adjacency matrix and $\boldsymbol{D}$ the diagonal degree matrix where each diagonal element $\boldsymbol{D_{jj}} = \sum_i w_{ij}$, the filtering operations in Equation 5.6 can be written as

$$\boldsymbol{x}_{out} = \boldsymbol{D}^{-1}\boldsymbol{W}\boldsymbol{x}_{in} \tag{5.8}$$

As mentioned in subsection 3.1.2, the decomposition of the Laplacian matrix into vectors $\boldsymbol{u_i}$ and eigenvalues λ_k allows to obtain its spectral representation, where the eigenvalues λ_k can be treated as graph frequencies, and the eigenvectors u_i the magnitude of the graph frequencies. Therefore, the Graph Fourier Transform ($\boldsymbol{GFT}$) of a signal $\boldsymbol{x}$ is defined as its projection onto the eigenvectors of the graph. $\widetilde{\boldsymbol{x}} = U^t\boldsymbol{x}$. The inverse $\boldsymbol{GFT}$ is given by $\boldsymbol{x} = U\widetilde{\boldsymbol{x}}$

Similar to conventional signal processing, graph spectral filtering is defined as

$$\widetilde{\boldsymbol{x}}_{out}(\lambda_i) = \boldsymbol{h}(\lambda_i)\widetilde{\boldsymbol{x}}_{in}(\lambda_i) \tag{5.9}$$

Where $\boldsymbol{h}(\lambda_i)$ is the spectral response of the filter. Using the definition of $\boldsymbol{GFT}$ and the diaginalized form of the normalized Laplacian matrix $\mathcal{L}$, we can write graph spectral filtering as

$$\boldsymbol{x}_{out} = Uh(\Lambda)U^t\boldsymbol{x}_{in} = h(\mathcal{L})\boldsymbol{x}_{in} \tag{5.10}$$

Considering Equation 5.8 we can rewrite BF as:

$$\begin{aligned} \boldsymbol{x}_{out} &= \boldsymbol{D}^{-1/2}\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}\boldsymbol{D}^{1/2}\boldsymbol{x}_{in} \\ \Rightarrow \boldsymbol{D}^{1/2}\boldsymbol{x}_{out} &= (\boldsymbol{I} - \mathcal{L})\boldsymbol{D}^{1/2}\boldsymbol{x}_{in} \end{aligned} \tag{5.11}$$

From this equation, we can see that the BF is a graph transform, similar to the one in Equation 5.10, operating on the normalized input signal $\hat{\boldsymbol{x}}_{in} = \boldsymbol{D}^{1/2}\boldsymbol{x}_{in}$ producing the normalized output $\hat{\boldsymbol{x}}_{out} = \boldsymbol{D}^{1/2}\boldsymbol{x}_{out}$. This normalization allows us to define the BF in terms on the non-negative definite matrix $\boldsymbol{\mathcal{L}}$ and thus have a spectral interpretation. It also ensures that a constant signal, when normalized, is an eigenvector of $\boldsymbol{\mathcal{L}}$ associated with zero eigenvalue. Following Equation 5.10 we have,

$$\hat{\boldsymbol{x}}_{out} = \boldsymbol{U}(\boldsymbol{I} - \boldsymbol{\Lambda})\boldsymbol{U}^t\hat{\boldsymbol{x}}_{in} \tag{5.12}$$

This shows that the BF is a frequency selective graph transform with a spectral response $h_{BF}(\lambda_i) = 1 - \lambda_i$ which corresponds to linear decay. The BF tries to preserve the low frequency components and attenuate the high frequency components. The BF is used in a iterative form in many applications. One method is by using fixed weights at each iteration as calculated from the initial image. Iterating preserves strong edges while removing weaker details. The BF iterated k-times can be written in matrix notation as

$$\boldsymbol{x}_{out} = (\boldsymbol{D}^{-1}\boldsymbol{W})^k\boldsymbol{x}_{in} = (\boldsymbol{I} - \mathcal{L}_r)^k\boldsymbol{x}_{in} \tag{5.13}$$

Where $\mathcal{L}_r = \boldsymbol{D}^{-1}\boldsymbol{L}$ is called the random walk Laplacian matrix. It can be shown that any graph transform $h(\mathcal{L}_r)$ can be written in terms of $\mathcal{L}$ as $h(\mathcal{L}_r) = \boldsymbol{D}^{-1/2}h(\mathcal{L})\boldsymbol{D}^{1/2}$. Using this, we can rewrite Equation 5.13 as

$$\hat{\boldsymbol{x}}_{out} = \boldsymbol{U}(\boldsymbol{I} - \boldsymbol{\Lambda})^k\boldsymbol{U}^t\hat{\boldsymbol{x}}_{in} \tag{5.14}$$

Therefore, the iterative application of the BF suppresses more of the high frequency component, that is, the higher the k, the more selective the filter will be.

5.2.4 Non-negative kernel regression (NNK) graph

In [4], Shekkizhar and Ortega present a modified version of the original NNK algorithm [28] used for graph construction in order to adapt this to image processing tasks such as filtering, denoising, and energy compaction. The NNK algorithm starts using a kernel-based filter, Ortega et. al use the bilateral filter in [4].

As mentioned above, the bilateral filtering model starts with the use of the kernel from the geometric and photometric distances between the pixels as shown in Equation 5.7. Note that $w_{i,j}$ can be viewed as the inner product of two kernel functions ϕ_i and ϕ_j. Therefore, the NNK proposes to solve:

$$\min_{\theta:\theta\geq 0} \quad \|\phi_i - \Phi_S\boldsymbol{\theta}\|_{l_2}^2 \tag{5.15}$$

Where ϕ_i is to be represented by a linear combination of atoms (with weights given by $\boldsymbol{\theta}$) from a dictionary obtained from a set S of neighbors (Φ_S) [4]. It is possible to associate a geometric interpretation to conditions that determine whether two nodes are connected (kernel ratio interval or KRI conditions). Thus, in contrast to K-Nearest Neighbor (KNN) graphs, even when window size w increases the number of connected nodes does not necessarily grow, so that NNK graphs tend to be better at reflecting the actual data topology. A key contribution of [4] is to adapt NNK to image data, and in particular taking into account the special characteristics of image to apply the KRI conditions more effectively. Thus, this model uses bilateral filtering, w-size windows and the KRI theorem to obtain a sparse graph representation of images. This can be seen in more detail in [4].

Their algorithm stands out due to its novelty and the efficiency it brings. They adapt the non-negative kernel regression techniques using a kernel-based filter —namely the Bilateral Filter— and take advantage of the pattern described by the pixel positions on images to present a sparser and more efficient graph representation of the image that can be computed faster than the original kernel-based filter. For that reason, it has big potential to be the chosen algorithm to do the integration with the GBF-CD in the graph learning stage.

The principal issue with this algorithm is the number of parameters we have to fix, specifically filter position parameter σ_d, filter intensity parameter σ_f, and window size ω. In order to simplify the experiments, we decide to keep the values used by Shekkizhar and Ortega for $\sigma_d = 2$ and $\omega = 5$ in their Matlab codes [4]. We let free the σ_f parameter and, as for the k-parameter of the **SEG-GSP** approach (smoothness model), we carried-out a grid-search varying it. The results are shown in the next sections.

Algorithm 5 *Precompute*(ω) :

Require: window size ω
Ensure: Ordered window pixels $\mathbb{S}$, Threshold factor Δ
1: $\boldsymbol{x}$ = pixel positions in $\omega \times \omega$, $\mathbb{S} = \{$pixels in $\omega \times \omega\}$
2: window center = i, $\mathbb{S} = \mathbb{S}$-{i}
3: $\mathbb{S}$ = sort $\mathbb{S}$ by $\|\boldsymbol{x}_j - \boldsymbol{x}_i\|^2 \quad \forall j \in \mathbb{S}$
4: **for** j,k in $\mathbb{S}$ **do**
5: $\quad \Delta_{j,k} = (\boldsymbol{x}_k - \boldsymbol{x}_j)^T(\boldsymbol{x}_j - \boldsymbol{x}_i)$
6: **end for**

We make use of Algorithm Algorithm 5 *Precompute* to construct the NNK graph as follows in Algorithm Algorithm 6:

Algorithm 6 Graph Learning NNK (**GL-NNK**) [4]

Require: Filter position parameter σ_d, Filter intensity parameter σ_f, window size ω
Ensure: Graph Adjacency $\mathbf{W}$

$\mu = \left(\frac{\sigma_f}{\sigma_d}\right)^2$, $[\mathbb{S}*, \mathbf{\Delta}] = Precompute(\omega)$
for each pixel i **do**
  $\mathbb{S} = \mathbb{S}*$, $\mathbb{P} = \{\}$
  for j in $\mathbb{S}$ **do**
    /* pick neighbors in spatial distance sorted order*/
    If j in $\mathbb{P}$ **then**:
      continue //skip if pruned
    $\mathbb{P} = \mathbb{P} + \{j\}$
    for pixel k in $\mathbb{S}$ with $\Delta_{j,k} \geq 0$ **do**
      /*consider pixels in same direction as j*/
      If $(\boldsymbol{f}_j - \boldsymbol{f}_k)^T(\boldsymbol{f}_j - \boldsymbol{f}_i) \leq \mu\Delta_{j,k}$ **then**
        $\mathbb{P} = \mathbb{P} + \{k\}$, $\mathbb{S} = \mathbb{S} - \{k\}$
    end for
  end for
  $\mathbf{W}_{i,\mathbb{S}} = \mathbf{K}_{i,\mathbb{S}}$, $\mathbf{W}_{i,\mathbb{S}^c} = 0$
end for

5.3 Spectral Filtering in Change Detection

To apply the spectral filtering approach in change detection (CD) task, we follow the comprehensive **GBF-SEG-GSP** algorithm (See Algorithm 4) using the **GL-NNK** algorithm (See Algorithm 6) which replaces the **GL-LogDeg** algorithm (See Algorithm 3) (smoothness approach) in the graph learning stage. We called the union of these two algorithms (Algorithm 4 and Algorithm 6) **GBF-SEG-NNK**.

We assess the performance of the **GBF-SEG-NNK** model for CD on 14 different datasets. These results and their corresponding analysis are elaborated in chapter 6.

Chapter 6

Results and discussion

From Figure 6.1 to Figure 6.14 we show the change map detected with respect to missed alarms (M), false alarms (FA) and correct changed pixels (C). Where each row corresponds to a data set, and each column from left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

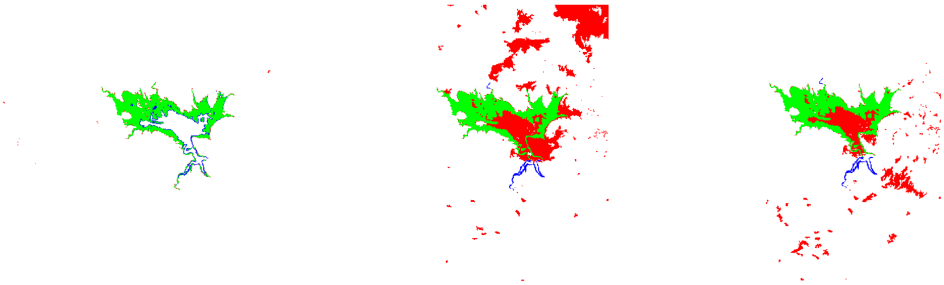


Figure 6.1: Sardania Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

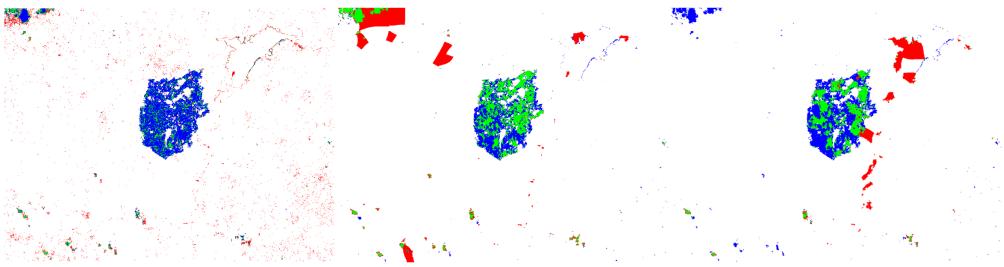


Figure 6.2: Omodeo Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

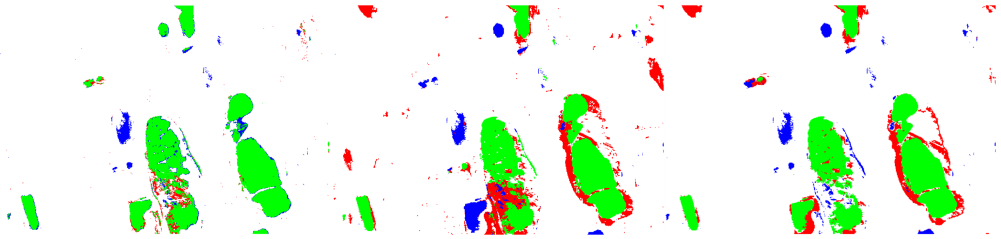


Figure 6.3: Alaska Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

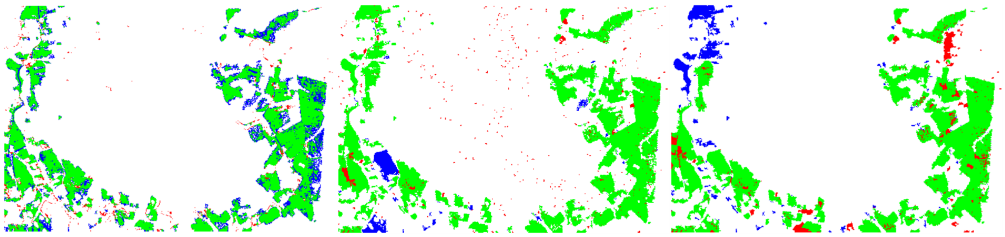


Figure 6.4: Madeirinha Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

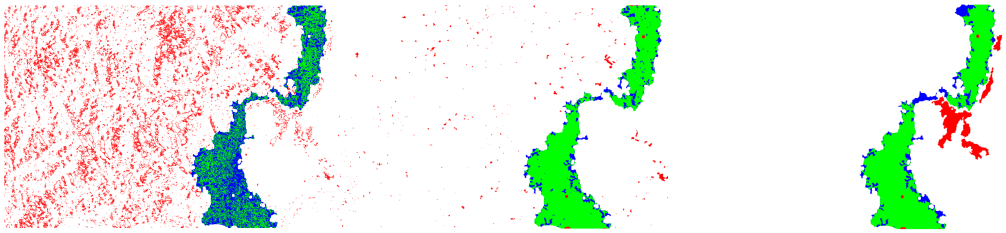


Figure 6.5: Katios Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

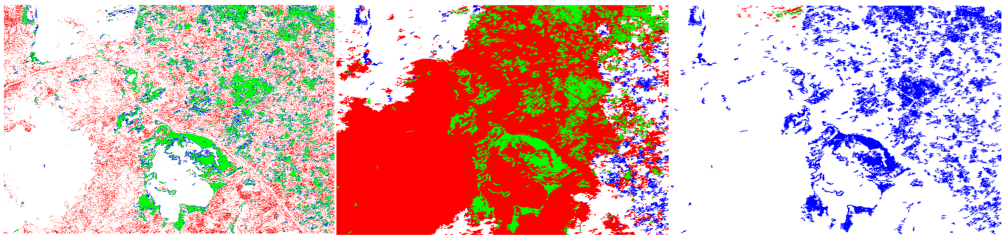


Figure 6.6: Dique Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

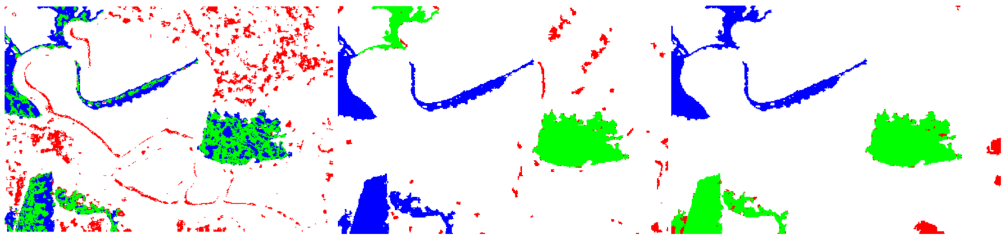


Figure 6.7: San Francisco Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

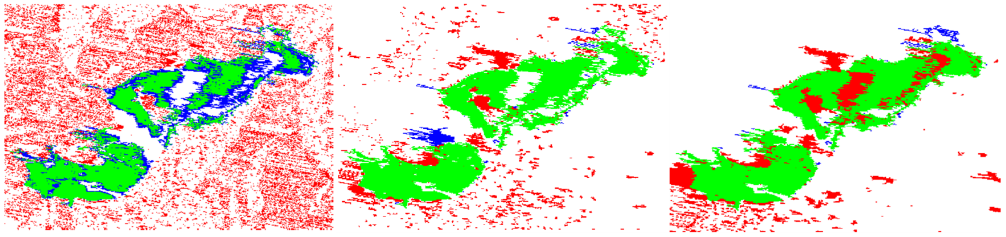


Figure 6.8: Wenchuan Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

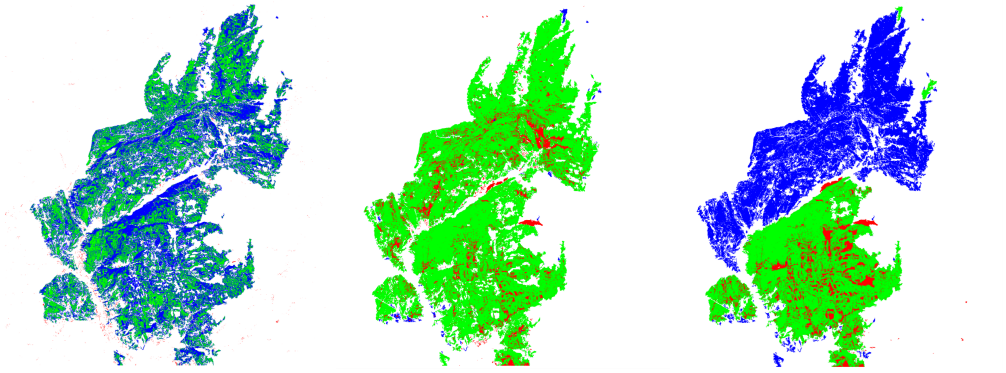


Figure 6.9: Canada Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

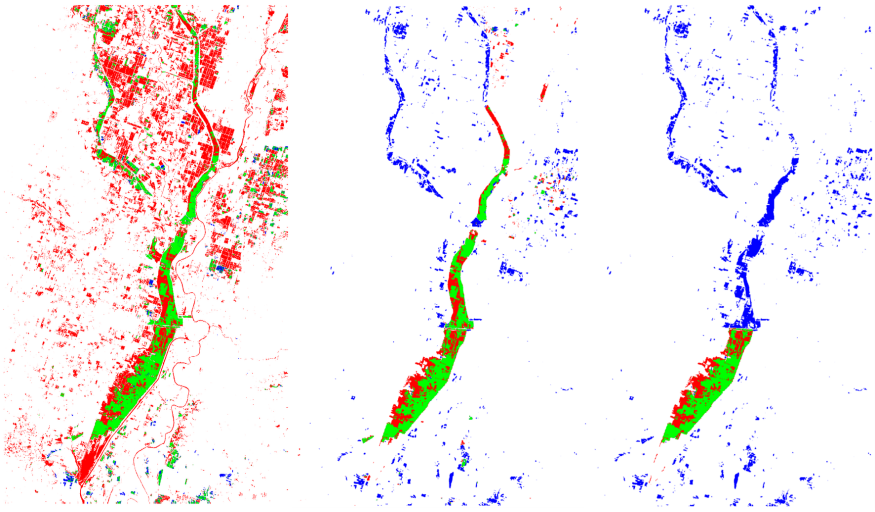


Figure 6.10: California Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

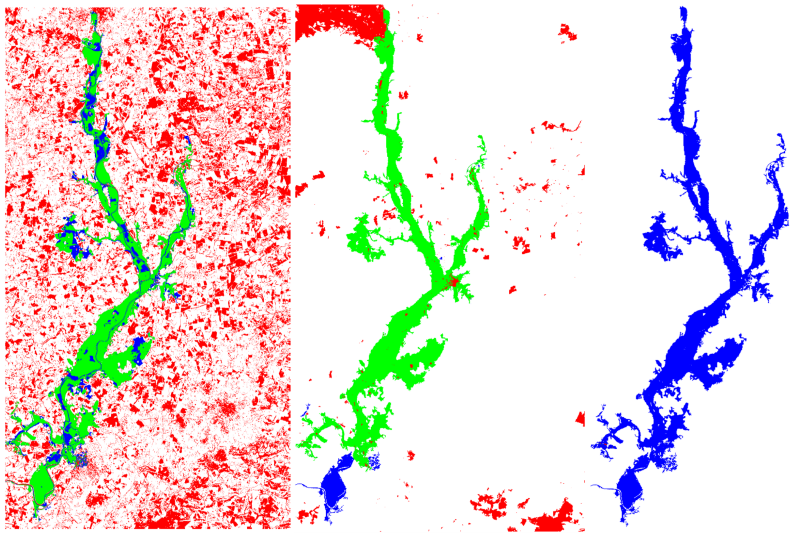


Figure 6.11: Contest Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

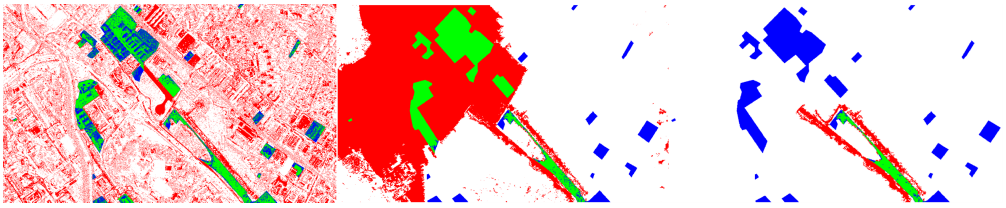


Figure 6.12: Toulouse Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

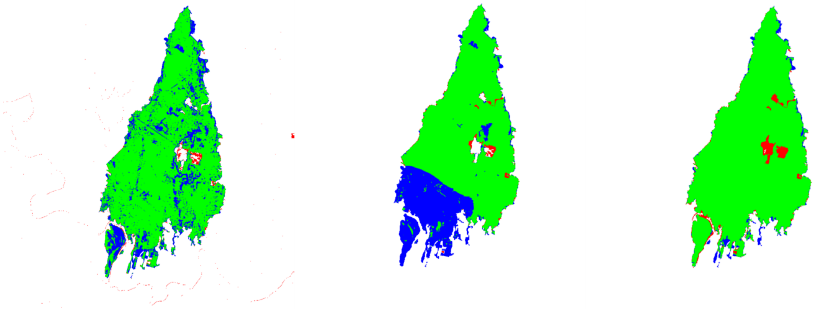


Figure 6.13: Bastrop Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

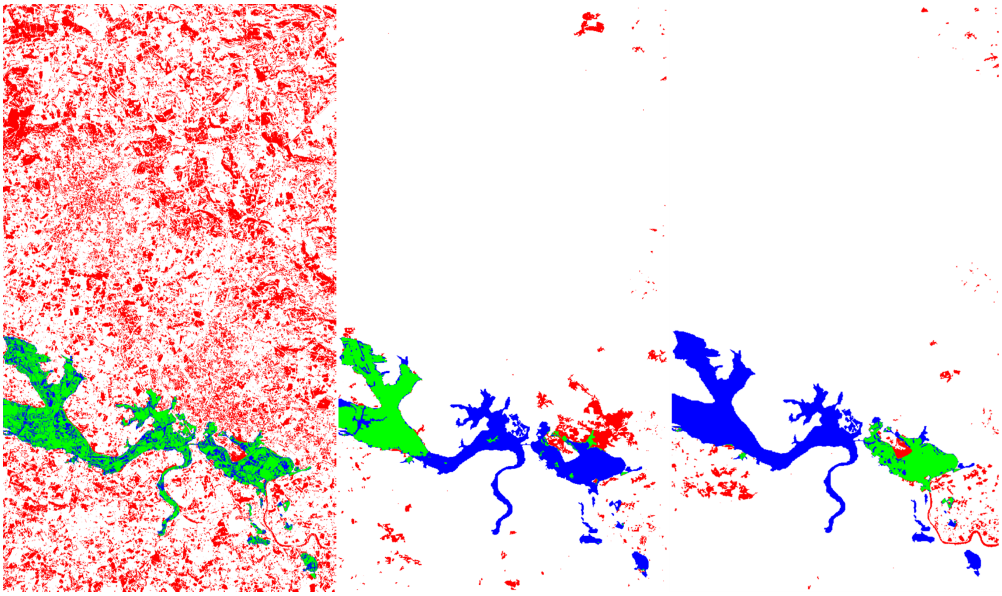


Figure 6.14: Gloucester Change Maps. From left to right to the **GBF-CD**, the **GBF-SEG-GSP** and **GBF-SEG-NNK** methods.

We should clarify that the results shown above are sensitive to the Matlab version used and were obtained in Matlab2018a. We found that the problem resides specifically in the graph cut segmentation, so we conclude that some Matlab libraries are responsible of it. Moreover, the results needed between 1-20 GB of RAM mainly due to the segmentation stage and after that between 6 and 8 GB of RAM. Furthermore, they were obtained in a server with the following characteristics detailed in the Table 6.1.

Feature	**Description**
Processor	2 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz Processors for 24 Physical Cores and 48 Process Threads
RAM	252 GB @2400MHz
Graphics	8 Graphics Cards Nvidia GTX 1080TI 11178MiB RAM "approx 11.72 GB"
Boot Drive	1x 977GB SDD for / and /home
Data Drive	3x 2TB HDD in Raid5 for /data

Table 6.1: Server specifications

As shown above in figures 6.1-6.14, the change maps obtained with the two models, **GBF-SEG-GSP** and **GBF-SEG-NNK**, visually have a better performance than the **GBF-CD** method overall.

Note that both novel methods does not work properly in some regions for datasets such as Dique and Toulouse, which have regions with "isolated" or small pixels of change, that we name "little islands". This no-detection failure can be a consequence of the segmentation step (Graph-cut method [5]) and can be studied in more detail in future works to reduce the Missed Alarms in such "little islands".

In contrast, we did some tests with different number of classes and we noticed that despite having fewer classes, it can be obtained a similar performance than with more classes. In some cases, it is worthwhile to work with fewer classes to avoid an increase in the computational cost without better performance.

In addition, it is important to mention that the segmentation process seems to divide the images in big chunks compared to the finer resolution which provides the **GBF-CD** method. Because of that, the individual false alarm patches are bigger in the version with fewer classes.

The comparative tables between the different approaches according to the metrics described in subsection 3.3.2 are presented below:

SARDANIA									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 93$	12.33	0.17	87.67	93.96	90.43	0.53	19.208
SEG-GSP (0.0001)	852	$k = 6$	5.20	5.51	94.81	34.43	48.27	5.50	114.713
SEG-NNK (0.0001)	852	$\sigma_f = 0.105$	5.20	2.71	94.81	51.65	65.55	2.78	53.6951

Table 6.2: Models performance for dataset Sardania

OMODEO									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 93$	77.00	1.26	22.99	47.26	28.73	4.83	91.62
SEG-GSP (0.015)	1281	$k = 1188$	44.83	2.08	55.17	56.78	53.82	4.09	458.54
SEG-NNK (0.015)	1281	$\sigma_f = 0.389$	62.40	1.45	37.60	56.16	42.89	4.32	413.59

Table 6.3: Models performance for dataset Omodeo

ALASKA									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 2$	11.66	0.87	88.34	92.36	89.17	2.02	3.62
SEG-GSP (0.001)	836	$k = 266$	14.98	5.47	85.02	65.06	70.09	6.49	122.17
SEG-NNK (0.001)	836	$\sigma_f = 0.213$	11.60	3.44	88.40	75.49	79.01	4.31	102.28

Table 6.4: Models performance for dataset Alaska

MADEIRINHA									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 9$	24.44	1.13	75.56	94.06	80.46	5.60	4.10
SEG-GSP (0.003)	511	$k = 353$	6.07	1.31	93.93	94.44	92.81	2.23	110.75
SEG-NNK (0.003)	511	$\sigma_f = 0.011$	10.58	2.23	89.42	90.51	87.60	3.83	102.21

Table 6.5: Models performance for dataset Madeirinha

KATIOS									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 60$	52.05	10.63	47.95	34.74	31.96	15.00	34.48
SEG-GSP (0.001)	1935	$k = 574$	12.69	0.66	87.31	93.97	89.44	1.93	1350.14
SEG-NNK (0.001)	1935	$\sigma_f = 0.113$	14.74	1.76	85.26	85.08	83.42	3.13	1258.39

Table 6.6: Models performance for dataset Katios

DIQUE									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 240$	30.42	13.69	69.57	48.11	47.26	16.27	103.911
SEG-GSP (0.000008)	2123	$k = 594$	18.48	71.24	81.52	17.28	4.08	63.09	311.1229
SEG-NNK (0.000008)	2123	$\sigma_f = 0.086$	99.48	0.21	0.52	30.97	0.52	15.53	205.4583

Table 6.7: Models performance for dataset Dique

SAN FRANCISCO									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 4$	48.82	7.64	51.17	49.34	42.85	12.87	3.21
SEG-GSP (0.000235)	209	$k = 140$	44.30	3.26	55.70	71.29	57.84	8.47	21.58
SEG-NNK (0.000235)	209	$\sigma_f = 0.033$	34.77	1.05	65.23	90.08	72.76	5.33	20.08

Table 6.8: Models performance for dataset San Francisco

WENCHUAN									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 39$	35.82	22.52	64.17	37.25	32.39	24.81	6.24
SEG-GSP (0.0006)	1091	$k = 525$	4.95	7.95	95.05	71.35	76.98	7.44	70.74
SEG-NNK (0.0006)	1091	$\sigma_f = 0.044$	2.97	9.26	97.03	68.58	75.39	8.18	49.58

Table 6.9: Models performance for dataset Wenchuan

CANADA									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 110$	54.10	0.38	45.90	97.86	54.42	15.27	240.74
SEG-GSP (0.0002)	2620	$k = 1662$	1.02	4.94	98.98	88.49	90.73	3.85	2446.50
SEG-NNK (0.0002)	2620	$\sigma_f = 0.003$	55.08	3.61	44.92	82.69	48.09	17.87	2277.12

Table 6.10: Models performance for dataset Canada

CALIFORNIA									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 270$	11.93	11.79	88.06	25.44	35.07	11.80	921.62
SEG-GSP (0.000115)	1374	$k = 336$	49.23	1.73	50.77	57.36	51.89	3.80	3999.13
SEG-NNK (0.000115)	1374	$\sigma_f = 0.347$	70.92	0.88	29.08	60.19	37.43	3.94	3960.73

Table 6.11: Models performance for dataset California

CONTEST									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 12$	23.80	26.57	76.19	21.26	22.86	26.33	96.46
SEG-GSP (0.00007)	1580	$k = 431$	8.53	3.95	91.47	68.59	76.04	4.34	6177.81
SEG-NNK (0.00007)	1580	$\sigma_f = 0.06$	99.99	0.00	0.01	23.62	0.02	8.61	6128.41

Table 6.12: Models performance for dataset Contest

TOULOUSE									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 96$	54.27	17.33	45.72	18.57	17.02	20.27	839.94
SEG-GSP (0.0000095)	2072	$k = 269$	22.46	36.13	77.55	15.66	14.77	35.04	14644.66
SEG-NNK (0.0000095)	2072	$\sigma_f = 0.481$	87.87	2.40	12.13	30.46	13.43	9.20	14550.70

Table 6.13: Models performance for dataset Toulouse

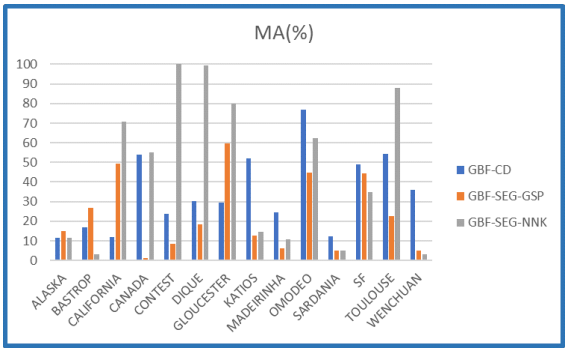
BASTROP									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 96$	16.83	0.23	83.16	97.71	88.75	1.99	109.35
SEG-GSP (0.0025)	578	$k = 11$	26.67	0.16	73.33	98.16	82.34	2.98	382.21
SEG-NNK (0.0025)	578	$\sigma_f = 0.003$	2.95	0.43	97.05	96.42	96.34	0.70	359.29

Table 6.14: Models performance for dataset Bastrop

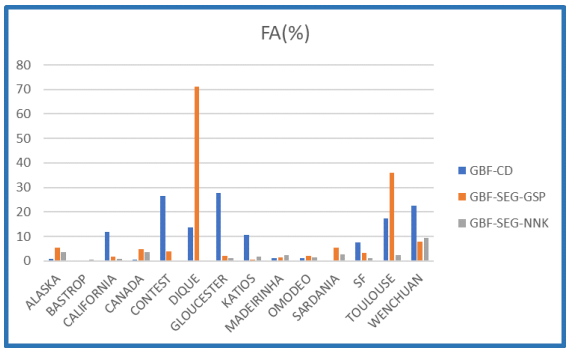
GLOUCESTER									
METHOD (stop)	**CLASSES**	**PAR. BEST VAL.**	**MA(%)**	**FA(%)**	**P(%)**	**R(%)**	**K(%)**	**OE(%)**	**ET (s)**
GBF-CD	-	$n = 76$	29.39	27.71	70.60	14.86	15.62	27.82	543.65
SEG-GSP (0.000015)	1774	$k = 218$	59.57	1.95	40.43	58.70	45.00	5.65	5721.93
SEG-NNK (0.000015)	1774	$\sigma_f = 0.415$	79.96	1.21	20.04	53.10	26.52	6.27	5656.97

Table 6.15: Models performance for dataset Gloucester

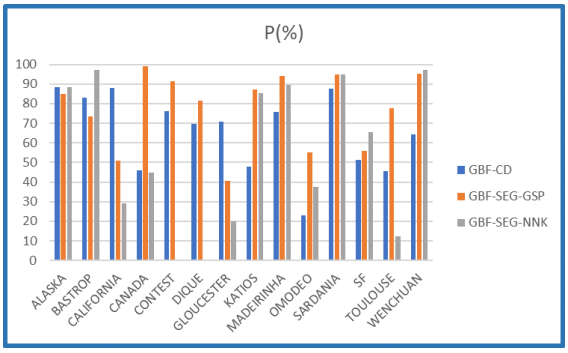
In tables Table 6.2-Table 6.15, we can see that the execution time needed to implement the **GBF-SEG-GSP** and **GBF-SEG-NNK** models is largely increased mainly attributed to the initial segmentation stage. In spite of that, we observe in the new models an improvement in terms of metrics in comparison to the original model (**GBF-CD**). To make this clearer, following we present the results through bar charts of each metric for the three models applied to all the datasets and elucidate some interpretations of them.



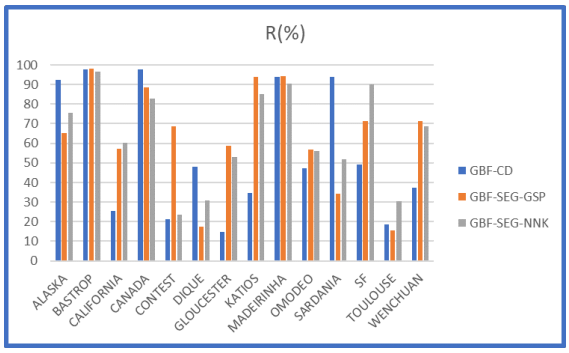
(a) Missed alarms



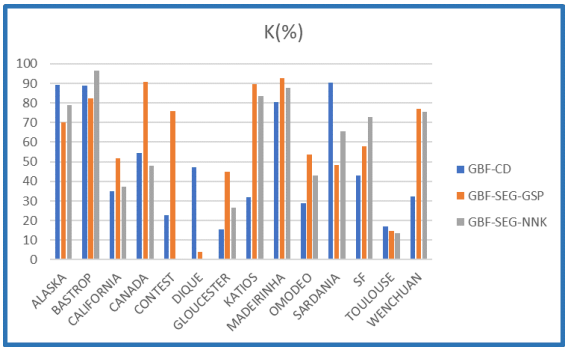
(b) False alarms



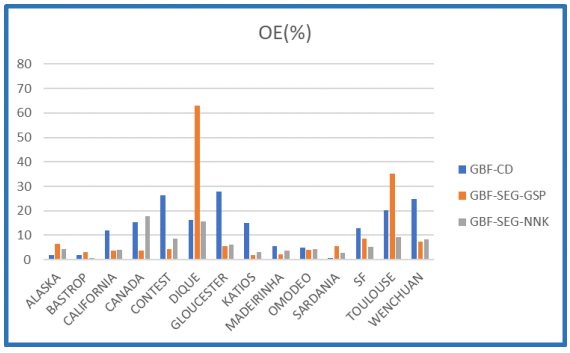
(c) Precision



(d) Recall



(e) Cohen's Kappa

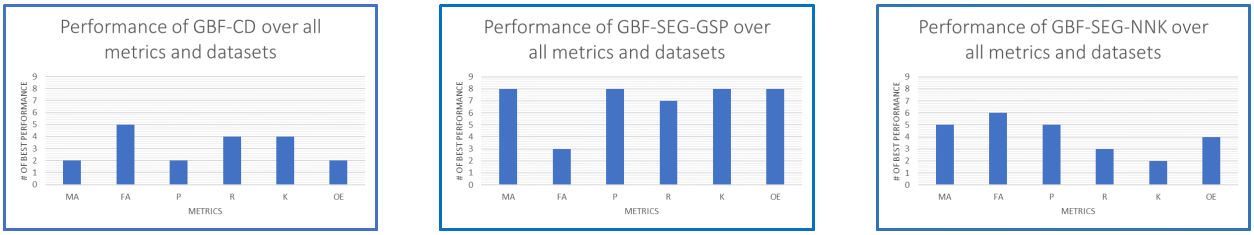


(f) Overall Error

Figure 6.15: Metrics of the three methods for the 14 datasets

The above graphs in Figure 6.15 show that the **GBF-SEG-GSP** algorithm had the best performance on almost all metrics, except FA. This is because, as previously stated, due to the segmentation required to reduce the computational cost, only the representative pixels of each class are analyzed. If one of those pixels is considered as a change zone, all pixels corresponding to that class will also be considered as such, so it is possible that false alarms increase. This can be seen more clearly in datasets such as Dique, for which the number of classes was not sufficient to surround the change zones, so it is possible that, by increasing the classes, we get a better performance in this

dataset. However, this could not be verified due to the high computational demand. This factor could be considered for future improvements of the model.



(a) GBF-CD (b) GBF-SEG-GSP (c) GBF-SEG-NNK

Figure 6.16: Model's performance - Bar charts that evaluate the performance of each method over all metrics and datasets. The count for each method in one of the six possible metrics means that in one dataset, the model outperformed all the competing methods in that metric.

In Figure 6.16, we observe that the sum of the best performance for the three different models in terms of MA and P is equal to 15, while the number of datasets is 14. That is because the results for both new models in Sardania dataset is equal in these two metrics (MA and P) and therefore, we counted it as a point in best performance for both of them.

As can be seen in Figure 6.16, the **GBF-SEG-GSP** algorithm obtained the best performance in all metrics except false alarms compared to the **GBF-CD** algorithm. In addition, considering Cohen's Kappa (K) as the most significant metric, the **GBF-SEG-GSP** managed to outperform the original model (**GBF-CD**) in 9 of the 14 datasets. On the other hand, the **GBF-SEG-NNK** algorithm, also had a higher performance than the original algorithm outperforming it in 8 of the 14 datasets. Therefore, it is possible to state that the application of the signal representation perspective in the graph learning stage together with the adaptation of the graph-cut segmentation method, improved the performance of the original model (**GBF-CD**).

Chapter 7

Conclusions

The addition of the two approaches based on signal representation perspectives (Signal Smoothness and Spectral Filtering) for the graph learning stage instead of applying a Gaussian kernel in the **GBF-CD** model, managed to improve the results. On one hand, the application of the Smoothness approach (**GBF-SEG-GSP**), managed to overcome the original model (**GBF-CD**) in 9 of the 14 datasets, considering the Kappa (K) as the most representative metric. On the other hand, the Spectral filtering approach (**GBF-SEG-NNK**), managed to surpass it in 8 of the 14 datasets. Considering that Jimenez et al. outperformed the **rR**, **rrR**, **KI**, and **U-CD-HPT** algorithms in 8 of the 14 datasets and that our both new approaches beat Jimenez's model, we can state that our results are promising. However, the use of Graph-cut method instead of using Nyström to reduce the computational cost, can lead to a reduction in performance in datasets that require a finer segmentation such as the so-called "little islands". One of the future improvements of the models is to automate the parameter's choice, so that no grid search is required to obtain the best performance. An additional task is to reduce the computational cost of the segmentation stage, testing another segmentation algorithms, and a final task is to find a better expression to calculate the prior, since, for example in datasets such as Sardania, Contest and Dique, this first approximation differs substantially from the ground truth.

Bibliography

[1] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[2] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 3, pp. 481–496, 2018.

[3] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, 2017.

[4] S. Shekkizhar and A. Ortega, "Efficient Graph Construction For Image Representation," vol. 2, no. 1, pp. 1956–1960, 2020.

[5] S. Gou and T. Yu, "Graph based sar images change detection," in *2012 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2152–2155, 2012.

[6] V. Kalofolias and N. Perraudin, "Large scale graph learning from smooth signals," *arXiv preprint arXiv:1710.05654*, 2019. Available: https://arxiv.org/pdf/1710.05654.pdf. [Accessed: 01- Aug- 2020].

[7] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014. Available at: https://epfl-lts2.github.io/gspbox-html/. [Accessed: 01- Aug- 2020].

[8] H. Ritchie, "Natural disasters," *Our World in Data*, 2014. Available: https://ourworldindata.org/natural-disasters. [Accessed: 01 - May -2020].

[9] M. Zanetti, F. Bovolo, and L. Bruzzone, "Rayleigh-rice mixture parameter estimation via em algorithm for change detection in multispectral images," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5004–5016, 2015.

[10] M. Zanetti and L. Bruzzone, "A theoretical framework for change detection based on a compound multiclass statistical model of the difference image," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 1129–1143, 2018.

[11] L. T. Luppino, F. M. Bianchi, G. Moser, and S. N. Anfinsen, "Unsupervised image regression for heterogeneous change detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 9960–9975, 2019.

[12] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern recognition*, vol. 19, no. 1, pp. 41–47, 1986.

[13] D. A. Jimenez-Sierra, H. D. Benítez-Restrepo, H. D. Vargas-Cardona, and J. Chanussot, "Graph-based data fusion applied to: Change detection and biomass estimation in rice crops," *Remote Sensing*, vol. 12, p. 2683, Aug 2020.

[14] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[15] M. D. Mura, S. Prasad, F. Pacifici, P. Gamba, J. Chanussot, and J. A. Benediktsson, "Challenges and opportunities of multimodality and data fusion in remote sensing," *Proceedings of the IEEE*, vol. 103, no. 9, pp. 1585–1601, 2015.

[16] L. Stankovic, D. P. Mandic, M. Dakovic, I. Kisil, E. Sejdic, and A. G. Constantinides, "Understanding the basis of graph signal processing via an intuitive example-driven approach [lecture notes]," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 133–145, 2019.

[17] NASA, "Landsat image gallery." [Online], 2020. Available: https://landsat.visibleearth.nasa.gov/. [Accessed: 02 - May - 2020].

[18] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering, keele university," *UK EBSE-2007-12007*, 2007.

[19] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph spectral image processing," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 907–930, 2018.

[20] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2017.

[21] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nyström method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 214–225, 2004.

[22] D. J. Bartholomew, M. Knott, and I. Moustaki, *Latent Variable Modelsand Factor Analysis: A Unified Approach*. Hoboken, NJ, USA: Wiley, 3rd ed., 2011.

[23] A. Basilevsky, *Statistical Factor Analysis and Related Methods*. Hoboken,NJ, USA: Wiley, 1994.

[24] V. Kalofolias, "How to learn a graph from smooth signals," in *The 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*, Journal of Machine Learning Research (JMLR), 2016.

[25] M. Grant and S. Boyd, "Cvx: Matlab software for disciplined convex programming, version 2.2," Jan. 2020. Available: http://cvxr.com/cvx.

[26] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to InformationRetrieval*. Cambridge Univ. Press, 2008.

[27] A. Gadde, S. K. Narang, and A. Ortega, "BILATERAL FILTER : GRAPH SPECTRAL INTERPRETATION AND EXTENSIONS Akshay Gadde , Sunil K Narang and Antonio Ortega Ming Hsieh Department of Electrical Engineering University of Southern California," pp. 1222–1226, 2013.

[28] S. Shekkizhar and A. Ortega, "Graph construction from data by non-negative kernel regression," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3892–3896, 2020.