Universidade Estadual de Campinas
Instituto de Computação

Filipe de Oliveira Costa

Image and Video Phylogeny Reconstruction

Reconstrução de Filogenias para Imagens e Vídeos

CAMPINAS
2016

Filipe de Oliveira Costa

Image and Video Phylogeny Reconstruction

Reconstrução de Filogenias para Imagens e Vídeos

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Anderson de Rezende Rocha
Co-supervisor/Coorientador: Prof. Dr. Zanoni Dias

Este exemplar corresponde à versão final da Tese defendida por Filipe de Oliveira Costa e orientada pelo Prof. Dr. Anderson de Rezende Rocha.

CAMPINAS
2016

Agência(s) de fomento e nº(s) de processo(s): FAPESP, 2013/05815-2; CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

C823i

Costa, Filipe de Oliveira, 1987-
Image and video phylogeny reconstruction / Filipe de Oliveira Costa. – Campinas, SP : [s.n.], 2016.

Orientador: Anderson de Rezende Rocha.
Coorientador: Zanoni Dias.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento de imagens. 2. Computação forense. 3. Visão por computador. I. Rocha, Anderson de Rezende,1980-. II. Dias, Zanoni,1975-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Reconstrução de filogenias para imagens e vídeos
Palavras-chave em inglês:
Image processing
Computer forensics
Computer vision
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Anderson de Rezende Rocha [Orientador]
Esther Luna Colombini
Sandra Eliza Fontes de Ávila
Agma Juci Machado Traina
Roberto Hirata Junior
Data de defesa: 14-06-2016
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Filipe de Oliveira Costa

Image and Video Phylogeny Reconstruction

Reconstrução de Filogenias para Imagens e Vídeos

Campinas, 14 de junho de 2016

Dedicatória

A meus pais e irmãos.

Nankurunaisa!

Acknowledgements

Four years. It seemed an eternity. But after a lot of headache, I realize it was worth it. And, of course, I could not fail to thank the people who were important to me all this time.

First, thanks to GOD, for my health, my family, my friends, and for never letting me give up.

Thanks to my advisors Prof. Anderson Rocha e Prof. Zanoni Dias, for their help, advice and, especially, for the superhuman patience they had with me. You are examples of professionals whom I always want to follow.

Thanks to my Italian new friends from *Politecnico di Milano*, for everything I learned there. *Grazie mille!*

Thanks to the Institute of Computing at Unicamp, all the professors and workers, for the opportunity for attending my graduate course.

Thanks to FAPESP (Process number 2013/05815-2) and CAPES (PDSE program - Process number 99999.003836/2014-02), for the financial support provided in this period.

Thanks to the people of Recod Lab, specially to Marina Oikawa and Alberto Oliveira, who directly participate with me in this journey. I have learned a lot with you guys. Thank you very much!

Thanks to the guys of “República Zero Bala” (Testa, Stoshi, Nerd, Bokita, Bonjour, Drugão, Guilherme, and, why not, my dog Tequila), for the friendship, for the support and for the occasional beers, because we are not iron-made. ¯_(ツ)_/¯

Thanks to my friends for the support, specially to my great friend Heloisa, for helping me keep my sanity in this period (or, at least, for trying to help me to keep my sanity).

Thanks to my girlfriend Mara, for helping me to be a better person, for supporting me in hard times and to be the best woman in the face of the Earth. I really love you!

Finally, I would like to thank the most important people of my life whom directly contribute with my education: my mother Maria José, my father Moisés, and my siblings Mariana e André. I am nothing without you. Thanks for everything! I love you all!

Resumo

Com o advento das redes sociais, documentos digitais (e.g., imagens e vídeos) se tornaram poderosas ferramentas de comunicação. Dada esta nova realidade, é comum esses documentos serem publicados, compartilhados, modificados e republicados por vários usuários em diferentes canais da Web. Além disso, com a popularização de programas de edição de imagens e vídeos, muitas vezes não somente cópias exatas de documentos estão disponíveis, mas, também, versões modificadas das fontes originais (duplicatas próximas). Entretanto, o compartilhamento de documentos facilita a disseminação de conteúdo abusivo (e.g., pornografia infantil), que não respeitam direitos autorais e, em alguns casos, conteúdo difamatório, afetando negativamente a imagem pública de pessoas ou corporações (e.g., imagens difamatórias de políticos ou celebridades, pessoas em situações constrangedoras, etc.). Muitos pesquisadores têm desenvolvido, com sucesso, abordagens para detecção de duplicatas de documentos com o intuito de identificar cópias semelhantes de um dado documento multimídia (e.g., imagem, vídeo, etc.) publicado na Internet. Entretanto, somente recentemente têm se desenvolvido as primeiras pesquisas para ir além da detecção de duplicatas e encontrar a estrutura de evolução de um conjunto de documentos relacionados e modificados ao longo do tempo. Para isso, é necessário o desenvolvimento de abordagens que calculem a dissimilaridade entre duplicatas e as separem corretamente em estruturas que representem a relação entre elas de forma automática. Este problema é denominado na literatura como Reconstrução de Filogenia de Documentos Multimídia. Pesquisas na área de filogenia de documentos multimídia são importantes para auxiliar na resolução de problemas como, por exemplo, análise forense, recuperação de imagens por conteúdo e rastreamento de conteúdo ilegal. Nesta tese de doutorado, apresentamos abordagens desenvolvidas para solucionar o problema de filogenias para imagens e vídeos digitais. Considerando imagens, propomos novas abordagens para tratar o problema de filogenia considerando dois pontos principais: (i) a reconstrução de florestas, importante em cenários onde se tem um conjunto de imagens semanticamente semelhantes, mas geradas por fontes ou em momentos diferentes no tempo; e (ii) novas medidas para o cálculo de dissimilaridade entre as duplicatas, uma vez que esse cálculo afeta diretamente a qualidade de reconstrução da filogenia. Os resultados obtidos com as soluções para filogenia de imagens apresentadas neste trabalho confirmam a efetividade das abordagens propostas, identificando corretamente as raízes das florestas (imagens originais de uma sequencia de evolução) com até 95% de acurácia. Para filogenia de vídeos, propomos novas abordagens que realizam alinhamento temporal nos vídeos antes de se calcular a dissimilaridade, uma vez que, em cenários reais, os vídeos podem estar desalinhados temporalmente, terem sofrido recorte temporal ou serem comprimidos, por exemplo. Nesse contexto, nossas abordagens conseguem identificar a raiz das árvores com acurácia de até 87%.

Abstract

Digital documents (e.g., images and videos) have become powerful tools of communication with the advent of social networks. Within this new reality, it is very common these documents to be published, shared, modified and often republished by multiple users on different web channels. Additionally, with the popularization of image editing software and online editor tools, in most of the cases, not only their exact duplicates will be available, but also manipulated versions of the original source (near duplicates). Nevertheless, this document sharing facilitates the spread of abusive content (e.g., child pornography), copyright infringement and, in some cases, defamatory content, adversely affecting the public image of people or corporations (e.g., defamatory images of politicians and celebrities, people in embarrassing situations, etc.). Several researchers have successfully developed approaches for the detection and recognition of near-duplicate documents, aiming at identifying similar copies of a given multimedia document (e.g., image, video, etc.) published on the Internet. Notwithstanding, only recently some researches have developed approaches that go beyond the near-duplicate detection task and aim at finding the ancestral relationship between the near duplicates and the original source of a document. For this, the development of approaches for calculating the dissimilarity between near duplicates and correctly reconstruct structures that represent the relationship between them automatically is required. This problem is referred to in the literature as Multimedia Phylogeny. Solutions for multimedia phylogeny can help researchers to solve problems in forensics, content-based document retrieval and illegal-content document tracking, for instance. In this thesis, we designed and developed approaches to solve the phylogeny reconstruction problem for digital images and videos. Considering images, we proposed approaches to deal with the phylogeny problem considering two main points: (i) the forest reconstruction, an important task when we consider scenarios in which there is a set of semantically similar images, but generated by different sources or at different times; and (ii) new measures for dissimilarity calculation between near-duplicates, given that the dissimilarity calculation directly impacts the quality of the phylogeny reconstruction. The results obtained with our approaches for image phylogeny showed effective, identifying the root of the forests (original images of an evolution sequence) with accuracy up to 95%. For video phylogeny, we developed a new approach for temporal alignment in the video sequences before calculating the dissimilarity between them, once that, in real-world conditions, a pair of videos can be temporally misaligned, one video can have some frames removed and video compression can be applied, for example. For such problem, the proposed methods yield up to 87% correct of accuracy for finding the roots of the trees.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Undoubtedly, multimedia documents (e.g., images and videos) are powerful communication tools living up to the classical adage comparing them to a thousand words when conveying any information. This communication power was multiplied significantly with the advent of social networks. Within this new reality, multimedia documents are published, shared, modified, and often republished effortlessly and, depending on their contents, they can easily *go viral*, being republished by many other users in different channels trough the Web. Additionally, with the popularization of image editing software and online editor tools, in most of the cases, not only their exact duplicates will be available, but also manipulated versions of the original source. This scenario easily leads to copyright infringement, sharing of illegal or abusive contents (e.g., child pornography) and, in some cases, negatively affect or impersonate the public image of people or corporations (e.g., a person in a bullying situation, fake and defamatory images of celebrities or politicians, etc.).

When small changes are applied during the redistribution, usually without interfering with their semantic meaning, they are called *near-duplicate* objects. Several approaches for near-duplicate detection and recognition (NDDR) have been developed targeting at different applications, such as photography collections arrangement [34, 61], multimedia file matching [71], copyright infringement detection [9, 70] and forgery detection [29, 36].

A far more challenging task, however, has been overlooked thus far in which we also want to find the ancestral relationship between the near duplicates and the original source (*root* or *patient zero*), estimating the transformations (e.g., geometric transformations, cropping, color changing, compression, etc.) that originally created the near duplicates in a set and reconstruct the order of them. The main idea is inspired by the evolutionary process observed in Biology, in which an organism inherits characteristics from its ancestors, and can also go through mutations over time, producing new species [41]. If we look at the complete structure relating this population, moving from the root to the leaves of the phylogeny tree, we are able to identify the ancestral lineage, the descendants of each ancestor and have an overall idea of their evolution timeline. This new research field is called *Multimedia Phylogeny* [24, 25].

A similar structure can be considered for a document that goes under slight modifications over time (e.g., geometric transformations, brightness and contrast corrections, compression, etc.). If we are able to reconstruct the phylogeny tree of a set of near-duplicate objects, we can easily visualize their past history and ancestry information. In the literature, there are approaches aiming at finding the structure of the evolution of images [23, 25, 37, 59]. Extensions to the original image phylogeny algorithm were also proposed for reconstructing the tree of evolution of a set of near-duplicate videos (Video Phylogeny) [24, 39] and the phylogeny of audio clips (Audio Phylogeny) [51].

There are several applications to multimedia phylogeny solutions [25], such as:

- **Security:** the relationship structure of a set of documents provides information of suspects' behavior, and points out the directions of content distribution.
- **Forensics:** better results can be achieved if the analysis is performed in the original document instead of in a near duplicate [30].
- **Copyright enforcement:** traitor tracing without the requirement of source control techniques such as watermarking or fingerprinting.
- **News tracking services:** the near-duplicate relationships can feed news tracking services with key elements for determining the opinion forming process across time and space.
- **Illegal content tracking:** We could use phylogeny to point out group's reuse of illegal material online.

The multimedia phylogeny problem can be separated into two basic steps: the *dissimilarity calculation* between the duplicates, in which a dissimilarity function is used to compare each pair of images, returning small values for similar images and large values for more distinct images, and the *phylogeny reconstruction*, considering one dissimilarity matrix that represents the dissimilarity between each pair of documents [25]. The definition of reliable dissimilarity measure is paramount for document phylogeny research, given that the dissimilarity calculation directly affects the result of the final phylogeny reconstruction.

For image phylogeny, researchers in this area mainly tried to solve this problem by considering only one tree for each set of near duplicates [19, 25, 47, 59]. However, in some cases, we may have multiple sets of semantically similar images, which are images with the same semantic content, but not necessarily near duplicates. In other words, the original images come from different cameras or from the same camera but from a slightly different point in space and time, and some near duplicates of them could be generated. Therefore, we are not interested in reconstructing a single structure (tree) but a *phylogeny forest*, a set of phylogeny trees, for representing their relationship. Approaches to finding phylogeny forests are useful for tracing the original documents within a large set of semantically-similar documents. In other words, if there is more than one original document (e.g., images with the same scene, but taken from another view point, for example), phylogeny forest approaches should be used for finding the correct trees of each subset of documents. Figure 1.1 depicts an example of the image phylogeny forest problem.

Considering videos, Dias et al. [24] proposed an initial approach to deal with the video phylogeny tree reconstruction problem. However, only temporally coherent videos

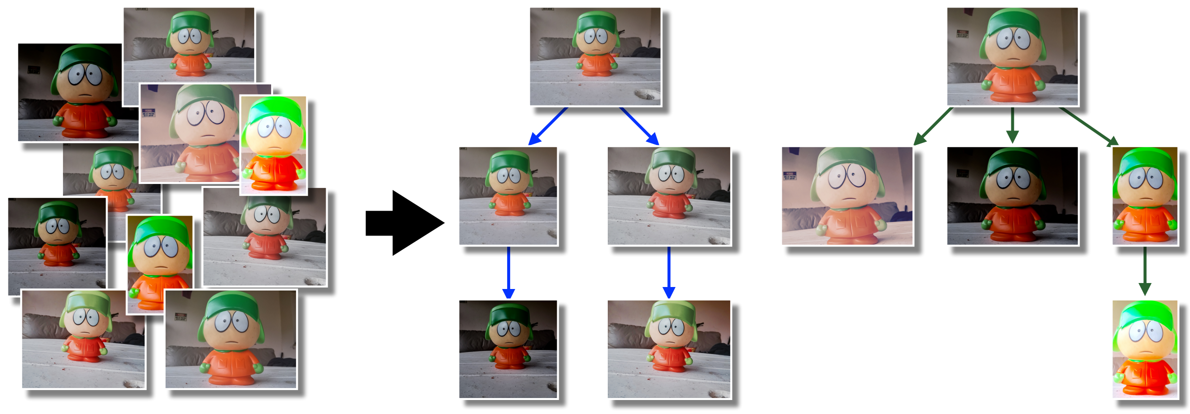


Figure 1.1: Image phylogeny forest problem. Given a set of semantically-similar images, our objective is to reconstruct a structure that represents the historical relationships among them. In this example, we have a forest with two trees, which means that the group of semantically-similar images has two original sources (with similar content) and each one spurs its own near duplicates (descendants).

(i.e., temporally aligned videos with the same number of frames) were considered thus far. Furthermore, the authors considered only videos compressed with the same standard and parameters without explicitly taking into account any other compression scheme in their reconstruction pipeline. These are somewhat limiting assumptions, given that video duplicates are typically encoded using different coding schemes and parameters and often are temporally misaligned.

In this thesis, we design and develop some solutions that aim at solving the multimedia phylogeny problem for images and videos. More specifically, for images, we present solutions that aim at reconstructing phylogeny trees and forests for representing the relationship between the duplicates. Moreover, we also develop new dissimilarity measures based on gradient and mutual information that significantly improve the quality of the phylogeny reconstruction process. Finally, for video phylogeny, we introduce new methods considering the temporal misalignment of the videos and different parameters of coding used for creating the near duplicates.

1.1 Image and Video Phylogeny: Hypothesis

Near duplicates of images and videos usually are created after some small transformations applied upon a source. A user can download this near duplicate, modify it again and republish it on the Internet effortlessly. These transformations can be done several times and in cascade, creating different near duplicates, as Figure 1.2 illustrates.

Previous work on the subject focused on a general hypothesis, which assumes that the transformations (e.g., color changing, compression, geometric transformation, cropping, etc.) used for creating near duplicates of a document (image or video) often leave irreversible artifacts in the data that allow us to point out the direction of the

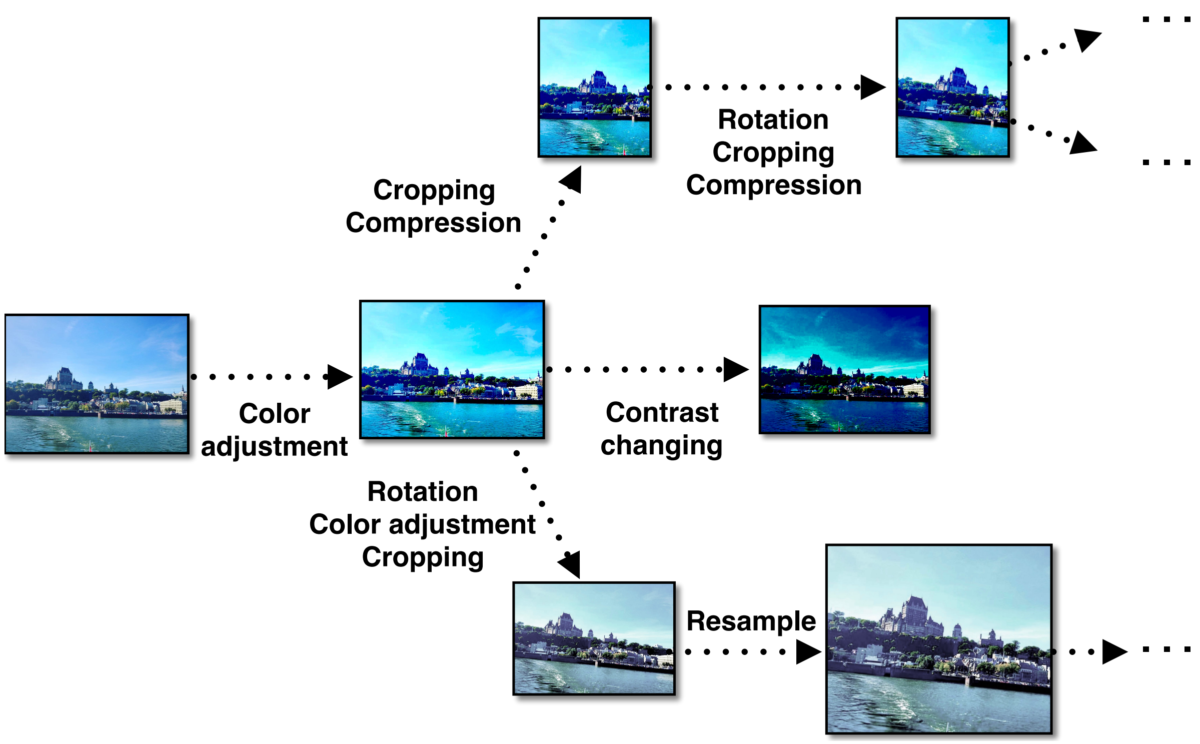


Figure 1.2: The generation process of near duplicates based upon a few image transformations.

transformations that the documents have undergone and, ultimately, create a phylogeny map coding the evolutionary structure of such a set of documents.

After studying and analyzing the way that the near-duplicate images and videos generally are created, our work herein further refines this hypothesis in four new ones:

H_1: Formulating the phylogeny reconstruction problem (finding the trees and reconstructing them from previously calculated dissimilarities) as an optimum-branching problem is more effective than using heuristic-based approaches such as those based on Minimum Spanning Trees (MST). Given its exact nature, Optimum Branching algorithm can generate more accurate phylogeny trees than greedy algorithms for image phylogeny forest reconstruction.

H_2: The different nature formulation of an Optimum branching formulation and an MST one lead to complementary properties explored during reconstruction and hence can be combined for improving the quality of the phylogeny forest reconstruction.

H_3: Due to possible errors of image registration, the comparison of the distributions of the pixel values of two near-duplicate images is more effective for the dissimilarity calculation, than their point-wise comparison.

H_4: While transformations in images consider three dimensions (width, height and depth), videos has a fourth dimension (time) upon which transformations can also be applied. For such cases, temporal alignment is paramount for a proper phylogeny reconstruction process.

1.2 Objectives

The main objective of this work is to design and develop solutions that allow us to identify the structure (phylogeny tree and phylogeny forest) that represents the generation process overtime of multimedia documents (images and videos).

Specific Objectives

- To design and develop solutions for image phylogeny reconstruction considering images of different sources and different phylogenies (forests);
- to design and develop new dissimilarity measures for images, aiming at improving the results of the phylogeny reconstruction
- to provide solutions for video phylogeny that deal with possible temporal misalignment and different compression parameters between the duplicates.

1.3 Scientific Contributions

This work is useful for targeting the solution of problems that involve legal tasks, such as those stressed out earlier involving security, forensics, and copyright enforcement. In addition, it has also application outside the realm of the legal system, as one could use it for tracking news overtime. More specifically, phylogeny solutions can be used for verifying, for instance, the source of illegal content improperly shared on the Internet (e.g., child pornography, defamatory content, etc.), given some near duplicates of such documents. In this vein, the main contributions of this work are:

- the creation of new near-duplicate datasets (which is public) and the validation of the proposed approaches in these datasets;
- new approaches for image phylogeny forest reconstruction, i.e., solutions considering documents of different sources, founded upon optimum branching methods and their combination with heuristic (greedy) ones;
- new dissimilarity measures for images, exploring gradient and mutual information for comparing them, instead of the standard approach based on point-wise comparison; and

- two new approaches for solving the video phylogeny problem, considering temporally misaligned and videos compressed with different parameters.

1.4 Publications

The research in this thesis has resulted in some important publications in top journals and conferences:

1. Filipe de O. Costa, Marina Oikawa, Zanoni Dias, Siome Goldenstein and Anderson Rocha; *Image phylogeny forests reconstruction.* IEEE Transactions on Information Forensics and Security (TIFS), vol. 9, no. 10, p. 1533-1546, 2014. [15]

2. Filipe de O. Costa, Alberto Oliveira, Pasquale Ferrara, Zanoni Dias, Siome Goldenstein and Anderson Rocha; *New Dissimilarity Measures for Image Phylogeny Reconstruction*, Springer Pattern Analysis and Applications (PAA) – Under review [17].

3. Filipe de O. Costa, Silvia Lameri, Paolo Bestagini, Zanoni Dias, Anderson Rocha, Marco Tagliasacchi and Stefano Tubaro, *Phylogeny reconstruction for misaligned and compressed video sequences*, IEEE International Conference on Image Processing (ICIP), p. 301-305, 2015. [14]

4. Filipe de O. Costa, Marina Oikawa, Zanoni Dias and Anderson Rocha, *Temporal alignment based on Hamming Distance for Video Phylogeny Reconstruction*, IEEE Workshop on Information Forensics and Security (WIFS), 2016 (Submitted).

1.5 Thesis Roadmap

This thesis is organized in seven chapters: Chapter 2 presents the related work in multimedia phylogeny. Then, the thesis is separated in two parts. Part I introduces the proposed solutions for image phylogeny, in Chapter 3, and new dissimilarity measures, in Chapter 4, along with the experimental results. In turn, Part II introduces the proposed solutions and experimental results for video phylogeny, in Chapters 5 and 6. Finally, we present the conclusions of this work as well as a brief analysis of their possible future developments and research ramifications in Chapter 7.

Chapter 2

Related Work

In this chapter, we present a view of the state of the art of multimedia phylogeny, showing a short review about the detection of near duplicates and presenting some approaches for the multimedia phylogeny reconstruction problem.

2.1 Near-Duplicate Detection and Recognition

One near duplicate is a modified version of one document that maintains its semantic features. For instance, one near-duplicate image is a version of one given image after we apply to it one or more transformations (e.g., cropping, rotation, color correction, etc.). In other words, the two images contain a kinship relationship.

Joly et al. [35] formally defined a near duplicate based on the concept of tolerated transformations. According to the authors, a multimedia document $\mathcal{D}_1$ is a near duplicate of other document $\mathcal{D}$ if, given a set of transformations $\mathcal{T}$, there exists at least one transformation $T_\beta \in \mathcal{T}$ such that $\mathcal{D}_1 = T_\beta(\mathcal{D})$. One family of transformations $\mathcal{T}$ can have several combinations of transformations that generates, for example, the near duplicate $\mathcal{D}_3 = T_3 \circ T_2 \circ T_1(\mathcal{D}), T_{\beta=1,2,3} \in \mathcal{T}$. Given the original document $\mathcal{D}$, we can build the near-duplicate tree, as Figure 2.1 shows.

A near duplicate is a pairwise equivalence relationship. This relationship links the original document (the root of the tree) to its variations generated trough some transformations [46]. Given an original document $\mathcal{D}$, if $\mathcal{D}_1$ is a near duplicate of $\mathcal{D}$ and $\mathcal{D}_2$ is a near duplicate of $\mathcal{D}_1$, then $\mathcal{D}_2$ is a near duplicate of $\mathcal{D}$.

Informally, the near-duplicate detection can be done trough marking-based approaches and content-based approaches. Marking-based approaches depend on a *signature* within the original document before its dissemination (e.g., *watermarking*, *fingerprinting*, etc.) [46]. With these approaches, it is possible to detect the original document by analyzing this signature and the changing of its pattern in other documents. In contrast, content-based methods depend on the analysis of the content of the documents, aiming at extracting relevant visual features. These methods identify when a set of features are close to the original document visual features. Figure 2.2 illustrates these philosophies for the near-duplicate detection and recognition (NDDR).

Several NDDR approaches have been developed in the last years for different

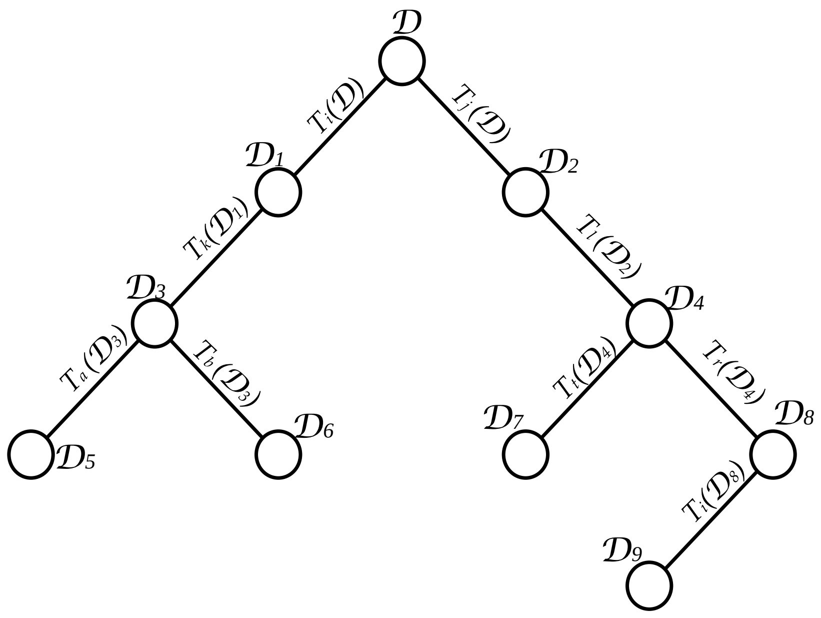


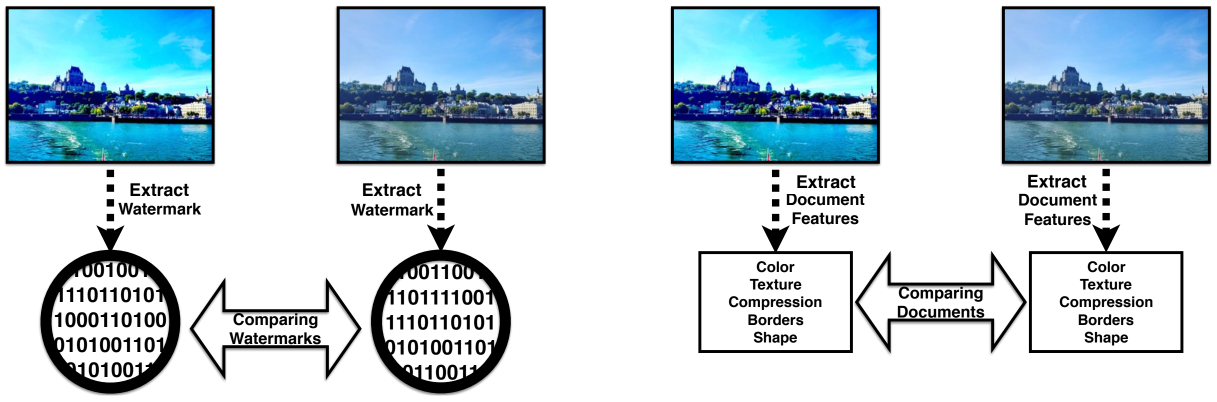
Figure 2.1: A near-duplicate tree of a document $\mathcal{D}$ and its transformations, according to Joly et al. [35]. Adapted from Dias et al. [23].

applications, such as organizing photography collections [34, 61], multimedia correspondence [11, 71], image and video copyright infringement detection [9, 70], image forgery detection [29, 36] and near-duplicate detection and temporal alignment for multi-view video sequences [48]. Such works focus on the near-duplicate identification without taking into account the structure of modifications or the transformations used for generating the near duplicates.

2.2 Multimedia Phylogeny

Multimedia Phylogeny is a new research area focused on finding the ancestry relationship structure of documents. In other words, the goal is to generate a structure of kinship relationships as a tree, whereby the root is the patient zero (the original document), the edges represent "father-child" relationships, and the leaves of the tree represent "terminal" documents that have more modifications than their ancestors. Once we trace the past history of the near duplicates, multimedia phylogeny can be useful for aiding (allied with additional side information) the discovery, for instance, of who was the first user that published an image online containing illegal or abusive content (e.g., fake and defamatory image of celebrities or politicians, child pornography, etc.), which were redistributed after being modified by different users.

In the literature, there are some works that aim at tracing the past history of near duplicates. In this section, we present some works related to multimedia phylogeny for images and videos, the focus of this thesis.


Extract Watermark
Extract Watermark
Comparing Watermarks
Extract Document Features
Extract Document Features
Color Texture Compression Borders Shape
Comparing Documents
Color Texture Compression Borders Shape

(a) Marking-based approach (b) Content-based approach

Figure 2.2: Near-duplicate detection approaches. Marking-based approach × content-based approach.

2.2.1 Image Phylogeny

Kennedy et al. [37] defined the problem of the "father-son" relationship between image pairs (Image Archeology) and proposed an approach for detecting, given a pair of images, which one is the original and which one is the near duplicate generated after some transformations. For this purpose, the authors developed a *Visual Migration Map* (VMM), representing a historical approximation of the image, aiming at distinguishing different types of image edits and their ideological perspective. However, the authors did not present discussions about how to find the parameters of the transformations used for generating the near-duplicate images nor discussed possible algorithms for the reconstruction of the phylogeny tree through the evolution process of the images.

The temporal verification of the transformations in images was also explored in Fan and Queiroz [27]. In that work, the authors identify the history of compression associated to one image. In another work, Mao et al. [45] discuss about some relevant information provided by the device that generated one image. Nevertheless, those works also do not discuss about approaches for finding the structure that represents the relationship between near duplicates.

On another front, De Rosa et al. [59] proposed an approach for detection of the dependencies of images in a set of near duplicates $\mathcal{I}$, considering the hypothesis that any image $\mathcal{I}_i \in \mathcal{I}$ can be described as the composition of two different components separately: image content-based components and image content-independent components. For verifying the dependency between two images $\mathcal{I}_A$ and $\mathcal{I}_B$, the authors consider that the mutual information of the images can be expressed as the sum of the mutual information between these components. The authors assume that, if there exists any dependency between the images $\mathcal{I}_A$ and $\mathcal{I}_B$, one image can be obtained approximately applying over the other image some processing functions. Thus, the image content is evaluated for estimating these functions.

After mapping one image onto the other's domain using the estimated image processing functions, the authors calculate the correlation coefficient between the images based on the content-independent components. In this case, the authors used the *Photo Responsivity*

Non-uniform Noise (PRNU) of the images, which is caused by the interaction of the light in the environment and the acquisition sensor of the camera at the moment of the generation of one image. The PRNU estimation is widely used for solving problems related to image source attribution, in which the goal is to identify if one given image was generated by a camera under investigation [13, 18, 42].

After analyzing all possible pairs of images in $\mathcal{I}$, the authors create a graph of dependencies whose nodes represent the images, the edges represent the relationship between two images and the weight of an edge represents the correlation value between them. For this, the authors evaluate each edge of the graph and discard all the edges that have a weight lower than a threshold. Then, they remove direct loops (it is impossible that one image $\mathcal{I}_i$ generates one image $\mathcal{I}_j$ and vise-versa at the same time.). Thereafter, other edges are discarded until each node of the graph has degree of entrance equal to 1 (assuming that the images were not generated by composition of two or more different sources). The process repeats until the method finds a tree.

In the same line of thought, Dias et al. [23, 25] formally defined the problem of Image Phylogeny following two steps: the calculation of the dissimilarity between each pair of near-duplicate images and the reconstruction of the phylogeny tree. Considering $\mathcal{T}$ a family of image transformations, T a transformation such that $T_{\overrightarrow{\beta}} \in \mathcal{T}$ parameterized by $\overrightarrow{\beta}$, and two near-duplicate images $\mathcal{I}_{src}$ (source) and $\mathcal{I}_{tgt}$ (target), the dissimilarity function $d(.,.)$ between them is defined as the lowest value of $d(\mathcal{I}_{src}, \mathcal{I}_{tgt})$, such that

$$d(\mathcal{I}_{src}, \mathcal{I}_{tgt}) = \min_{T_{\overrightarrow{\beta}} \in \mathcal{T}} |\mathcal{I}_{tgt} - T_{\overrightarrow{\beta}}(\mathcal{I}_{src})|_{\text{ point-wise comparison }} \mathcal{L}. \tag{2.1}$$

Equation 2.1 calculates the dissimilarity between the best transformation mapping $\mathcal{I}_{src}$ onto $\mathcal{I}_{tgt}$ parameterized by $\overrightarrow{\beta}$, according to the family of transformations $\mathcal{T}$. The comparison between the images can be performed by any point-wise comparison method $\mathcal{L}$.

Given a set of near duplicates, the estimation of the transformation T, parameterized by $\overrightarrow{\beta}$ used to map an image $\mathcal{I}_{src}$ onto an image $\mathcal{I}_{tgt}$'s domain follows a three-step method generating $\mathcal{I}'_{src} = T_{\overrightarrow{\beta}}(\mathcal{I}_{src})$:

1. **Geometric matching:** also known as Image Registration. Among several different approaches known in the literature [73], the image registration is calculated by finding interest points in each pair of images, using SURF (Speeded-Up Robust Features) [3], which will be used to estimate warping and cropping parameters robustly using RANSAC [28];

2. **Color matching:** it is performed for adjusting the color of the source image $\mathcal{I}_{src}$ to the target image $\mathcal{I}_{tgt}$ by normalizing each channel of $\mathcal{I}_{src}$ by the mean and standard deviation of the respective channel in $\mathcal{I}_{tgt}$ [58];

3. **Compression matching:** the image $\mathcal{I}_{src}$ is compressed with $\mathcal{I}_{tgt}$'s JPEG compression parameters.

Then, a comparison between the estimated $\mathcal{I}'_{src} = T_{\overrightarrow{\beta}}(\mathcal{I}_{src})$ and $\mathcal{I}_{tgt}$ is performed point-wise. There are also different approaches for calculating the point-wise dissimilarity

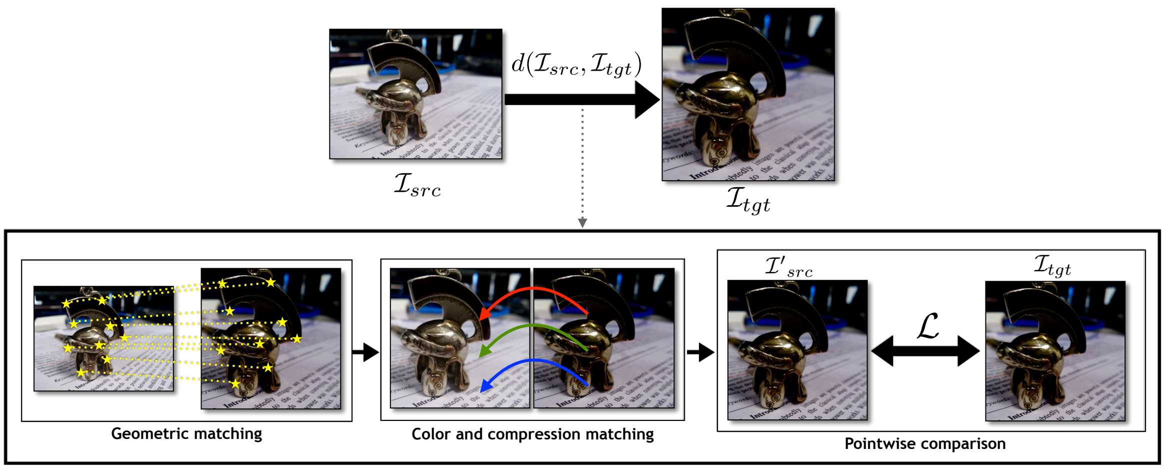


Figure 2.3: Dissimilarity calculation process. The mapping of image $\mathcal{I}_{src}$ onto $\mathcal{I}_{tgt}$'s domain involves a three-step process: geometric, color and compression matching. Afterwards, it is possible to directly compare the images using any point-wise comparison algorithm.

between two images [32] and the authors opted to estimate it using the Mean Squared Error (MSE) technique. Figure 2.3 depicts this dissimilarity calculation process.

Note that the dissimilarity is not a metric. Once the transformation T used for mapping $\mathcal{I}_{src}$ to $\mathcal{I}_{tgt}$ can insert irreversible artifacts in $\mathcal{I}_{src}$ for mapping it to $\mathcal{I}_{tgt}$ (e.g., remove pixels after spatial cropping, JPEG compression, etc.) the inverse transformation T^{-1} will not recover the lost information when performing the mapping in the opposite way. Thus, $d(\mathcal{I}_{src}, \mathcal{I}_{tgt}) \neq d(\mathcal{I}_{tgt}, \mathcal{I}_{src})$. Low values of $d(\mathcal{I}_{src}, \mathcal{I}_{tgt})$ denote a good transformation and the resultant image $\mathcal{I}'_{src}$ is a close approximation of $\mathcal{I}_{tgt}$, which is a strong evidence that $\mathcal{I}_{src}$ is the father of $\mathcal{I}_{tgt}$ in the phylogeny tree.

After calculating the dissimilarity for each pair of images, we have a dissimilarity matrix $M_{n \times n}$, where n is the number of near duplicates and each position of the matrix represents the dissimilarity between one pair of images.

For reconstructing the *Image Phylogeny Tree* (IPT) associated with the dissimilarity matrix, the authors first proposed the Oriented Kruskal algorithm (OK), an extension of the classic Kruskal Minimal Spanning Tree algorithm [38], adapted for oriented graphs. Considering a graph G that represents the dissimilarity matrix, the OK algorithm starts considering each vertex (near duplicate) of G as one root of the phylogeny tree. Then, the algorithm sorts all edges (i, j) of G and analyzes each position according to the sorted order, joining different trees and checking whether or not one edge can be added to the tree.

Figure 2.4 illustrates one example of the OK algorithm. After sorting all the edges of the dissimilarity matrix M, the algorithm selects the edges in ascending order, according to their weights. The edge will be discarded if it is an entrance edge of a node that already has entrance degree equal to 1 (Steps 3 and 7 in this example) or if it will create a circuit (Step 6). The algorithm stops when it finds a phylogeny tree.

The method presents effective results for reconstructing phylogeny trees for a small near-duplicate image dataset. The results were also confirmed on a large-scale test

M	1	2	3	4	5	6
1	-	5	6	12	46	32
2	26	-	23	23	31	34
3	22	29	-	32	25	39
4	8	35	12	-	16	13
5	31	7	44	19	-	27
6	19	25	31	44	10	-

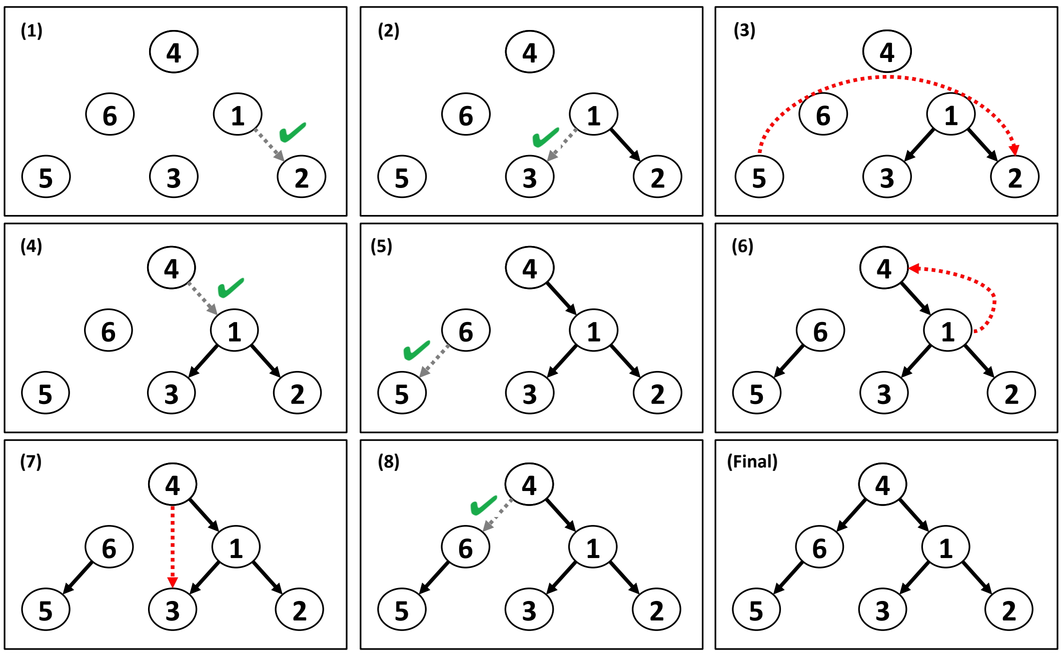


Figure 2.4: Oriented Kruskal Algorithm.

performed and presented in [21], with over two million test cases, with synthetic and real data.

Although the field has been developing significantly over the past years, thus far researchers mainly focused on proposing different phylogeny reconstruction approaches often using a standard methodology for dissimilarity calculation, as originally proposed in [25]. Dias et al. [19] presented other approaches for IPT reconstruction, comparing their performance with the Oriented Kruskal algorithm. The first method is a variation of the algorithm for minimum spanning tree of Prim [57], adapted for oriented graphs. This Oriented Prim approach initially receives the dissimilarity matrix M and, for each possible root out of n, it reconstructs the tree rooted at the chosen node and calculates the cost for building such tree. After evaluating all possible roots, the final IPT is the one with the lowest reconstruction cost, which is calculated by summing up the weight of the edges in the tree. However, this algorithm presented low effectiveness for tree reconstruction, considering the ancestry relationship among the images.

The second approach presented in [19] is based on the Optimum Branching (OB) algorithm developed independently by Chu and Liu [10], Edmonds [26] and Bock [6]. Considering $G = (V, E)$ to be the input graph and r one node of the phylogeny tree, the first step in this algorithm consists of assuming the node r as a possible root, as done in

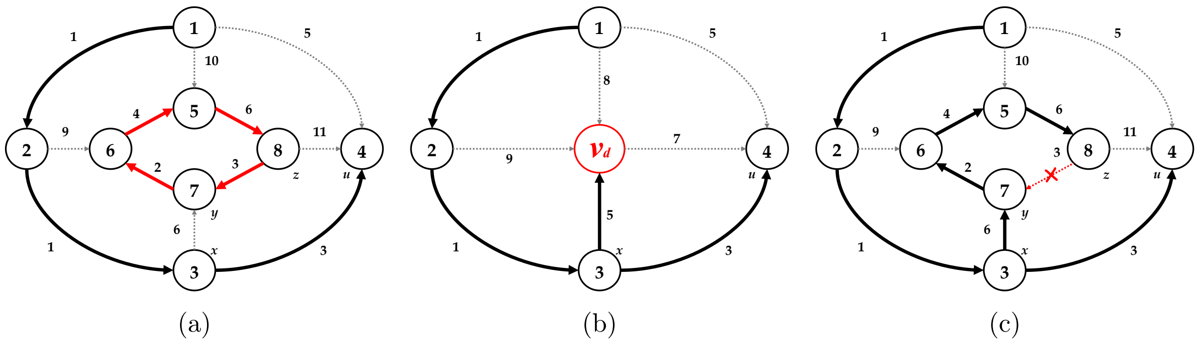


Figure 2.5: Optimum branching algorithm simulation.

the Oriented Prim Algorithm. As an example, suppose that vertex 1 in Figure 2.5 was chosen as the initial root r. Then, for each node $v \in V \mid v \neq r$, the edge arriving at v with the lowest cost is selected. If there is no circuit in the found graph, the algorithm returns such branching. Otherwise, it deals with it by creating a dummy node v_d. For each edge connecting a node x outside the circuit and a node y in the circuit, the weight w is updated by taking into account $w(x, y)$ plus the lowest edge weight in the circuit minus the weight of the edge arriving at y in the circuit. In this example, $w(3, 7) = 6$, the lowest edge weight inside the cycle is $w(7, 6) = 2$, and the weight of the edge that arrives at the vertex $y = 7$ is $w(8, 7) = 3$. Thus, $w(3, v_d) = 6 + 2 - 3 = 5$. On the other hand, to update each edge connecting a node z inside the circuit and a node u outside the circuit, it is necessary to take into account $w(z, u)$ plus the lowest edge weight in the circuit minus the weight of the edge arriving at z inside the circuit. For instance, $w(v_d, 4) = w(8, 4) + w(7, 6) - w(5, 8) = 11 + 2 - 6 = 7$.

After processing all nodes within and outside the circuit, the algorithm recursively finds the optimal branching A_{i+1} in this graph, resulting in the updated graph depicted in Figure 2.5(b). Next, the algorithm needs to cope with the circuit itself. This is achieved by first discarding the edge in the current optimum branching connecting a node x to the dummy node v_d and updating such edge with the correct one in the original graph. In this case, the edge $(3, v_d)$ is replaced by edge $(3, 7)$ in the final branching. In addition, the algorithm updates all edges in the optimum branching within the circuit that do not arrive in y (in the example, the edge $(8, 7)$ is removed). Finally, the algorithm just needs to update possible edges from nodes within the circuit to outside that are part of the optimum branching. The algorithm ends with the graph in Figure 2.5(c) with solid lines highlighting the edges that belong to the optimum branching.

The algorithm's complexity is given by the recurrence $T(V, E) = T(V_0, E_0) + O(V, E)$, where $O(V, E)$ is the complexity of trying to generate one tree. As in each recursive call the number of vertices is reduced in, at least, one unit, $|V_0| < |V|$, we have that $T(V, E) = O(V(V + E))$. In our case, as $E = \Theta(V^2)$ (complete graph), then $T(V, E) = O(V^3)$. A faster implementation provided by Tarjan [66] also exists, which runs in $O(V^2)$, for dense graphs [19].

This new approach outperformed the Oriented Kruskal algorithm, being more effective for finding the roots and the ancestry relationships, either considering complete trees or with missing links in the phylogeny tree. This happens due to the different properties presented by each algorithm. The Oriented Kruskal algorithm is a greedy algorithm, that is, it makes a local optimal choice (the lowest cost edge) at each stage to find the minimum branching. On the other hand, the Optimum Branching algorithm is an exact algorithm, always finding one optimum global solution (it considers the whole dissimilarity matrix to make decisions about the phylogeny reconstruction), which leads to better results in most of the cases. More details about the optimum branching algorithm can be found in [19].

Different from previous works, Melloni et al. [47] proposed the use of two dissimilarity measures: $d'(\mathcal{I}_{src}, \mathcal{I}_{tgt})$, which is the same as proposed by Dias et al. [25] and described in Equation 2.1, and $d''(\mathcal{I}_{src}, \mathcal{I}_{tgt})$, which indicates the randomness part of the image, obtained by extracting the noise of an image using a wavelet-based denoising approach [49]. With two dissimilarity measures, the approach aims at reducing the ambiguity in determining the parent of an image that has low dissimilarity with more than one image. The authors calculate two dissimilarity matrices: one rank matrix, in which low values mean that the images are dissimilar to only one parent, and the reconstruction matrix, which is the dissimilarity matrix described by Dias et al. [25]. In the end, the authors use the two dissimilarity matrices and a modified version of the Oriented Kruskal algorithm to reconstruct the phylogeny trees.

More recently, Bestagini et al. [5] have developed an approach for image phylogeny based on region analysis. The proposed approach differs in the dissimilarity calculation, in which the authors compare image pairs considering not the whole image, but regions of the image that might me modified (e.g., near-duplicate with logo insertion). In this way, the method can automatically localize areas that have been locally modified, if it happened, and spot the differences between near-duplicates. For example, let $\mathcal{I}_1$ and $\mathcal{I}_2$ two images, in which $\mathcal{I}_2$ is son of $\mathcal{I}_1$ and was generated after a logo insertion. With the difference of the two images, it is possible to identify the region where the logo was inserted in $\mathcal{I}_2$. With this mask and comparing both images, it is possible to know that the logo only exists in $\mathcal{I}_2$. Thus, the method automatically defines that the dissimilarity will be calculated only mapping $\mathcal{I}_1$ to $\mathcal{I}_2$ and not in the opposite way (once the logo insertion operation is irreversible). After comparing all the near-duplicate images, the authors generate a dissimilarity tree and reconstruct the phylogeny tree considering the OB algorithm.

Milani et al. [50] have propose the using of processing age metrics, which are sets of features that can blindly model the processing age of an image. By exploiting these features for the dissimilarity calculation, the authors improves the quality of IPT reconstruction and also reduce the computational complexity of the process.

Image Phylogeny Forest (IPF)

In some cases in multimedia phylogeny, instead of one tree, we may find m trees (or an *Image Phylogeny Forest* – IPF) representing the ancestry relationship in a set of near-duplicate images. This happens when we have multiple images with the same semantic content, but that are not directly related to each other. For instance, images from the same scene taken with different cameras, or with the same camera but using different settings and in slightly different positions. In this case, each tree in the forest represents the structure of transformations and the evolution of one subset of near-duplicate images, while the forest comprises distinct subsets of near-duplicate images which are semantically similar [22].

A first approach for reconstructing an IPF from a set of semantically similar images extended the idea of reconstructing an IPT from a set of near-duplicate images using the Oriented Kruskal algorithm. In this approach, Dias et al. [25] considered that to reconstruct m trees, the algorithm would stop when the structure of relationship reached $n - m$ edges where n is the number of images in the set. Although providing good results, the algorithm required from the user the number of trees to look for in the forest. However, this information is not always available, and without knowing the number of trees to reconstruct, the performance of the algorithm decreased with the number of trees, specially regarding the correct number of roots and the ancestry relationship in the IPF [22].

To enable automatic IPF reconstructions, Dias et al. presented in [22] a modified version of the Oriented Kruskal algorithm originally used to reconstruct IPTs. To create the new approach, named *Automatic Oriented Kruskal* (AOK), three parameters are required as input: the number of semantically similar images n, an $n \times n$ dissimilarity matrix M built upon these images and a parameter γ_{AOK}, calculated beforehand and defined as the number of standard deviations used to limit the number of edges to be included in the forest.

The AOK algorithm keeps track of the variance of processed edges and only adds a new one to the forest if the weight of the current edge is lower than γ_{AOK} times the standard deviation of the processed edges up to that point. This parameter γ_{AOK} is related to a threshold point τ_{AOK} that selects only edges that belong to valid trees. To define its value, a study about the behavior of the dissimilarity values of valid trees and forests was performed. It was observed that a Log-Normal distribution can reasonably describe the data regardless of the number of trees in the forest and the type of image capture (single/multiple cameras). The threshold $\tau_{\text{AOK}} = \mu_{\text{AOK}} + \gamma_{\text{AOK}} \times \sigma_{\text{AOK}}$ was used, where μ_{AOK} represents the average and σ_{AOK} the standard deviation of the weight of the edges already selected. After testing for different values, it was defined that $\gamma_{\text{AOK}} = 2$. In terms of complexity, and considering the same notation used for the analysis of OK algorithm, the AOK algorithm has the same complexity of the original Kruskal algorithm: $O(V^2 \log V)$.

More recently, Oikawa et al. [52] presented an approach that aims at taking advantage of data clustering techniques in the multimedia analysis context. The authors describe how to find the image phylogeny forests based on images that inherit content from a single parent. A new approach using manifold learning and spectral clustering was devised to obtain a better representation of the data points distribution and, hence, produce good image clusters. After clustering the images, the authors reconstruct one phylogeny tree for each group.

Not rarely, it is necessary to deal with scenarios in which an object inherits content from multiple sources (parents) instead of a single one. For instance, images created by

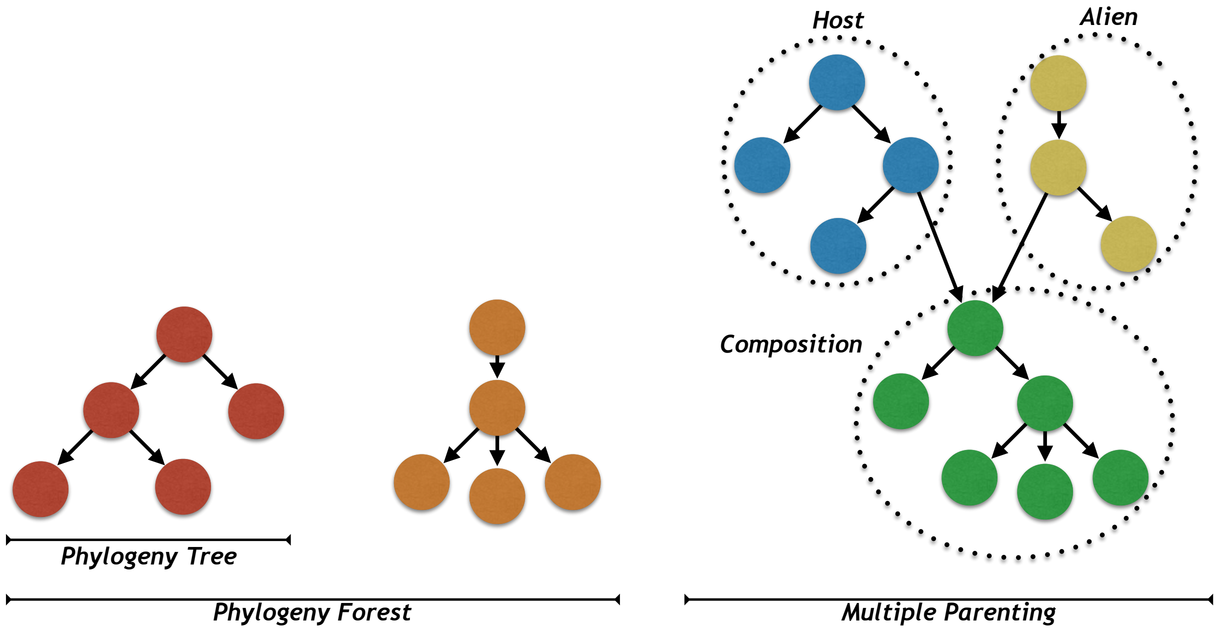


Figure 2.6: Phylogeny Tree × Phylogeny Forest × Multiple parenting reconstruction

means of composition of pieces from two different sources. These scenarios, referred to as *Multiple parenting phylogeny*, is an extension upon the IPF reconstruction problem. Oliveira et al. [53, 54] proposed the first solution for the multiple parenting phylogeny, considering images. The authors sought to find the relationships not only between images with essentially the same content (near duplicates), but also between the shared part of their content, being foreground or background. First, the authors try to separate the images into three groups: *hosts*, *alien* and *composition* (images created by the composition of the background of one *host* image and the foreground of one *alien* image). Then, a phylogeny tree is reconstructed for each one of these groups and, afterwards, the authors try to find the host and the alien images that created the root of the composition tree (considering that the root is the original composition, and the other nodes are near duplicates created from it). Figure 2.6 illustrates one example of the multiple parenting reconstruction problem.

2.2.2 Video Phylogeny

In the video phylogeny problem, starting from a pool of near-duplicate videos, we seek to reconstruct the relationship between every video pair. This problem, deeply studied for images is typically solved in two basic steps: (i) the calculation of the dissimilarity between each pair of near-duplicate objects; and (ii) the reconstruction of the phylogeny tree based on some dissimilarity measures (as presented in Section 2.2.1). As far as we know, there are just a few works related to video phylogeny reconstruction in the literature.

Dias et al. [24] proposed an initial approach to deal with the video phylogeny tree reconstruction problem. Basically, the authors follow the same steps for IPT reconstruction, but adapting the dissimilarity calculation. For this step, the authors consider that all videos have the same number of frames and all the frames are temporally coherent (it is, the i-th frame of both videos have the same semantic content). Then, the authors calculate the dissimilarity considering each frame of all the videos separately. For example, if all the videos have n frames, the method calculates n values of dissimilarity for each pair of videos, one for each frame. Thus, the process ends with n dissimilarity matrices.

The phylogeny reconstruction is done for each dissimilarity matrix with the Oriented Kruskal algorithm. In the end of this process, the authors generate a "parenthood matrix", that contains the frequency of the edges in each one of the trees previously calculated. Then, the final tree is given by the most frequent edges. The authors report results in which the method can find the root of the duplicates in 91% of 16 test cases. However, this approach has some limitations. First, only temporally coherent videos (i.e., temporally aligned videos with the same number of frames) were considered. In a more realistic scenario, videos can suffer temporal transformations (e.g., we can remove some frames of the video), which prevent us from checking all the frames without a previous temporal alignment. Furthermore, the authors considered only videos compressed with the same standard and parameters without explicitly taking into account any other compression scheme in their reconstruction pipeline. In a real world setup, video duplicates are typically encoded using different coding schemes and parameters (e.g., quality, size of the group of pictures (GoP), codec, etc.).

On a similar front, Lameri et al. [39] have developed an approach to reconstruct one video sequence through all the near duplicates in a set. The main idea consists of reconstructing parent sequences given a set of partially overlapped near-duplicate video shots used in other sequences. First, the authors use a robust hash algorithm for detecting if a group of frames is a near duplicate of other groups of frames (for instance, if the beginning of two video sequences have the same content). After this, they use an algorithm for automatically finding near duplicates by matching between multiple parts of multiple video sequences. In the end, the authors perform a temporal alignment based on an approach for sequence alignment using the difference of luminance between subsequent frames in all video duplicates. Although the goal of the method is to recover the original content of a video when it is no longer available, by combining its available near duplicates, the authors did not perform a reconstruction of a complete phylogeny tree.

2.2.3 The Phylogeny Problem in Other Media

The concepts about image and video phylogeny can be applied for finding the relationship between near duplicates of other kinds of documents. Andrews et al. [1] proposed a classification method for identifying *Name Phylogeny*. First, the authors assume that one name was not simply created, but it is derived from other similar names. Then, they developed a method based on Expectation/Maximization (EM) for reestimating the phylogeny and the parameters of the classifier. With this, the authors can find variants of a name in a set of names (e.g., "Filipe de Oliveira Costa", "Filipe de Oliveira" , "Filipe Costa", etc.).

Nucci et al. [51] presented a method to deal with the *Audio Phylogeny Reconstruction* problem. For this, the authors considered a set of near-duplicate audio samples comprising transformations such as compression, clipping, fade in and fade out. The dissimilarity between a pair of near-duplicate audio samples S_A and S_B is calculated through the *Signal to Noise Ratio* (SNR), according to the following equation

$$SNR(S_A, S_B) = 20log_{10}\left(\frac{||S_B||_2}{||S_A - S_B||_2}\right). \tag{2.2}$$

Finally, the authors reconstruct the audio phylogeny using the Oriented Kruskal algorithm.

Part I

Image Phylogeny

Chapter 3

Image Phylogeny Forest Reconstruction

In some cases in multimedia phylogeny, instead of one tree, we may find m trees representing the ancestry relationship in a set of near-duplicate images. This happens when we have multiple images with the same semantic content, but that are not directly related to each other. For instance, images from the same scene taken with different cameras, or with the same camera but using different settings and in slightly different positions. In this case, each tree in the forest represents the structure of transformations and the evolution of one subset of near-duplicate images, while the forest comprises distinct subsets of near-duplicate images which are semantically similar [22].

A first approach to reconstruct an IPF from a set of semantically similar images extended the idea of reconstructing an IPT from a set of near-duplicate images using the Oriented Kruskal algorithm. In this approach, Dias et al. [25] considered that to reconstruct m trees, the algorithm would stop when the structure of relationship reached $n - m$ edges, where n is the number of images in the set. Although yielding good results, the algorithm required, from the user, the number of trees to look for in the forest. However, this information is not always available, and without knowing the number of trees to reconstruct, the performance of the algorithm decreased with the number of trees, specially regarding the correct number of roots (trees) and the ancestry relationship in the IPF [22].

To enable automatic IPF reconstructions, Dias et al [22] presented a modified version of the Oriented Kruskal algorithm originally used to reconstruct IPTs. To create the new approach, named *Automatic Oriented Kruskal* (AOK), three parameters are required as input: the number of semantically similar images n, an $n \times n$ dissimilarity matrix M built upon these images and a parameter γ_{AOK}, calculated beforehand and defined as the number of standard deviations used to limit the number of edges to be included in the forest.

The AOK algorithm keeps track of the variance of processed edges and only adds a new one to the forest if the weight of the current edge is lower than γ_{AOK} times the standard deviation of the processed edges up to that point. This parameter γ_{AOK} is related to a threshold point τ_{AOK} that selects only edges that belong to valid trees. To define its value, a study about the behavior of the dissimilarity values of valid trees and forests was performed. It was observed that a Log-Normal distribution can reasonably describe the data regardless of the number of trees in the forest and the type of image capture (single/multiple cameras). The threshold $\tau_{\text{AOK}} = \mu_{\text{AOK}} + \gamma_{\text{AOK}} \times \sigma_{\text{AOK}}$ was used, where μ_{AOK} represents the average and σ_{AOK}, the standard deviation of the weight of the edges already selected. After testing for different values, it was defined that $\gamma_{\text{AOK}} = 2$. In terms of complexity, and considering the same notation used for the analysis of OB algorithm, AOK algorithm has the same complexity of the original Kruskal algorithm: $O(V^2 \log V)$. Further details and a step-by-step algorithm can be found in [22].

3.1 Automatic Reconstruction of Image Phylogeny Forests

Without user intervention, the IPF reconstruction algorithm relies on the choice of a good threshold point to correctly decide the number of trees in the forest in advance. In this section, we propose new methods to automatically decide the number of trees in a forest, through the analysis of the dissimilarity matrix values.

3.1.1 Automatic Optimum Branching (AOB)

Following a successful approach to automatically decide the number of trees in an IPF using the Oriented Kruskal algorithm, one might wonder what happens if we apply a similar process to the optimum branching algorithm explained in Section 2.2.1. Thus, in this section, we extend upon this idea, proposing the Automatic Optimum Branching algorithm (AOB), with the necessary modifications to deal with its particularities.

Algorithm 1 describes our proposed implementation, which requires the following parameters: the number n of semantically similar images, an $n \times n$ dissimilarity matrix M built upon these images, and the parameter γ_{AOB}, which is the number of standard deviations used to limit the number of edges to be included in the forest.

Lines 1-3 initialize the vector $forest$ with n initial trees, each tree containing a vertex representing an image. In our implementation, we used the same tree representation introduced in [22], in which each position from $forest[i]$ denotes the parent of node i; $i = \{0..n-1\}$. For instance, $forest[0] = 3$ means that edge $(3, 0)$ exists in the forest. Lines 4-6 initialize the auxiliary variables n_{edges}, which is used to keep track of the number of accepted edges, the variables x_1 and x_2 to calculate the standard deviation of accepted edges, and G to represent the graph constructed using the values from the dissimilarity matrix M. In Line 7, G is used as input for the Optimum Branching (OB) algorithm, returning its minimum branching in B. Subsequently, the edges of B are sorted into non-decreasing order in Line 8 and used to decide the number of trees in this forest in the `for` loop in Lines 9-21.

In Line 10, the evaluation of the standard deviation only starts when the number of processed edges is greater than half of the edges of the minimum branching. This was done to avoid removing edges that are still valid to our approach, since with fewer values used for the calculation of the standard deviation, it is not possible to obtain a stable result.

In the next step, the algorithm updates the number of accepted edges and their

Algorithm 1 Automatic Optimum Branching

```
Input: number of $n$ semantically similar images, $n \times n$ dissimilarity matrix $M$ and the number of standard
    deviations $\gamma_{\text{AOB}}$ used as limit (default: $\gamma_{\text{AOB}} = 2$)
Output: reconstructed IPF $forest$
 1: for $i \in [1..n]$ do                                        ▷ Initialization
 2:     $forest[i] \leftarrow i$
 3: end for
 4: $n_{edges} \leftarrow 0$                                      ▷ Number of processed edges
 5: $x_1 \leftarrow x_2 \leftarrow 0$                             ▷ Variables to calculate $\sigma$ dynamically
 6: $G \leftarrow Graph(M)$
 7: $B \leftarrow OB_{min}(G)$                                    ▷ Minimum branching
 8: $B' \leftarrow$ Sort edges $(i, j)$ of $B$ by weight in non-decreasing order
 9: for $(i, j) \in B'$ do
10:     if $(n_{edges} > |B'|/2)$ then
11:         $\sigma_{\text{AOB}} \leftarrow \sqrt{\frac{x_1 - \left(\frac{x_2^2}{n_{edges}}\right)}{n_{edges} - 1}}$
12:         if $(w(i, j) - last > \gamma_{\text{AOB}} \times \sigma_{\text{AOB}})$ then
13:
14:             return $forest$;
15:         end if
16:     end if
17:     $n_{edges} \leftarrow n_{edges} + 1$                      ▷ Updates auxiliary variables
18:     $last \leftarrow w(i, j)$                                  ▷ Last processed edge
19:     $x_1 \leftarrow x_1 + w(i, j)^2$
20:     $x_2 \leftarrow x_2 + w(i, j)$
21:     $forest[j] \leftarrow i$                                   ▷ Adds new edge to the forest
22: end for
23: return $forest$                                               ▷ Returns the final forest
```

standard deviation, includes the new edge into the *forest* vector, and goes back to the beginning of the loop. Otherwise, the algorithm leaves the loop and the final result stored in the vector *forest* is returned. In our approach, we use $\gamma_{\text{AOB}} = 2$ as the default value (for more details about how this value was obtained, please refer to Section 3.3.1).

3.1.2 Extended Automatic Optimum Branching (E-AOB)

Up to this point, it is possible to obtain a first result for the IPF using AOB algorithm. However, after some experiments, we noticed that the IPF reconstruction could be further improved by also executing the OB algorithm on each tree belonging to this forest recursively. One important thing to notice in this process is the fact that AOB algorithm considers all edges to construct the minimum branching. Once we remove some edges to build the forest, we create several partitions that are independent of each other. If these partitions are analyzed separately, the OB algorithm will choose edge connections that are optimal considering only the edges that belong to the current partition. This re-execution characterizes our Extended Automatic Optimum Branching (E-AOB) algorithm. It is also important to notice that, if we apply this re-execution using the Automatic Oriented Kruskal (AOK), due to its greedy heuristic, the result will not change, with AOK selecting the same edges to generate the forest.

The implementation of E-AOB in Algorithm 2 requires as input the dissimilarity matrix M representing the relationship among the images and the initial forest obtained

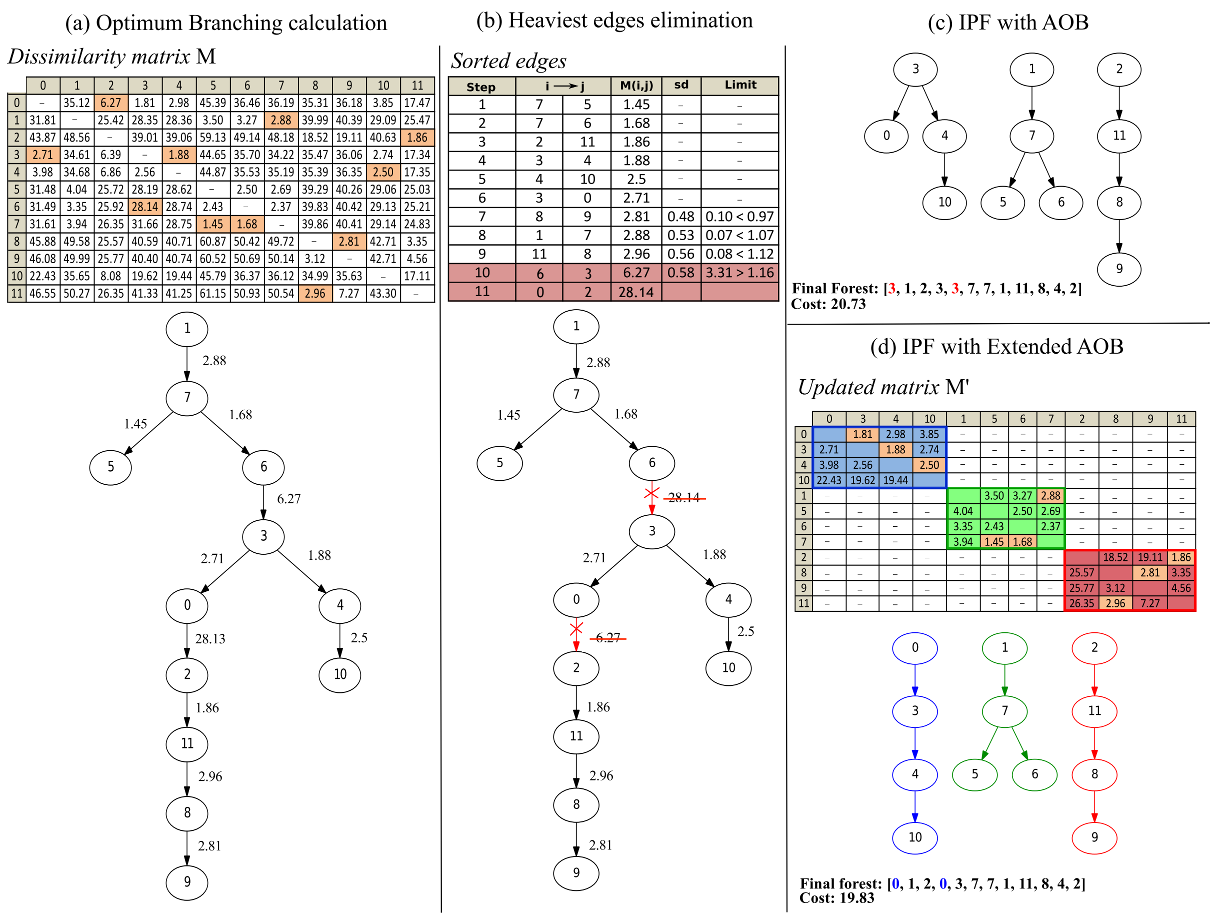
Algorithm 2 Extended Automatic Optimum Branching

```
Input: number of $n$ semantically similar images, $n \times n$ dissimilarity matrix $M$ and the vector $forest$
    with the IPF reconstructed using AOB
Output: reconstructed forest $finalForest$
 1: for $i \in [1..n]$ do                                            ▷ Initialization
 2:     $finalForest[i] \leftarrow i$
 3: end for
 4: $G \leftarrow Graph(M)$
 5: for $i \in [1..n]$ do
 6:     for $j \in [1..n]$ do                                        ▷ Keeps valid connections
 7:         if $(Roots(forest, j) = Roots(forest, i))$ then
 8:             $G'(i, j) \leftarrow G(i, j)$
 9:         end if
10:     end for
11: end for
12: $B'' \leftarrow OB_{min}(G')$                                    ▷ Execute OB again
13: for $(i, j) \in B''$ do
14:     $finalForest[j] \leftarrow i$
15: end for
16: return $finalForest$                                              ▷ Returns the final forest
```

with the execution of AOB algorithm. Similar to the AOB algorithm, Lines 1-3 initialize the vector $finalForest$ with n initial trees, each tree containing a vertex representing an image, and Line 4 initializes the variable G used to represent the graph constructed using the values from the dissimilarity matrix M. Lines 5-11 uses an auxiliary variable G' to store only the weights of the edges connected in the current forest, obtained from the vector $forest$ and representing subgraphs of graph G. In Line 14, the AOB algorithm is executed again, returning the optimum branching for each of the trees in this forest. Lines 13-14 updates the variable $finalForest$ with this new forest, which is returned by the algorithm in Line 16.

Figure **??** illustrates the execution of our proposed algorithm for a toy example with $n = 12$ semantically similar images related by the dissimilarity matrix M shown in Figure 3.1(a). After executing the OB algorithm once, we obtain the minimum branching B shown in Figure 3.1(a). We sort its edges from the lowest to the highest weight to decide which edges should be deleted to construct the forest. The table in Figure 3.1(b) shows the dynamic calculation of the standard deviation, which starts from Step 7 with edge $(8, 9)$, since $|B'|/2 = 12/2 = 6$. In this case, the current standard deviation of the edges is $\sigma_{\text{AOB}} \approx 0.48$. Since $w(8, 9) - w(3, 0) = 2.81 - 2.71 = 0.10$ and $0.10 < 2 \times 0.48 \approx 0.97$, then this edge can be accepted and the algorithm continues to the next edge. In Step 8, the edge $(1, 7)$ updates σ to 0.53 and since $w(1, 7) - w(8, 9) = 2.88 - 2.81 = 0.07$ and $0.07 < 2 \times 0.53 \approx 1.07$, the algorithm proceeds to the next edge until it reaches edge $(0, 2)$. For this edge, the standard deviation is $\sigma_{\text{AOB}} \approx 0.58$ and $w(0, 2) - w(11, 8) = 6.27 - 2.96 = 3.31$ and $3.31 > 2 \times 0.25 \approx 1.16$, which is above the allowed limit. Hence, this edge is discarded and the algorithm stops, returning the forest depicted in Figure 3.1(c).

Based on the forest reconstructed in this first part, we can apply the E-AOB method, updating the dissimilarity matrix to keep only the edges connected in the current forest and executing the OB algorithm again. The final result for the forest is depicted in



Dissimilarity matrix M

	0	1	2	3	4	5	6	7	8	9	10	11
0	–	35.12	6.27	1.81	2.98	45.39	36.46	36.19	35.31	36.18	3.85	17.47
1	31.81	–	25.42	28.35	28.36	3.50	3.27	2.88	39.99	40.39	29.09	25.47
2	43.87	48.56	–	39.01	39.06	59.13	49.14	48.18	18.52	19.11	40.63	1.86
3	2.71	34.61	6.39	–	1.88	44.65	35.70	34.22	35.47	36.06	2.74	17.34
4	3.98	34.68	6.86	2.56	–	44.87	35.53	35.19	35.39	36.35	2.50	17.35
5	31.48	4.04	25.72	28.19	28.62	–	2.50	2.69	39.29	40.26	29.06	25.03
6	31.49	3.35	25.92	28.14	28.74	2.43	–	2.37	39.83	40.42	29.13	25.21
7	31.61	3.94	26.35	31.66	28.75	1.45	1.68	–	39.86	40.41	29.14	24.83
8	45.88	49.58	25.57	40.59	40.71	60.87	50.42	49.72	–	2.81	42.71	3.35
9	46.08	49.99	25.77	40.40	40.74	60.52	50.69	50.14	3.12	–	42.71	4.56
10	22.43	35.65	8.08	19.62	19.44	45.79	36.37	36.12	34.99	35.63	–	17.11
11	46.55	50.27	26.35	41.33	41.25	61.15	50.93	50.54	2.96	7.27	43.30	–

Sorted edges

Step	i ⟶ j		M(i,j)	sd	Limit
1	7	5	1.45	–	–
2	7	6	1.68	–	–
3	2	11	1.86	–	–
4	3	4	1.88	–	–
5	4	10	2.5	–	–
6	3	0	2.71	–	–
7	8	9	2.81	0.48	0.10 < 0.97
8	1	7	2.88	0.53	0.07 < 1.07
9	11	8	2.96	0.56	0.08 < 1.12
10	6	3	6.27	0.58	3.31 > 1.16
11	0	2	28.14		

Final Forest: [3, 1, 2, 3, 3, 7, 7, 1, 11, 8, 4, 2]
Cost: 20.73

Updated matrix M'

	0	3	4	10	1	5	6	7	2	8	9	11
0		1.81	2.98	3.85	–	–	–	–	–	–	–	–
3	2.71		1.88	2.74	–	–	–	–	–	–	–	–
4	3.98	2.56		2.50	–	–	–	–	–	–	–	–
10	22.43	19.62	19.44		–	–	–	–	–	–	–	–
1	–	–	–	–		3.50	3.27	2.88	–	–	–	–
5	–	–	–	–	4.04		2.50	2.69	–	–	–	–
6	–	–	–	–	3.35	2.43		2.37	–	–	–	–
7	–	–	–	–	3.94	1.45	1.68		–	–	–	–
2	–	–	–	–	–	–	–	–		18.52	19.11	1.86
8	–	–	–	–	–	–	–	–	25.57		2.81	3.35
9	–	–	–	–	–	–	–	–	25.77	3.12		4.56
11	–	–	–	–	–	–	–	–	26.35	2.96	7.27	

Final forest: [0, 1, 2, 0, 3, 7, 7, 1, 11, 8, 4, 2]
Cost: 19.83

Figure 3.1: Simulation of AOB and E-AOB algorithm to reconstruct an IPF with three trees and 12 semantically-similar images, from a 12 × 12 dissimilarity matrix.

Figure 3.1(d), where each tree is represented by the same color of its corresponding sub-matrix used for the reconstruction. With the execution of this additional step in AOB, it is possible to correct the position of nodes 0 and 3, whose parent-child relationship was reversed in the initial IPF reconstructed by AOB.

3.2 Exploring multiple combinations

As mentioned in the previous sections, there are basically three methods currently being explored for the reconstruction of IPFs: the Automatic Oriented Kruskal (AOK) and the two extensions we introduced in Section 3.1, AOB and E-AOB, which are extensions of the Optimum Branching algorithm proposed in [19].

Considering a dissimilarity matrix M, AOK algorithm follows a greedy heuristic, while AOB and E-AOB searches for the best global solution for the phylogeny reconstruction. However, all these algorithms assume a perfect dissimilarity calculation, which is not always true. Thus, in some cases, a greedy heuristic may present better results than a global solution. In this section, aiming at exploring these different properties and their complementarity, we propose a combination among the results given by each approach, in such a way that errors introduced by one method can be fixed by other method(s).

3.2.1 Fusion methodology for IPFs reconstruction

Given n semantically similar images and the reconstructed IPFs of two or more methods to be combined, our idea consists of finding their most common elements to construct the final IPF. In the first part of the implementation, we calculate (a) the number of roots returned by each forest, (b) the number of times each node appears as a root, and (c) the number of times each edge connecting two nodes in the forest appears. In the last part, we implement the fusion scheme, constructing a new forest whose roots and edges of the trees are the ones with the highest number of votes calculated in the previous step.

However, it is known that the calculation of the dissimilarity among the images is not an exact estimation, given that one of the steps include matching of images, which is not perfect. In our implementation, we take advantage of this flaw in the dissimilarity calculation, and apply different amounts of perturbations through noise addition to the dissimilarity matrix M relating a set of images, generating 100 different variations of M and using them to reconstruct the IPFs. The number of variations was chosen in such a way that it is enough to analyze the consistency among the results and to achieve statistical significance in the analysis.

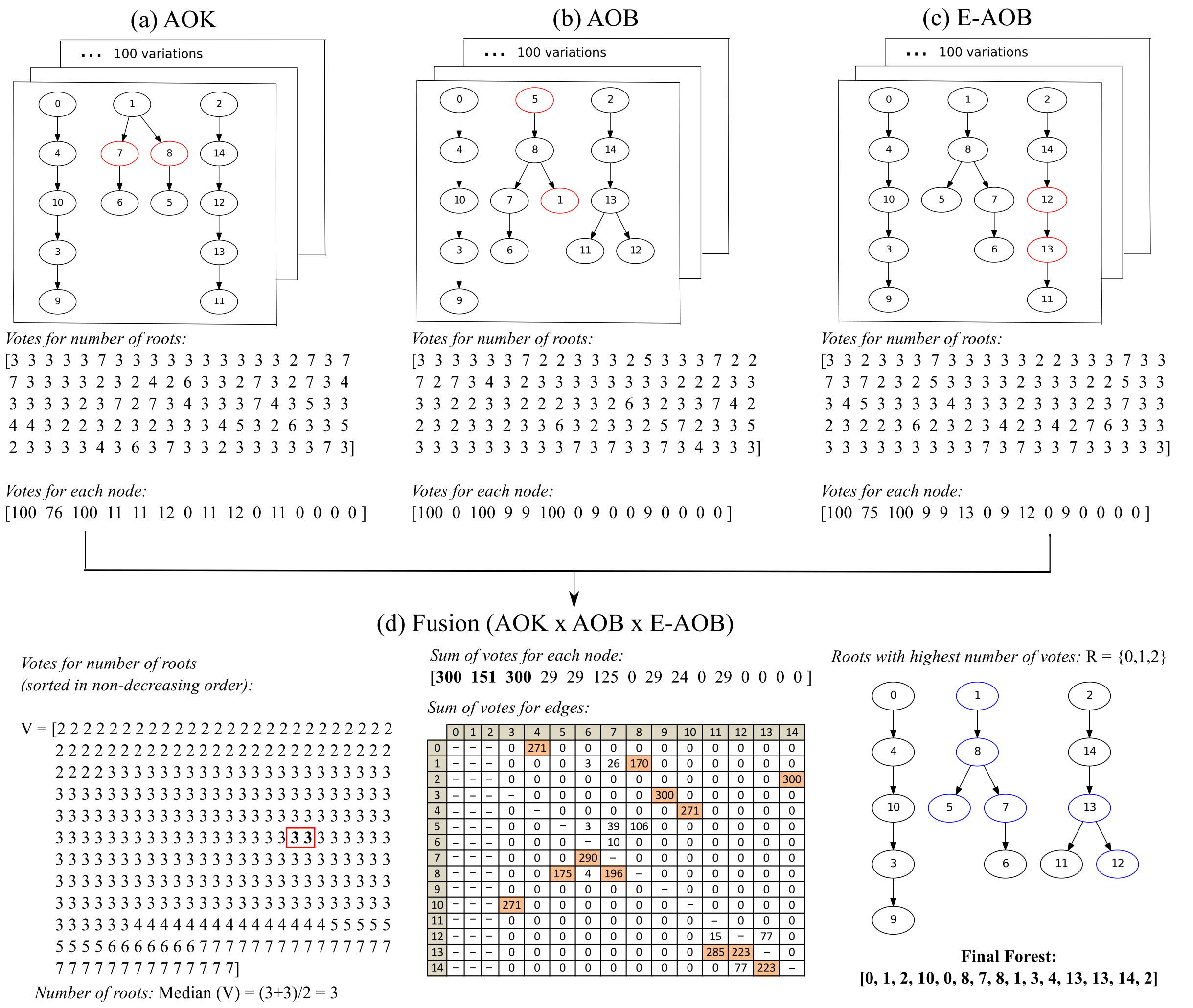
The noise was created by first calculating the standard deviation σ_M of all values in the dissimilarity matrix M relating each set of images. Then, for each value $m_{ij} \in M$, a different value between $[-k \times \sigma_M, +k \times \sigma_M]$ was randomly chosen using uniform distribution and added to the corresponding m_{ij} entry. In our implementation, $k = 1.5\%$ (for more details about this value, please refer to Section 3.3.3).

Once the number of roots and edges for all forests have been found, we calculate the final number of roots r by choosing the median of the number of votes received by all methods in each of the 100 executions.[1] Then, to decide which nodes are the roots, we select the r nodes having the highest number of votes. In case there is a tie among the votes, we randomly choose one or some of them, depending on how many roots are yet to be decided. For the edges selection, we sum up the number of times each edge connecting two nodes in each forest appears, constructing a matrix of votes for edges. Once the roots are chosen, we fix these roots and give the matrix of votes for edges as input to a maximum branching algorithm, resulting in the sought combined forest.

For instance, consider the example depicted in Figure 3.2 with $n = 15$ semantically similar images. The first set shown in Figure 3.2(a) represents all 100 possibilities of reconstructed IPFs using AOK, with the first IPF illustrating one out of 100 possible reconstructions. The nodes highlighted in red denote a node in the wrong position when compared to the ground-truth forest. Similarly, Figure 3.2(b) and (c) represent the IPFs using AOB and E-AOB, respectively. Each set of IPFs is followed by the votes for the number of roots and the number of votes each root received.

Let V be a vector storing the number of roots returned by each method for each

[1]There are several alternative strategies to calculate r, such as the average and the most frequent values. In our experiments, the median presented better results during training, but there is still room for further exploration.



(a) AOK

Votes for number of roots:
[3 3 3 3 3 7 3 3 3 3 3 3 3 3 3 3 2 7 3 7
7 3 3 3 3 2 3 2 4 2 6 3 3 2 7 3 2 7 3 4
3 3 3 3 2 3 7 2 7 3 4 3 3 3 7 4 3 5 3 3
4 4 3 2 2 3 2 3 2 3 3 3 4 5 3 2 6 3 3 5
2 3 3 3 3 4 3 6 3 7 3 3 2 3 3 3 3 3 7 3]

Votes for each node:
[100 76 100 11 11 12 0 11 12 0 11 0 0 0 0]

(b) AOB

Votes for number of roots:
[3 3 3 3 3 3 7 2 2 3 3 3 2 5 3 3 3 7 2 2
7 2 7 3 4 3 2 3 3 3 3 3 3 3 3 2 2 2 3 3
3 3 2 2 3 3 2 2 2 3 3 2 6 3 2 3 3 7 4 2
2 3 2 3 3 2 3 3 6 3 2 3 3 2 5 7 2 3 3 5
3 3 3 3 3 3 3 3 3 7 3 7 3 3 7 3 4 3 3 3]

Votes for each node:
[100 0 100 9 9 100 0 9 0 0 9 0 0 0 0]

(c) E-AOB

Votes for number of roots:
[3 3 2 3 3 3 7 3 3 3 3 3 2 2 3 3 3 7 3 3
7 3 7 2 3 2 5 3 3 3 3 2 3 3 3 2 2 5 3 3
3 4 5 3 3 3 3 4 3 3 3 2 3 3 3 2 3 7 3 3
2 3 2 2 3 6 2 3 2 3 4 2 3 4 2 7 6 3 3 3
3 3 3 3 3 3 3 3 3 7 3 7 3 3 7 3 3 3 3 3]

Votes for each node:
[100 75 100 9 9 13 0 9 12 0 9 0 0 0 0]

(d) Fusion (AOK x AOB x E-AOB)

Votes for number of roots
(sorted in non-decreasing order):

V = [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 **3 3** 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5
5 5 5 5 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 7 7]

Number of roots: Median (V) = (3+3)/2 = 3

Sum of votes for each node:
[**300 151 300** 29 29 125 0 29 24 0 29 0 0 0 0]

Sum of votes for edges:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	–	–	–	0	271	0	0	0	0	0	0	0	0	0	0
1	–	–	–	0	0	0	3	26	170	0	0	0	0	0	0
2	–	–	–	0	0	0	0	0	0	0	0	0	0	0	300
3	–	–	–	–	0	0	0	0	0	300	0	0	0	0	0
4	–	–	–	0	–	0	0	0	0	0	271	0	0	0	0
5	–	–	–	0	0	–	3	39	106	0	0	0	0	0	0
6	–	–	–	0	0	0	–	10	0	0	0	0	0	0	0
7	–	–	–	0	0	0	290	–	0	0	0	0	0	0	0
8	–	–	–	0	0	175	4	196	–	0	0	0	0	0	0
9	–	–	–	0	0	0	0	0	0	–	0	0	0	0	0
10	–	–	–	271	0	0	0	0	0	0	–	0	0	0	0
11	–	–	–	0	0	0	0	0	0	0	0	–	0	0	0
12	–	–	–	0	0	0	0	0	0	0	0	15	–	77	0
13	–	–	–	0	0	0	0	0	0	0	0	285	223	–	0
14	–	–	–	0	0	0	0	0	0	0	0	0	77	223	–

Roots with highest number of votes: R = {0,1,2}

Final Forest:
[0, 1, 2, 10, 0, 8, 7, 8, 1, 3, 4, 13, 13, 14, 2]

Figure 3.2: Reconstructed IPFs for algorithms (a) AOK, (b) AOB, and (c) E-AOB methods used separately. Highlighted in red, the nodes in the wrong position with respect to the ground-truth forest, which can happen in different positions within the 100 variations. In (c) the final forest after fusion of IPF reconstruction algorithms, with the corrected nodes highlighted in blue.

variation of the IPF. To calculate the number of roots the final IPF should have, we choose the median of V, that is, $Median(V) = 3$. To decide which nodes are the roots, we select the nodes with the highest number of votes, which in this case are Roots $= \{0, 1, 2\}$.

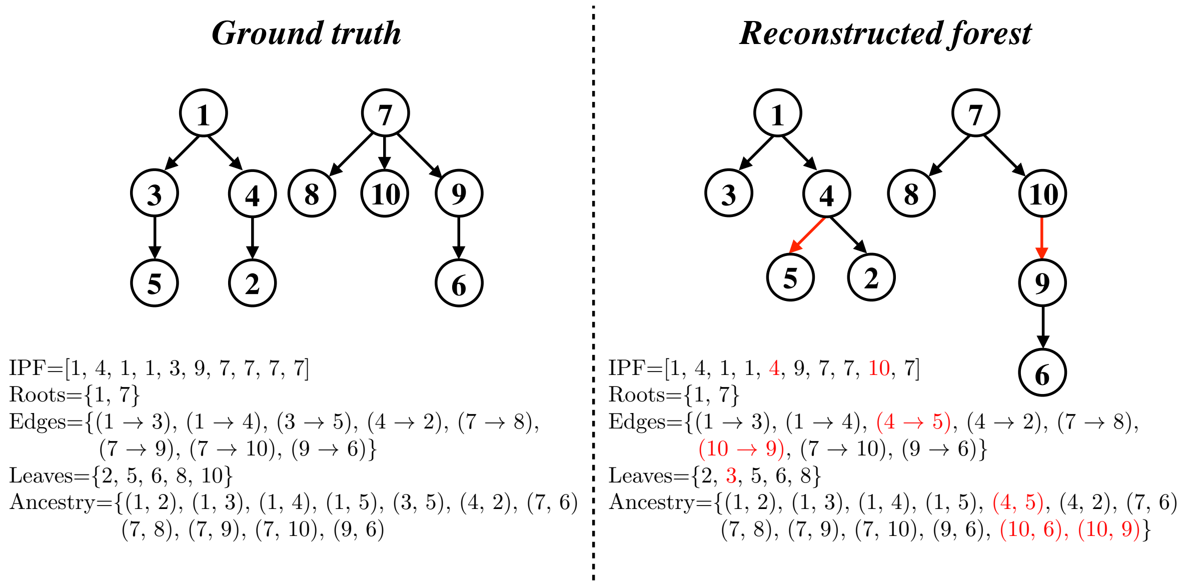
To count the votes for edges, we go through all edges (except for the ones pointing to the roots), increasing their score in a matrix of votes every time the edge (i, j) appears, and summing up the votes for the edges in all forests being combined. Figure 3.2(d) shows the sum of votes for roots and edges obtained with the combination of AOK $\times$ AOB $\times$ E-AOB, and the result after the fusion. The nodes highlighted in blue in the final forest represent the nodes that were previously in the wrong position (highlighted in red in the forest returned by the first execution of each of the methods), and were corrected after applying our fusion algorithm. Since the forest obtained after the fusion is the same forest than the one used to create this example, it is possible to see the fusion algorithm was able to correct the roots as well as the position of nodes $\{1, 5, 7, 8, 12, 13\}$.

3.3 Experimental Setup

In this section, we present the experiments that were performed for image phylogeny forest reconstruction, the obtained results and some discussions.

3.3.1 Evaluation

For evaluating the reconstructed IPFs, we consider the same quantitative metrics (*roots, dges, leaves* and*ancestry*) introduced by Dias et al. [22], considering scenarios where the ground truth is available. The *roots* metric measures if the reconstructed forest contains exactly the same roots as the ground-truth forest, i.e., the algorithm was able to find the very original images used to start the near-duplicate generation processes. *Edges* and *ancestry* measure how well the algorithm finds the kinship relationships along time. While edges assess this information only locally and independently, ancestry assesses the entire evolutionary process of a given image (a full branch in the tree). Finally, the *leaves* metric compares the leaves (most modified images in a given branch of the tree) found by an algorithm with the original ones in the ground-truth forest. Figure 3.3 shows an example of the calculation of these evaluation metrics.



IPF=[1, 4, 1, 1, 3, 9, 7, 7, 7, 7]
Roots={1, 7}
Edges={(1 $\rightarrow$ 3), (1 $\rightarrow$ 4), (3 $\rightarrow$ 5), (4 $\rightarrow$ 2), (7 $\rightarrow$ 8), (7 $\rightarrow$ 9), (7 $\rightarrow$ 10), (9 $\rightarrow$ 6)}
Leaves={2, 5, 6, 8, 10}
Ancestry={(1, 2), (1, 3), (1, 4), (1, 5), (3, 5), (4, 2), (7, 6) (7, 8), (7, 9), (7, 10), (9, 6)

IPF=[1, 4, 1, 1, 4, 9, 7, 7, 10, 7]
Roots={1, 7}
Edges={(1 $\rightarrow$ 3), (1 $\rightarrow$ 4), (4 $\rightarrow$ 5), (4 $\rightarrow$ 2), (7 $\rightarrow$ 8), (10 $\rightarrow$ 9), (7 $\rightarrow$ 10), (9 $\rightarrow$ 6)}
Leaves={2, 3, 5, 6, 8}
Ancestry={(1, 2), (1, 3), (1, 4), (1, 5), (4, 5), (4, 2), (7, 6) (7, 8), (7, 9), (7, 10), (9, 6), (10, 6), (10, 9)}

Figure 3.3: Evaluation metrics: roots, edges, leaves and ancestry. We represent the IPF as a vector, where IPF$[v] = u$ means that there exists the edge $(u \rightarrow v)$ in the forest, and one node v is a root only if $IPF[v] = v$. The differences between the reconstructed forest and the ground truth forest are highlighted in red.

The evaluation metrics are defined by:

Root: $R(\text{IPF}_1, \text{IPF}_2) = \frac{|R_1 \cap R_2|}{|R_1 \cup R_2|}$

Edges: $E(\text{IPF}_1, \text{IPF}_2) = \frac{|E_1 \cap E_2|}{|E_2|}$,

Leaves: $L(\text{IPF}_1, \text{IPF}_2) = \frac{|L_1 \cap L_2|}{|L_1 \cup L_2|}$,

Ancestry: $A(\text{IPF}_1, \text{IPF}_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|}$.

where N is the number of nodes in a tree, IPF_1 is the reconstructed forest with elements represented by R_1 (roots), E_1 (edges), L_1 (leaves) and A_1 (ancestry), IPF_2 is the forest ground truth with elements R_2, E_2, L_2 and A_2. The roots, leaves and ancestry metrics calculate the intersection of the results returned by IPF_1 with respect to the reference forest IPF_2, and normalizes it by the union of both sets, while the edges metric calculate the score of correct edges by normalizing the intersection of the result returned by IPF_1 with respect to the reference forest IPF_2 by the ground truth set. For instance, in the example of Figure 3.3, the root metric yields $R(\text{IPF}_1, \text{IPF}_2) = 2/2 = 100\%$, the edges metric yields $E(\text{IPF}_1, \text{IPF}_2) = 6/8 = 75\%$, the leaves metric yields $L(\text{IPF}_1, \text{IPF}_2) = 4/6 = 66.6\%$, and ancestry metric yields $A(\text{IPF}_1, \text{IPF}_2) = 10/14 = 71.4\%$.

3.3.2 Dataset

For our experiments in a controlled scenario, we consider images taken with a single camera (OC) and with multiple cameras (MC) having similar scene semantics (the main content of the image is the same, but with small variations in the camera parameters, such as viewpoint, zoom, etc.). Figure 3.4 depicts some examples of scenes we considered in this thesis. Three different datasets were used in the experiments, with images in the JPEG format:

- **Training Dataset**: it represents a small exploratory set containing images from OC and MC scenarios, taken from three different cameras, three different scenes, three images per camera, four forest sizes $|F| = \{2..5\}$, one topology[2], and 10 random variations of parameters for creating the near-duplicate images, totaling $2 \times 3^3 \times 4 \times 1 \times 10 = 2,160$ forests.

- **Dataset A**: it comprises images from OC and MC scenarios, three different scenes, three different cameras, three images per camera, four forest sizes $|F| = \{2..5\}$, four different tree topologies, and ten random variations of parameters for creating the near-duplicate images. Therefore, the dataset comprises $2 \times 3^3 \times 4 \times 4 \times 10 = 8,640$ forests. The Training Dataset as well as Dataset A are the same used by [22].

- **Dataset B**: it comprises images randomly selected from a set of 20 different scenes, 10 different cameras, 10 images per camera, 10 different tree topologies, 10 random variations of parameters for creating the near duplicate images, and forests with 10 trees each. For each of the cases, OC and MC, a total of 2,000 forests within this set were randomly selected with forests of size $|F| = \{2..10\}$. Therefore, this set comprises $2 \times 2,000 \times 9 = 36,000$ forests.

The image transformations used to create the near duplicates are the same used in [25]: geometric transformations, brightness and contrast adjustment, and lossy compression using the standard lossy JPEG algorithm. Table 3.1 details the transformations and their

[2]A topology refers to the form of the trees in a forest.

Figure 3.4: Examples of pictures present in the datasets described in this work.

Table 3.1: Transformations and their operational ranges for creating controlled data set

Resampling (Up/Down)	$[90\%, 110\%]$
Rotation	$[-5^o, 5^o]$
Scaling by axis	$[90\%, 110\%]$
Off-diagonal correction	$[0.95, 1.05]$
Cropping	$[0\%, 5\%]$
Brighness Adjustment	$[-10\%, 10\%]$
Contrast Adjustment	$[-10\%, 10\%]$
Gamma Correction	$[0.9, 1.1]$
Re-compression	$[50\%, 100\%]$

operational ranges for creating the controlled data set. We performed the near-duplicate generation process using the algorithms implemented in the ImageMagick Library[3].

The experiments have two parts: first, we analyze the effects of choosing different values for the parameter γ_{AOB} (threshold calculation), and the value of the parameter k (amount of noise to be introduced in all variations of the dissimilarity matrix). Second, we evaluated the robustness of the approaches proposed in this work in a controlled scenario. For the tests using the OB algorithm, we consider a black-box implementation[4] that follows Tarjan's description [66].

[3]http://www.imagemagick.org/script/index.php

[4]Available at http://edmonds-alg.sourceforge.net

3.3.3 First part: calculation of parameters $\gamma_{\mathbf{AOB}}$ and k

The experiments described in this section were performed using the Training Dataset previously described. To find the value of parameter γ_{AOB}, we studied the behavior of AOB using Algorithm 1, varying the thresholds in the interval $[\mu_{\text{AOB}} - 2\sigma_{\text{AOB}}, \mu_{\text{AOB}} + 2\sigma_{\text{AOB}}]$, separated by steps of 0.1. From this experiment, we defined $\tau_{\text{AOB}} = \mu_{\text{AOB}} + (2 \times \sigma_{\text{AOB}})$, and, as a consequence, $\gamma_{\text{AOB}} = 2$.

The parameter k used to add noise in the dissimilarity matrices were chosen according to the formula described in Section 3.2.1. The parameter k was tested in the interval $[1\%, 10\%]$, separated by steps of 0.5%. The best results were achieved for $k = 1.5\%$ of the σ_M value (recall that σ_M is the standard deviation of the input dissimilarity matrix).

3.3.4 Second part: validation in a controlled scenario

In the second part of the experiments, we analyzed the robustness of AOK, AOB, and E-AOB in two rounds: (a) using each method separately and (b) their possible combinations C = {AOK × AOB, AOK × E-AOB, AOB × E-AOB, AOK × AOB × E-AOK} for Datasets A and B.

To directly compare the algorithms, the error variation $\Delta error$ was calculated with respect to each metric (roots, edges, leaves and ancestry), using the same equation introduced by Dias et al. [19]:

$$\Delta error_{metric}(\text{M1,M2}) = \left(\frac{1 - \text{M1}_{metric}}{1 - \text{M2}_{metric}}\right) - 1 \tag{3.1}$$

where M1 represents the method being evaluated in comparison to method M2. For instance, in Table 3.2, in the column $\Delta error$(E-AOB, AOK), M1 represents E-AOB, M2 represents AOK, and each of the columns **Roots, Edges, Leaves** and **Ancestry** represents the results for $\Delta error_{roots}$, $\Delta error_{edges}$, $\Delta error_{leaves}$, and $\Delta error_{ancestry}$, respectively.

A $\Delta error < 0$ value means that algorithm M1 outperformed M2, being able to reduce the error. Table 3.2 compares the methods AOB and E-AOB against AOK, with E-AOB algorithm outperforming the other two methods in both datasets, A and B, regardless of the number of trees in the forest. On average, E-AOB algorithm reduces the error for finding the roots in 41% for the metric roots, 7% for edges, 10% for leaves and 11% for ancestry in dataset A. For dataset B, the average of the error reduction is approximately 20% for the metrics roots, leaves and ancestry, and 18% for edges.

A Wilcoxon signed-rank test was performed for all metrics in dataset B, comparing (i) AOK with AOB, and (ii) AOK with E-AOB. In Table 3.2, the results for this test are described in its last row, with the blue dots indicating that the difference among the results are statistically significant, at 95% confidence level, in favor of AOB or E-AOB, while the red crosses represent statistical difference in favor of the baseline AOK. In case (i), a statistical difference in favor of AOK was found for metric roots in the OC scenario, and for metrics roots and ancestry in the MC scenario. These results show that AOB is only able to improve the results of AOK regarding the metrics edges and leaves. On the other hand, when comparing AOK and E-AOB (Scenario ii), all differences are

Table 3.2: Comparison among AOK [22] and the variations of AOB algorithm

(a) Semantically similar images from the scenario using a single camera (OC)

	AOK				**AOB**				**E-AOB**				$\Delta error$ **(E-AOB, AOK)**			
								Dataset A								
$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.834	0.788	0.818	0.740	0.842	0.805	0.834	0.755	0.911	0.809	0.839	0.773	-46.60%	-9.80%	-11.50%	-12.60%
3	0.882	0.823	0.827	0.798	0.898	0.833	0.842	0.814	0.938	0.838	0.847	0.829	-47.40%	-8.20%	-11.70%	-15.50%
4	0.870	0.831	0.820	0.826	0.861	0.835	0.827	0.828	0.914	0.839	0.833	0.843	-33.80%	-4.70%	-6.80%	-10.00%
5	0.883	0.780	0.812	0.762	0.887	0.788	0.824	0.778	0.930	0.791	0.829	0.787	-39.80%	-5.10%	-8.60%	-10.70%
								Dataset B								
$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.803	0.852	0.838	0.806	0.786	0.865	0.857	0.822	0.833	0.872	0.864	0.841	-15.44%	-14.02%	-16.26%	-17.80%
3	0.850	0.874	0.856	0.845	0.808	0.887	0.882	0.852	0.885	0.896	0.891	0.879	-23.17%	-17.52%	-24.16%	-21.60%
4	0.854	0.878	0.859	0.848	0.799	0.893	0.886	0.856	0.892	0.903	0.897	0.884	-25.85%	-20.51%	-26.62%	-24.00%
5	0.854	0.879	0.859	0.844	0.768	0.892	0.884	0.843	0.889	0.903	0.896	0.878	-23.91%	-20.14%	-26.27%	-21.74%
6	0.849	0.880	0.859	0.842	0.760	0.893	0.881	0.839	0.879	0.904	0.894	0.872	-20.29%	-20.11%	-24.53%	-19.22%
7	0.832	0.880	0.856	0.830	0.750	0.894	0.878	0.833	0.868	0.905	0.891	0.864	-21.49%	-20.84%	-24.40%	-20.31%
8	0.799	0.883	0.856	0.817	0.723	0.899	0.880	0.824	0.841	0.910	0.894	0.855	-20.84%	-22.81%	-26.16%	-20.87%
9	0.778	0.884	0.856	0.802	0.704	0.899	0.879	0.810	0.822	0.910	0.892	0.840	-19.58%	-22.36%	-25.51%	-19.28%
10	0.755	0.883	0.854	0.784	0.682	0.898	0.877	0.793	0.802	0.908	0.890	0.823	-19.12%	-21.30%	-24.31%	-18.20%
	Wilcoxon				×	●	●	●	●	●	●	●				

(b) Semantically similar images from the scenario using multiple cameras (MC)

	AOK				**AOB**				**E-AOB**				$\Delta error$ **(E-AOB, AOK)**			
								Dataset A								
$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.830	0.787	0.817	0.739	0.837	0.805	0.833	0.755	0.908	0.809	0.839	0.773	-45.80%	-10.30%	-12.00%	-12.80%
3	0.883	0.822	0.822	0.801	0.873	0.831	0.837	0.811	0.936	0.837	0.845	0.832	-45.00%	-8.60%	-12.70%	-15.80%
4	0.887	0.830	0.817	0.833	0.835	0.830	0.822	0.821	0.925	0.838	0.832	0.846	-34.00%	-4.70%	-8.10%	-8.30%
5	0.898	0.782	0.814	0.775	0.868	0.786	0.824	0.777	0.937	0.791	0.831	0.794	-38.30%	-4.30%	-9.10%	-8.30%
								Dataset B								
$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.810	0.859	0.838	0.818	0.765	0.865	0.848	0.819	0.830	0.875	0.856	0.844	-10.42%	-11.03%	-11.19%	-14.69%
3	0.854	0.874	0.852	0.843	0.782	0.879	0.863	0.840	0.876	0.891	0.875	0.872	-14.93%	-13.46%	-15.24%	-18.75%
4	0.872	0.878	0.854	0.855	0.781	0.883	0.863	0.850	0.890	0.896	0.876	0.884	-14.53%	-14.48%	-15.14%	-20.19%
5	0.882	0.877	0.853	0.856	0.776	0.884	0.867	0.849	0.903	0.897	0.879	0.887	-17.63%	-16.89%	-18.11%	-21.07%
6	0.895	0.877	0.854	0.863	0.772	0.885	0.866	0.852	0.911	0.898	0.880	0.890	-15.47%	-17.69%	-18.19%	-19.61%
7	0.897	0.880	0.855	0.867	0.778	0.889	0.868	0.859	0.917	0.903	0.883	0.896	-19.53%	-18.79%	-19.33%	-22.32%
8	0.900	0.886	0.856	0.872	0.785	0.895	0.871	0.866	0.922	0.908	0.886	0.901	-22.59%	-19.73%	-20.85%	-23.16%
9	0.893	0.887	0.860	0.867	0.774	0.897	0.874	0.860	0.915	0.909	0.889	0.896	-20.58%	-19.43%	-20.77%	-21.28%
10	0.885	0.888	0.862	0.859	0.768	0.896	0.875	0.852	0.909	0.908	0.889	0.886	-20.48%	-18.27%	-19.42%	-19.24%
	Wilcoxon				×	●	●	×	●	●	●	●				

statistically significant in favor of E-AOB, confirming this method has better performance than the AOK method.

Using as baseline the results presented in Table 3.2, better results for the fusion approach were found for the combination (AOK $\times$ AOB $\times$ E-AOB). Although the combination using only AOK and E-AOB seems to be a better choice, the results of this fusion presented lower results compared to the one using the three methods together.

In the experiments, we noticed that the AOB method is important in the fusion approach, mainly because the dissimilarity calculation is not perfect. For instance, in some cases, the dissimilarity between a pair of images may be exchanged: node A is the real parent of node B, but the dissimilarity $d(B, A) < d(A, B)$. This will lead AOK to choose the wrong direction between these images, since it follows a greedy heuristic. Complementarily, by taking into account all dissimilarities, AOB may choose the correct

Table 3.3: Results for Fusion (AOK × AOB × E-AOB) and the $\Delta error$ in comparison to AOK and E-AOB.

(a) Semantically similar images from the scenario using a single camera (OC)

	Fusion (AOK × AOB × E-AOB)				**$\Delta error$ (Fusion, AOK)**				**$\Delta error$ (Fusion, E-AOB)**			
						Dataset A						
$\|F\|$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.917	0.811	0.841	0.777	-50.25%	-10.83%	-12.52%	-14.22%	-6.89%	-1.09%	-1.11%	-1.86%
3	0.943	0.840	0.85	0.833	-52.03%	-9.75%	-13.60%	-17.42%	-8.75%	-1.67%	-2.16%	-2.28%
4	0.926	0.843	0.838	0.848	-43.32%	-6.98%	-10.09%	-12.84%	-14.31%	-2.38%	-3.49%	-3.16%
5	0.932	0.794	0.831	0.789	-42.23%	-6.20%	-9.97%	-11.60%	-3.97%	-1.18%	-1.46%	-1.06%
						Dataset B						
$\|F\|$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.867	0.884	0.879	0.862	-32.34%	-22.02%	-25.61%	-28.64%	-19.99%	-9.31%	-11.17%	-13.18%
3	0.900	0.900	0.896	0.886	-33.08%	-21.03%	-27.74%	-26.15%	-12.90%	-4.25%	-4.71%	-5.81%
4	0.901	0.905	0.898	0.889	-32.19%	-22.25%	-27.83%	-26.82%	-8.55%	-2.19%	-1.65%	-3.70%
5	0.893	0.905	0.897	0.879	-26.25%	-21.61%	-26.84%	-22.20%	-3.07%	-1.84%	-0.78%	-0.60%
6	0.883	0.905	0.894	0.873	-22.45%	-20.89%	-24.73%	-19.83%	-2.71%	-0.97%	-0.27%	-0.75%
7	0.870	0.906	0.892	0.865	-22.88%	-21.59%	-24.78%	-20.86%	-1.77%	-0.95%	-0.51%	-0.70%
8	0.837	0.911	0.894	0.853	-18.95%	-23.59%	-26.40%	-19.68%	2.38%	-1.02%	-0.33%	1.50%
9	0.818	0.911	0.893	0.838	-18.00%	-23.41%	-26.07%	-18.29%	1.96%	-1.36%	-0.76%	1.23%
10	0.796	0.909	0.890	0.820	-16.62%	-22.13%	-24.76%	-16.62%	3.09%	-1.05%	-0.59%	1.93%
		Wilcoxon			●	●	●	●	—	●	●	—

(b) Semantically similar images from the scenario using multiple cameras (MC)

	Fusion (AOK × AOB × E-AOB)				**$\Delta error$ (Fusion, AOK)**				**$\Delta error$ (Fusion, E-AOB)**			
						Dataset A						
$\|F\|$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.916	0.810	0.840	0.778	-50.90%	-10.97%	-12.83%	-14.70%	-9.35%	-0.70%	-0.96%	-2.15%
3	0.944	0.840	0.848	0.837	-52.24%	-10.34%	-14.34%	-18.23%	-13.13%	-1.86%	-1.87%	-2.86%
4	0.932	0.841	0.835	0.851	-40.42%	-6.62%	-10.16%	-10.97%	-9.77%	-2.01%	-2.21%	-2.96%
5	0.943	0.793	0.833	0.798	-43.95%	-5.23%	-10.21%	-10.20%	-9.16%	-0.98%	-1.26%	-2.09%
						Dataset B						
$\|F\|$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.871	0.889	0.871	0.869	-32.12%	-21.24%	-20.20%	-28.25%	-24.23%	-11.48%	-10.15%	-15.90%
3	0.900	0.898	0.88	0.884	-31.52%	-18.45%	-18.89%	-26.08%	-19.51%	-5.77%	-4.30%	-9.02%
4	0.905	0.899	0.879	0.891	-25.81%	-17.50%	-17.63%	-24.48%	-13.21%	-3.52%	-2.93%	-5.38%
5	0.916	0.900	0.882	0.891	-28.84%	-18.82%	-19.94%	-24.20%	-13.61%	-2.32%	-2.24%	-3.96%
6	0.921	0.901	0.883	0.895	-25.00%	-19.65%	-19.82%	-22.78%	-11.27%	-2.37%	-1.99%	-3.94%
7	0.928	0.905	0.885	0.900	-30.30%	-20.38%	-20.39%	-25.20%	-13.38%	-1.95%	-1.32%	-3.71%
8	0.931	0.910	0.888	0.904	-31.63%	-21.05%	-22.02%	-25.45%	-11.68%	-1.65%	-1.48%	-2.98%
9	0.926	0.911	0.891	0.900	-31.10%	-21.00%	-22.22%	-24.34%	-13.24%	-1.95%	-1.83%	-3.89%
10	0.917	0.910	0.891	0.890	-28.03%	-19.68%	-20.96%	-21.53%	-9.50%	-1.72%	-1.91%	-2.84%
		Wilcoxon			●	●	●	●	●	●	●	●

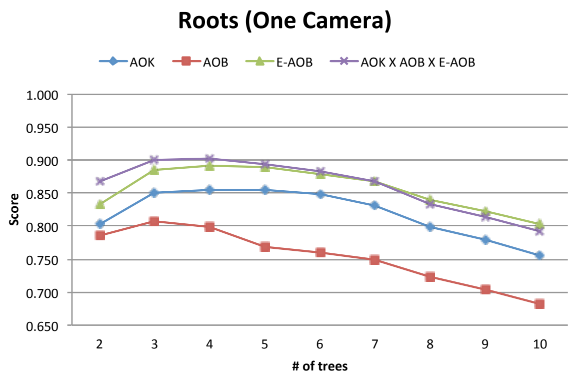
direction for some of these cases. With the re-execution performed by E-AOB (on each tree separately), some edges may change their direction to obtain the optimum branching. In some cases, E-AOB chooses edges that AOK also chooses erroneously. Therefore, in the fusion (AOK × E-AOB), since we take into account only the votes of both methods, if both of them have a large number of votes for some wrong edge, it will lead to the wrong result. However, when we include the votes from AOB, combined to the votes correctly obtained by E-AOB, we may be able to keep the right edge direction. If the dissimilarity calculation can be further improved, there is a possibility that the fusion AOK $\times$ E-AOB may present better results than the fusion of AOK $\times$ AOB $\times$ E-AOB. Improvements on this dissimilarity calculation is not included herein, but it is a challenge to be addressed by future research of the community. Results for the fusions (AOK $\times$ AOB), (AOK $\times$ E-AOB), and (AOB $\times$ E-AOB), can be found in the Appendix A.

Table 3.3 shows the results for the fusion (AOK $\times$ AOB $\times$ E-AOB) and the error variation in comparison to AOK and to the current best performing algorithm, E-AOB. In this fusion, the OC scenario had lower performance only in Dataset B, introducing more error for metrics roots and ancestry when the forest has more than eight trees. However, after running a Wilcoxon signed-rank test for these results, no statistical difference was found among them (represented by the green dashes in the last row of the table). For the other metrics in OC, and all metrics in the MC scenario, differences are statistically significant at 95% confidence level in favor of the fusion approach.

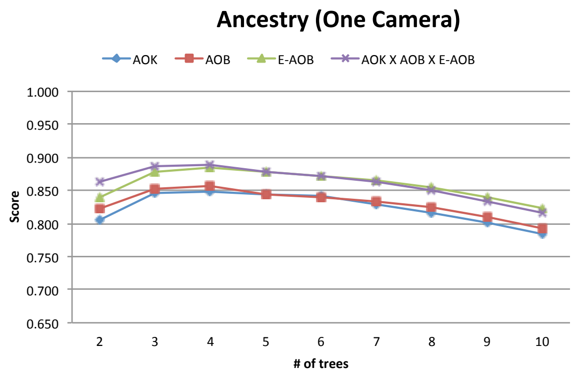
The charts in Figure 3.5 depict a comparison among the performance of all methods described in Table 3.2 and the fusion. The metrics root and ancestry are the ones featured in these graphics because they are usually the most important in a forensic scenario. For instance, roots are important to find the source of the illegal activity, and with the ancestry relationship, we can trace back the users involved in the chain of transformations. In the Figure 3.5, it becomes clear the improvement obtained with E-AOB and the fusion in comparison to AOK and AOB.

Additionally, Figure 3.6 shows a comparison of the error reduction ($-\Delta error$) between methods E-AOB and the fusion AOK $\times$ AOB $\times$ E-AOB, in comparison to AOK, described in Table 3.3. A closer analysis of the results shows that the MC scenario has better performance than OC. In a forensic scenario, this is advantageous since, in most of the cases, more than one user may be producing and spreading some illegal content from the same scene, and using different cameras. In this case, it is important to find enough evidence to frame all users involved in the illegal activity (e.g., cases of child pornography). Furthermore, the MC case is also more common in other scenarios. Consider, for instance, that we want to find all images related to a certain topic. In this case, the images will come from different photographers and higher are the chances they will be using different cameras. Therefore, the correct identification and separation of each group of images plays an important role.

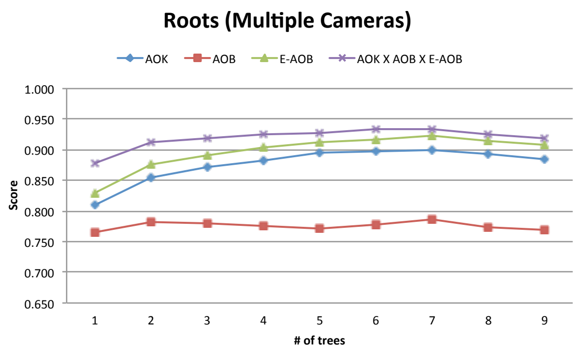
Regarding the methods' running time, the biggest bottleneck corresponds to the dissimilarity calculation, as reconstructing the forest and performing the fusion is much faster. Considering a forest with 100 semantically-similar images, the dissimilarity function step spends 38.31 minutes, on average, for all cases. On the other hand, the time to perform the fusion of three methods (including the time spent in 100 perturbations of each method, the votes counting, and the final execution of the fusion) takes, on average, 0.478 seconds. Tables 3.4 and 3.5 detail the running time for all forest sizes. These experiments were performed in a machine with processor Intel Xeon E5645, 2.40GHz, with 16GB of memory, and running Ubuntu 12.04.4 LTS (GNU Linux 3.5.0-49-generic



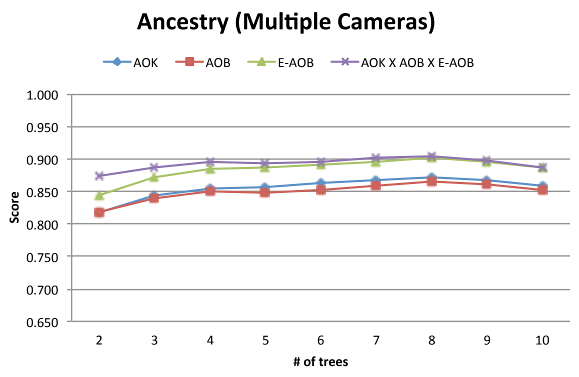
(a) Metric: Roots (OC)



(b) Metric: Ancestry (OC)



(c) Metric: Roots (MC)



(d) Metric: Ancestry (MC)

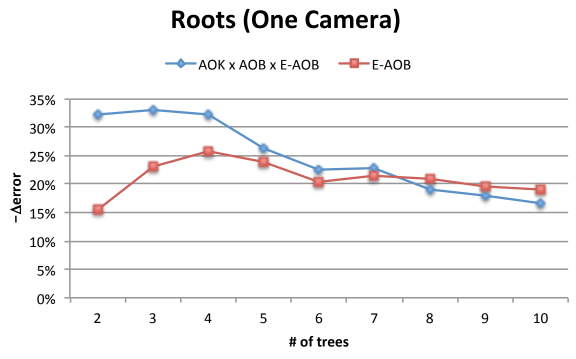
Figure 3.5: Comparison among the proposed approaches for scenarios with a single camera (OC) and with multiple cameras (MC), featuring the metrics roots and ancestry.

Table 3.4: Average time (*minutes*) for the dissimilarity calculation and the forest reconstruction (single execution).

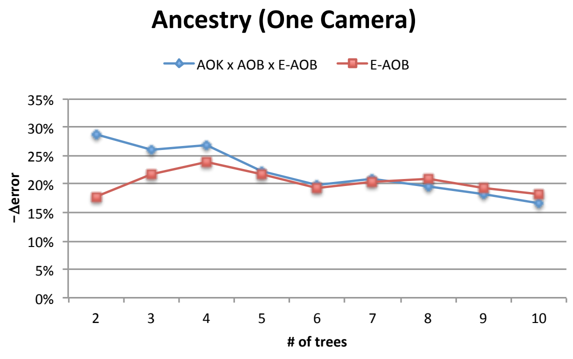
$\|F\|$	**Dissimilarity**	**AOK**	**AOB**	**E-AOB**
2	1.77	0.2×10^{-5}	30.6×10^{-5}	55.4×10^{-5}
3	4.28	0.5×10^{-5}	46.2×10^{-5}	76.7×10^{-5}
4	6.45	0.9×10^{-5}	57.7×10^{-5}	108.6×10^{-5}
5	10.23	1.5×10^{-5}	77.9×10^{-5}	136.1×10^{-5}
6	18.40	2.3×10^{-5}	95.6×10^{-5}	170.7×10^{-5}
7	23.12	3.1×10^{-5}	117.7×10^{-5}	200.6×10^{-5}
8	24.42	3.9×10^{-5}	140.2×10^{-5}	234.7×10^{-5}
9	32.13	5.4×10^{-5}	164.2×10^{-5}	272.5×10^{-5}
10	38.31	6.3×10^{-5}	191.9×10^{-5}	312.7×10^{-5}

x86_64). The source code was implemented using the OpenCV library, version 2.4.10 [5].

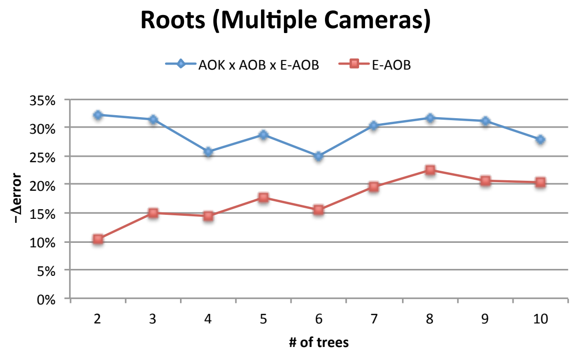
[5]`http://opencv.org/`



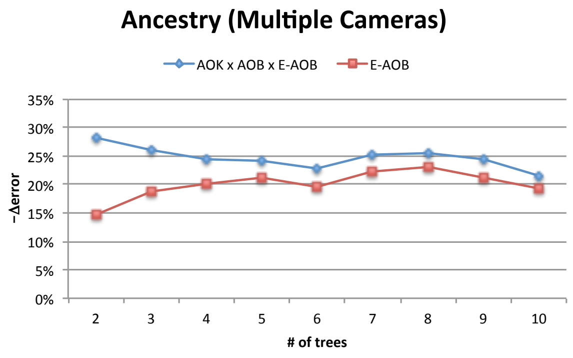
(a) Metric: Roots (OC)



(b) Metric: Ancestry (OC)



(c) Metric: Roots (MC)



(d) Metric: Ancestry (MC)

Figure 3.6: Comparison among the error reduction ($-\Delta_{error}$) of methods E-AOB and the Fusion (AOK $\times$ AOB $\times$ E-AOB) in comparison to AOK, for the metrics roots and ancestry. In (a) and (b), the graphs represent the scenario with images from a single camera (OC), while (c) and (d) show the results for images from the multiple camera (MC) scenario.

Finally, to ensure reproducibility of all experiments, all datasets used in the experiments and the entire source code are freely available. The datasets are registered on the address: `http://figshare.com/articles/Image_Phylogeny_Forests_Reconstruction/1012816` under the accession number `http://dx.doi.org/10.6084/m9.figshare.1012816`. The source-code and documentation are available in a public repository in the following address: `http://repo.recod.ic.unicamp.br/public/projects`.

3.4 Publication

The main findings of the research presented in this chapter resulted in one publication.

- **F. O. Costa, M. Oikawa, Z. Dias, S. Goldenstein and A. Rocha**; *Image phylogeny forests reconstruction.* IEEE Transaction on Information Forensics and Security (TIFS), v. 9, n. 10, p. 1533-1546, 2014. [15]

Table 3.5: Average time (*milliseconds*) considering 100 executions to perform the Fusion (AOK × AOB × E-AOB).

$\|F\|$	**AOK**	**AOB**	**E-AOB**	**Votes**	**Fusion**	**Total**
2	0.30	18.44	33.78	16.99	17.87	87.38
3	0.55	27.53	48.47	22.41	26.73	125.69
4	0.87	37.11	66.10	27.43	35.79	167.30
5	1.28	47.64	81.12	35.34	46.52	211.91
6	1.77	59.14	100.55	40.95	56.77	259.17
7	2.32	72.92	123.14	47.23	68.90	314.51
8	2.94	79.80	143.51	53.56	80.40	360.20
9	3.63	100.08	164.21	63.09	92.32	423.32
10	4.65	115.9	183.41	64.46	109.35	477.77

Chapter 4

New dissimilarity measures for Image Phylogeny Forest Reconstruction

Although the field of Image Phylogeny has been developing significantly over the past years, thus far researchers mainly focused on proposing different phylogeny reconstruction approaches [15, 19, 20, 22–25] often using a standard methodology for dissimilarity calculation as originally proposed in [25]. This dissimilarity calculation involves the estimation of the transformations that map a source image onto a target image, followed by their comparison in a point-wise fashion. As the estimation of transformations is not exact, the point-wise comparison method $\mathcal{L}$ (Equation 2.1) is strongly affected by artifacts generated in these processes (Sections 2.2.1 and 3.2.1). Given that the dissimilarity calculation directly affects the result of the final phylogeny reconstruction [25], the definition of reliable dissimilarity measure is paramount for the image phylogeny research field.

Aiming at solving those problems and increasing the quality of the phylogeny reconstruction, in this chapter, we introduce new methods to perform the dissimilarity calculation between images for the phylogeny reconstruction process. Firstly, we employ a histogram-based method to match color histograms between two near-duplicate images, better capturing possible color differences between them. Secondly, we develop a new comparison metric working on image gradients, rather than directly on the pixel domain, and, finally, we use the mutual information to compare them. The new comparison metrics aim at better tackling possible image misalignments during the mapping process of one image onto another's domain.

4.1 New Dissimilarity Calculation Techniques

As described in Section 2.2.1, the estimation of the transformation T, parameterized by $\overrightarrow{\beta}$ used to map an image $\mathcal{I}_{src}$ onto an image $\mathcal{I}_{tgt}$'s domain follows a three-step method generating $\mathcal{I}'_s = T_{\overrightarrow{\beta}}(\mathcal{I}_{src})$: geometric matching, color matching and compression matching. Then, a comparison between the estimated $\mathcal{I}'_{src} = T_{\overrightarrow{\beta}}(\mathcal{I}_{src})$ and $\mathcal{I}_{tgt}$ is performed point-wise. This point-wise comparison is usually performed using MSE calculation.

Differently from this standard dissimilarity pipeline, here we propose improvements for the dissimilarity calculation process. For that, we propose the replacement of the color matching step, which was not very accurate, and also the metric used for performing the comparison between two images. We now turn our attention to these new approaches for improving the dissimilarity calculation.

4.1.1 Histogram Color Matching

The second step of the transformation estimation T (after geometric matching) consists of mapping the color space of the source image $\mathcal{I}_{src}$ onto the target's image $\mathcal{I}_{tgt}$ color space. Previous work on image phylogeny [15,19,22,23,47] performed the color matching between two images by normalizing each color channel of $\mathcal{I}_{src}$ by the mean and standard deviation of the $\mathcal{I}_{tgt}$'s corresponding channel [58]. This method, although simplistic, works reasonably well when the color changes are minor. However, it leads to some problems when the transformations applied to the image when generating a child are stronger, specially in the case of contrast changes, gamma correction, or non-linear color mappings, which affect the distribution of pixel intensities throughout the image.

For a better color matching step, we propose to use a histogram matching technique [31]. This technique transforms the source image colors in such a way that their distribution acquires a form closer to the color distribution of the target image, by using the target image's color distribution information. Figure 4.1 shows two examples of color matching algorithms.

To match the histograms of two images $\mathcal{I}_{src}$ and $\mathcal{I}_{tgt}$, we compute their histograms, H_{src} and H_{tgt} and compute their *Cumulative Distribution Function* (CDF) [43]. For a gray-scale image $\mathcal{I}$, with L gray levels, the gray level i has the probability of

$$p^{\mathcal{I}}(i) = \frac{n_i}{n}, \quad 0 \leq i < L \tag{4.1}$$

where n is the number of pixels in the image and n_i is the number of pixels of gray value i in the histogram of the image. The CDF of an image $\mathcal{I}$ is

$$\mathcal{C}^{\mathcal{I}}(i) = \sum_{k=0}^{i} p^{\mathcal{I}}(k). \tag{4.2}$$

With $\mathcal{C}^{\mathcal{I}_{src}}$ and $\mathcal{C}^{\mathcal{I}_{tgt}}$, the CDFs for $\mathcal{I}_{src}$ and $\mathcal{I}_{tgt}$, respectively, we find a transformation $\mathcal{M}$ that maps $\mathcal{C}^{\mathcal{I}_{src}}$ onto $\mathcal{C}^{\mathcal{I}_{tgt}}$. For each gray level i of $\mathcal{I}_{src}$, we find the gray level j of $\mathcal{I}_{tgt}$ whose $\mathcal{C}^{\mathcal{I}_{tgt}}(j)$ is the closest in $\mathcal{C}^{\mathcal{I}_{tgt}}$ to $\mathcal{C}^{\mathcal{I}_{src}}(i)$. Once the mapping is found, each pixel with gray level i in $\mathcal{I}_{src}$ has its value replaced by j.

We treat each color channel of these images independently, matching their histograms individually. Figure 4.2 ilustrate the histogram matching process.

4.1.2 Gradient Comparison

Image gradients describe the value and direction of pixel intensity variation. They can be used to extract different information about the image, such as texture and location of edges. Here we filter an image by using a convolution with a *Sobel* [63] kernel for gradient

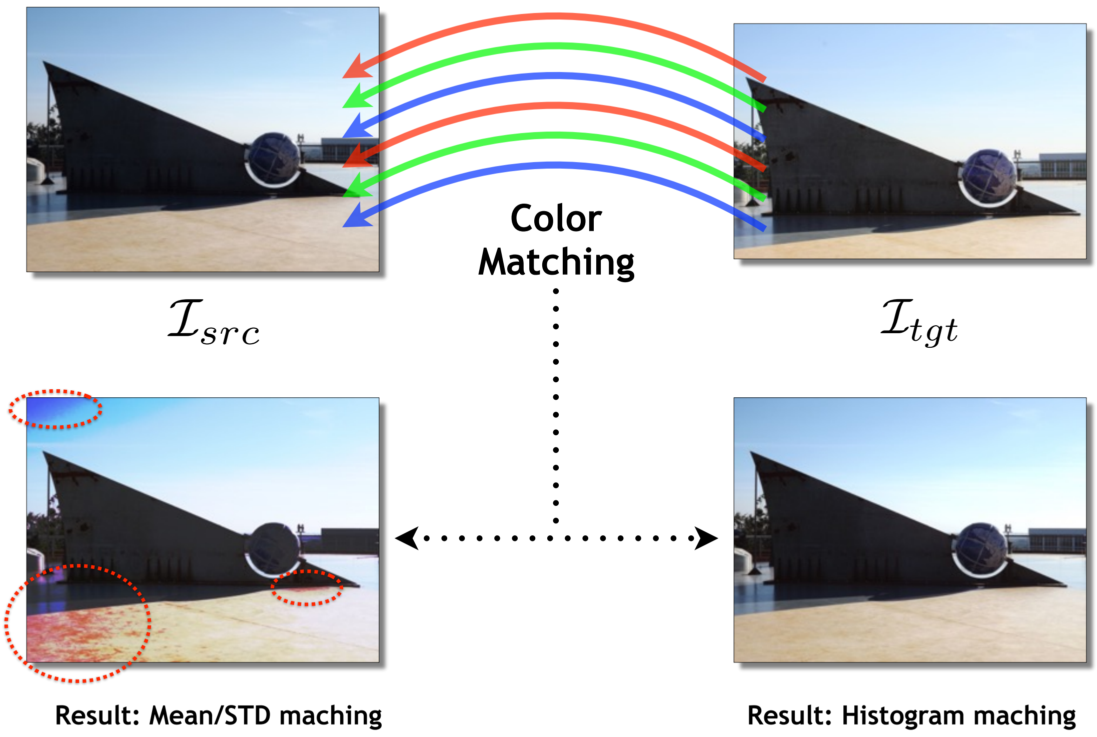


Figure 4.1: Matching the colors of the source image according to the color of the target image. The result of the color matching algorithm based on mean and standard deviation normalization [58] presents undesired artifacts that cannot be simply neglected, as can be noted in the marked regions of the picture. This problem is diminished when we perform a better color matching through histogram analysis.

estimation [31]. The convolution of an image $\mathcal{I}(x, y)$ with an $m \times n$ kernel $K(x, y)$ is given by:

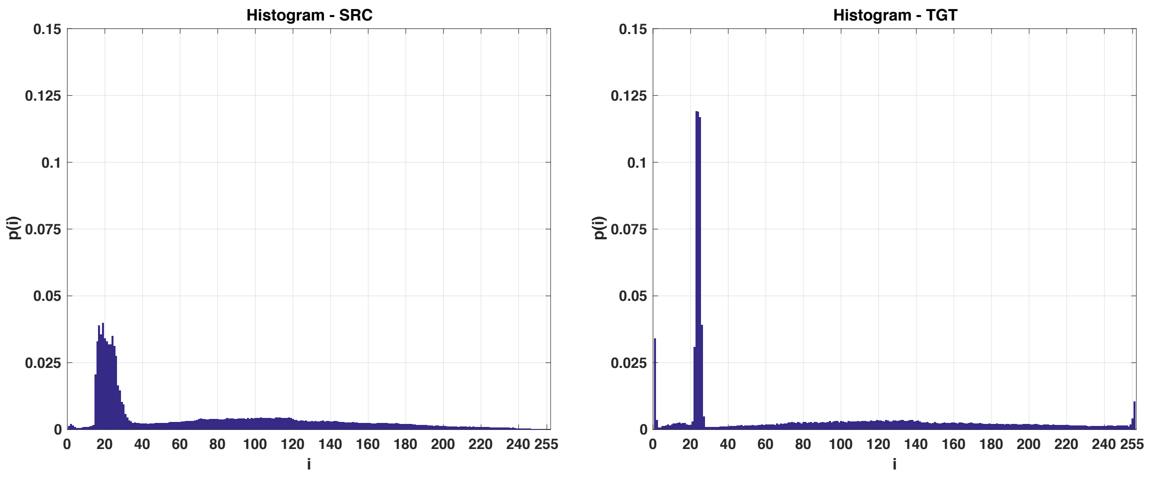
$$K(x,y) * \mathcal{I}(x,y) = \sum_{i=\lfloor -m/2 \rfloor}^{\lfloor m/2 \rfloor} \sum_{j=\lfloor -n/2 \rfloor}^{\lfloor n/2 \rfloor} K(i,j)\mathcal{I}(x-i, y-j) \tag{4.3}$$

where '$*$' denotes the convolution operator. This equation is evaluated for all values of displacement variables x and y [31].

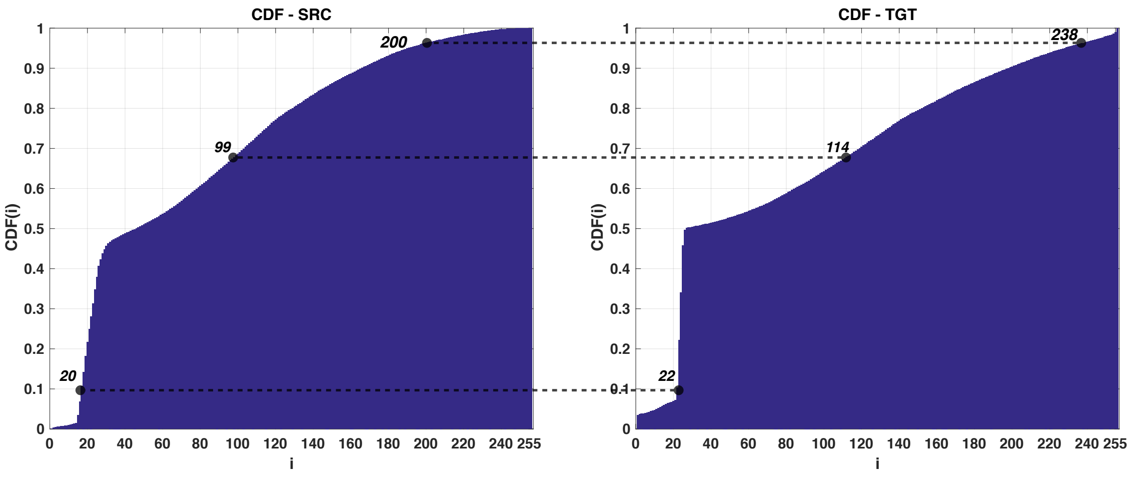
As contrast enhancement and color transformations are often used when creating near duplicates, directly affecting the gradients of the image, this becomes an important information to add to the dissimilarity calculation. By comparing the gradients of a transformed image $\mathcal{I}'_{src}$ and $\mathcal{I}_{tgt}$, it is possible to compare both the intensity values (encoded in the gradient), as well as their variation throughout the image.

While the image comparison metric $\mathcal{L}$ stays the same (i.e., Minimum Square Error), we first compute the gradients in the horizontal and vertical directions, by convolving the images to be compared with the 3×3 Sobel kernels S_h (horizontal direction) and S_v (vertical direction)[1]. The R, G and B channels of $\mathcal{I}'_{src}$ and $\mathcal{I}_{tgt}$ are treated separately

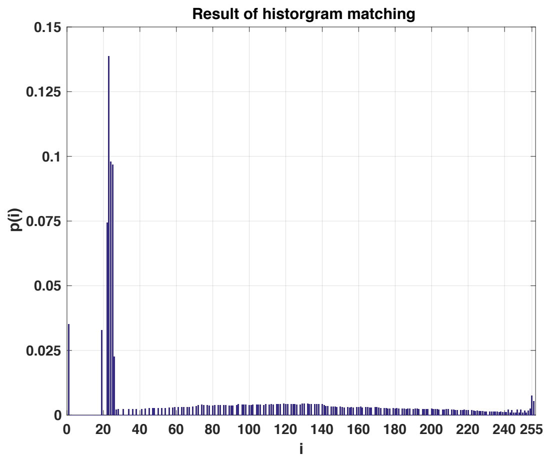
[1]In our experiments, we have used the 3×3 Sobel kernel. We performed some exploratory tests with other kernel sizes (e.g., 3×3, 5×5 and 7×7) but their performance was similar for the problem herein.



(a)



(b)



(c)

Figure 4.2: Histogram matching process. a) The histogram of the images $\mathcal{I}_{\text{src}}$ and $\mathcal{I}_{\text{tgt}}$; b) The Cumulative Distribution Function of the two histograms. The values of $\mathcal{C}^{\mathcal{I}_{src}}$ are mapped to $\mathcal{C}^{\mathcal{I}_{\text{tgt}}}$; c) The resultant histogram of $\mathcal{I}_{\text{src}}$ after mapping it to the domain of $\mathcal{I}_{\text{tgt}}$.

resulting in a total of six gradient images (two directions per color channel). The image comparison metric $\mathcal{L}$ is applied to each respective pair of gradient images of $\mathcal{I}'_{src}$ and $\mathcal{I}_{tgt}$, and the mean of the six values obtained in each position is taken as the final dissimilarity value.

4.1.3 Mutual Information Comparison

In Information Theory, mutual information (MI) is a measure of statistical dependency of two random variables, which represents the amount of information that one random variable contains about the other [62]. The mutual information between two random variables X and Y is given by:

$$MI(X,Y) = H(Y) - H(Y|X) = H(X) - H(X|Y), \tag{4.4}$$

where $H(X) = -E_x[\log(P(X))]$ is the entropy (i.e., the expected value of the information associated to a random variable) of X and $P(X)$ is the probability distribution of X. In the case of discrete random variables, MI is defined as:

$$MI(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right), \tag{4.5}$$

where $p(x,y)$ is the joint Probability Distribution Function (PDF) [43] of X and Y, and both $p(x)$ and $p(y)$ are the marginal PDFs of X and Y, defined, respectively, as:

$$p(x) = \sum_{y} p(x,y), \tag{4.6}$$

$$p(y) = \sum_{x} p(x,y). \tag{4.7}$$

MI has been widely employed in several image applications such as gender identification [65], multi-modal data fusion [7], feature selection [2], and in image registration problems [44,68] as a similarity measure (or cost function) to maximize when aligning two images (or volumes).

Applying MI to images means that the two random variables are the image $X = \mathcal{I}'_{src}$ and the image $Y = \mathcal{I}_{tgt}$ and x and y are the values of two pixels belonging to $\mathcal{I}'_{src}$ and $\mathcal{I}_{tgt}$, respectively. Thus, $p(x,y)$ is the joint PDF of the images $\mathcal{I}'_{src}$ and $\mathcal{I}_{tgt}$, evaluated for the values (x,y), where $x,y \in [0\ldots255]$.

Clearly, the previous definitions involve the knowledge of the PDFs of pixels and, in particular, the joint PDF $p(x,y)$, from which it is easy to obtain $p(x)$ and $p(y)$ by marginalization (Equations 4.6 and 4.7). In general, such joint PDF is not known *a priori*, and needs to be estimated. Several methods [8] have been conceived to estimate the PDF of one or more random variables from a finite set of observations, such as the approximation of the joint PDF by the joint histogram

$$\hat{p}(x,y) = \frac{h(x,y)}{\sum_{x,y} h(x,y)}, \tag{4.8}$$

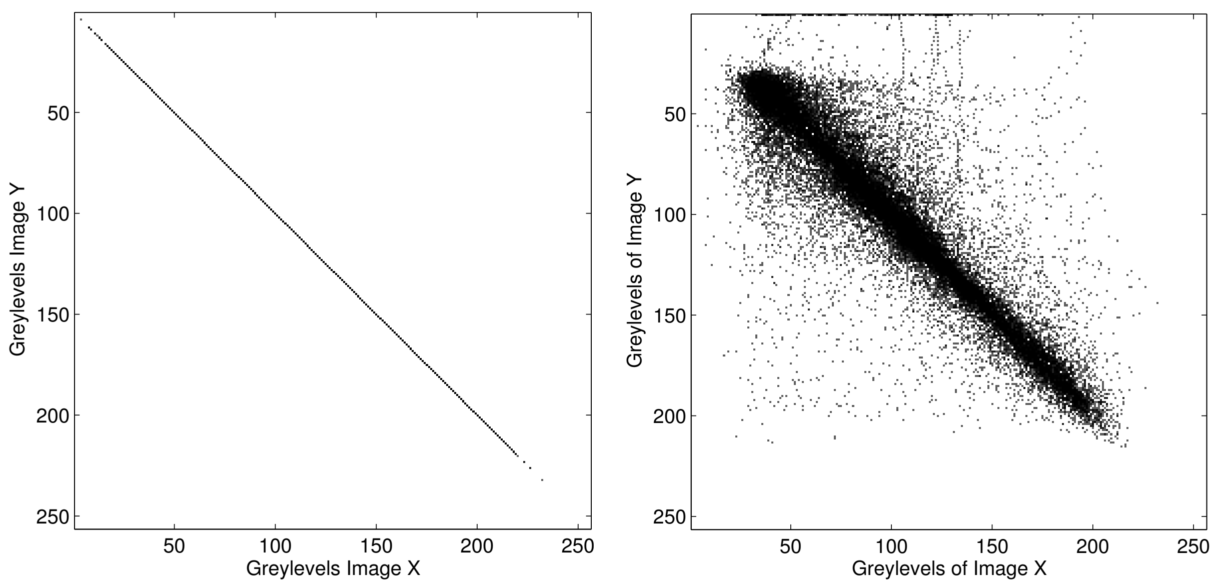


Figure 4.3: Bi-dimensional representation of two joint histograms. White pixels mean zero values while the other pixels represent values greater than zero (the images were inverted for viewing purposes). (a) Joint histogram of two (gray-scale) images perfectly aligned. (b) Joint histogram of two slightly misaligned images. Note that perfect alignment is nearly impossible unless the images are equal.

where $h(x,y)$ is the joint histogram of the images X and Y, namely the number of occurrences for each couple of gray level values (x,y), evaluated on the same (i,j) position on both images. MI has the following property: given two images $\mathcal{I}'_{src}$ and $\mathcal{I}_{tgt}$, $MI(\mathcal{I}'_{src}, \mathcal{I}_{tgt})$ is bounded as

$$0 \leq MI(\mathcal{I}'_{src}, \mathcal{I}_{tgt}) \leq \min(H(\mathcal{I}'_{src}), H(\mathcal{I}_{tgt})). \tag{4.9}$$

It can be demonstrated that MI is maximum when the two images are completely aligned (in terms of geometrical, color and compression transformation). Figure 4.3(a) shows a perfectly aligned case. If we assume a perfect transformation $T_{\vec{\beta}}$ that maps an image $\mathcal{I}_{src}$ onto an image $\mathcal{I}_{tgt}$'s domain, the mutual information $MI(T_{\vec{\beta}}(\mathcal{I}_{src}), \mathcal{I}_{tgt})$ is maximum. However, since each transformation is not completely reversible, if we apply the inverse transformation $T_{\vec{\beta}}^{-1}$ to $\mathcal{I}_{tgt}$ to obtain $\mathcal{I}_{src}$, their joint histogram is similar to Figure 4.3(b).

4.1.4 Gradient Estimation and Mutual Information Combined

The Gradient and Mutual Information comparison, presented in Sections 4.1.2 and 4.1.3, respectively, can be further combined into a single form of computing the dissimilarity value between two images. First, we calculate the gradient of the images $\mathcal{I}'_{src}$ and $\mathcal{I}_{tgt}$ as we described in Section 4.1.2. Afterwards, we compare each correspondent gradient of both images with mutual information, instead of using the image comparison metric $\mathcal{L}$ based on the standard Minimum Square Error. The final dissimilarity is the average of mutual information values for each gradient image.

With this approach, we aim at better capturing the information about variation in certain directions of the image (gradient information), as well as at seeking to avoid effects caused by slight misalignments during the mapping (mutual information estimation). This method also takes into consideration the amount of texture information preserved between two near duplicates for calculating the dissimilarity.

Unfortunately, the combined method slightly increases the computational cost of the dissimilarity calculation, given that we need to estimate the mutual information six times after the gradient calculation. However, this method yields better reconstruction results as we shall discuss in Section 4.3. Finally, these two methods can also be combined with a better color matching approach (c.f., Section 4.1.1) further improving the dissimilarity calculation between pairs of images.

4.2 Experimental Setup

In this section, we discuss the evaluation setup and the used datasets and metrics for validating the methods discussed in this chapter.

Dataset: For validation, we employed the freely available image dataset introduced by Costa et al. [15] and discussed on Section 3.3.2. The Training Dataset (for defining the parameter $\gamma_{\text{E-AOB}}$) and the Dataset B are considered herein. As previous described, Dataset B comprises cases for *One Camera* and *Multiple Cameras* scenarios, considering $||\mathcal{F}|| = 1..10$ and generating 2000 test cases for each forest size, finishing with $2 \times 2,000 \times 10 = 40,000$ test cases. As we evaluate each dissimilarity measure and each color matching approach, in this dataset, the final number of test cases is 320,000.

Phylogeny reconstruction: As the actual phylogeny reconstruction is not a focus herein, after estimating the dissimilarity matrix, we apply an algorithm for reconstructing the phylogeny forest. For that, we use the Extended Automatic Optimum Branching (E-AOB) algorithm described in Section 3.1.2. The parameter $\gamma_{\text{E-AOB}}$ as defined in Section 3.3.3, was found considering the training dataset and each one of the proposed dissimilarity measures. From these experiments, we also defined $\tau_{\text{AOB}} = \mu_{\text{AOB}} + (2.0 \times \sigma_{\text{AOB}})$, and as a consequence, $\gamma_{\text{E-AOB}} = 2.0$.

Evaluation metrics: For a better assessment of the proposed methods, we consider scenarios in which the *ground truth* is available. We used the evaluation metrics introduced by Dias et al. [22] to evaluate the proposed approach: *Roots, Edges, Leaves* and *Ancestry*, described in Section 3.3.1.

Real cases: We also performed experiments and qualitative analysis considering two real datasets available in the literature.

- *The Situation Room* [22]: It comprises an image taken on May 1st, 2011, by the White House photographer Peter Souza and its variants, collected from the Internet. We performed the dissimilarity matrix calculation and the phylogeny reconstruction considering 98 near-duplicate images collected through Google Images[2] and manually classified them in different groups considering (a) cases of inserting the Italian soccer player Mario Balotelli, (b) text overlay, (c) watermarking, (d) face swapping, (e) insertion of a joystick, (g) hats, and (n) changes in the image size without splicing operations.

- *The Ellen DeGeneres' selfie* [52]: this dataset comprises near-duplicate images related to the *selfie* taken by the TV host Ellen DeGeneres and some famous actors on March 2nd, 2014, during the 86th Academy Awards. The original image became viral after it was published on her Twitter account. Since then, it has been copied, modified and republished several times, with cases of text overlay, insertion of other people and animals in the picture and face swap. The dataset has 44 pictures from the internet and it is divided in five groups:

  (a) Edited versions of the original image posted at DeGeneres' Twitter account (@TheEllenShow[3]);

  (b) The moment that the picture has been taken but from a different point of view (another camera);

  (c) Group similar to group (b), but with small differences on the posture of the people in the picture;

  (d) Similar to groups (b) and (c), but with small differences on the facial expression and posture of the people;

  (e) The moment before the acquisition of the *selfie* when the artists were still gathering for taking the picture.

4.3 Results and discussion

In this section, we show the performed experiments to compare the proposed methods with the state-of-the-art MSE method, which has been the "de facto" dissimilarity calculation method thus far for image phylogeny [15,19,22–25]. We analyze the impacts of calculating the dissimilarities using image gradients instead of image intensities, the replacement of the standard point-wise comparison metric minimum squared error with a mutual information dissimilarity calculation, and the incorporation of color matching for better representing the mapping of a source image onto a target image before actually calculating the dissimilarity.

4.3.1 Quantitative Experiments

Figures 4.4 and 4.5 depicts the results for the different approaches considered herein for calculating the dissimilarities for OC and MC scenarios, respectively. In all cases, the geometrical mapping of one source image onto a target image is performed following the

[2]`https://www.google.com.br/imghp`

[3]https://twitter.com/TheEllenShow/status/440322224407314432/photo/1

procedure discussed in the beginning of Section 2.2.1. The phylogeny reconstruction part uses the E-AOB algorithm for all methods, regarded as the state of the art in the literature for the reconstruction part [15, 19].

The baseline dissimilarity calculation considered is the MSE, the state of the art, which compares two images point-wise using the pixel intensities. The proposed modifications are:

1. Gradient estimation (GRAD), which still compares the images point-wise but using image gradients instead of pixel intensities (Section 4.1.2);

2. Mutual information (MINF), which replaces the point-wise comparison using pixel intensities with the mutual information calculation of pixel intensities (Section 4.1.3);;

3. Gradient estimation plus comparison with mutual information (GRMI), incorporating the calculus of dissimilarities using mutual information of image gradients (Section 4.1.4); and, finally,

4. Histogram color matching plus gradient estimation with mutual information (HGMI), extending upon GRMI to incorporate a better color matching before comparison.

First of all, the dissimilarity calculation does not benefit directly from the replacement of point-wise pixel intensity comparison by a point-wise comparison of image gradients, as the results show MSE outperforming GRAD for OC and MC scenarios. The gradient itself only captures directional variations and small misalignments when comparing two gradient images affect the results more than when comparing the images through pixel intensities.

If we change the point-wise comparison method to mutual information but still use the pixel intensities, we have MINF outperforming MSE for the MC case. With MINF, small misalignments are not as important as for the GRAD case. One interesting behavior, however, is the improved performance for the OC case (Root and Ancestry metrics). In the OC case, as all of the images come from the same camera, the color matching for such images should be more refined than just the mapping using the mean and standard deviation to differentiate an image and its descendant. A point-wise comparison, in this case, is more effective for small differences (MSE method).

The results improve when combining the gradient calculation with mutual information (GRMI). The first reason is that, by not comparing the pixel intensities directly, the color information artifacts are not as strong. Second, the comparison is not done in a point-wise fashion but rather, in a probability distribution-like form, better capturing the different variations of the gradient images as well as accounting for possible small misalignments. Finally, the combination of the histogram color matching technique with gradient estimation and mutual information yields the final method HGMI, which solves the former color matching problem when using MINF. As we can see, HGMI outperforms the MSE baseline for all cases. With HGMI, we can reduce the dissimilarity errors by better matching the color transformations involved in the process of near-duplicate

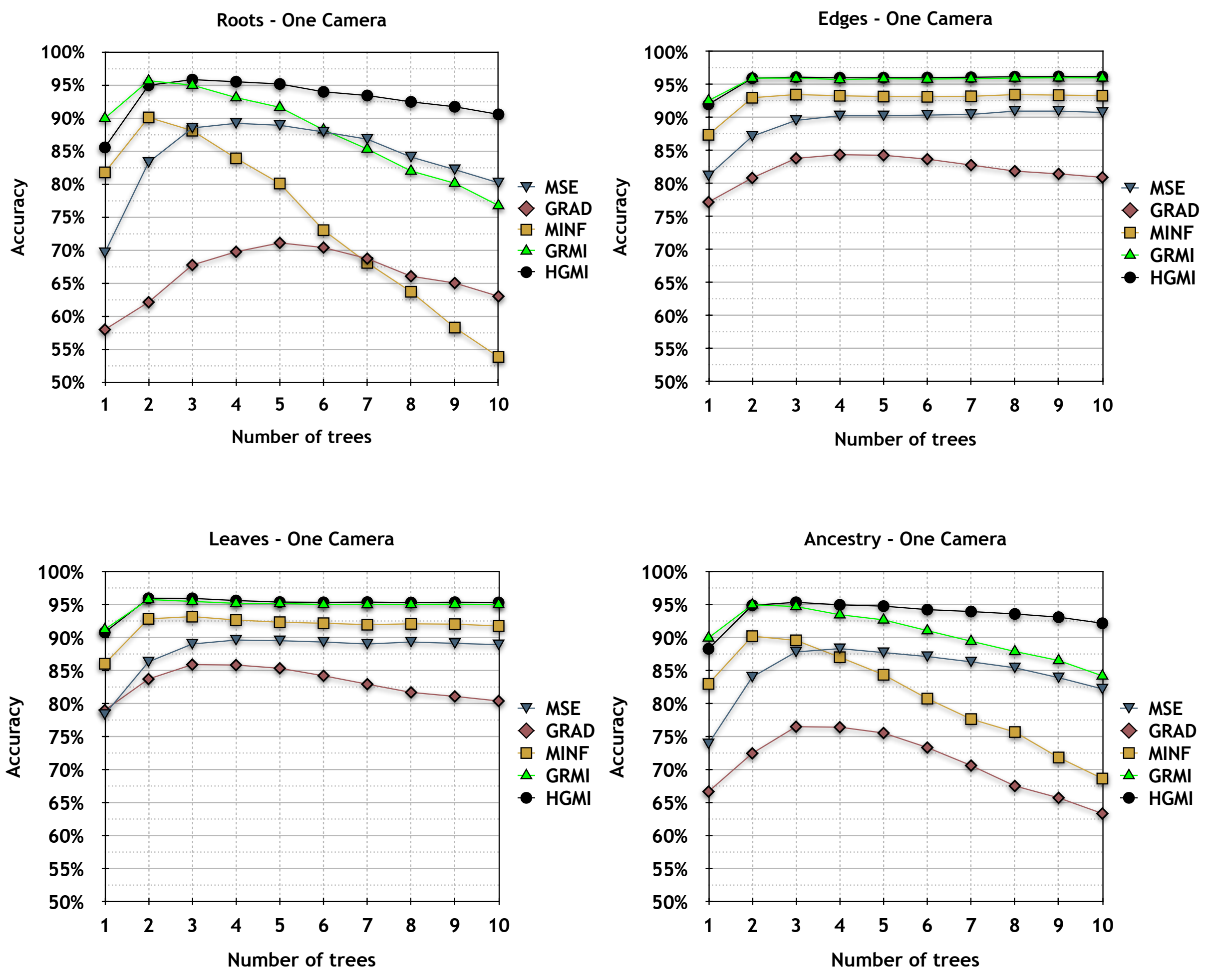


Figure 4.4: Results of forest reconstruction in the one camera (OC) scenario, considering the metrics Roots, Edges, Leaves and Ancestry.

generation, by comparing the images using gradients instead of pixel intensities and in a distribution-like form instead of a point-wise one.

The other possible combinations of the methods discussed herein are presented on Appendix B. Note that none of them is more effective than the ones presented and discussed in Figures 4.4 and 4.5.

4.3.2 Efficiency

In addition to comparing different methods in terms of effectiveness, it is also worth assessing the efficiency of the different methods described in Section 4.3.1. For that, all the experiments reported in this chapter were performed in a machine with an Intel Xeon E5645 processor, 2.40GHz, 16GB of memory, and running Ubuntu 12.04.5 LTS. The source code was implemented using the OpenCV library, version 2.4.10.

To compare a pair of typical images (each with about one megapixel), including the time to register both images and performing the comparison in both directions ($\mathcal{I}'_i \to \mathcal{I}_j$ and $\mathcal{I}'_j \to \mathcal{I}_i$), MSE takes about 0.6s, GRAD takes 0.8s, and MINF takes 0.7s. The best performing methods GRMI and HGMI take both about 1.7s. However, all methods can

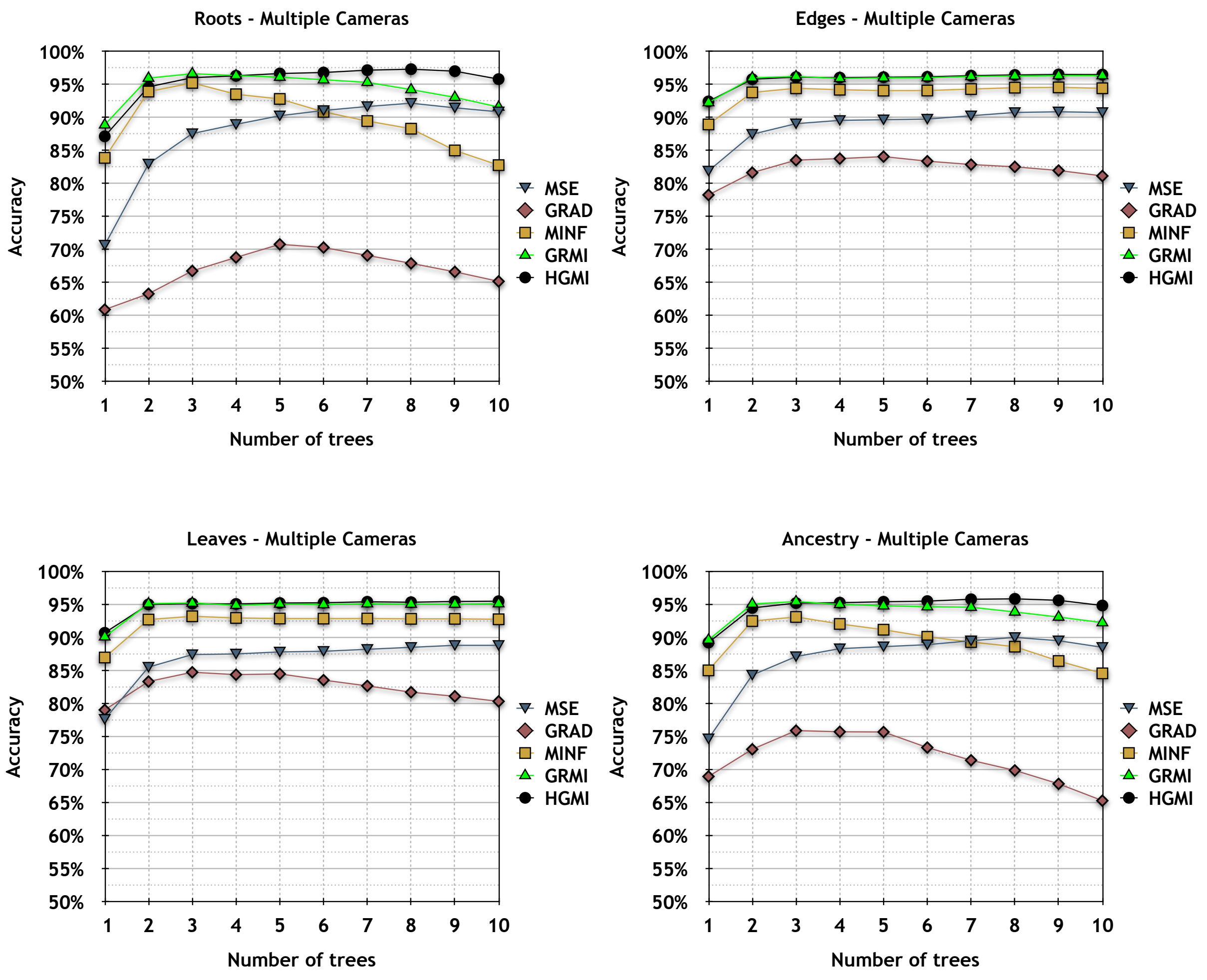


Figure 4.5: Results of forest reconstruction in the multiple cameras (MC) scenario, considering the metrics Roots, Edges, Leaves and Ancestry.

be optimized to compensate their additional computational requirement using GPUs and parallel computing.

Registration efficiency

Although the efficiency of the dissimilarity calculation is not the primary focus of this work, we can also optimize the dissimilarity calculation process by selecting, for instance, a faster keypoint detector and descriptor for the registration step. Taking this into account, we performed a performance test comparing two descriptor extractors: SURF (that was used in this work and has been the standard in image phylogeny solutions thus far) and ORB (*Oriented Fast and Rotated BRIEF*), a binary descriptor extractor based on the Harris corner detector [60].

For the performance test, we considered 50 toy examples, comprising trees with 10 nodes each. We evaluate, for these examples, the time (in seconds) of each step of the dissimilarity calculation process. Table 4.1 shows the time spent by each step of the dissimilarity calculation, comparing the descriptor extraction using SURF and the descriptor extraction made using ORB. For this test, we considered the HGMI dissimilarity calculation, which was the best approach presented in Section 4.3.1.

Table 4.1 shows that the ORB descriptor extractor is more efficient than SURF for finding the keypoints and for describing them. However, its efficiency does not influence the other steps.

Table 4.1: Time analysis (in seconds) of each step of HGMI dissimilarity calculation, considering SURF and ORB for the descriptor matching in the registration step.

	SURF	ORB
Keypoints and descriptors extraction (for each image)	0.831	0.030
Descriptors cross-check matching (for each pair of images)	0.077	0.101
Image registration ($\mathcal{I}_{src} \to \mathcal{I}_{tgt}$)	0.138	0.166
Color and compression matching ($\mathcal{I}_{src} \to \mathcal{I}_{tgt}$)	0.100	0.102
Dissimilarity calculation ($\mathcal{I}_{src} \to \mathcal{I}_{tgt}$)	0.911	0.965
Total execution time (full 10×10 matrix)	112.790	105.410

To analyze the effectiveness of the phylogeny reconstruction, we used 1,000 samples of case tests (500 for the OC scenario and 500 for the MC scenario), considering the HGMI method for dissimilarity calculation and forests with 10 trees. Figure 4.6 shows the difference in the quality of reconstruction, for all the evaluation metrics, considering different γ_{E-AOB} parameters for the phylogeny forest reconstruction.

Figure 4.6 shows that the registration step using SURF as the descriptor extractor is better then using ORB. While SURF is invariant to rotation, scale and color changing, ORB is only invariant to rotation and Gaussian noise. Considering the family of transformations presented in the datasets, it is natural to expect SURF to outperform ORB in the registration step and, consequently, in the phylogeny forest reconstruction.

4.3.3 Error Reduction

To directly compare the approaches, we also calculate the error variation, $\Delta error$, with respect to each metric (roots, edges, leaves and ancestry), using the Equation 3.1. Figure 4.7 depicts the average error reduction for HGMI when compared to the baseline MSE. In this case, there is an error reduction of about 45% in the OC scenario and more than 50% in the MC scenario for all evaluation metrics, clearly showing that the proposed HGMI dissimilarity measure is remarkably superior to the standard MSE procedure.

A Wilcoxon signed-rank test [69] shows that the best proposed approach, HGMI, is statistically better than the state-of-the-art MSE method for all cases and metrics, with 95% of confidence and a p-value of 0.002.

4.3.4 Effects of Dissimilarity Errors on the Reconstruction

The dissimilarity errors directly affect the selection of the edges by the E-AOB reconstruction algorithm, as this process is done by comparing the difference of edge weights and the standard deviation of edges already selected, considering that the forest needs to have 90 edges[4]. However, this event does not happen (on average) for GRAD-

[4]For cases with $n = 100$ images, the initial branching has $n - 1 = 99$ edges. For creating a forest $\mathcal{F}$ where $|\mathcal{F}| = 10$ trees, the number of total edges is $n - |\mathcal{F}| = 100 - 10 = 90$.

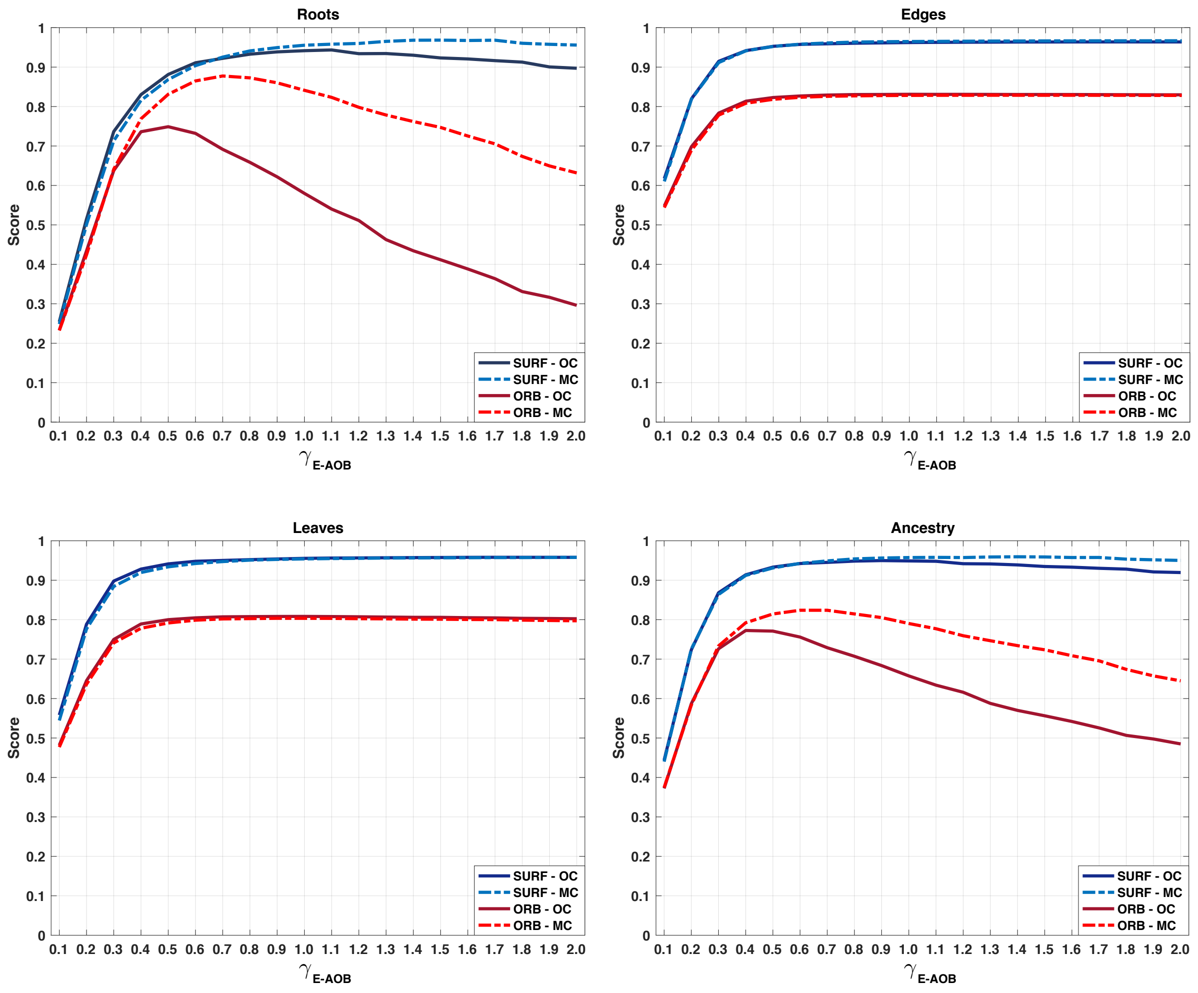


Figure 4.6: Phylogeny reconstruction test, considering ORB and SURF for the reconstruction step.

MC, GRAD-OC and MINF-OC, showing that a wrong number of trees is calculated for these cases, as Figure 4.8 shows. Note that, for GRMI and HGMI cases, in most of the cases, a correct number of trees is selected. Specifically for the HGMI case, the correct size of the forests outperform the baseline (MSE) in approximately 10 percentage points in the MC scenario and 20 percentage points in the OC scenario.

4.3.5 Qualitative Experiments with Real Cases

We now turn our attention to assessing the behavior of the best performing method (HGMI) considering two real cases from the internet: *The Situation Room* [22] and *The Ellen DeGeneres' selfie* [52]

For real cases, the feedback of a forensic expert for evaluating the quality of an algorithm is essential as there is no ground-truth. In this case, we empirically define the γ parameter of the E-AOB algorithm for each case ($\gamma_{E-AOB} = 2.0$ for the case in *The Situation Room* and $\gamma_{E-AOB} = 0.5$ for the case of *The Ellen DeGeneres' selfie*). Figures 4.9 and 4.10 show the reconstructed forests for these cases.

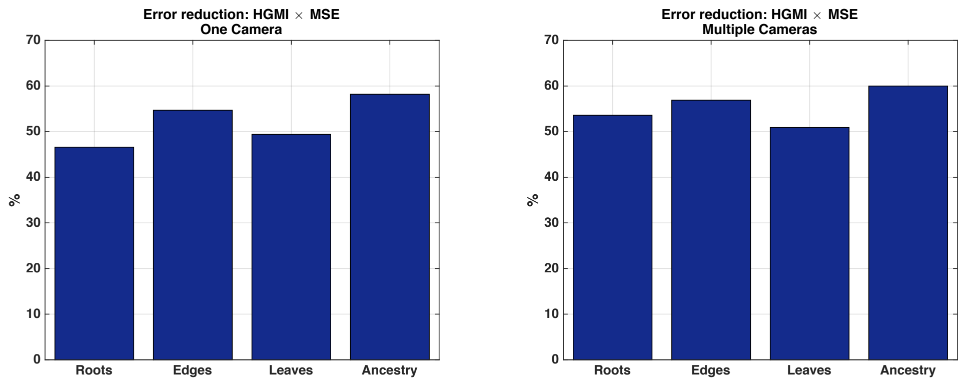


Figure 4.7: Error reduction: HGMI × MSE.

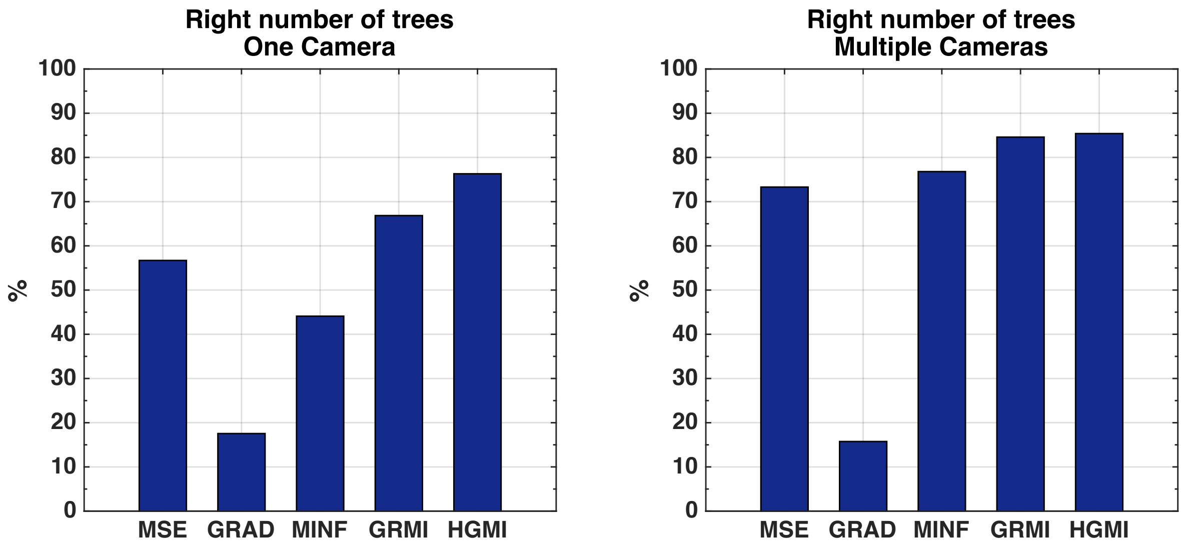


Figure 4.8: Average result (%) of correct number of trees calculated by the E-AOB algorithm, for 2,000 test cases, considering forests with 10 trees.

For *The Situation Room* scenario, the algorithm correctly identified the image with ID *0000* (the White House version) as the root of the tree. Furthermore, as we expected, the result was that all images were grouped under the same tree (with image *0000* as the root). Although there are some images in wrong groups (sub-trees) in the reconstructed phylogeny, it is important to note that this dataset is mostly composed by images generated by splicing operations, which is, in fact, a special case of IPFs (multiple parenting phylogeny [53]). However, the E-AOB could separate these groups in different sub-trees with good effectiveness.

Considering the *Ellen DeGeneres' selfie* scenario, we have a forest with five trees. The near duplicates are correctly organized according to their groups. The node *a00* is the picture originally posted at the DeGeneres' Twitter account, and it was not selected here as the root of the group. However, the node is only two-edges of distance to the root. The tree with images *a09, a10, a11* and *a12* should also be placed as a child of node *a00*, but it has a splicing of a cat in the picture, and the algorithm ended up classifying *a09* and *a10* as ancestors of *a00* and the nodes *a11* and *a12* as nodes not related to *a00*.

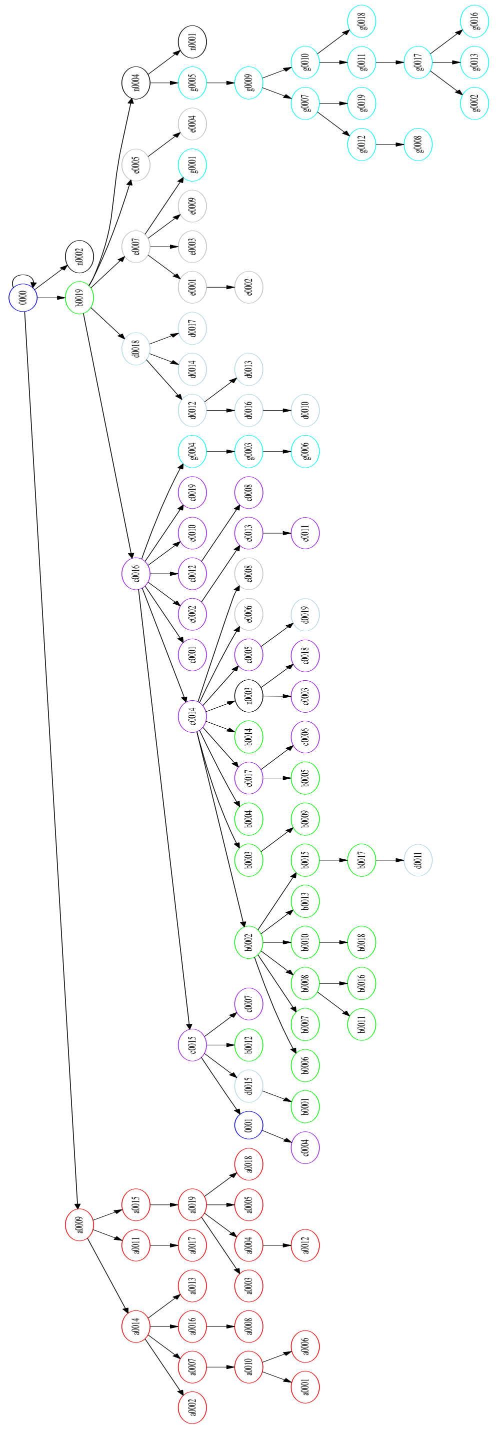
The nodes *a09*, *a10*, *a11* and *a12* are correctly grouped, since image *a09* is actually a montage also extracted from a Twitter's official account (@RealGrumpyCat[5]). The images *a10*, *a11*, and *a12* are all variants of this image. The image *a03* also should be classified as a child of *a00*, but it was separated in a single tree. However, this image was generated by splicing, in which all the faces in the picture were replaced by DeGeneres' face. Groups *b*, *c* and *d* are the hardest to analyze, since there is a subtle difference among them. As we can see, group *d* was correctly separated in a different tree. Although the groups *b* and *c* are placed on the same tree, it is possible to note that most of the images that belong to the same group are together (with the exception of image *c01*, which is in a single tree). As mentioned before, the groups *b* and *c* are very close, semantically speaking, which can explain why they are grouped in the same tree. This structure certainly would help the work of a forensics expert. The group *e* was also correctly classified in a different tree.

4.4 Publication

The main findings of the research presented in this chapter resulted in one article submitted to an international journal:

- **F. O. Costa, A. Oliveira, P. Ferrara, Z. Dias, S. Goldenstein and A. Rocha**; *New Dissimilarity Measures for Image Phylogeny Reconstruction*, Springer Pattern Analysis and Applications (PAA) – Under review [17].

[5]https://twitter.com/RealGrumpyCat/status/440335332265848835/photo/1



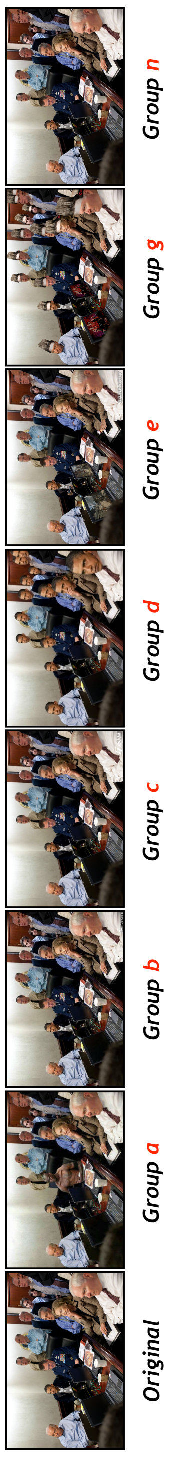


Figure 4.9: Reconstructed phylogeny for *The Situation Room* scenario.

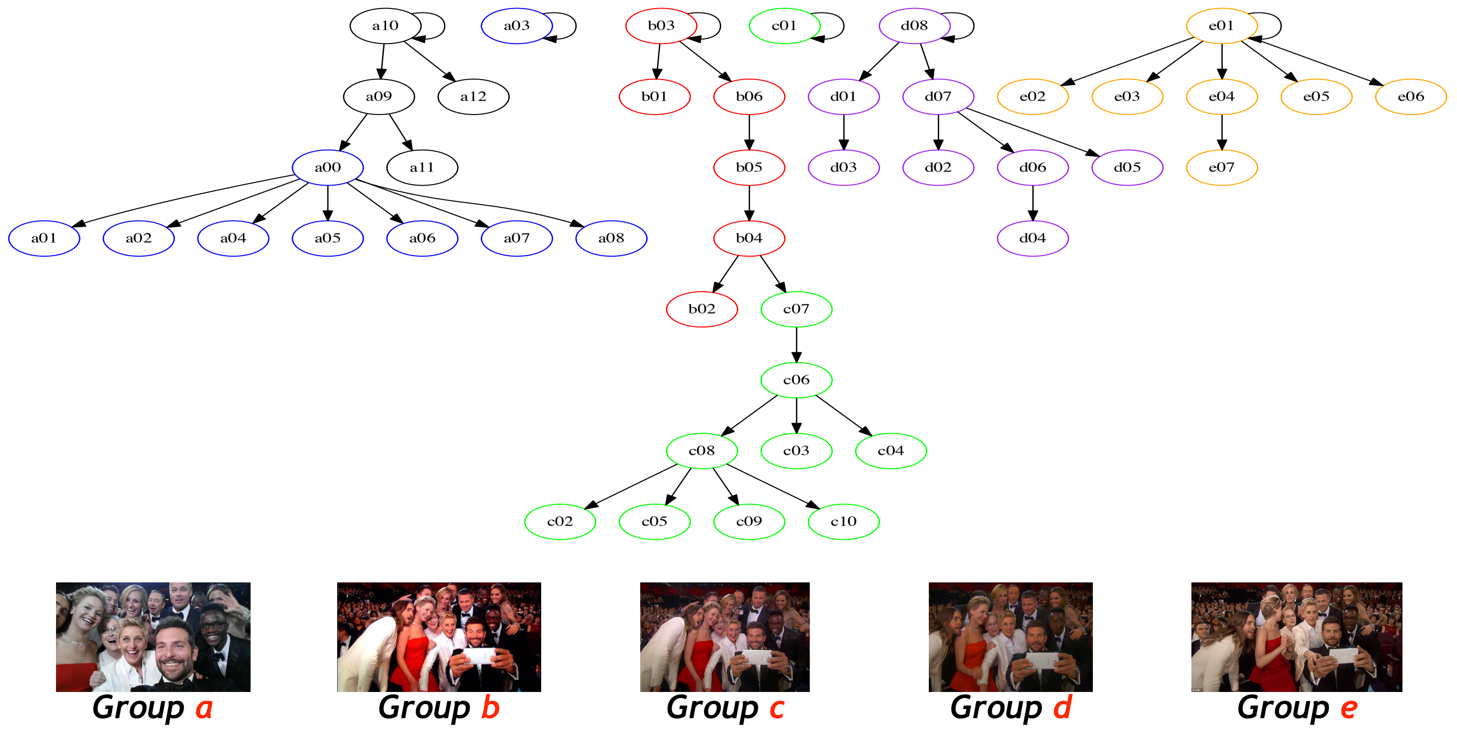


Figure 4.10: Reconstructed phylogeny for *The Ellen DeGeneres' selfie* scenario.

Part II

Video Phylogeny

Chapter 5

Phylogeny Reconstruction for Misaligned and Compressed Video Sequences

In this chapter, we deal with the problem of reconstructing the *Video Phylogeny Tree* (VPT) associated with a set of videos. Starting from a pool of near-duplicate videos, we seek to reconstruct the relationship between every video pair. This problem, deeply studied for images [23, 47, 59], is typically solved in two steps: (i) the calculation of the dissimilarity between each pair of near-duplicate objects; and (ii) the reconstruction of the phylogeny tree based on the aforementioned dissimilarity measures [23, 25].

Dias et al. [24] proposed an initial approach to deal with the video phylogeny tree reconstruction problem. However, only temporally coherent videos (i.e., temporally aligned videos with the same number of frames) were considered. Furthermore, the authors considered only videos compressed with the same standard and parameters without explicitly taking into account any other compression scheme in their reconstruction pipeline. These are somewhat limiting assumptions, since video duplicates are typically encoded using different coding schemes and parameters and often are temporally misaligned.

In order to deal with this more challenging setups, we propose a modification to the pipeline used in [24] for solving the problem of VPT reconstruction. Our algorithm considers temporal clipping and coding as possible transformations for the generation of near duplicates, in addition to geometric transformations and brightness/contrast correction. More specifically, we (i) make use of a video alignment procedure; and (ii) in the video registration procedure, we explicitly consider that the coding is applied to each video under analysis; the coding algorithm and parameters are, in general, different for each video.

5.1 Problem formulation

As described in Section 2.2.1, the dissimilarity between two near duplicates is computed for each pair of documents, and these values are stored as entries of a matrix. This dissimilarity matrix can be interpreted as an oriented graph, where each node is an image, for instance, and the direction of links between two images indicates which one generated the other. The phylogeny tree can then be reconstructed using a phylogeny reconstruction algorithm [19, 23]. This reconstructed tree represents the relationship between all near duplicates, where the root of the tree is the original image (the patient zero), each edge represents a set of transformations T between two images and leaves are the most modified duplicates in a given path from the root.

To the best of our knowledge, the only work proposing to reconstruct the video phylogeny tree (rather than the image phylogeny one) is [24]. Basically, the authors follow the same pipeline used for images. In this case, the dissimilarity between two videos $\mathcal{V}_{\text{src}}$ and $\mathcal{V}_{\text{tgt}}$ is calculated for each pair of corresponding frames, assuming that the videos are temporally coherent. Thus, the authors obtain F dissimilarity matrices, where $F = ||\mathit{Frames}(\mathcal{V}_{\text{src}})|| = ||\mathit{Frames}(\mathcal{V}_{\text{tgt}})||$, $\mathit{Frames}(\mathcal{V})$ is the set of frames of a video $\mathcal{V}$ and $|| \cdot ||$ computes the cardinality. The phylogeny tree is then reconstructed using different strategies, i.e., merging all the dissimilarity matrices into a single matrix in order to estimate a single tree, or reconstructing a tree from each dissimilarity matrix and then merging the trees.

A drawback of the method proposed in [24] is that the authors do not consider the case of temporally misaligned sequences nor different coding parameters. In a more realistic scenario, these operations are commonly applied to videos. Indeed, videos are routinely distributed in compressed format in order to reduce storage space. Moreover, videos edited by different users and downloaded from web sharing platforms are hardly aligned, as some frames may have been removed, and no absolute temporal reference is given.

Figure 5.1 depicts an example of the challenges introduced in a real-world scenario. Given a video sequence $\mathcal{V}_A$ and some duplicates, it is possible to note that there are no correspondent frames between the duplicates $\mathcal{V}_B$ and $\mathcal{V}_C$. The duplicates $\mathcal{V}_D$, $\mathcal{V}_E$, and $\mathcal{V}_F$ have correspondent frames with all duplicates and only the duplicate $\mathcal{V}_F$ has all frames corresponding exactly with the original video $\mathcal{V}_A$. Moreover, each duplicate may have undergone a different coding step. Therefore, the choice of frames to be compared for dissimilarity computation is not straightforward, as nothing guarantees the existence of a subset of correspondent frames contemplating all video sequences. Additionally, each codec leaves peculiar footprints [4] that, if not taken into account, may lead to an incorrect dissimilarity computation.

5.2 Video Phylogeny Tree Reconstruction

To cope with the aforementioned problems, we propose an algorithm devised in four steps:

1. Temporal alignment of video pairs;
2. Frame registration (geometric transformation, color correction and compression);
3. Dissimilarity computation;
4. Phylogeny tree reconstruction.

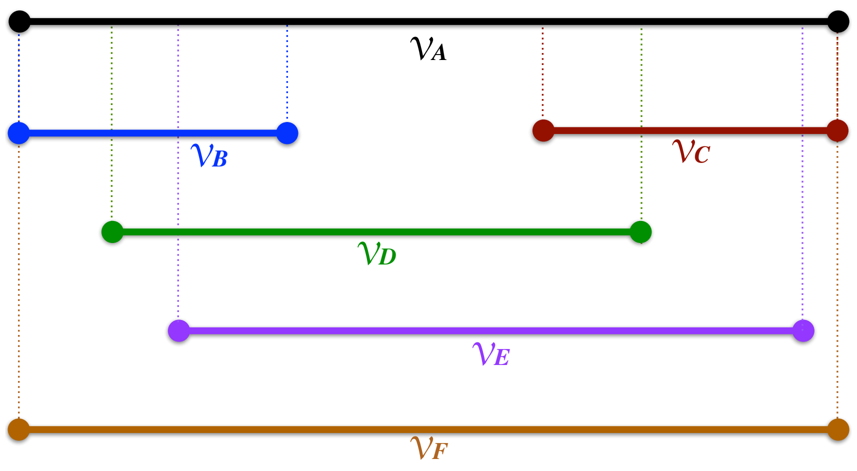


Figure 5.1: Example of video sequences with different sizes. The dotted lines represent the region of the video sequences that are correspondents to $\mathcal{V}_A$.

In the following, we present a detailed description of each step. Fig. 5.3 depicts a block diagram of this scheme.

5.2.1 Video alignment and frame selection

Given a pair of video duplicates $\mathcal{V}_{\text{src}}$ and $\mathcal{V}_{\text{tgt}}$, the first step consists in aligning them in order to be able to compare the correct frame pairs. For this, we rely on the procedure detailed in [39]. We resort to a 1-dimensional description of a video over time, obtained through computing the difference between the average of luminance values of adjacent frames of a video $\mathcal{V}$ as

$$LD_{\mathcal{V}}(i) = avgluma(\mathcal{V}(i)) - avgluma(\mathcal{V}(i-1)), \tag{5.1}$$

where $avgluma(\mathcal{V}(i))$ extracts the average of the luminance component of the i-th frame $\mathcal{V}(i), \forall i \in \{2, 3, ..., ||Frames(\mathcal{V})||\}$. The alignment between two sequences of frames is performed by looking at the position $\hat{i}$ of the highest peak of the phase-correlation between $LD_{\mathcal{V}_{\text{src}}}(i)$ and $LD_{\mathcal{V}_{\text{tgt}}}(i)$, given by

$$\hat{i} = \arg\max_i \mathcal{F}^{-1}\left[\frac{L_{\text{src}} \cdot L^*_{\text{tgt}}}{|L_{\text{src}} \cdot L^*_{\text{tgt}}|}\right](i), \tag{5.2}$$

where $L_{\text{src}} = \mathcal{F}[LD_{\mathcal{V}_{\text{src}}}]$ and $L_{\text{tgt}} = \mathcal{F}[LD_{\mathcal{V}_{\text{tgt}}}]$ are the Fourier transforms of $LD_{\mathcal{V}_{\text{src}}}$ and $LD_{\mathcal{V}_{\text{tgt}}}$, respectively, '$*$' indicates the complex conjugate operator and '$\cdot$' the element-wise product. The value of $\hat{i}$ indicates the delay in frames between the analyzed sequences. Given the mutual delay, the selection of common frames is straightforward.

Fig. 5.4 shows an example of $LD_{\mathcal{V}}(i)$ for two video sequences. The y axis represents the values of LD, while the x axis represents the number of frames. In the first graph (top), the sequences are misaligned. The second graph (middle) shows the result of the alignment. Note that some frames of the longest sequence (blue line) should be discarded for obtaining the best alignment between the video sequences. Fig. 5.4 (bottom) also shows the result of correlation (5.2), where the pronounced peak is marked with a black dot.

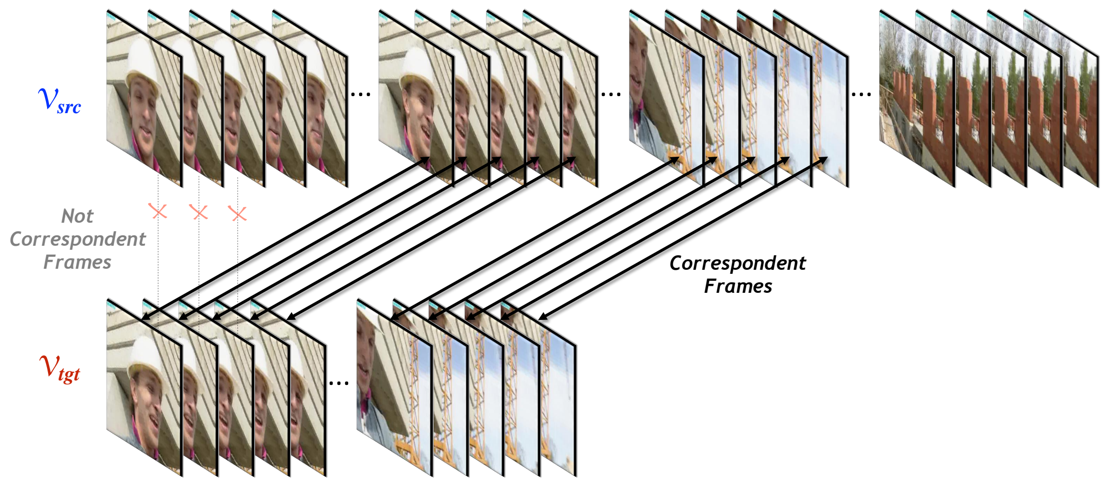


Figure 5.2: Example of two misaligned video sequences. In this example, to evaluate the correspondent frames of $\mathcal{V}_{src}$ and $\mathcal{V}_{tgt}$, a temporal alignment is necessary, removing the frames at the beginning and at the end of $\mathcal{V}_{src}$.

5.2.2 Frame registration and dissimilarity calculation

The goal of this step is to find the best transformation T that maps $\mathcal{V}_{\text{src}}$ onto $\mathcal{V}_{\text{tgt}}$. We estimate the transformation $T_{\vec{\beta}}$ for each pair of frames in three steps:

- **Geometric correction:** considering that the geometric transformation applied for creating one video is exactly the same for every frame (e.g., frames are all resized with the same factor), we estimate the geometric transformation mapping $\mathcal{V}_{\text{src}}(idx)$ to $\mathcal{V}_{\text{tgt}}(idx)$ and apply it to every frame of $\mathcal{V}_{\text{src}}$ as done for images (described in Section 2.2.1), where idx is the index of the first valid correspondent frame in both videos (e.g., not a black frame).

- **Color correction:** for color correction, we use the color transfer algorithm based on histogram matching [31] previously described in Section 4.1.1.

- **Coding matching:** we encode (re-encode) $\mathcal{V}_{\text{src}}$ by using the same coding scheme and parameters used by $\mathcal{V}_{\text{tgt}}$. This means that we recover the used codec, quantization parameter (QP), and group of pictures (GoP) size (i.e., the distance between consecutive intra-coded frames) from the bitstream of $\mathcal{V}_{\text{tgt}}$, and apply them to $\mathcal{V}_{\text{src}}$. Considering these parameters, we perform *intra-frame* compression (that defines the quality of compression of one frame) and *inter-frame* compression (that defines the quality of the video considering also the estimation of some frames based on the GoP size). The obtained sequence is denoted as $\mathcal{V}'_{\text{src}}$. The QP parameter sometimes is trivially extracted from the video header (e.g., for H.264 videos). When this information is not available, one option is to estimate the parameter [67]. In this work, we consider the situation in which it is possible to reliably recover such parameters without errors.

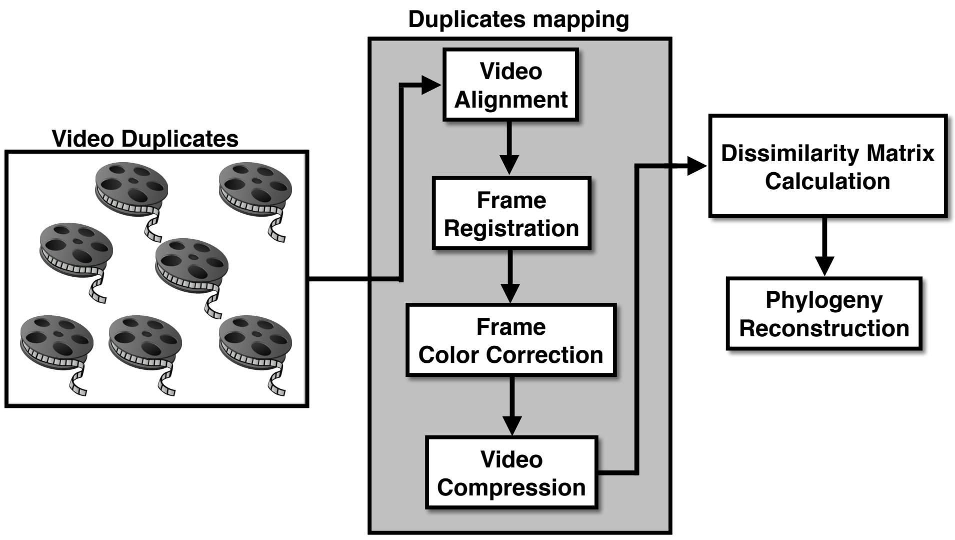


Figure 5.3: Pipeline of the proposed video phylogeny approach.

In [24], the best results reported by the authors were obtained calculating one dissimilarity matrix for each pair of frames common to all the analyzed sequences. That was possible due to the fact that all the videos had the same length and were temporally aligned.

Considering the case in which we have temporal clipped videos, we opt to calculate the frame-wise dissimilarity considering only the corresponding frames of the aligned sub-sequences of the videos $\mathcal{V}'_{\text{src}}$ and $\mathcal{V}_{\text{tgt}}$, then we calculate the average of these values. The dissimilarity between two frames is obtained using the mean squared error metric that was the same metric used by Dias et al. [24]. Although the dissimilarity measure based on gradient estimation and the comparison of them with mutual information showed better effectiveness for image phylogeny than MSE measure, this method is computationally more expensive (as mentioned in Chapter 4), than MSE-based methods. Dealing with videos, this cost increases substantially, since we have to perform it for each frame of the videos. Because of this, we opt to use MSE in these experiments, but the new approach can be further investigated for video phylogeny.

5.2.3 Tree reconstruction

After calculating the dissimilarity matrix, we use an algorithm for reconstructing the phylogeny tree. Considering that the dissimilarity matrix represents an oriented graph, we do not have a complete graph in this case, as some dissimilarity values were not computed due to the temporal clipping constraint. For the tree reconstruction task, we use the Optimum Branching (OB) algorithm [6,10,26]. As presented in Section 2.2.1, this algorithm was already proposed for phylogeny reconstruction [15,19] and outperforms the Oriented Kruskal proposed in [24].

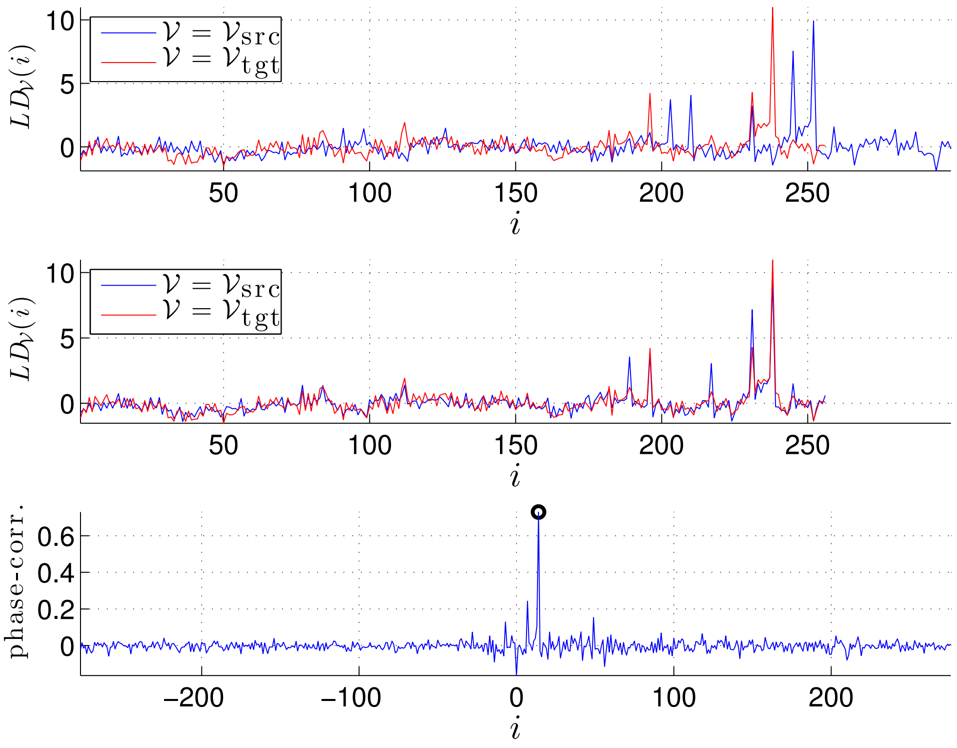


Figure 5.4: Example of misaligned $LD_{\mathcal{V}}(i)$ (top), aligned $LD_{\mathcal{V}}(i)$ (middle) and correlation computed as in Eq. (5.2) (bottom).

5.2.4 Parenthood matrix for misaligned video sequences

As mentioned in Section 5.2.2, Dias et al. [24] deal with the video phylogeny tree reconstruction problem calculating the dissimilarity considering each frame of all the videos separately, generating n dissimilarity matrices (where n is the number of frames of all videos), reconstruct n video phylogeny trees, generate a “parenthood matrix”, that contains the frequency of the edges in each one of the trees previously calculated and reconstruct the final tree considering the most frequent edges. Although this approach has some limitations, the authors report better results when the parenthood matrix is reconstructed than calculating the average of the dissimilarity values for each pair of frames (as we described in the last section).

Here, we proposed an approach for the parenthood matrix reconstruction considering misaligned video sequences. Given a set of near-duplicate videos, some of the duplicates can be temporally clipped when they were generated. Taking into account that this operation removes some frames of the videos, it is common to find pairs of temporal misaligned video sequences, as Figure 5.1 shows.

To create a parenthood matrix considering the scenario with misaligned video sequences, we decided to align them temporally. First, we select one video, randomly, and label each frame according to its position in the video. For example, consider a video sequence $\mathcal{V}_i$ with n frames. The first frame has $id = 1$, the second frame has $id = 2$, and so on. Then, we perform the method described in Section 5.2.1 between $\mathcal{V}_i$ and each video sequence $\mathcal{V}_j, j \neq i$. After this alignment, we label each frame of $\mathcal{V}_j$ considering its position in the video and also the alignment performed with $\mathcal{V}_i$.

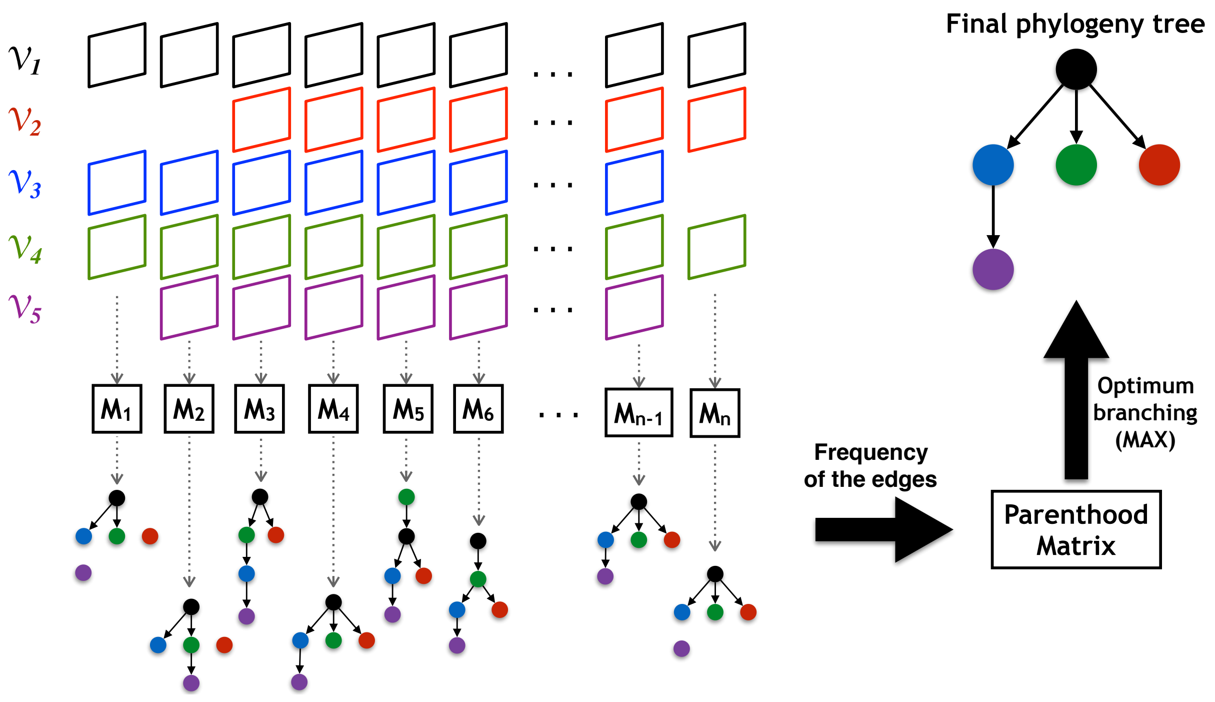


Figure 5.5: Process of phylogeny reconstruction after the calculation of a parenthood matrix.

After this, we calculate one dissimilarity matrix for each frame with one main difference. Given the time t, the dissimilarity matrix M_t will consider only the videos that have the frame $\mathcal{V}[t]$. The other fields of the dissimilarity matrix are filled with ∞. For each matrix, one phylogeny tree is reconstructed using the OB algorithm. The frequency of the edges of all phylogeny trees is calculated, and stored in a parenthood matrix. Finally, we generate the final phylogeny tree with the OB algorithm, aiming at generating one branching with maximum total weight. Figure 5.5 illustrates this process.

5.3 Experiments and results

In this section, we present details about the experiments that were performed for video phylogeny reconstruction, the obtained results and some discussions.

5.3.1 Dataset

The validation of the proposed video phylogeny tree reconstruction algorithm was carried out on a set with 200 phylogeny trees comprising a total number of 2,000 near-duplicate videos. More specifically, we started from eight well known uncompressed sequences at CIF resolution (i.e., 352×288 pixels) of 300 frames each, namely: *city*, *crew*, *deadline*, *foreman*, *hall*, *mobile*, *mother* and *paris*[1]. Then, for creating the duplicates, we considered the following possible transformations: contrast enhancement, brightness adjustment, spatial cropping and spatial resizing in any combination. As video codecs, we selected

[1]Available at `https://media.xiph.org/video/derf/`

Figure 5.6: Representative frames of the videos *city*, *crew*, *deadline*, *foreman*, *hall*, *mobile*, *mother*, and *paris*.

MPEG-2, MPEG-4 Part 4, and H.264/AVC[2]. Figure 5.6 shows one representative frame of each video used in this work.

Starting from these videos, we generated different phylogeny trees. Each tree was generated with a random structure starting from a randomly chosen video sequence. Near-duplicate videos (i.e., nodes of a tree) were generated from their parent sequences in three steps: (i) we applied a random transformation, randomly choosing the transformation parameters; (ii) we optionally (i.e., with probability p_{clip}) applied time clipping by removing a random number of frames from the beginning and/or the end of the video sequence; and (iii) we encoded (re-encoded) the sequence choosing a random codec, a random compression rate and a random GoP size.

Using this procedure, we generated two datasets differing in the probability p_{clip} of applying time clipping to each node. Table 5.1 shows the specific information for each dataset. Using such different datasets allows us to better study the behavior of the algorithm regarding different compression schemes, different tree sizes as well as to simulate a real-world scenario (i.e., mixing every possible transformation and codec with or without time clipping).

5.3.2 Evaluation metrics

To estimate the effectiveness of the estimated phylogeny trees, we used the following metrics as in [23]: (i) *Root* – correct identification of the tree root; (ii) *Edges* – correct orientation of children-parents relationships; (iii) *Leaves* – correct identification of the furthest sons in the tree; and (iv) *Ancestry* – correct reconstruction of all children-relatives relationships, from the root to the most distant sons. More formally, if VPT_i and VPT_j

[2]`libavcodec` implementations at `https://libav.org/`

Table 5.1: Parameters used to generate the used datasets. If $p_{\text{clip}} = 0$, time-clipping is not applied.

Dataset	$\mathcal{D}^{\text{no clip}}$	$\mathcal{D}^{\text{clip border}}$
N. of trees	100	100
N. of nodes	10	10
Frame rate	25 FPS	25 FPS
GoP	1 – 15	1 – 15
QP	1 –12	1 –12
Crop	±10%	±10%
Resize	±10%	±10%
Bright	±10%	±10%
Contrast	±10%	±10%
p_{clip}	0%	50%
Clip size	–	5–20%

are two trees to compare (i.e., the estimated one and the ground truth), these metrics are defined as

Root: $R(\text{VPT}_i, \text{VPT}_j) = \begin{cases} 1, & \text{if } R_i = R_j \\ 0, & \text{otherwise,} \end{cases}$

Edges: $E(\text{VPT}_i, \text{VPT}_j) = \frac{|E_i \cap E_j|}{N-1}$,

Leaves: $L(\text{VPT}_i, \text{VPT}_j) = \frac{|L_i \cap L_j|}{|L_i \cup L_j|}$,

Ancestry: $A(\text{VPT}_i, \text{VPT}_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}$,

where N is the number of nodes in a tree, R_i, E_i, L_i and A_i are the root and sets of edges, leaves and ancestry of VPT_i and R_j, E_j, L_j and A_j are the root and the sets of edges, leaves and ancestry of VPT_j, respectively. We also used a metric *depth*, that measures the distance, in number of edges, between the original root of the ground truth and the root of the reconstruct tree. In other words, if $R_i = R_j$, $depth(\text{VPT}_i, \text{VPT}_j) = 0$. Otherwise, $depth(\text{VPT}_i, \text{VPT}_j)$ is equal to the number of edges in the path from the root of the reconstructed tree and the node R_j.

5.3.3 Results and discussion

Turning our attention to the experiments, we show the importance of taking care of video coding and temporal alignment in the VPT reconstruction process. Figure 5.7 shows the average results obtained on trees from the $\mathcal{D}^{\text{no clip}}$ and $\mathcal{D}^{\text{clip border}}$ datasets when coding and temporal alignment are considered or not. The presented results are generated in different scenarios: not considering coding matching nor temporal alignment (blue), considering only the temporal alignment (red) and considering temporal alignment and coding matching (gray). All the results were obtained considering the average of the dissimilarities for each correspondent frame, considering two video sequences. All the experiments presented in this chapter were performed in a machine with an Intel Xeon

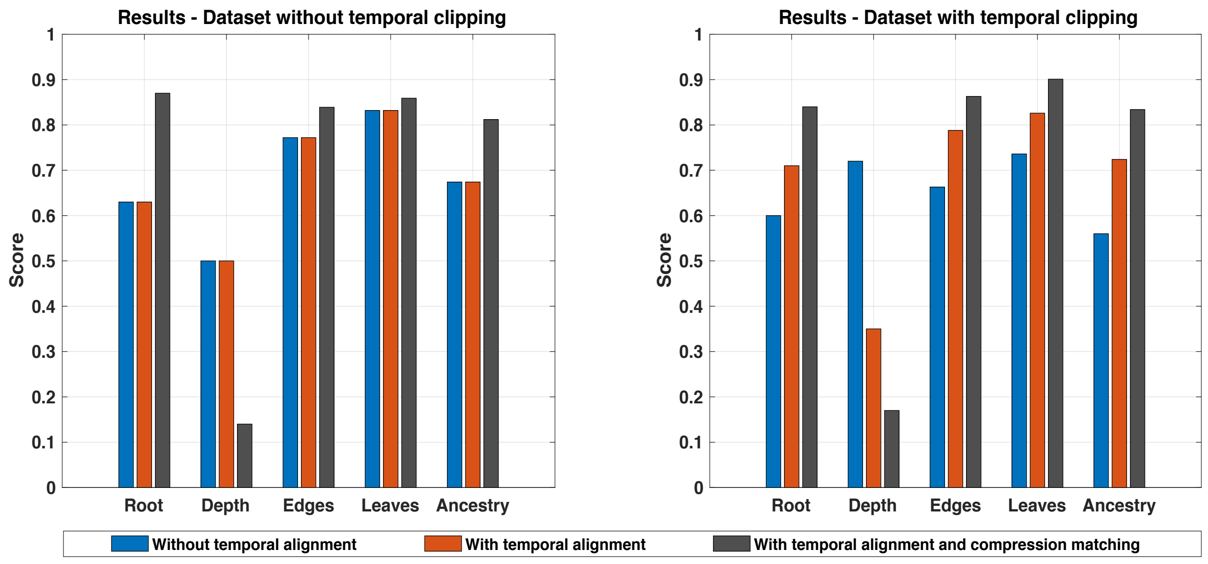


Figure 5.7: Results obtained with near-duplicate videos with ($\mathcal{D}^{\text{clip border}}$) and without temporal clipping ($\mathcal{D}^{\text{no clip}}$), considering or not coding matching and alignment.

E5645 processor, 2.40GHz, 16GB of memory, and running Ubuntu 12.04.5 LTS. The source code was implemented using the libraries OpenCV v2.4.10 and FFMPEG 2.8.6.

The results presented in Figure 5.7 show the robustness of the algorithm in a real-world case, where the video codec can change from node to node and time clipping can be applied. They clearly demonstrate the need for explicitly considering the temporal alignment of the video sequences and also the coding matching in the registration procedure when dealing with videos that have been clipped and coded (as in real-world setups).

The results without any kind of alignment for the dataset with temporal clipping were obtained when it was possible to calculate the frame registration. In cases in which one of the video sequences had been clipped in the beginning and without performing any temporal alignment, the frame registration will be done considering frames that are not correspondent. In this case, the frame registration is performed considering wrong matches of keypoints, resulting in a completely wrong frame mapping, as Figure 5.8 shows.

Aiming at showing the difference between the dissimilarity calculations for the phylogeny tree reconstruction, Figure 5.9 shows the average results obtained on trees from the $\mathcal{D}^{\text{no clip}}$ and $\mathcal{D}^{\text{clip border}}$ datasets. For this case, we always consider the temporal alignment. The results are separated for each dataset and for each methodology of dissimilarity calculation (average of MSE of each pair of correspondent frames $\times$ parenthood matrix).

The results presented in Figure 5.9 show that the reconstruction of one parenthood matrix instead to considering the average of dissimilarities correspondent frames of two video sequences slightly improves the quality of the phylogeny reconstruction. As expected, the compression matching in the frame registration step improves the results and is an important step. Unfortunately, it is not easy to obtain all compression parameters from videos (as is done for images), and, as far as we know, there is no reliable method for estimating it for all different codecs that are available in the market. However, for scientific

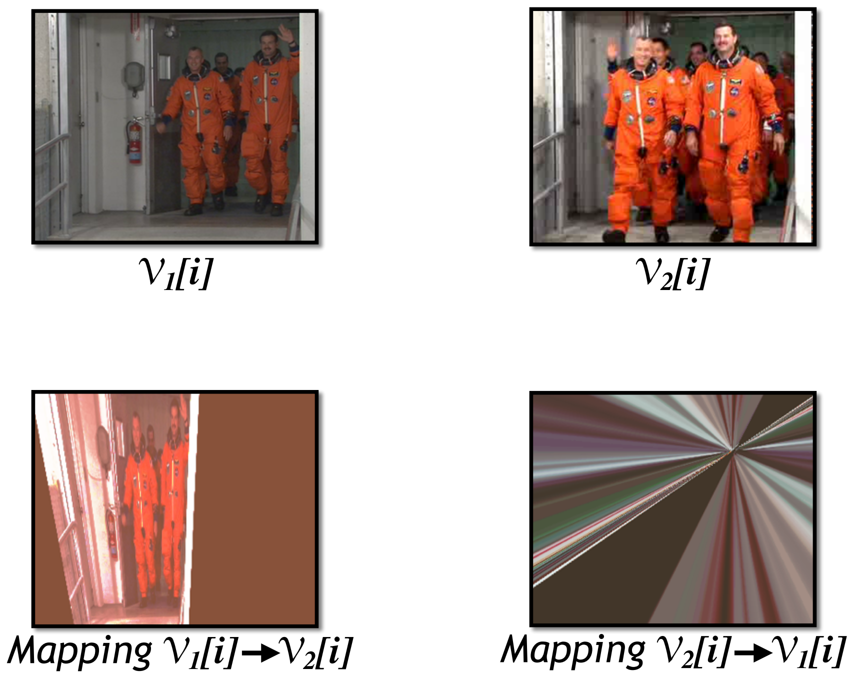

$\mathcal{V}_1[i]$
$\mathcal{V}_2[i]$
Mapping $\mathcal{V}_1[i] \rightarrow \mathcal{V}_2[i]$
Mapping $\mathcal{V}_2[i] \rightarrow \mathcal{V}_1[i]$

Figure 5.8: Example of wrong frame registration, considering the i-th frame of two video sequences $\mathcal{V}_1$ and $\mathcal{V}_2$ without temporal alignment.

purposes, the results with coding matching as we presented in this chapter are valid and shows the need for further investigations on how to retrieve/estimate compression parameters from video standards directly.

Considering the results presented in this section, we can conclude that the experiments clearly show the importance of the temporal alignment, given that the video sequences can be temporally misaligned and with different number of frames (e.g., temporally clipped), allowing us to compare only correspondent frames of two near-duplicate videos. Furthermore, the step of coding matching is also important when mapping one video onto another for calculating the dissimilarity matrix, improving the quality of the phylogeny reconstruction.

5.4 Publication

The main findings of the research presented in this chapter resulted in one article published in an international conference:

- **F. O. Costa, S. Lameri, P. Bestagini, Z. Dias, A. Rocha, M. Tagliasacchi and S. Tubaro**, *Phylogeny reconstruction for misaligned and compressed video sequences*, IEEE Intl. Conference on Image Processing (ICIP), pp. 301-305, 2015. [14]

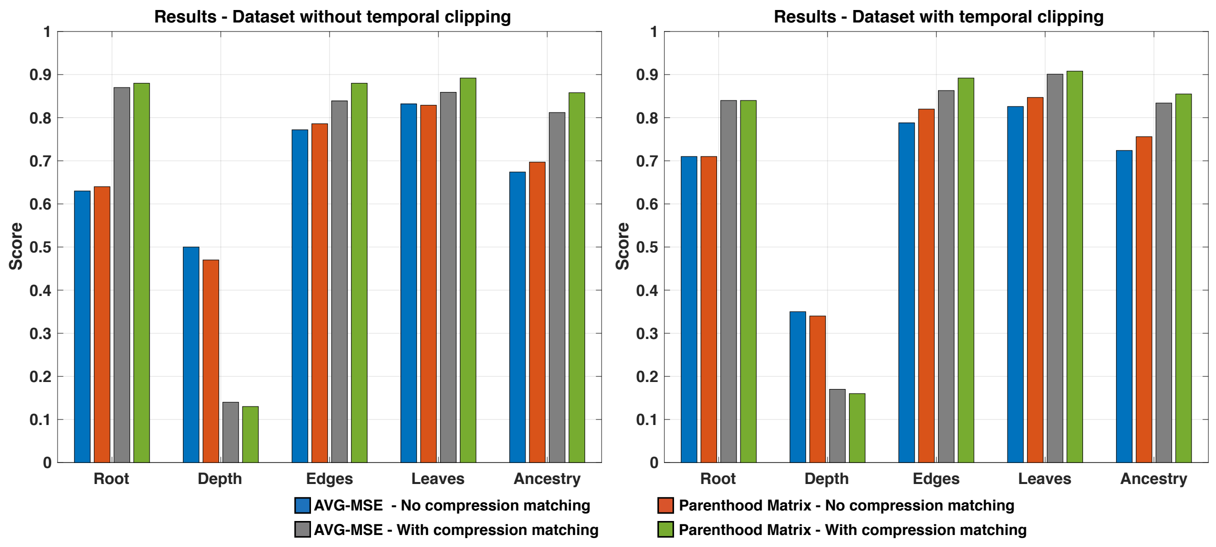


Figure 5.9: Results: Average of MSE (frame-wise) vs. Parenthood matrix.

Chapter 6

Hashing-based Temporal Alignment for Video Phylogeny Reconstruction

As showed in Chapter 5, it is important to perform a temporal alignment between two video sequences before performing the mapping from one video sequence onto another, in order to compare only correspondent frames, given that near-duplicate videos can have a different number of frames and may not be always temporally coherent. Not performing this temporal alignment can result in problems at the frame registration step, as showed before in Figure 5.8. Furthermore, to compare only correspondent frames is important to turn the dissimilarity calculation more accurate.

The approach presented in the last chapter considered only videos that might be clipped at the beginning and/or at the end of the stream. However, one common operation present in real-world video editing consist of also performing temporal clipping in the middle of the video sequence. If we apply only the temporal alignment based on difference of luminance (described in Section 5.2.1) in near-duplicate videos generated by temporal clipping in the middle of one video sequence, only part of the video sequence will be correctly aligned, leading to a possible comparison of non-correspondent frames, which negatively affects the dissimilarity calculation, as Figure 6.1 illustrates.

In this chapter, we present a new method for dealing with the video phylogeny reconstruction problem that takes into account video sequences that can be generated by temporal clipping not only at the beginning/end of the stream, but at anywhere. The performed experiments show promising results for this more challenging video phylogeny problem.

6.1 Temporal alignment based on Hamming distance

The temporal alignment presented in this chapter was first proposed by Lameri et al. [39] and it is related to a previous work in the field of video fingerprinting / robust hashing [12, 55]. Hashing-based methods are used mainly to determine the presence of near-duplicate content to support other applications such as content-based video authentication [40] and content-based video retrieval [64, 72].

The alignment approach we consider herein aims at reconstructing the content of the

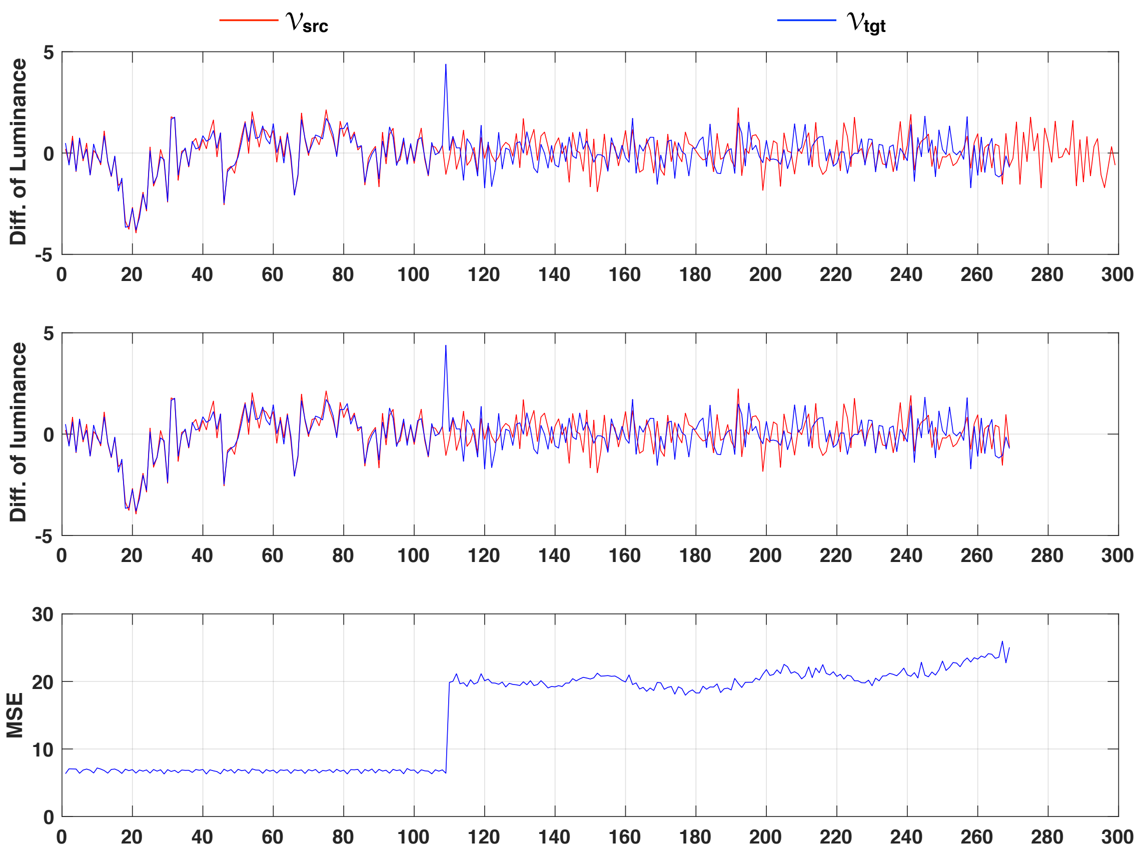


Figure 6.1: Temporal alignment based on difference of luminance, considering that one video was generated by temporal clipping in the middle of the video: (top) The sequences are not aligned; (middle) temporal alignment based on difference of luminance (Chapter 5) is applied; (bottom) dissimilarity between $\mathcal{V}_{src}$ and $\mathcal{V}_{tgt}$; the values remain approximate when we compare two correspondent frames of the sequences, but it increases significantly when the comparison is made between two non-correspondent frames.

original source videos, i.e., a parent sequence, given a set of partially overlapped near-duplicate video shots used in other sequences. Given two video sequences, this approach works considering the following steps:

1. Near-duplicate matching: the method detects whether or not the video sequences share common frames (more details in Section 6.1.1);

2. Near-duplicate extraction: if the sequences have common content, the method reveals which parts of the sequences are near duplicates;

3. Near-duplicate alignment: after the common part of the video sequences are extracted, we perform an additional step of fine temporal alignment between the shots, considering the alignment based on difference of luminance, presented in Section 5.2.1;

4. Shot extraction: the common shots are identified and temporally sorted with respect to each other, considering scene changes when the average luminance suddenly

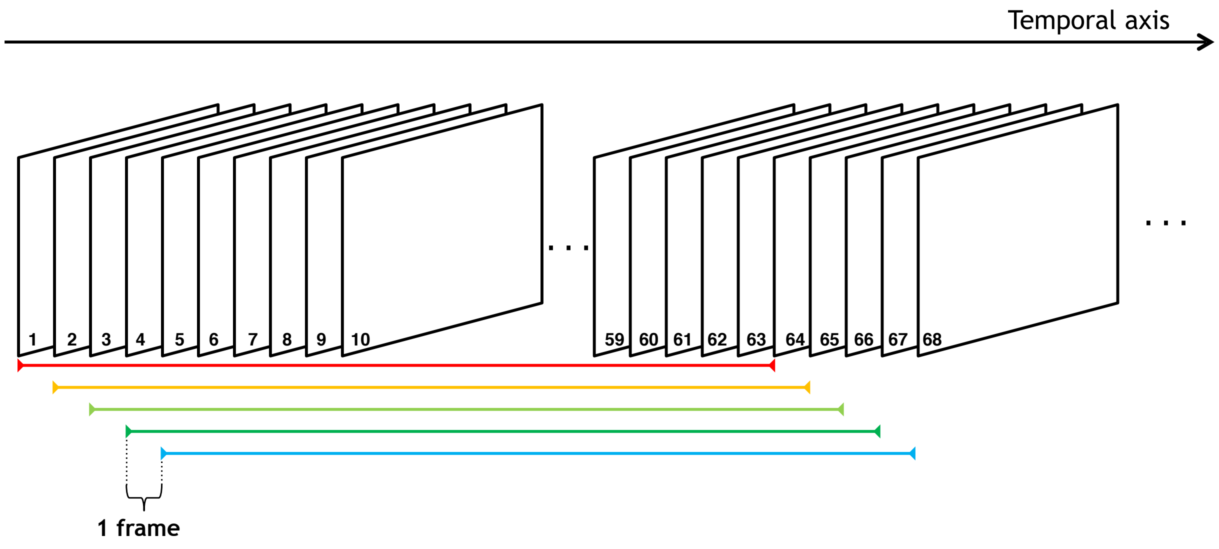


Figure 6.2: Groups of 64 frames, overlapped by 63 frames.

changes from a frame to another;

5. Parent reconstruction: the method groups shots belonging to the same parent sequence, linking together pairs of video sequences whose corresponding shots share a common subsequence of frames.

In our work, we perform the Steps 1, 2 and 3 and use the output of Step 3 for performing the same approach we proposed for video phylogeny in Chapter 5 for further refining the alignment. We describe these steps below.

6.1.1 Near-duplicate matching

The first step aims at detecting whether two video sequences share common frames. As we are not considering composed videos and different video content (video phylogeny forest), we perform this step for finding the parts of the sequences that have similar content.

For this purpose, we make use of the hashing algorithm proposed in [55] for estimating a distance matrix between the video sequences. Initially, all the frames of the sequences are re-scaled to 32×32 pixels and split into groups of 64 frames, overlapped by 63 frames, as Figure 6.2 illustrates. With this, each video sequence $\mathcal{V}$ is split into $n = ||\mathcal{V}|| - 63$ groups of 64 frames.

Then, a $3D$ Discrete Cosine Transform (DCT) is applied to each group $\mathcal{C}$, obtaining $32 \times 32 \times 64$ DCT coefficients. After this, the DCT coefficients $c_{x,y,z}|x, y, z \in [1, ..., 4]$ are extracted (the coefficient $c_{0,0,0}$ is the DC coefficient), obtaining a total of 64 coefficients for each group of 64 frames. Figure 6.3 depicts the DCT coefficient selection.

After calculating the DCT coefficients for all group of 64 frames, a binary hash h_k^n is computed for the n-th group of frames $\mathcal{C}_n$ of the video sequence $\mathcal{V}_k$. This binary hash is composed by a 64-bit string obtained binarizing the 64 DCT coefficients according to a threshold $med_{\mathcal{C}_n}$ = the median value of $DCT_{\mathcal{C}_n}$ (the DCT coefficients of $\mathcal{C}_n$). In other

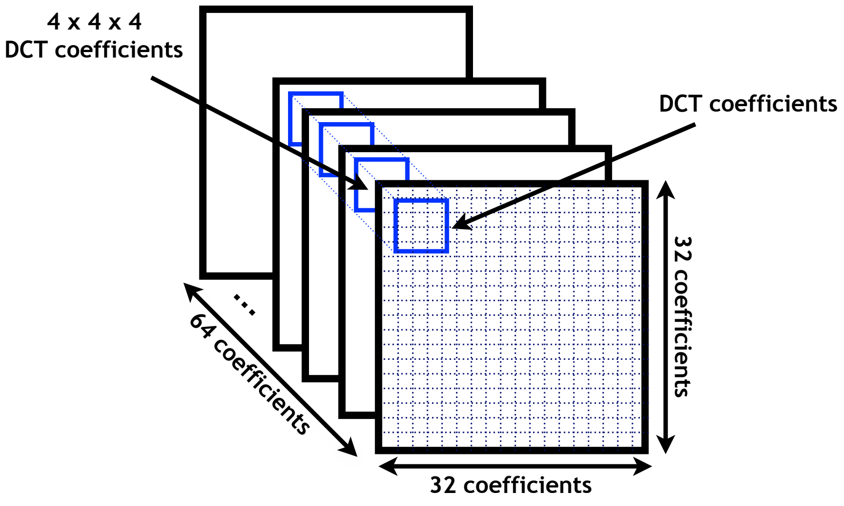


Figure 6.3: DCT coefficient selection. After calculating all $32 \times 32 \times 64$ DCT coefficients for one group of frames $\mathcal{C}$, we select the $4 \times 4 \times 4$ coefficients $c_{x,y,z} | x, y, z \in [1, ..., 4]$ (blue region).

words, this process is done according to the following equation.

$$h_k^n[i] = \begin{cases} 1, & \text{if } \mathrm{DCT}_{\mathcal{C}_n}[i] > med_{\mathcal{C}_n}; \\ 0, & \text{otherwise}; \end{cases} i = 1, 2, ..., 64. \tag{6.1}$$

Once the binary hash is computed for all clusters, they are compared pairwise using the Hamming distance [33]. Given two binary hashes $h_{k_1}^{n_1}$ and $h_{k_2}^{n_2}$, the Hamming distance H between them is calculated by

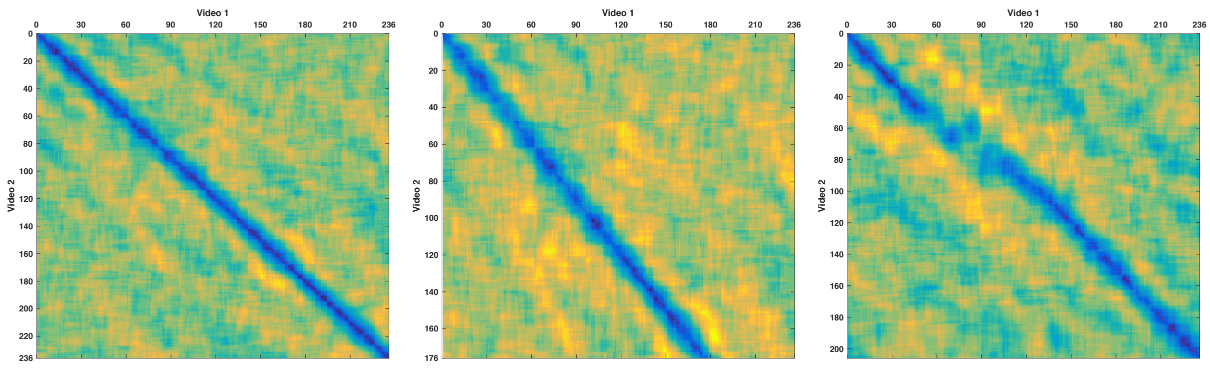
$$\mathrm{H}(h_{k_1}^{n_1}, h_{k_2}^{n_2}) = \sum_{x=1}^{64} h_{k_1}^{n_1}[x] \oplus h_{k_2}^{n_2}[x], \tag{6.2}$$

where $\oplus$ is the Exclusive OR (XOR) operator. Low values of H indicate that the hashes are similar, which means that the clusters $\mathcal{C}_{n_1}$ and $\mathcal{C}_{n_2}$ are groups of near-duplicate frames.

By calculating the Hamming distance for every pair of clusters of two sequences, we obtain a distance matrix $M_{Hamming}$ in which it is stored the Hamming distances of all pairs of clusters, respecting their temporal order. Formally, given two video sequences $\mathcal{V}_i$ and $\mathcal{V}_j$ that comprise n_1 and n_2 groups of 64 frames, respectively, the values of $M_{Hamming}$ are given by

$$M_{Hamming}[x, y] = \mathrm{H}(h_i^x, h_j^y) \forall x = 1, 2, ..., n_1; y = 1, 2, ..., n_2. \tag{6.3}$$

Figure 6.4 shows examples of such distance matrices. The axes represent two video sequences, and darker colors mean lower values for Hamming distance between them.



(a) (b) (c)

Figure 6.4: Example of distance matrices, in which (a) both video sequences do not have temporal clipping and (b) Sequence 2 has temporal clipping in the end; and (c) Sequence 2 has temporal clipping in the middle. The axes represents the index of the groups of 64 frames for each video.

Figure 6.4 shows that low values in the matrices are found to be aligned along a segment. The higher the number of matching frames, the longer the "dark blue" segment. Given one distance matrix $M_{Hamming}$, the method automatically detects the presence of one or more matching blocks of near duplicates as follows: first, the values of $M_{Hamming}$ are binarized considering a defined threshold τ set to discriminate between similar and non-similar groups of frames. Then, a morphological opening is applied to the image representation of the distance matrix to discard spurious areas of local low values and identify the connected components (the parameters τ and the structuring elements used for the opening operation are described in Section 6.2.1). Finally, the connected components are identified by determining the coordinates of the minimum value along every row and column and fitting a line passing through these minima. This line represents the block of near-duplicate frames that are common. Figure 6.5 shows the matching blocks detection.

6.1.2 Near-duplicate extraction and alignment

The previous analysis allows us to identify which parts of two video sequences are near duplicates. Given the distance matrix $M_{Hamming}$, let the point $M_{Hamming}[x_1, y_1]$ be the start point of the segment and $M_{Hamming}[x_2, y_2]$ the end of the segment. This means that the frames $\mathcal{V}_1\{x_1, ..., x_2+63\}$ are near duplicates of $\mathcal{V}_2\{y_1, ..., y_2+63\}$, where the constant value 63 considers the fact that the alignment was done considering hashes computed by clusters of 64 frames overlapped by 63 frames.

After extracting these blocks, each correspondent block is temporally aligned once more, considering the temporal alignment based on luminance difference, presented in Section 5.2.1. Finally, once we have the blocks of near duplicates of two video sequences, we can calculate the dissimilarity matrix considering all correspondent frames of them, as presented in Section 5.2.2.

Figure 6.6 shows the results of the different approaches presented in this thesis for the video temporal alignment, considering two misaligned video sequences $\mathcal{V}_1$ and $\mathcal{V}_2$.

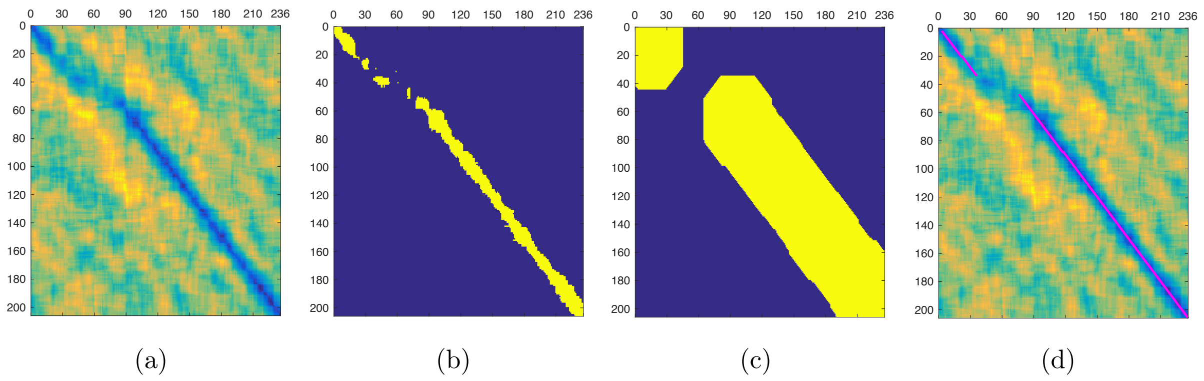


Figure 6.5: Matching blocks detection. (a) Alignment of two video sequences; (b) The Hamming distance values are binarized according to a threshold τ; (c) Result of the opening operation; (d) Matching blocks (represented by the pink lines).

In this example, $\mathcal{V}_2$ was generated by applying temporal clipping in its parent sequence and it has two blocks of near-duplicate frames with $\mathcal{V}_1$ (Figure 6.6.a). The first chart in Figure 6.6.b shows the temporal misalignment of the sequences by the difference of luminance of them. If we apply only the temporal alignment considering the correlation between the differences of luminance (the second chart of Figure 6.6.b), only part of the video sequences will be correctly temporally aligned. However, if we perform the hashing-based temporal alignment, extract the blocks of near-duplicate frames and align them separately, we have a better temporal alignment (the third chart of Figure 6.6.b).

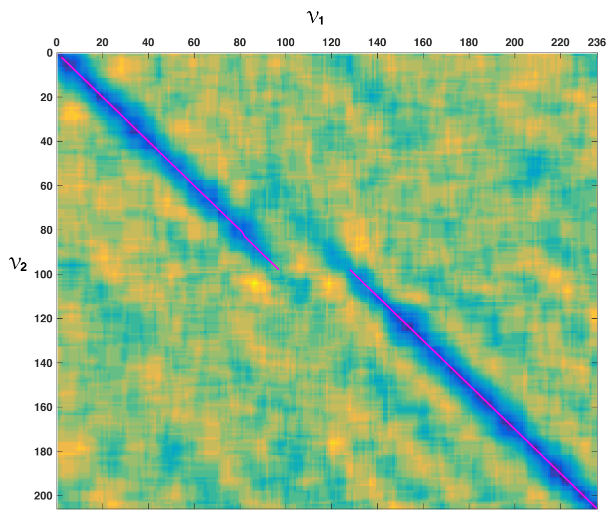
6.2 Experiments and results

In this section, we present the experiments performed with the hashing-based temporal alignment, the obtained results and some discussions.

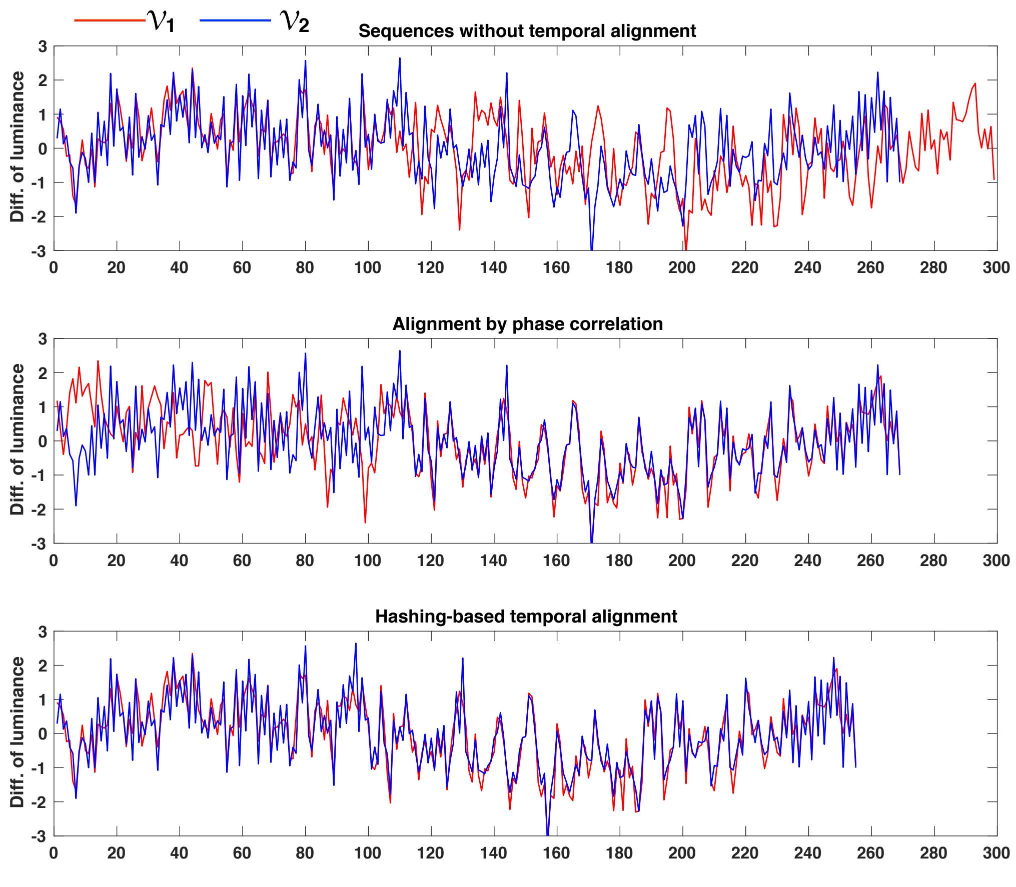
6.2.1 Datasets and evaluation

The validation of the proposed temporal alignment for video phylogeny reconstruction was carried out on a set of 300 phylogeny trees comprising a total number of 3,000 near-duplicate videos. More specifically, we used the datasets $\mathcal{D}^{\text{no clip}}$ and $\mathcal{D}^{\text{clip border}}$ presented in Section 5.3 and created another dataset $\mathcal{D}^{\text{clip any}}$, considering the same conditions of the others but including different temporal clippings. In this dataset, the near duplicates can be generated applying temporal clipping not only in the beginning/end of the video sequences, but also in the middle of them. The probability of temporal clipping is also $p_{\text{clip}} = 0.5$. For the video alignment, we considered the threshold $\tau = 16$ for binarizing the distance matrices and the opening operation was performed using a disk with radius $= 5$ pixels for erosion and another with radius $= 30$ pixels for dilation. The chosen parameters were the same used in [39] for the step of near-duplicate detection[1].

[1]Performing a small exploratory test, we conclude that these parameters are the best parameters also for our work.



(a)



(b)

Figure 6.6: Results of video temporal alignment approaches: (a) Distance matrix of the sequences; (b) *top:* video sequences $\mathcal{V}_1$ and $\mathcal{V}_2$ are temporally misaligned; *middle:* the video sequences aligned through the correlation between the differences of luminance of their frames; *bottom:* video sequences aligned with the hashing-based approach proposed in this chapter with a further refinement step using the correlation between the differences of luminance proposed in Chapter 5.

We considered cases with and without coding matching and OB algorithm for the phylogeny reconstruction. To evaluate the reconstructed trees, we considered the same evaluation measures presented in Section 5.3: *Root*, *Depth*, *Edges*, *Leaves* and *Ancestry*.

6.2.2 Results and discussion

Table 6.1 shows the average results obtained with trees from the $\mathcal{D}^{\text{no clip}}$, $\mathcal{D}^{\text{clip border}}$ and $\mathcal{D}^{\text{clip any}}$ dataset not considering coding matching (NOT-COD) and considering coding matching (WITH-COD). In both cases, we compare the temporal alignment based on difference of luminance (LUMA) and the hashing-based temporal alignment (HASH). The experiments presented in this chapter were performed in a machine with an Intel Xeon E5645 processor, 2.40GHz, 16GB of memory, and running Ubuntu 12.04.5 LTS. The source code was implemented using the libraries OpenCV v2.4.10 and FFMPEG 2.8.6.

Table 6.1: Results obtained on the $\mathcal{D}^{\text{no clip}}$, $\mathcal{D}^{\text{clip border}}$ and $\mathcal{D}^{\text{clip any}}$ dataset, comparing different approaches for temporal alignment of the video sequences.

Method	Dataset	Alignment	*Root*	*Depth*	*Edges*	*Leaves*	*Ancestry*
NOT-COD	$\mathcal{D}^{\text{no clip}}$	LUMA	0.63	0.50	0.772	0.832	0.674
NOT-COD	$\mathcal{D}^{\text{no clip}}$	HASH	0.65	0.44	0.777	0.828	0.702
WITH-COD	$\mathcal{D}^{\text{no clip}}$	LUMA	0.87	0.14	0.839	0.859	0.812
WITH-COD	$\mathcal{D}^{\text{no clip}}$	HASH	0,69	0.40	0.778	0.831	0.711
NOT-COD	$\mathcal{D}^{\text{clip border}}$	LUMA	0.71	0.35	0.788	0.826	0.724
NOT-COD	$\mathcal{D}^{\text{clip border}}$	HASH	0.71	0.33	0.807	0.846	0.740
WITH-COD	$\mathcal{D}^{\text{clip border}}$	LUMA	0.84	0.17	0.863	0.901	0.834
WITH-COD	$\mathcal{D}^{\text{clip border}}$	HASH	0.66	0.40	0.780	0.824	0.722
NOT-COD	$\mathcal{D}^{\text{clip any}}$	LUMA	0.62*	0.56*	0.724*	0.758*	0.628*
NOT-COD	$\mathcal{D}^{\text{clip any}}$	HASH	0.67	0.45	0.755	0.789	0.683
WITH-COD	$\mathcal{D}^{\text{clip any}}$	LUMA	0.72*	0.39*	0.801*	0.828*	0.696*
WITH-COD	$\mathcal{D}^{\text{clip any}}$	HASH	0.66	0.47	0.765	0.810	0.696

Table 6.1 shows that the hashing-based temporal alignment has similar performance when compared with the temporal alignment based on the difference of luminance, considering the cases when temporal clipping was not applied over the near duplicates (dataset $\mathcal{D}^{\text{no clip}}$) or was applied only in the beginning/end of the video sequences (dataset $\mathcal{D}^{\text{clip border}}$). It also presents similar results for the cases when we have duplicates created by temporal clipping in the middle of the sequences, when we do not consider the coding matching step.

The phylogeny reconstruction for the cases considering the coding matching has lower performance, compared to the alignment based on difference of luminance. One justification for these results is the nature of video compression. Contrary to the image compression matching, when we just compress the source image considering the JPEG quantization table of the target image, the video compression is more complex and also considers the dimension of time, mainly for performing *inter-frame* compression. Once we are assuming that the alignment is not perfect, some frames can be discarded, mainly at the beginning and at the end of each block of frames in the frames selection step. When we perform the video encoding step over the video sequences in this situation, the coding may select different frames to perform the prediction, comparing to the frames that were selected when the near duplicate was generated. It may therefore affect the results of compression matching, once we are performing a different *intra-frame* compression.

The results for the dataset $\mathcal{D}^{\text{clip any}}$ considering the temporal alignment based on distance of matrix has similar performance when compared to the results for $\mathcal{D}^{\text{no clip}}$ and $\mathcal{D}^{\text{clip border}}$. It means that the robustness of the new temporal alignment remains when we have videos with temporal clipping in the middle of the sequences. It is important to note that the results for the dataset $\mathcal{D}^{\text{clip any}}$ for the alignment based on difference of luminance were generated only for comparison. For these cases, the average of dissimilarity is not robust, once we are performing a comparison between frames that are not correspondent to each other, resulting in the problem depicted previously in Figure 6.1.

Concluding, the results presented in this section confirm that the hashing-based temporal alignment is important when we are dealing with videos with temporal clipping in the middle of the sequences. This alignment is more robust than the previous alignment based on difference of luminance (Chapter 5), once it better guarantees that only correspondent frames will be compared for the dissimilarity calculation. Table 6.2 shows a comparative analysis between the two temporal alignment approaches described in this thesis.

6.3 Publication

The main findings of this research resulted in one submission to an international conference:

- **Filipe de O. Costa, Marina Oikawa, Zanoni Dias and Anderso Rocha**, *Temporal alignment based on Hamming Distance for Video Phylogeny Reconstruction*, IEEE Intl. Workshop on Information Forensics and Security (WIFS) – Submitted. [16]

Table 6.2: Comparing the temporal alignment approaches: Difference of luminance vs. Hashing-based approach.

Method	Based on difference of luminance	Hashed-based
Robustness when not considering coding matching step	Good	Good
Robustness when considering coding matching step	Good	Bad
Robustness against videos without temporal clipping	Good	Good
Robustness against videos with border temporal clipping	Good	Good
Robustness against videos with any temporal clipping	Bad	Good
Conclusions	• Faster than the Hashed-based method • Better effectiveness when considering coding matching • Good effectiveness when the videos are not clipped or are clipped at the beginning/end of the stream • Does not work when videos are clipped anywhere in the stream	• More robust alignment • Deal with videos with any kind of temporal clipping • Lower performance when considering coding matching step

Chapter 7

Conclusions and Future Work

In this thesis, we presented new solutions for image and video phylogeny reconstruction, a more challenging problem than the near-duplicate detection problem. In image and video phylogeny, we want to find the ancestral relationship between the near duplicates and the original source. Solutions for image and video phylogeny are useful for helping to solve problems of copyright enforcement, illegal content tracking and forensics, among others.

Image Phylogeny: For image phylogeny, our contributions focus on solving two main problems: the phylogeny forest reconstruction and dissimilarity calculation for each pair of compared images.

We introduced three new different approaches to deal with image phylogeny forests. First, we extended upon the approach developed for phylogeny trees by Dias et al. [19], using Optimum Branching and applying a similar idea used for automatic reconstruction of IPFs that the authors used for their Oriented Kruskal algorithm. By finding a decision point after analysis of the behavior of valid forests, we made possible the use of the optimum branching algorithm to deal with forests, resulting in the AOB method.

Results were further improved with the proposal of E-AOB, which employs an important additional step of re-execution of the OB algorithm in each tree of the IPF found by AOB, further refining the initial results. Both approaches outperformed the state-of-the-art AOK method presented in the literature.

We have also explored the idea of combining the best matches among the proposed methods, aiming at reducing the errors introduced by them when they are used independently, and improve the score of all phylogeny-related metrics. Thus, we proposed a novel approach through the combination of the reconstructed forests of AOK, AOB and E-AOB methods. Results for this new approach showed better or equivalent performance to the best result achieved thus far (E-AOB method), being a competitive solution for the image phylogeny problem.

The introduced fusion algorithm explores a view of the values of a dissimilarity matrix as random variables. Since the relative ordering between causal pairs of images can change when their difference is small enough, our method introduces robustness to statistical errors in the estimation of the image dissimilarities, and takes away some of the importance of extra fine tuning the dissimilarity calculation. The method can be easily extended to an arbitrary number of algorithms.

We have also explored novel approaches for computing the dissimilarity between two images, applied to the problem of image phylogeny forest reconstruction. Our method rely upon the incorporation of a different color matching approach for better estimating the involved changes during the generation of near duplicates and the comparison between two images using gradient calculation and mutual information estimation.

This work shows that comparing distributions is better than direct point-wise comparisons (with mutual information outperforming MSE as the comparison approach), gradient distributions are more appropriate than direct color distributions (with gradient-based comparisons outperforming pixel-based comparisons when combined with mutual information), and it also shows that a more powerful family of color transformations enables a better tree reconstruction at the end of the dissimilarity calculation pipeline (with the incorporation of the histogram matching approach).

As discussed earlier, we provide direct comparisons, using the Wilcoxon signed-rank test, between the GRMI/HGMI and all combinations of these methods. These improvements are not marginal and certainly will significantly boost the current existing image phylogeny solutions as the dissimilarity calculation step, although overlooked thus far, is as important to the whole process as is the actual tree reconstruction step. The HGMI method also presented good results in real-case setups, with good separation of different groups of near-duplicate images showing good potential for real-world deployment when analyzing the relationship among images.

Video Phylogeny: In video phylogeny, we have a much more challenging setup. In this case, we introduced approaches to video phylogeny tree reconstruction starting from the analysis of a pool of near-duplicate video sequences. The proposed methods are based on the same pipeline presented in [24], but they accommodate the case of time clipped, misaligned and compressed video sequences. Indeed, videos are often distributed in compressed format and cannot be considered aligned in a real-world scenario. Dealing with more complex videos allow us to solve the phylogeny problems considering real scenarios, in which we do not know which transformations were used for creating the near-duplicates.

We proposed two temporal alignment for the video sequences. The first is based on the difference of luminance of subsequent frames and the correlation between these differences. The second is based on the generation of a distance matrix of two video sequences, by means of 3D-DCT calculation, Hamming distance and opening operation. The first temporal alignment approach can solve the phylogeny problems when we consider near-duplicate videos that were clipped in the beginning or in the end of the sequence, but the second approach can also deal with phylogeny problems when the videos were clipped anywhere in the sequence.

The performed analysis clearly shows the need of explicitly considering coding, temporal clipping and temporal alignment in the reconstruction process. As a matter of fact, when these operations are not taken into account, the reconstruction accuracy drops significantly. Therefore, the proposed methods can be regarded as an important step toward video phylogeny of real-world video sequences, once that, as far as we know, there is only one work in the literature [24] addressing this problem thus far. In addition, when the coding leaves more pronounced artifacts in the generated videos, finding the parent-child relationships is slightly easier (inter vs. intra-coded cases), considering that the video compression is an irreversible operation.

Future work: As future work for image phylogeny, one could further analyze the behavior of the phylogeny algorithms using other statistical measures for the IPF reconstruction. For instance, kurtosis is a statistical measure of extreme variation, such that higher values denote a few extreme deviations in the data, while lower values show more frequent, smaller deviations. Variations on the kurtosis pattern have been previously studied in image forensics to detect, for instance, noise added to the image at various stages of production, which causes changes in the kurtosis [74], or image splicing, by identifying any portions of the image with significantly different kurtosis values [56]. By exploring an analogous possibility, we hypothesize that it is possible to find a relationship between the variation of the kurtosis of the edges chosen by the phylogeny forest algorithms while reconstructing the forest, and the choice of the number of trees this forest should have.

Another branch of research could also consider other types of image transformations, such as *blurring*, *sharpness changing* and *content insertion* (e.g., logos), which are very common in the creation of near-duplicate images.

For video phylogeny, we see the results are still below those obtained with images. Therefore, future work could be devoted to extending the considered working scenarios. The first extension could consider videos with different frame rates. On top of that, one could work toward improving the geometric registration step, extending the geometric transformation estimation method to consider more frames. Investigating different tree reconstruction strategies exploiting frame-wise dissimilarity information rather than the sole dissimilarity average, could also be a path worth pursuing and with promising rewards. Another branch could be the segmentation of the videos into shots before comparison instead of comparing them frame-wise. Furthermore, a better coding matching step when the temporal alignment based on distance matrices is applied can be investigated.

Certainly, another extension worth exploring consists of adapting the existing methods to cope with videos generated by composition between two different video sequences, the multiple-parenting scenario [53] mentioned previously. Furthermore, one could also evaluate the impacts of new dissimilarity calculations to phylogeny estimation for different videos.

Finally, considering all the experiments on video phylogeny presented in this thesis were developed considering a somewhat controlled setup, evaluations taking into account real-world cases would be invaluable. In this case, we would need to design and develop robust methods to reliably estimate the compression parameters directly from the video stream, given that some parameters (e.g., quantization parameter) are readily available only for some specific codes (e.g., H264).

Bibliography

[1] Nicholas Andrews, Jason Eisner, and Mark Dredze. Name phylogeny: a generative model of string variation. In *Proc. of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 344–355, 2012.

[2] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks (TNN)*, 5(4):537–550, 1994.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Elsevier Computer Vision Image Understanding*, 110(3):346–359, 2008.

[4] Paolo Bestagini, Ahmed Allam, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro. Video codec identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2257–2260, 2012.

[5] Paolo Bestagini, Marco Tagliasacchi, and Stefano Tubaro. Image phylogeny tree reconstruction based on region selection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2059–2063, 2016.

[6] Frederick Bock. An algorithm to construct a minimun directed spanning tree in a directed network. *Developments in operations research*, 1(1):29–44, 1971.

[7] Roger Bramon, Imma Boada, Anton Bardera, Joaquim Rodriguez, Miquel Feixas, Josep Puig, and Mateu Sbert. Multimodal data fusion based on mutual information. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18(9):1574–1587, 2012.

[8] Kennedy. A. Brownlee. *Statistical theory and methodology in science and engineering*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1965.

[9] Xiangang Cheng and Liang-Tien Chia. Stratification-based keyframe cliques for removal of near-duplicates in video search results. In *ACM International Conference on Multimedia Information Retrieval*, pages 313–322, 2010.

[10] Yoeng-Jin Chu and Tseng-Hong Liu. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396, 1965.

[11] Ondrej Chum, James Philbin, and Andrew Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. 810:812–815, 2008.

[12] Baris Coskun, Bulent Sankur, and Nasir Memon. Spatio–temporal transform based video hashing. *IEEE Transactions on Multimedia*, 8(6):1190–1208, 2006.

[13] Filipe de O. Costa, M. Eckmann, W. J. Scheirer, and Anderson Rocha. Open set source camera attribution. In *IEEE Conference on Graphics, Pattern and Images (SIBGRAPI)*, pages 71–78, 2012.

[14] Filipe de O. Costa, Silvia Lameri, Paolo Bestagini, Zanoni Dias, Anderson Rocha, Marco Tagliasacchi, and Stefano Tubaro. Phylogeny reconstruction for misaligned and compressed video sequences. In *IEEE International Conference on Image Processing (ICIP)*, pages 301 – 305, 2015.

[15] Filipe de O. Costa, Marina Oikawa, Zanoni Dias, Siome Goldenstein, and Anderson Rocha. Image phylogeny forests reconstruction. *IEEE Transactions on Information Forensics and Security (TIFS)*, 9(10):1533–1546, 2014.

[16] Filipe de O. Costa, Marina Oikawa, Zanoni Dias, and Anderson Rocha. Temporal alignment based on hamming distance for video phylogeny reconstruction. In *Submitted to IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–5, 2016.

[17] Filipe de O. Costa, Alberto Oliveira, Pasquale Ferrara, Zanoni Dias, Siome Goldenstein, and Anderson Rocha. New dissimilarity measures for image phylogeny reconstruction. Technical Report IC-15-07, Institute of Computing, University of Campinas, 2015. In English, 20 pages.

[18] Filipe de O. Costa, Ewerton Silva, Michael, Walter J. Scheirer, and Anderson Rocha. Open set source camera attribution and device linking. *Pattern Recognition Letters*, 39:92–101, 2014.

[19] Zanoni Dias, Siome Goldenstein, and Anderson Rocha. Exploring heuristic and optimum branching algorithms for image phylogeny. *Elsevier Journal of Visual Coimmunication and Image Representation*, 24:1124–1134, 2013.

[20] Zanoni Dias, Siome Goldenstein, and Anderson Rocha. Large-scale image phylogeny: Tracing back image ancestry relationships. *IEEE Multimedia*, 20:58–70, 2013.

[21] Zanoni Dias, Siome Goldenstein, and Anderson Rocha. Large-scale image phylogeny: Tracing image ancestral relationships. *MultiMedia, IEEE*, 20(3):58–70, 2013.

[22] Zanoni Dias, Siome Goldenstein, and Anderson Rocha. Toward image phylogeny forests: Automatically recovering semantically similar image relationships. *Elsevier Forensic Science International (FSI)*, 231:178–189, 2013.

[23] Zanoni Dias, Anderson Rocha, and Siome Goldenstein. First steps toward image phylogeny. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2010.

[24] Zanoni Dias, Anderson Rocha, and Siome Goldenstein. Video phylogeny: Recovering near-duplicate video relationships. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2011.

[25] Zanoni Dias, Anderson Rocha, and Siome Goldenstein. Image phylogeny by minimal spanning trees. *IEEE Transactions on Information Forensics and Security (TIFS)*, 7(2):774–788, 2012.

[26] Jack Edmonds. Optimum branchings. *Journal of Research of National Institute of Standards and Technology*, 71B:48–50, 1967.

[27] Zhigang Fan and Ricardo L. De Queiroz. Identification of bitmap compression history: Jpeg detection and quantizer estimation. *IEEE Transactions on Image Processing*, 12(2):230–235, 2003.

[28] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *ACM Communications*, 24(6):381–395, 1981.

[29] Jessica Fridrich, David Soukal, and Jan Lukas. Detection of copy-move forgery in digital images. In *Digital Forensics Research Conference*, pages 134 –137, 2003.

[30] Siome Goldenstein and Anderson Rocha. High-profile forensic analysis of images. In *International Conference on Imaging for Crime Detection and Prevention (ICDP)*, pages 1–6. Citeseer, 2009.

[31] Rafael Gonzalez and Richard Woods. *Digital Image Processing.* Prentice-Hall, 3 edition, 2007.

[32] Ardeshir A. Goshtasby. *Image Registration: principles, tools and methods*. Advances in Computer Vision and Pattern Recognition - Springer, 1 edition, 2012.

[33] Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[34] Alejandro Jaimes, Shih-Fu Chang, and Alexander C. Loui. Duplicate detection in consumer photography and news video. In *ACM Workshop on Multimedia and Security*, pages 423–424, 2002.

[35] Alexis Joly, Olivier Buisson, and Carl Frélicot. Content-based copy retrieval using distortion-based probabilistic similarity search. *IEEE Transactions on Multimedia*, 9(2):293–306, 2007.

[36] Yan Ke, Rahul Sukthankar, and Larry Huston. Efficient near-duplicate detection and subimage retrieval. In *ACM Workshop on Multimedia and Security*, pages 869–876, 2004.

[37] Lyndon Kennedy and Shih-Fu Chang. Internet image archaeology: Automatically tracing the manipulation history of photographs on the web. In *ACM International Conference of Multimedia*, pages 349–358, 2008.

[38] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[39] Silvia Lameri, Paolo Bestagini, Ambra Melloni, Simone Milani, Anderson Rocha, Marco Tagliasacchi, and Stefano Tubaro. Who is my parent? Reconstructing video sequences from partially matching shots. In *IEEE International Conference on Image Processing (ICIP)*, pages 5342–5346, 2014.

[40] Sunil Lee, Chang D. Yoo, and Ton Kalker. Robust video fingerprinting based on symmetric pairwise boosting. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(9):1379–1388, 2009.

[41] Benjamin Lewin. *Genes VI.* Oxford University Press, 1997.

[42] Jan Lukáš, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security (TIFS)*, 1(2):205–214, 2006.

[43] James G. MacKinnon. *Numerical distribution functions for unit root and cointegration tests.* Institute for Economic Research, Queen's University, 1995.

[44] Frederik Maes, Andre Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens. Multimodality image registration by maximization of mutual information. *Medical Imaging, IEEE Transactions on*, 16(2):187–198, 1997.

[45] Junwen Mao, Orhan Bulan, Gaurav Sharma, and Suprakash Datta. Device temporal forensics: an information theoretic approach. In *IEEE International Conference on Image Processing*, pages 1485–1488, 2009.

[46] Yannick Maret. *Efficient Duplicate Detection Based on Image Analysis.* Ph.D. thesis, École Polytechinique Fédérale de Lausanne, 2007.

[47] Ambra Melloni, Paolo Bestagini, Simone Milani, Marco Tagliasacchi, Anderson Rocha, and Stefano Tubaro. Image phylogeny through dissimilarity metrics fusion. In *IEEE European Workshop on Visual Information Processing (EUVIP)*, pages 1–6, 2014.

[48] Ambra Melloni, Silvia Lameri, Paolo Bestagini, Marco Tagliasacchi, and Stefano Tubaro. Near-duplicate detection and alignment for multi-view videos. In *IEEE International Conference on Image Processing (ICIP)*, pages 1–4, 2015.

[49] M. Kivanc Mihcak, Igor Kozintsev, Kannan Ramchandran, and Pierre Moulin. Low-complexity image denoising based on statistical modeling of wavelet coefficients. *IEEE Signal Processing Letters*, 6(12):300–303, 1999.

[50] Simoni Milani, Marco Fontana, Paolo Bestagini, and Stefano Tubaro. Phylogenetic analysis of near-duplicate images using processing age metrics. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2054–2058. IEEE, 2016.

[51] Matteo Nucci, Marco Tagliasacchi, and Stefano Tubaro. A phylogenetic analysis of near-duplicate audio tracks. In *IEEE International Workshop on Multimedia Signal Processing*, pages 99–104, 2013.

[52] Marina Oikawa, Zanoni Dias, Anderson Rocha, and Siome Goldenstein. Manifold learning and spectral clustering for image phylogeny forests. *IEEE Transactions on Information Forensics and Security*, 11(1):5–18, 2016.

[53] Alberto Oliveira, Pasquale Ferrara, Alessia De Rosa, Alessandro Piva, Mauro Barni, Siome Goldenstein, Zanoni Dias, and Anderson Rocha. Multiple parenting identification in image phylogeny. In *IEEE International Conference on Image Processing (ICIP)*, pages 5347–5351, 2014.

[54] Alberto Oliveira, Pasquale Ferrara, Alessia De Rosa, Alessandro Piva, Mauro Barni, Siome Goldenstein, Zanoni Dias, and Anderson Rocha. Multiple parenting phylogeny relationships in digital images. *IEEE Transactions on Information Forensics and Security (TIFS)*, 11(2):328–343, 2016.

[55] Job C. Oostveen, Ton Kalker, and Jaap Haitsma. Visual hashing of digital video: applications and techniques. In *International Symposium on Optical Science and Technology*, pages 121–131. International Society for Optics and Photonics, 2001.

[56] Xunyu Pan, Xing Zhang, and Siwei Lyu. Exposing image splicing with inconsistent local noise variances. In *IEEE International Conference on Computational Photography*, pages 1 – 10, 2012.

[57] Robert C. Prim. Shortest connection networks and some generalizations. *Bell system technical journal*, 36(6):1389–1401, 1957.

[58] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics Applications*, 21:34–41, 2001.

[59] Alessia De Rosa, Francesca Uccheddu, Andrea Costanzo, Alessandro Piva, and Mauro Barni. Exploring image dependencies: a new challenge in image forensics. *SPIE Media Forensics and Security*, 7541(2):1 – 12, 2010.

[60] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[61] Frederik Schaffalitzky and Andrew Zisserman. Multi-view matching for unordered image sets, or "how do I organize my holiday snaps?". In *European Conference on Computer Vision*, pages 414–431, 2002.

[62] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[63] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[64] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Jiebo Luo. Effective multiple feature hashing for large-scale near-duplicate video retrieval. *IEEE Transactions on Multimedia*, 15(8):1997–2008, 2013.

[65] Juan E. Tapia and Claudio A. Perez. Gender classification based on fusion of different spatial scale features selected by mutual information from histogram of LBP, intensity, and shape. *IEEE Transactions on Information Forensics and Security (TIFS)*, 8(3):488–499, 2013.

[66] Robert E. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

[67] Giuseppe Valenzise, Marco Tagliasacchi, and Stefano Tubaro. Estimating QP and motion vectors in H. 264/AVC video from decoded pixels. In *ACM Workshop on Multimedia in Forensics, Security and Intelligence*, pages 89–92, 2010.

[68] Paul Viola and William M. Wells. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24:137–154, 1997.

[69] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, pages 80–83, 1945.

[70] Zhihua Xu, Hefei Ling, Fuhan Zou, Zhengding Lu, and Ping Li. Robust image copy detection using multi-resolution histogram. In *ACM International Conference on Multimedia Information Retrieval*, pages 129–136, 2010.

[71] Dong-Qing Zhang and Shih-Fu Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *ACM Workshop on Multimedia and Security*, pages 877–884, 2004.

[72] John R. Zhang, Jennifer Y. Ren, Fangzhe Chang, Thomas L. Wood, and John R. Kender. Fast near-duplicate video retrieval via motion time series matching. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 842–847, 2012.

[73] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.

[74] Daniel Zoran and Yair Weiss. Scale invariance and noise in natural images. In *IEEE International Conference on Computer Vision*, pages 2209 – 2216, 2009.

Appendix A

Results for all possible fusions

In this appendix, we present the results of our experiments for image phylogeny forest reconstruction, considering the combinations between two methods in comparison to the fusion with three methods.

For a better comparison among the fusion results, Table A.1 presents the results for all possible combinations of AOK, AOB, and E-AOB. Note that the results of the fusion with the three methods are more effective than the other combinations between two methods.

Table A.1: Results for all possible fusions

(a) Semantically similar images from OC

Dataset A

	Fusion (AOK×AOB×E-AOB)				**Fusion (AOK × AOB)**				**Fusion (AOK × E-AOB)**				**Fusion (AOB × E-AOB)**			
$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.917	0.811	0.841	0.777	0.893	0.811	0.841	0.774	0.916	0.812	0.842	0.779	0.895	0.809	0.839	0.770
3	0.943	0.840	0.850	0.833	0.929	0.841	0.850	0.831	0.938	0.842	0.851	0.833	0.931	0.838	0.846	0.828
4	0.926	0.843	0.838	0.848	0.904	0.843	0.838	0.843	0.920	0.846	0.840	0.849	0.908	0.839	0.833	0.840
5	0.932	0.794	0.831	0.789	0.923	0.795	0.832	0.790	0.931	0.796	0.833	0.791	0.927	0.792	0.829	0.786

Dataset B

$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.867	0.884	0.879	0.862	0.850	0.883	0.878	0.856	0.860	0.885	0.879	0.860	0.850	0.881	0.876	0.854
3	0.900	0.900	0.896	0.886	0.873	0.898	0.892	0.877	0.894	0.901	0.896	0.885	0.873	0.897	0.892	0.876
4	0.901	0.905	0.898	0.889	0.867	0.902	0.894	0.878	0.895	0.905	0.898	0.888	0.867	0.901	0.895	0.877
5	0.893	0.905	0.897	0.879	0.848	0.901	0.891	0.865	0.888	0.905	0.895	0.878	0.850	0.900	0.893	0.865
6	0.883	0.905	0.894	0.873	0.841	0.901	0.889	0.860	0.879	0.905	0.893	0.873	0.842	0.901	0.890	0.860
7	0.870	0.906	0.892	0.865	0.832	0.902	0.887	0.854	0.867	0.906	0.891	0.865	0.832	0.902	0.888	0.854
8	0.837	0.911	0.894	0.853	0.797	0.907	0.889	0.841	0.834	0.911	0.893	0.853	0.802	0.907	0.890	0.842
9	0.818	0.911	0.893	0.838	0.772	0.907	0.888	0.824	0.812	0.911	0.892	0.837	0.781	0.907	0.889	0.826
10	0.796	0.909	0.890	0.820	0.749	0.905	0.886	0.805	0.789	0.910	0.890	0.817	0.762	0.905	0.886	0.810

(b) Semantically similar images from MC

Dataset A

	Fusion (AOK × AOB × E-AOB)				**Fusion (AOK × AOB)**				**Fusion (AOK × E-AOB)**				**Fusion (AOB × E-AOB)**			
$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.916	0.810	0.840	0.778	0.893	0.810	0.839	0.773	0.914	0.811	0.841	0.777	0.897	0.809	0.838	0.771
3	0.944	0.840	0.848	0.837	0.926	0.841	0.848	0.833	0.943	0.843	0.851	0.838	0.936	0.837	0.843	0.831
4	0.932	0.841	0.835	0.851	0.903	0.841	0.834	0.844	0.931	0.843	0.837	0.853	0.907	0.837	0.830	0.842
5	0.943	0.793	0.833	0.798	0.923	0.794	0.833	0.794	0.940	0.796	0.834	0.799	0.926	0.791	0.830	0.792

Dataset B

$\lvert F\rvert$	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
2	0.871	0.889	0.871	0.869	0.851	0.887	0.869	0.860	0.867	0.890	0.872	0.868	0.846	0.884	0.866	0.859
3	0.900	0.898	0.880	0.884	0.871	0.894	0.878	0.873	0.895	0.898	0.882	0.883	0.866	0.893	0.875	0.872
4	0.905	0.899	0.879	0.891	0.872	0.895	0.876	0.879	0.904	0.900	0.881	0.891	0.871	0.894	0.875	0.878
5	0.916	0.900	0.882	0.891	0.872	0.896	0.878	0.878	0.914	0.901	0.883	0.891	0.869	0.895	0.877	0.877
6	0.921	0.901	0.883	0.895	0.875	0.896	0.877	0.880	0.919	0.901	0.883	0.894	0.874	0.896	0.877	0.880
7	0.928	0.905	0.885	0.900	0.881	0.900	0.879	0.886	0.924	0.905	0.885	0.900	0.885	0.900	0.879	0.887
8	0.931	0.910	0.888	0.904	0.886	0.905	0.883	0.891	0.927	0.910	0.888	0.904	0.885	0.905	0.882	0.890
9	0.926	0.911	0.891	0.900	0.875	0.906	0.886	0.885	0.920	0.911	0.891	0.899	0.877	0.906	0.885	0.884
10	0.917	0.910	0.891	0.890	0.863	0.905	0.886	0.875	0.912	0.910	0.891	0.890	0.867	0.905	0.885	0.874

Appendix B

Other combinations of dissimilarity methods

In this appendix, we present the other possible combinations of the methods discussed on Chapter 4 and the comparison between all the methods and the best method HGMI with Wilcoxon signed-rank test. The results are presented in Tables B.1 and B.2. Note that none of them is more effective than the ones presented and discussed in Chapter 4, Section 4.3.1.

Table B.1: Results of the phylogeny reconstruction, considering the color matching algorithm based on the average and standard deviation analysis. P-values obtained with the Wilcoxon Signed Rank test, comparing the methods to HGMI.

	Multiple Cameras (MC)				One Camera (OC)			
	MSE				MSE			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.707	0.819	0.777	0.748	0.696	0.812	0.784	0.740
2	0.830	0.875	0.856	0.844	0.833	0.872	0.864	0.841
3	0.876	0.891	0.875	0.872	0.885	0.896	0.891	0.879
4	0.890	0.896	0.876	0.884	0.892	0.903	0.897	0.884
5	0.903	0.897	0.879	0.887	0.889	0.903	0.896	0.878
6	0.911	0.898	0.880	0.890	0.879	0.904	0.894	0.872
7	0.917	0.903	0.883	0.896	0.868	0.905	0.891	0.864
8	0.922	0.908	0.886	0.901	0.841	0.910	0.894	0.855
9	0.915	0.909	0.889	0.896	0.822	0.910	0.892	0.840
10	0.909	0.908	0.889	0.886	0.802	0.908	0.890	0.823
P-value	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	GRAD				GRAD			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.609	0.783	0.791	0.690	0.580	0.772	0.791	0.668
2	0.633	0.817	0.834	0.732	0.621	0.809	0.838	0.725
3	0.668	0.836	0.848	0.760	0.678	0.838	0.860	0.766
4	0.688	0.838	0.845	0.758	0.698	0.844	0.859	0.765
5	0.708	0.841	0.846	0.758	0.711	0.843	0.854	0.756
6	0.703	0.834	0.836	0.734	0.704	0.837	0.843	0.734
7	0.691	0.829	0.828	0.715	0.687	0.828	0.830	0.707
8	0.680	0.826	0.818	0.700	0.661	0.819	0.818	0.676
9	0.667	0.820	0.812	0.679	0.650	0.815	0.812	0.658
10	0.652	0.812	0.804	0.654	0.630	0.810	0.805	0.634
P-value	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	MINF				MINF			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.839	0.890	0.871	0.851	0.818	0.874	0.861	0.831
2	0.940	0.938	0.928	0.926	0.901	0.930	0.929	0.903
3	0.953	0.945	0.933	0.932	0.881	0.935	0.932	0.897
4	0.936	0.942	0.930	0.921	0.839	0.933	0.927	0.871
5	0.928	0.941	0.930	0.913	0.801	0.932	0.924	0.844
6	0.909	0.941	0.930	0.902	0.730	0.932	0.922	0.808
7	0.895	0.943	0.930	0.894	0.681	0.932	0.920	0.777
8	0.883	0.946	0.929	0.887	0.637	0.935	0.922	0.758
9	0.850	0.946	0.929	0.865	0.583	0.934	0.921	0.719
10	0.828	0.945	0.928	0.847	0.539	0.933	0.918	0.687
P-value	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	GRMI				GRMI			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.889	0.923	0.902	0.897	0.900	0.926	0.913	0.900
2	0.960	0.960	0.952	0.952	0.956	0.960	0.958	0.950
3	0.967	0.963	0.953	0.955	0.950	0.960	0.956	0.948
4	0.964	0.960	0.950	0.951	0.931	0.958	0.953	0.935
5	0.962	0.961	0.952	0.949	0.916	0.959	0.952	0.928
6	0.958	0.961	0.951	0.948	0.882	0.959	0.951	0.911
7	0.954	0.963	0.952	0.947	0.853	0.959	0.951	0.895
8	0.943	0.963	0.951	0.940	0.820	0.960	0.951	0.880
9	0.931	0.964	0.951	0.932	0.801	0.960	0.951	0.866
10	0.916	0.964	0.952	0.923	0.768	0.960	0.951	0.843
P-value	0.2324	0.1602	0.009766	0.04883	0.02734	0.1055	0.06445	0.01953

Table B.2: Results of the phylogeny reconstruction, considering the color matching algorithm based on histogram matching. P-values obtained with Wilcoxon Signed Rank test, comparing the methods to HGMI.

	Multiple Cameras (MC)				One Camera (OC)			
	HMSE				HMSE			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.716	0.810	0.806	0.747	0.663	0.795	0.803	0.717
2	0.823	0.864	0.857	0.832	0.837	0.862	0.872	0.836
3	0.857	0.875	0.862	0.856	0.895	0.882	0.887	0.873
4	0.876	0.879	0.862	0.869	0.905	0.886	0.884	0.882
5	0.891	0.879	0.865	0.873	0.908	0.883	0.879	0.877
6	0.903	0.884	0.869	0.881	0.906	0.884	0.877	0.877
7	0.909	0.888	0.870	0.887	0.905	0.885	0.875	0.876
8	0.915	0.892	0.871	0.892	0.901	0.890	0.877	0.878
9	0.919	0.894	0.875	0.891	0.895	0.889	0.878	0.873
10	0.922	0.891	0.875	0.888	0.884	0.888	0.877	0.863
P-value	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	HGRAD				HGRAD			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.585	0.769	0.777	0.671	0.560	0.759	0.783	0.648
2	0.618	0.808	0.829	0.721	0.612	0.806	0.839	0.719
3	0.651	0.828	0.843	0.751	0.665	0.829	0.854	0.753
4	0.675	0.833	0.842	0.757	0.686	0.836	0.852	0.756
5	0.697	0.837	0.844	0.756	0.696	0.836	0.848	0.750
6	0.694	0.830	0.836	0.735	0.691	0.829	0.835	0.728
7	0.685	0.825	0.829	0.718	0.672	0.818	0.823	0.698
8	0.673	0.822	0.819	0.700	0.650	0.811	0.812	0.671
9	0.659	0.816	0.812	0.677	0.639	0.807	0.806	0.653
10	0.645	0.808	0.806	0.653	0.630	0.805	0.803	0.638
P-value	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	HMINF				HMINF			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.824	0.893	0.873	0.854	0.799	0.884	0.869	0.841
2	0.929	0.939	0.928	0.923	0.902	0.933	0.934	0.909
3	0.949	0.946	0.934	0.934	0.901	0.939	0.939	0.911
4	0.946	0.946	0.934	0.929	0.882	0.940	0.936	0.896
5	0.943	0.945	0.935	0.924	0.858	0.939	0.932	0.878
6	0.939	0.945	0.936	0.923	0.817	0.939	0.932	0.857
7	0.927	0.948	0.937	0.918	0.781	0.940	0.930	0.835
8	0.925	0.951	0.937	0.915	0.747	0.942	0.930	0.820
9	0.908	0.951	0.938	0.904	0.696	0.941	0.930	0.786
10	0.890	0.949	0.937	0.889	0.664	0.940	0.927	0.761
P-value	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	HGMI				HGMI			
#TREE	**Roots**	**Edges**	**Leaves**	**Ancestry**	**Roots**	**Edges**	**Leaves**	**Ancestry**
1	0.872	0.925	0.908	0.893	0.856	0.920	0.909	0.884
2	0.947	0.958	0.951	0.946	0.950	0.960	0.960	0.950
3	0.961	0.961	0.952	0.953	0.958	0.961	0.960	0.954
4	0.964	0.961	0.952	0.954	0.955	0.961	0.957	0.950
5	0.967	0.961	0.953	0.955	0.952	0.961	0.955	0.948
6	0.968	0.962	0.953	0.956	0.940	0.961	0.954	0.943
7	0.972	0.964	0.955	0.959	0.934	0.961	0.954	0.940
8	0.974	0.965	0.954	0.959	0.924	0.962	0.954	0.937
9	0.971	0.966	0.955	0.957	0.917	0.962	0.954	0.932
10	0.958	0.965	0.956	0.949	0.906	0.962	0.954	0.922
Best Approach	–	–	–	–	–	–	–	–