Inexact graph matching : application to 2D and 3D Pattern Recognition

Kamel Madi

▶ **To cite this version:**

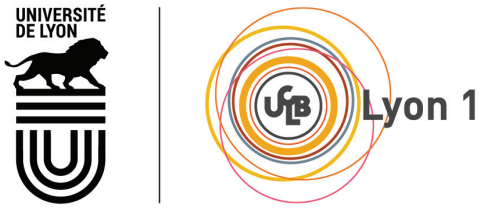
Kamel Madi. Inexact graph matching : application to 2D and 3D Pattern Recognition. Computer Vision and Pattern Recognition [cs.CV]. Université de Lyon, 2016. English. NNT : 2016LYSE1315 . tel-01493118

HAL Id: tel-01493118
https://theses.hal.science/tel-01493118

Submitted on 21 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



N° d'ordre NNT : 2016LYSE1315

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée au sein de

l'Université Claude Bernard Lyon 1

École Doctorale ED 512
Informatique et Mathématiques (InfoMaths)

Spécialité de doctorat : Informatique

Soutenue publiquement le 13/12/2016, par :

Kamel MADI

Inexact graph matching: application to $2D$ and $3D$ Pattern Recognition

---

Devant le jury composé de :

Amel Bouzeghoub, Professeur, Télécom SudParis	Présidente, Examinatrice
Daniela Grigori, Professeur, Université Paris Dauphine	Rapporteure
Olivier Togni, Professeur, Université de Bourgogne	Rapporteur
Hamamache Kheddouci, Professeur, Université Lyon 1	Directeur de thèse
Hamida Seba, Maitre de Conférences HDR, Université Lyon 1	Co-directeur de thèse
Charles-Edmond Bichot, MCF, Ecole Centrale de Lyon	Co-directeur de thèse
Eric Paquet, Professeur, National Research Council Canada	Invité
Olivier Barge, Ingénieur de recherche, Archéorient	Invité

UNIVERSITE CLAUDE BERNARD - LYON 1

Président de l'Université	**M. le Professeur Frédéric FLEURY**
Président du Conseil Académique	M. le Professeur Hamda BEN HADID
Vice-président du Conseil d'Administration	M. le Professeur Didier REVEL
Vice-président du Conseil Formation et Vie Universitaire	M. le Professeur Philippe CHEVALIER
Vice-président de la Commission Recherche	M. Fabrice VALLÉE
Directrice Générale des Services	Mme Dominique MARCHAND

COMPOSANTES SANTE

Faculté de Médecine Lyon Est – Claude Bernard	Directeur : M. le Professeur G.RODE
Faculté de Médecine et de Maïeutique Lyon Sud – Charles Mérieux	Directeur : Mme la Professeure C. BURILLON
Faculté d'Odontologie	Directeur : M. le Professeur D. BOURGEOIS
Institut des Sciences Pharmaceutiques et Biologiques	Directeur : Mme la Professeure C. VINCIGUERRA
Institut des Sciences et Techniques de la Réadaptation	Directeur : M. X. PERROT
Département de formation et Centre de Recherche en Biologie Humaine	Directeur : Mme la Professeure A-M. SCHOTT

COMPOSANTES ET DEPARTEMENTS DE SCIENCES ET TECHNOLOGIE

Faculté des Sciences et Technologies	Directeur : M. F. DE MARCHI
Département Biologie	Directeur : M. le Professeur F. THEVENARD
Département Chimie Biochimie	Directeur : Mme C. FELIX
Département GEP	Directeur : M. Hassan HAMMOURI
Département Informatique	Directeur : M. le Professeur S. AKKOUCHE
Département Mathématiques	Directeur : M. le Professeur G. TOMANOV
Département Mécanique	Directeur : M. le Professeur H. BEN HADID
Département Physique	Directeur : M. le Professeur J-C PLENET
UFR Sciences et Techniques des Activités Physiques et Sportives	Directeur : M. Y.VANPOULLE
Observatoire des Sciences de l'Univers de Lyon	Directeur : M. B. GUIDERDONI
Polytech Lyon	Directeur : M. le Professeur E.PERRIN
Ecole Supérieure de Chimie Physique Electronique	Directeur : M. G. PIGNAULT
Institut Universitaire de Technologie de Lyon 1	Directeur : M. le Professeur C. VITON
Ecole Supérieure du Professorat et de l'Education	Directeur : M. le Professeur A. MOUGNIOTTE
Institut de Science Financière et d'Assurances	Directeur : M. N. LEBOISNE

To my mother and father.
To my sisters and brothers.
To my fiancee.

Acknowledgments

It would not have been possible to achieve this thesis and write this manuscript without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

First of all, I wish to express my deepest gratitude to my advisors, Prof. Hamamache Kheddouci and Dr. Hamida Seba, for their valuable guidance, caring, patience and consistent encouragement throughout this research work. I thank them for giving me an interesting research topic related to graph matching field. I thank them also for continuous support and encouragement on both my research and other matters of life during my PhD.

I'm very thankful to Prof. Eric Paquet for introducing me to the field of 3D Pattern Recognition. I thank him for receiving me as visiting researcher at Ottawa University and National Research Council Canada during six months. I thank him for the knowledge and skills he imparted through our collaboration. He was always willing to help and give his best suggestions. Working with him is so enjoyable. I also appreciate his help to improve the quality of my dissertation. I also thank him for being a member of my PhD jury and for the time he spent to examine my thesis.

I'm also very thankful to all the members of KITE project. I thank Mr. Olivier Barge and all the archeologists' team of the Archéorient laboratory, for their expertise, help and collaboration which were necessary to realize and achieve our work on Kite recognition in satellite images. I'm also thankful to Dr. Charles-Edmond Bichot for his collaboration.

I'm also very thankful to all the members of my PhD jury. Prof. Daniela Grigori and Prof. Olivier Togni for the time they spent for reviewing my thesis manuscript, and for their valuable feedback, comments and suggestions. Prof. Amel Bouzeghoub for spending her valuable time as my thesis examiner.

I'm also very thankful to Prof. Mohand-Said Hacid, the head of LIRIS laboratory, for his help, availability and valuable guidance.

I would like to thank my parents, my sisters, my brothers, my fiancee and all my family for their continuous moral support, understanding, patience and encouragement with their best wishes. Their love accompanies me wherever I go. Without them, this thesis would not have been possible. I'm very grateful to my mother and father, for all their love, support, and patience over the years. I'm also very thankful to my future wife. Words cannot describe how lucky I am to have her in my life.

I would like also to thank my friends, from Bejaia (Algeria), Lyon, Paris and Marseille (France), Ottawa (Canada) and everywhere. I prefer to not mention names, since the list is very long. I thank them for their continuous moral support and encouragement. I would also like to thank my friends and colleagues form LIRIS Laboratory and Lyon 1 University with whom I spent wonderful moments during my thesis.

I am greatly indebted to all the teachers and professors, in Béjaia (Algeria), Marseille, Montpellier, Paris, Lyon (France) and Ottawa (Canada), from whom I have learned over the years.

I wish, through this thesis, to pay tribute to my cousin Ghilas Madi, who left us very younger at the age of 21 years, at February 14th, 2016. He was a wonderful and very respectful guy and a brilliant student in Computer Science.

Finally, I would like to dedicate this thesis dissertation to my parents, my sisters, my brothers, my fiancee and all my family.

Abstract: Graphs are powerful mathematical modeling tools used in various fields of computer science, in particular, in Pattern Recognition. Graph matching is the main operation in Pattern Recognition using graph-based approach. Finding solutions to the problem of graph matching that ensure optimality in terms of accuracy and time complexity is a difficult research challenge and a topical issue. In this thesis, we investigate the resolution of this problem in two fields: $2D$ and $3D$ Pattern Recognition. Firstly, we address the problem of geometric graphs matching and its applications on $2D$ Pattern Recognition. Kite (archaeological structures) recognition in satellite images is the main application considered in this first part. We present a complete graph based framework for Kite recognition on satellite images. We propose mainly two contributions. The first one is an automatic process transforming Kites from real images into graphs and a process of generating randomly synthetic Kite graphs. This allowing to construct a benchmark of Kite graphs (real and synthetic) structured in different level of deformations. The second contribution in this part, is the proposition of a new graph similarity measure adapted to geometric graphs and consequently for Kite graphs. The proposed approach combines graph invariants with a geometric graph edit distance computation. Secondly, we address the problem of deformable $3D$ objects recognition, represented by graphs, *i.e.*, triangular tessellations. We propose a new decomposition of triangular tessellations into a set of substructures that we call *triangle-stars*. Based on this new decomposition, we propose a new algorithm of graph matching to measure the distance between triangular tessellations. The proposed algorithm offers a better measure by assuring a minimum number of triangle-stars covering a larger neighbourhood, and uses a set of descriptors which are invariant or at least oblivious under most common deformations. Finally, we propose a more general graph matching approach founded on a new formalization based on the stable marriage problem. The proposed approach is optimal in term of execution time, *i.e.* the time complexity is quadratic $\mathcal{O}(n^2)$ and flexible in term of applicability ($2D$ and $3D$). The analyze of the time complexity of the proposed algorithms and the extensive experiments conducted on Kite graph data sets (real and synthetic) and standard data sets (2D and 3D) attest the effectiveness, the high performance and accuracy of the proposed approaches and show that the proposed approaches are extensible and quite general.

Keywords: Graph matching, Graph edit distance, Graph decomposition, Graph based modelling, Pattern recognition, $3D$ object recognition, Deformable object recognition, Kites, Satellite image.

Résumé: Les Graphes sont des structures mathématiques puissantes constituant un outil de modélisation universel utilisé dans différents domaines de l'informatique, notamment dans le domaine de la reconnaissance de formes. L'appariement de graphes est l'opération principale dans le processus de la reconnaissance de formes à base de graphes. Dans ce contexte, trouver des solutions d'appariement de graphes, garantissant l'optimalité en termes de précision et de temps de calcul est un problème de recherche difficile et d'actualité. Dans cette thèse, nous nous intéressons à la résolution de ce problème dans deux domaines : la reconnaissance de formes $2D$ et $3D$. Premièrement, nous considérons le problème d'appariement de graphes géométriques et ses applications sur la reconnaissance de formes 2D. Dance cette première partie, la reconnaissance des Kites (structures archéologiques) est l'application principale considérée. Nous proposons un "framework" complet basé sur les graphes pour la reconnaissance des Kites dans des images satellites. Dans ce contexte, nous proposons deux contributions. La première est la proposition d'un processus automatique d'extraction et de transformation de Kites à partir d'images réelles en graphes et un processus de génération aléatoire de graphes de Kites synthétiques. En utilisant ces deux processus, nous avons généré un benchmark de graphes de Kites (réels et synthétiques) structuré en 3 niveaux de bruit. La deuxième contribution de cette première partie, est la proposition d'un nouvel algorithme d'appariement pour les graphes géométriques et par conséquent pour les Kites. L'approche proposée combine les invariants de graphes au calcul de l'édition de distance géométrique. Deuxièmement, nous considérons le problème de reconnaissance des formes 3D où nous nous intéressons à la reconnaissance d'objets déformables représentés par des graphes *c.à.d.* des tessellations de triangles. Nous proposons une décomposition des tessellations de triangles en un ensemble de sous structures que nous appelons triangle-étoiles. En se basant sur cette décomposition, nous proposons un nouvel algorithme d'appariement de graphes pour mesurer la distance entre les tessellations de triangles. L'algorithme proposé assure un nombre minimum de structures disjointes, offre une meilleure mesure de similarité en couvrant un voisinage plus large et utilise un ensemble de descripteurs qui sont invariants ou au moins tolérants aux déformations les plus courantes. Finalement, nous proposons une approche plus générale de l'appariement de graphes. Cette approche est fondée sur une nouvelle formalisation basée sur le problème de mariage stable. L'approche proposée

est optimale en terme de temps d'exécution, *c.à.d.* la complexité est quadratique $\mathcal{O}(n^2)$, et flexible en terme d'applicabilité (2D et 3D). Cette approche se base sur une décomposition en sous structures suivie par un appariement de ces structures en utilisant l'algorithme de mariage stable. L'analyse de la complexité des algorithmes proposés et l'ensemble des expérimentations menées sur les bases de graphes des Kites (réelle et synthétique) et d'autres bases de données standards ($2D$ et $3D$) attestent l'efficacité, la haute performance et la précision des approches proposées et montrent qu'elles sont extensibles et générales.

Mots clés: Appariement de graphes, Distance d'édition de graphes, Décomposition de graphes, Modélisation à base de graphes, Reconnaissance de formes, Reconnaissance d'objets 3D, Reconnaissance d'objets déformables, Kites, Images satellites.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

General Introduction

Graph theory is an important mathematical field. Its origin dates back to 1735 when the Swiss mathematician Leonard Euler solved the problem of the Seven Bridges of Königsberg, called also, the *Königsberg bridge problem* [6]. The city of Königsberg in Prussia (now Kaliningrad) was built on both sides of the Pregel River, and included two islands connected to each other and the mainland by seven bridges (See Figure 1.1). The problem was to find a continuous tour through the city of Königsberg that would cross each bridge exactly once, come back at the same point from which it began. The Swiss mathematician, Leonard Euler, demonstrated that no such tour was possible. He gave an abstract model of the problem by representing land masses with points and bridges with links between pairs of points (See Figure 1.2). This abstract description of the problem was the introduction to the graph notion, and its solution is often referred as the first theorem in graph theory [6].

Graphs are useful and powerful mathematical tools allowing both the description of properties of an object and the relationships between a set of objects. The object properties are described by means of nodes and the relationships between objects are represented by means of edges. In this context, graphs constitute an universal modeling tool and receive considerable attention from the whole scientific community, allowing their use in various fields of computer science, in particular the field of Pattern Recognition.

Pattern recognition describes the act allowing to determine the category or the class to which a given pattern belongs. The term "*pattern*" means an observation in the real world. Pattern recognition is one of the important capabilities of humans, with which intuitively we can recognize the face of a friend, a category of an animal, a written sentence, a spoken word, a car in the street, an object in a specific place, an image, etc. The aim of pattern recognition as a field of computer science is to propose algorithms allowing the imitation as possible of the human capacity of perception and recognition [7].

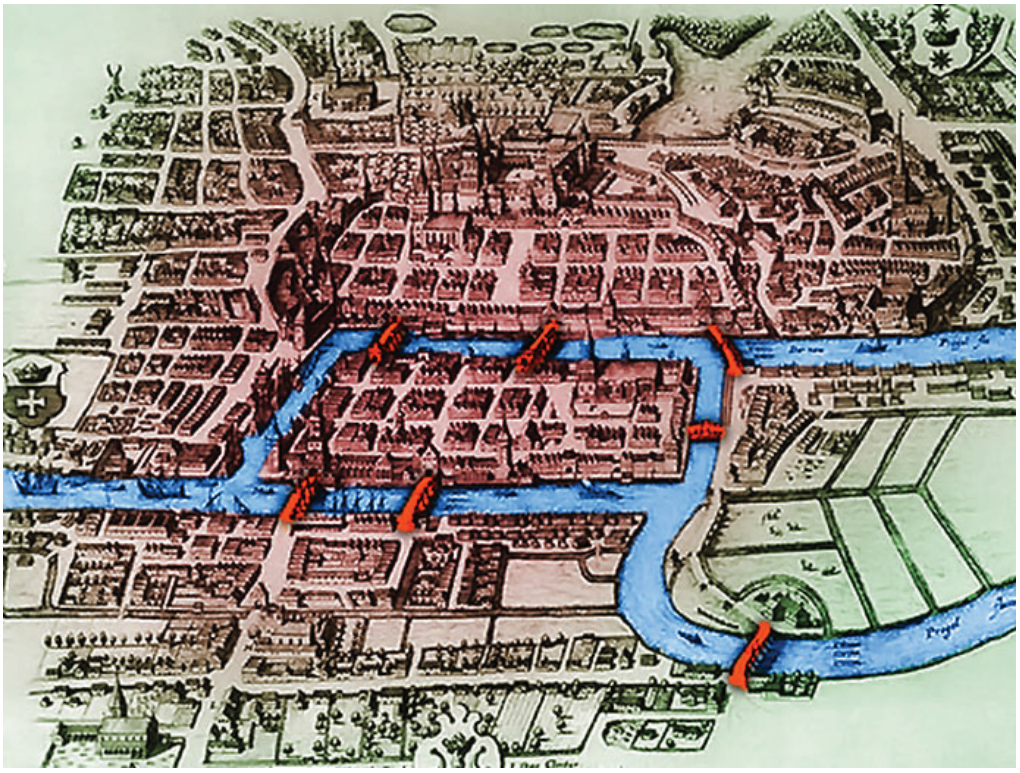


Figure 1.1: The Seven Bridges of Königsberg city [1].

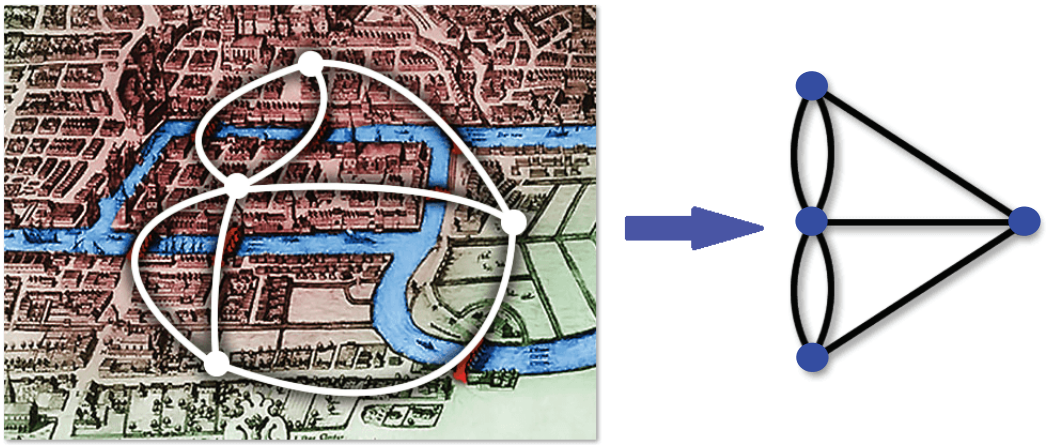


Figure 1.2: Graph representation of Königsberg bridge problem [1].

Over the years, the use of graphs in pattern recognition has gained popularity and has obtained a growing attention from the scientific community. Consequently, graph-based approaches are used in various fields [2], like 2D and 3D pattern recognition (*eg.*, face recognition, street recognition, rigid and non-rigid object recognition), document analysis (*eg.*, handwriting recognition, digit and symbols recog-

nition), biometric identification (*eg.*, fingerprint recognition), video analysis (*eg.*, human gesture recognition), ... , etc.

Graph matching and more generally graph comparison is the main operation in the process of pattern recognition using a graph-based approach. Graph matching is the process of finding a correspondence between vertices and edges of two graphs that satisfies a certain number of constraints ensuring that substructures in one graph are mapped to similar substructures in the other. Graph matching solutions are classified into two wide categories: exact approaches and inexact approaches.

In this thesis, Inexact graph matching and its applications for $2D$ and $3D$ Pattern Recognition are investigated. Thus, the thesis is divided into 8 chapters organized into two parts. Each part contains two chapters. First of all, in Chapter 2, we give some preliminaries and introduce some notations needed in the rest of the thesis. Then, we give in Chapter 3, an overview of the related work concerning graph matching and its applications in Pattern Recognition.

Part I: Geometric graph matching: application to 2D pattern recognition

In the field of Pattern Recognition, it is often required to compare objects, and the question how to represent those objects in a formal way is a fundamental issue. Graphs are popular and powerful mathematical modeling tools. Graph based techniques for pattern recognition aim to solve mainly two major problems. The first is to find an optimal way to represent the considered patterns by graphs. The second problem is to find the adequate method to compare the objects represented by graphs. In this context, finding solutions to the problem of graph modeling and graph matching that ensure optimality in terms of accuracy and time complexity is a difficult research challenge and a topical issue. The related work about graph modeling, graph matching and its applications in Pattern Recognition are detailed in Chapter 3. In this part, we address the issue of geometric graphs matching and its applications on 2D Pattern Recognition. Kite recognition in satellite images is the main application considered in this part. Kites are huge archaeological structures of stone visible from satellite images. Because of their important number and their wide geographical distribution, automatic recognition of these structures on images is an important step towards understanding these enigmatic remnants. In this part, we present a complete identification tool relying on a graph representation

of the Kites. As Kites are naturally represented by graphs, graph matching methods are thus the main building blocks in the Kite identification process. However, Kite graphs are disconnected geometric graphs for which traditional graph matching methods are useless. This part of this thesis is realized within the **KITE project**, consequently, Kite recognition in satellite images, is the main application of the geometric graph matching part. **KITE project** allowed us to collaborate with a team of archeologists expert on Kites. The archeologists provided us the satellite images (ground truth data) needed to the construction of the Kite database. They have also checked and validated the experiments steps that we realized and the obtained results. This part contains two chapters, in the first one (Chapter 4), we present the process of Kite graphs construction from real images, and the process of generating a synthetic data set of Kite graphs generated randomly. The two data sets (real and synthetic) are used to validate our algorithms. In the second one (Chapter 5), we propose a new graph similarity measure adapted to geometric graphs and consequently for Kite graphs. The proposed approach combines graph invariants with a geometric graph edit distance computation leading to an efficient Kite identification process. In Chapter 5, we analyze the time complexity of the proposed algorithms and conduct extensive experiments both on real and synthetic Kite graph data sets to attest the effectiveness of the approach. We also perform a set of experimentations on other data sets in order to show that the proposed approach is extensible and quite general.

Part II: Inexact graph matching for 3D objects recognition

Object recognition is one of the fundamental challenges in computer vision, which has been studied for more than four decades [8]. In the last years, there has an increasing interest on the 3D objects analysis. The high advances in different fields of technology and specially in the field of 3D, engender a high growing need of automated methods for 3D objects recognition. Using triangular tessellations, 3D objects may be compared with graph matching techniques. This part addresses the problem of comparing deformable or non-rigid 3D objects (such as human and animal bodies). The shapes considered are represented by graphs, *i.e.*, triangular tessellations. We propose a new distance for comparing deformable 3D objects. This distance is based on the decomposition of triangular tessellations into a set of substructures that we call *triangle-stars*. A triangle-star is a connected component formed by

the union of a triangle and its neighborhood. The proposed decomposition offers a parameterizable triangle-stars depending on the degree of the considered neighborhood. The number of triangle-stars obtained is much smaller than the number of nodes and the number of classic stars [9, 10] and, as a result, the computational complexity is reduced. Furthermore, triangle-stars are local structures that cover a larger neighborhood than classic stars decomposition [9, 10]. Consequently, the proposed dissimilarity measure assures an optimal approximation. This is justified by the fact that optimal methods are based on graph's global structures and, consequently, a larger local structure allows to be closer to the global one. The proposed approach uses a set of descriptors which are invariant or at least oblivious under most common deformations. This part contains two chapters, in the first one (Chapter 6), we present the proposed decomposition of triangular tessellations into triangle-stars. We describe the proposed distance (dissimilarity measure). We prove that the proposed distance is a pseudo-metric and we analyse its time complexity. In the second chapter (Chapter 7), we describe the experimentations that we undertook to evaluate our approach. We present the different databases that we use in our experiments, some state of the art shape-matching algorithms to compare with, the evaluation criteria and the experimental results. The analysis of the time complexity and our experimental results on three standard databases (TOSCA, SHREC09 and SHREC11) confirm the high performance and accuracy of our algorithm. The set of experimentations and the obtained results on SHREC09 database show that the proposed approach is efficient also for the 3D objects sub-matching, which prove that our method is extensible and quite general.

Finally, Chapter 8 concludes the thesis with a summary of the contributions and some suggestions for further research. One of the promising ideas that we project to realize in further research is described in this last chapter. We propose a more general graph matching approach founded on a new formalization based on the stable marriage problem [11]. The proposed approach is optimal in term of execution time, *i.e.* the time complexity is quadratic $\mathcal{O}(n^2)$. The proposed algorithm is flexible in term of applicability ($2D$ and $3D$).

The set of publications arising from this thesis are listed in Appendix A.

Chapter 2

Preliminary Notions

Contents

In this chapter, we first introduce some useful definitions related to graphs, and present a short overview of the notations used in this thesis. Secondly, we present some common concepts and typical classes of graphs. Finally, we present some aspects of graph matching. Definitions and notations related to a particular chapter can be found in the corresponding chapter. We refer the reader to ([12], [13], [14] and [15]) for more background information on graph theory.

2.1 Basic definitions

In this section, we introduce some useful definitions and notation relating to graphs.

Graph: A graph is mathematical construct that models a relationship between a set of items. The set of items represents a set of objects. A link between the two items represents a relationship between two objects. The items are called vertices or nodes and the links are called edges. Thus, a graph $G(V, E)$ is a set of nodes connected by a set of edges. Formally, a graph G is a four tuple $G = (V, E, \alpha, \beta)$, where V is a finite not empty set of nodes or vertices. $E \subseteq V \times V$ is the set of edges. $\alpha : V \to L_V$ is the node labelling function, $\beta : E \to L_E$ is the edge labelling function while L_V and L_E are the sets of labels associated with the nodes and edges respectively. The cardinality of the node set $V(G)$ is called the *order* of G, commonly denoted by $|V(G)|$. The cardinality of the edge set $E(G)$ is the *size* of G, commonly denoted by $|E(G)|$. A graph can be *directed* or *undirected*. In a

directed graph, edges are ordered pairs (u, v) connecting the *source* node u to the *target* node v, commonly denoted by $\overrightarrow{uv}$ and called *arcs*. In an undirected graph, edges are unordered pairs $\{u, v\}$ and connect the two nodes in both directions. The notation uv is used to indicate the edge $\{u, v\}$. In an undirected graph G, two distinct nodes u and v are *adjacent* or *neighbors* if there exists an edge $uv \in E(G)$ that connects them. An edge uv is said to be *incident* to the nodes u and v. Two edges are adjacent if they are incident to a same node (they share a node). A *loop* at a node, links the node to itself and makes it its own neighbor. A graph is *simple* if there is at most one edge between every two nodes.
In this thesis, unless it is specified, the graphs which are considered will be finite simple undirected graphs, having no loops.

Neighborhood and Degree: The set of all neighbors of a node v in a graph G, is denoted by $N(v)$. The number of neighbors of v is called the *degree* of v and it is denoted by $deg(v)$. A node v is an *isolated node* if $deg(v) = 0$, which means that v is without any neighbors. A node of degree one ($deg(v) = 1$) is called a *leaf* or a *pendant node*. The *minimum degree* of a graph G is $\delta(G) = min\{deg(v) : v \in V(G)\}$ and the *maximum degree* of a graph G is denoted by $\Delta(G) = max\{deg(v) : v \in V(G)\}$.

Path and Cycle: A *path* is in an undirected graph G defined as a sequence of nodes $(v_1, v_2, \ldots, v_k)$ such that each pair v_i, v_{i+1} is an edge in $E(G)$. A path is called *simple* if all its nodes are distinct. Figure 2.1 shows an example of a simple graph *path*. A *cycle* is in an undirected graph G defined as a sequence of nodes $(v_0, v_1, \ldots, v_k)$ such that the set of edges $E(G)$ is $v_i v_{i+1}$ and the edge $v_k v_0$, where $i \in \{0, \ldots, k-1\}$. In other words, a *cycle* is a closed path staring and finishing with the same node. Figure 2.2 shows an example of a *cycle*.

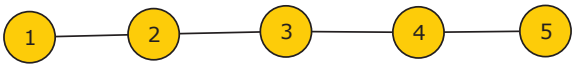


Figure 2.1: Example of a graph path

Connectivity and Clique: A graph G is *connected* if there exists a path between any two distinct vertices of G (see Figure 2.3). Otherwise, the graph G is disconnected. A *clique* in a graph G is a subset S of $V(G)$ such that every two nodes in

S are adjacent.

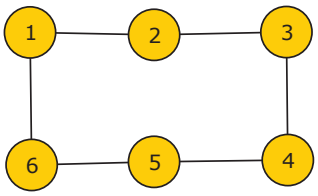


Figure 2.2: Example of a cycle

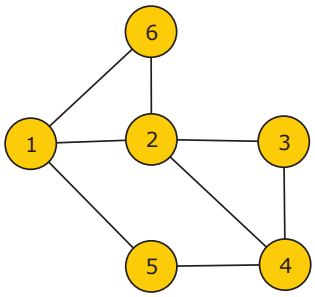


Figure 2.3: Example of a connected graph

Subgraph and Supergraph: A graph H is called a *subgraph* of a graph G, written as $H \subseteq G$, if every node of the graph H is also a node of the graph G and every edge of the graph H is an edge of the graph G. Formally, $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Which means also that the graph G is a *supergraph* of the graph H.

2.2 Some special graphs

Several graph classes have been defined and considered in the graph theory literature, in order to model specific problems or to take advantage of the theoretical properties of those classes. In this section, we present some typical and important classes that will be considered in this thesis.

Planar graph: A graph is *planar* if it can be drawn in the plane without any edges crossing, which means that edges intersect only at their common nodes. Figure 2.4 illustrates an example of a *planar* graph.

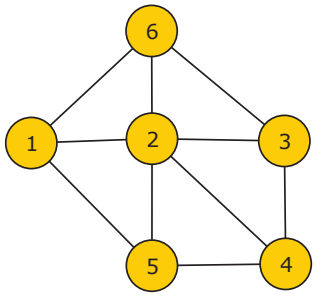


Figure 2.4: Example of a planar graph

Bipartite graph: A graph $G = (V, E)$ is *bipartite* if its set of nodes V can be split into two disjoint subsets V_1 and V_2 such that every edge of E connects a node in V_1 to another node in V_2. Figure 2.5 illustrates an example of a *bipartite* graph.

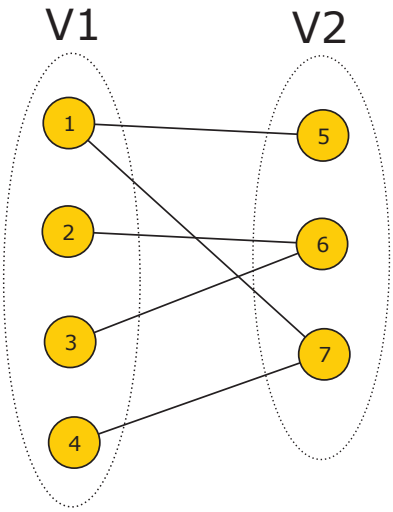


Figure 2.5: Example of a bipartite graph

Tree: A *tree* T is a connected graph which has not any cycles. An Example is given in Figure 2.6. A tree can be *rooted* or *unrooted*. A rooted tree is a tree in which one vertex has been distinguished as the *root*.

Star: A star S is a labelled, single-level, tree which can be represented by a 3-tuple $S = (r, L, l)$, where r is the root node, L is the set of leaves (mono-degree nodes) and l is a labelling function. Edges exist only between the root node r and any nodes in L [10]. Figure 2.7 illustrates an example of a *star*. Figure 2.8 shows the obtained stars from a given graph G.

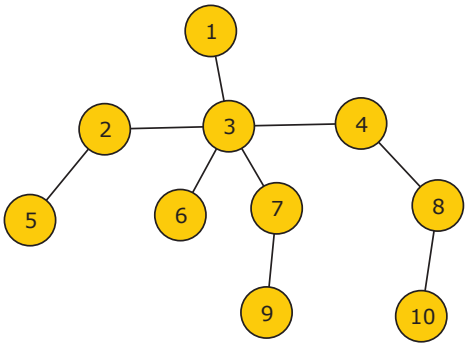


Figure 2.6: Example of a tree.

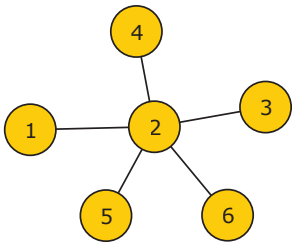


Figure 2.7: Example of a star.

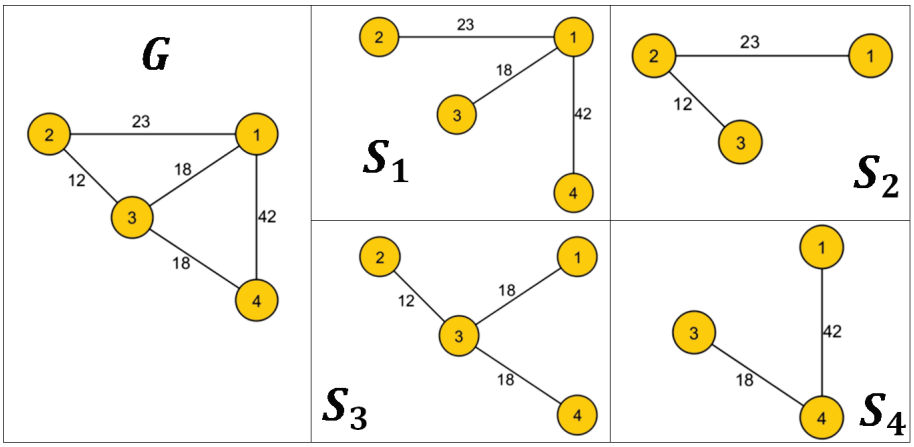


Figure 2.8: Example of stars of a graph

2.3 Graph matching

Graph matching is the process of finding a correspondence between nodes and edges of two graphs. In this section we present some important definitions related to graph matching. We present also the Graph edit distance (GED) which is one of the most famous and powerful fault-tolerant graph matching method.

Graph homomorphism: A graph *homomorphism* f from a graph G to a graph H, written as $G \to H$, is a mapping from $V(G)$ to $V(H)$ with edge preserving, which means that, if two nodes are adjacent in G, their images by f are adjacent in H. However, more than one node of G may be mapped to the same node in H. Formally, $uv \in E(G) \Rightarrow f(u)f(v) \in E(H)$.

Graph isomorphism: A graph *isomorphism* f between two graphs G and H, written as $G \simeq H$, is a bijection between their sets of nodes $V(G)$ and $V(H)$ with edge preserving. In other words, a graph *isomorphism* is a graph *homomorphism* with one-to-one correspondence between $V(G)$ and $V(H)$. Figure 2.9 illustrates an example of an *isomorphism* f between two graphs G and H.

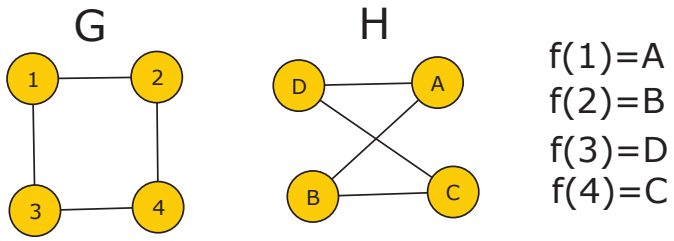


Figure 2.9: Example of a graph isomorphism.

Subgraph isomorphism: A graph *isomorphism* f between a graph G and a subgraph H' of the graph H ($G \simeq H'$) is called *subgraph isomorphism* between G and H. Figure 2.10 illustrates an example of a *subgraph isomorphism* f between two graphs G and H.

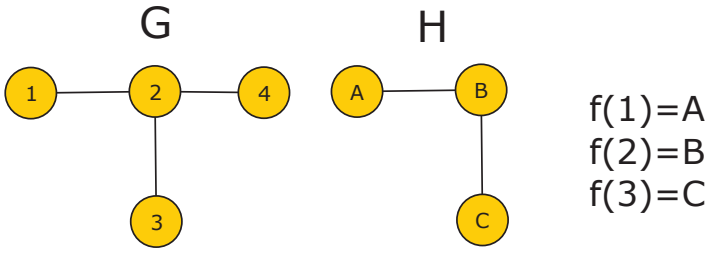


Figure 2.10: Example of a subgraph isomorphism.

Graph monomorphism: A graph *monomorphism* is a relaxed instance of subgraph isomorphism, in which the mapped subgraph allows additional edges. In other words, additional edges are allowed between nodes in the larger graph.

Maximum common subgraph isomorphism: A graph *isomorphism* f between the largest subgraph G' of the graph G and a subgraph H' of the graph H ($G' \simeq H'$) is called *Maximum common subgraph isomorphism* between G and H. In other words, the *Maximum common subgraph* is the largest isomorphic part of two graphs. In general, in the literature, this problem is related to the maximum clique problem.

Graph Edit Distance (GED) The Graph Edit Distance [9] between two graphs G_1 and G_2 is the minimum number of edit operations (minimum cost) to transform a graph G_1 into a graph G_2. A set of edit operations is given by insertions, deletions and substitutions (or relabeling) of graph elements (nodes and/or edges). We denote the substitution of two elements u and v by $(u \rightarrow v)$, the deletion of the element u by $(u \rightarrow \varepsilon)$, and the insertion of the element v by $(\varepsilon \rightarrow v)$. A cost is associated to each edit operation. A sequence of edit operations $e_1, ..., e_k$ transforming G_1 into G_2 is called an edit path between G_1 and G_2. However, for every pair of graphs G_1 and G_2, several edit paths transforming G_1 into G_2 exist with different total costs. The edit distance of two graphs is then defined as the minimum cost edit path between the two graphs G_1 and G_2. Figure 2.11 gives an example of the process *graph edit distance (GED)* transforming the graph G_1 into the graph G_2.
Formally, Let $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ be the source and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ be the target graph. The graph edit distance between G_1 and G_2 is defined as following:

$$\lambda(G_1,\ G_2)\ =\ \min_{(e_1,...,e_k)\ \in\ \gamma(G_1,G_2)}\ \sum_{i=1}^{k} c(e_i)$$

where $\gamma(G_1, G_2)$ denotes the set of edit paths transforming G_1 into G_2, and c denotes the cost function measuring the strength $c(e_i)$ of edit operation e_i.

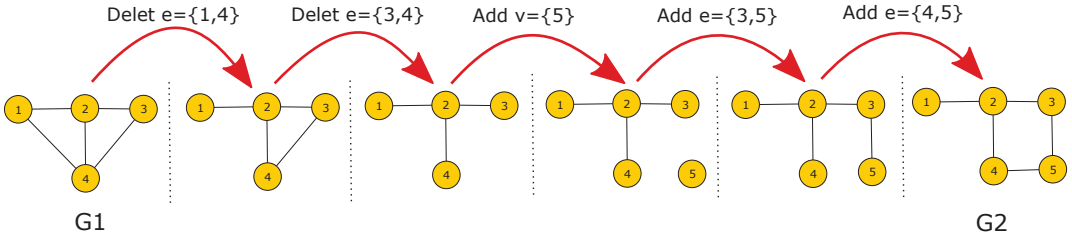


Figure 2.11: Example of applying the Graph Edit Distance (GED).

Chapter 3

Related work: Graph matching and its applications

Contents

Inexact graph matching and its applications on 2D and 3D Pattern Recognition is the main problem that we consider in this thesis. In this chapter, firstly, we present and discuss a state of the art related to graph matching algorithms. Due to the huge number of graph matching algorithms proposed since the late 1970s, it is not possible to review all the available algorithms in the literature. Thus, we provide juste the necessary state of the art needed for understanding this thesis and placing it within the appropriate context. For more exhaustive surveys, we refer the reader to [2, 16, 3, 17, 18]. In the state of the art that we present, we survey some graph matching algorithms available in the literature, especially the recent ones and we describe the different classes of graph matching algorithms based mainly on the taxonomy introduced in [2, 3]. Secondly, we present some applications of graph-based approaches used in Pattern Recognition. A special attention is given to 2D and 3D applications which are the main applications in this thesis.

3.1 Introduction

Graphs are a powerful representation tool and a famous mathematical formalism used in many applications of structural Pattern Recognition and classification [3, 17]. Graphs constitute an universal and a flexible modeling tool allowing both the description of properties of an object and the relationships between a set of objects. Since the late 1970s, the use of graphs in Pattern Recognition gained popularity and obtained a growing attention from the scientific community. Consequently, graph-based approaches are used in various fields. This wide utilisation is due to the technological advancement of new computer generations offering a high computational power, allowing the use of graph-based algorithms which have in the majority of cases a high computational complexity. Graph based techniques for Pattern Recognition aim to solve mainly two major problems. The first one is to find an optimal way to represent objects by graphs. The second problem is to find the appropriate method to compare and/or classify the objects represented by graphs.

In this chapter, we survey some efficient and recent graph matching algorithms and graph-based techniques in Pattern Recognition. We also present various applications of graph-based approaches in Pattern Recognition. The state of the art that we present is organized on two taxonomies. In the first one (Section 3.2), we present some efficient graph matching algorithms and graph-based techniques in Pattern Recognition available in the literature, especially the recent ones, and we describe their associated classes of graph matching algorithms. A special attention is given to two main approaches that we use in the rest of this thesis: Graph Edit Distance (GED) and graph invariants. The second taxonomy (Section 3.3) presents various applications of graph-based approaches in Pattern Recognition. We focus on the 2D and 3D Pattern Recognition which are the two main applications in this thesis. Section (3.4) concludes the chapter.

3.2 Graph matching methods

It is often required in many applications to compare objects. When objects are represented by graphs, graph matching and, more generally, graph comparison is a fundamental issue. Graph matching is the process of finding a correspondence between vertices and edges of two graphs that satisfies a certain number of constraints, ensuring that substructures in one graph are mapped to similar substructures in the

other. In this section, we present some graph matching and, more generally, graph comparison algorithms that have been proposed and utilized in Pattern Recognition. Figure 3.1 presents the taxonomy and the classification considered of the graph matching techniques and the graph-based approaches in Pattern Recognition.

Graph matching solutions are classified into two wide categories: exact approaches and inexact approaches. Exact approaches, such as those that test for graph isomorphism or sub-graph isomorphism, refer to the methods that look for an exact mapping between the vertices and the edges of a query graph and the vertices and the edges of a target graph. Inexact graph matching computes a distance between the compared graphs. This distance measures how similar (or dissimilar) are the graphs and deals with the errors that are introduced by the processes needed to model objects by graphs. In sections 3.2.1 and 3.2.2, we present respectively exact and inexact graph matching algorithms. In Section 3.2.3, we present some graph-based algorithms (used in Pattern Recognition) which are not exactly forms of graph matching. However, they are related to graph matching either because they present a way of comparing graphs or because they use graphs in the process of classification.

3.2.1 Exact graph matching

Exact graph matching is a mapping between the nodes and the edges of a query graph and the nodes and the edges of a target graph. With exact graph matching, edge-preserving must be ensured. This means that adjacent nodes in the query graph are mapped to adjacent nodes in the target graph. Graph isomorphism represents the most strict form of graph matching, in which the edge-preserving is satisfied in both directions and the mapping is a bijective (one-to-one) correspondence. A graph isomorphism between one of the two graphs and a subgraph of the other graph is known as subgraph isomorphism. Other forms of exact graph matching exist, in which the constraint of edge-preserving in both directions is dropped, giving rise to other type of graph matching like: graph *monomorphism*, graph *homomorphism*, *maximum common subgraph* (MCS). See Chapter 2 for detailed definitions. All the forms of exact graph matching, that we cited before, belong to the NP-complete class. Graph isomorphism constitutes the only exception for which it has not yet been demonstrated if it belongs to the NP class or not [2]. Recently, the authors of [19] show that graph isomorphism can be solved in quasi-polynomial time ($\exp((\log\ n)^{O(1)})$). Some algorithms for graph isomorphism with polynomial time

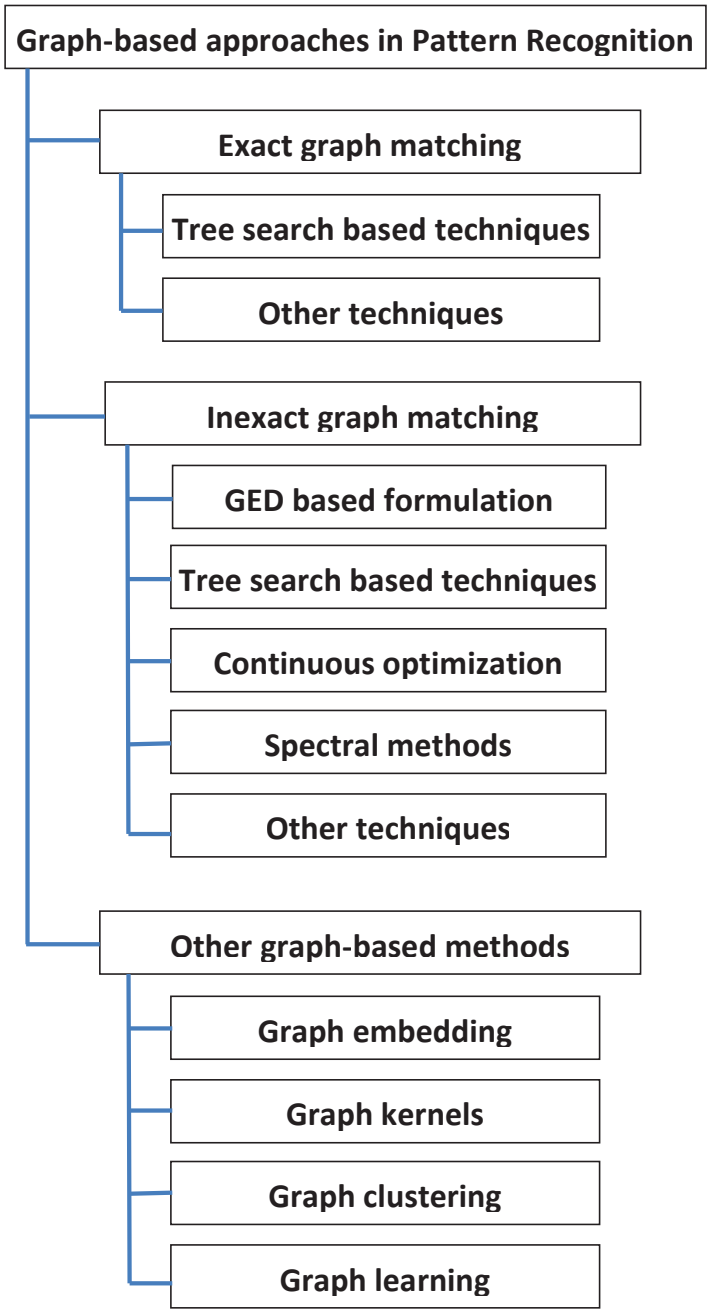


Figure 3.1: Taxonomy of Graph-based approaches in Pattern Recognition based on the taxonomies introduced in [2, 3].

complexity have been proposed for special kinds of graphs (such as: *e.g.* trees [20] and planar graphs [21]). However, until now, no polynomial algorithms are proposed for the general case. Exact graph matching has exponential time complexity in the worst case. Consequently, the exact methods are known to be only able to deal with graphs having a small number of nodes. The exact methods are mainly aimed to reduce the computational time of the matching process deriving from the exponential complexity of the exact matching problem. In the following we briefly review some typical methods for exact graph matching.

3.2.1.1 Tree search based techniques

Tree search based techniques constitute the main pillar of the most of existing algorithms for exact graph matching. These techniques use principally backtracking process in addition to some heuristics. The key idea in tree search based techniques is the following: a partial matching is constructed starting with an empty mapping set and iteratively enriched by adding a new couple of mapped nodes, with possibility of backtracking, usually using some heuristics to cut as soon as possible unfruitful search paths in order to avoid the complete exploration of the research space of all the possible matchings.

Various algorithms based on tree search techniques have been proposed in the literature. Ullmann's algorithm [22], is the first important algorithm based on tree search techniques. The algorithm is one of the most popular graph matching algorithm and still largely used until now despite of its age. Ullmann's algorithm addresses mainly graph isomorphism, subgraph isomorphism and monomorphism problems. However Ullmann explain a way to use the algorithm for maximum clique detection and consequently for the *Maximum common subgraph* (MCS) problem. Ullmann proposes a procedure called *refinement procedure* in order to cut the search space (unfruitful matches). The proposed procedure uses a matrix of possible future mapped couples of nodes to remove. In 1998, the authors of [23] propose the VF algorithm which is an algorithm for both isomorphism and subgraph isomorphism. The authors propose a fast heuristic which analyses the nodes adjacent to the ones already added in the partial matching. In 2004, the same authors propose an enhanced version of the algorithm [24], called VF2, in which they reduce the memory requirement from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, where n is the number of nodes in the graphs. An improved version of VF2 for biological graphs is presented in [25].

More recent tree search based algorithms have been proposed. In 2007, the authors of [26] propose an enhanced algorithm for finding the *Maximum Clique* and by the same way the *Maximum Common Subgraph* (MCS), called *MaxCliqueDyn*. In order to prune unfruitful matches, the proposed algorithm uses branch and bound combined with approximate graph coloring for finding tight bounds. In 2011, Ullmann in [27] presents an important enhancement of his own very well-known isomorphism algorithm from 1976 [22]. The new algorithm is based on the Binary Constraint Satisfaction Problem. Other algorithms based on Constraint Satisfaction Problem (CSP) have been proposed (*eg.*, [28] and the improved version in [29]). A more improved algorithm has been proposed by Solnon [30], in which the author use a better filtering based on the *AllDifferent* constraint. In 2016, the authors of [31] proposed a new exact maximum clique algorithm for large and massive sparse graphs. The authors used a branch-and-bound algorithm with a novel sparse encoding for the adjacency matrix.

3.2.1.2 Other techniques

Other algorithms addressing the problem of exact graph matching, not based on tree search techniques, have been proposed in the literature. One of the most efficient, fastest and interesting algorithm addressing the problem of graph isomorphism, not based on tree search techniques, is *Nauty*'s algorithm which is proposed in 1981 by McKay [32]. Using some results coming from group theory, Nauty's algorithm constructs an automorphism group of each graph. The algorithm associates a canonical form to each graph, consequently, two graphs are isomorphic if their canonical forms are equal. The equality between two canonical forms can be verified in $\mathcal{O}(n^2)$ time, however the canonical form can be constructed in exponential time in the worst case, while in the average case, Nauty's algorithm achieves a good performance.

Other kind of approaches are graph invariants, which have been efficiently used to solve the graph comparison problem in general and the graph isomorphism problem in particular. They are used for example in Nauty [32]. A vertex invariant, for example, is a number $i(v)$ assigned to a vertex v such that if there is an isomorphism that maps v to v' then $i(v) = i(v')$. Examples of invariants are the degree of a vertex, the number of cliques of size k that contain the vertex, the number of vertices at a given distance from the vertex, etc. Graph invariants are also the basis of graph probing [33], where a distance between two graphs is defined as the norm of their

probes. Each graph probe is a vector of graph invariants. A generalization of this concept is also used in [34] to compare biological data.

Other algorithms not based on tree search have been proposed. In [35], the authors propose an isomorphism algorithm that is based on Random Walks. The authors of [36] discuss the matching problem (graph isomorphism, subgraph isomorphism and maximum common subgraph) for the special case of graphs having unique node labels. In 2012, the authors of [37] propose a technique for speeding up existing exact subgraph isomorphism algorithms on large graphs.

3.2.2 Inexact graph matching

Identical structure with edge preserving in both directions are the constraints required by the exact graph matching so that two graphs will be isomorphic. Moreover, exact graph matching algorithms require a high computational complexity. All these strict constraints make exact graph matching only usable in few applications. The handled graphs in various graph-based applications, are subject to deformations due to several causes, (eg, the non-rigidity of the patterns, the noise in the acquisition process and the errors introduced by the modeling processes, etc). Consequently, the obtained graphs in many cases are different from the graph reference models. Hence, the matching process must be fault tolerant and this by allowing the structural difference between the compared graphs. The matching process must also able to find the solution in acceptable time, even without guarantee to find the optimal solution, but at least, find a good approximate solution. All these reasons and needs that we cited above have prompted the authors to propose an important number of inexact graph matching algorithms. The ignoring of the identical structure with edge preserving constraint imposed in the exact graph matching algorithms, is replaced by considering a system of penalizing which is used when the edge preserving is not respected in the inexact graph matching process. Indeed, a specific cost is associated to the edges not satisfying the edge preserving constraint. Thus, inexact graph matching algorithms aim to find a matching between the compared graphs that minimizes (or maximizes) the matching cost. In other words, inexact graph matching algorithms aim to compute a distance between the compared graphs. This distance measures how similar (or dissimilar) are the graphs.

Inexact graph matching algorithms are mainly classified into two classes: optimal and approximate algorithms.

Optimal inexact matching algorithms always find a solution if it exists. The solution found is exact and represents a global minimum of the matching cost. Consequently, these algorithms are considered as a generalization of exact matching algorithms. Optimal inexact matching algorithms not only require an exponential time complexity as exact graph matching algorithms but they are generally more expensive, which makes these algorithms not useable for many applications.

Suboptimal or approximate matching algorithms find a local minimum of the matching cost. The local minimum found is generally not far from the global one. However, there are no guarantees to reach the global minimum and to be able to find an exact solution if it exists [2]. The main advantage of these algorithms is their time complexity, which is usually polynomial. Hence, approximate matching algorithms are widely used in many applications.

A large number of inexact graph matching approaches have been proposed in the literature. In the following, we review some important inexact graph matching approaches, classified following the kind of the matching algorithm used.

3.2.2.1 Approaches based on Graph Edit Distance formulation

Many inexact graph matching algorithms are formulated as an approximation approach to compute the Graph Edit Distance (GED). Graph edit distance (GED) is one of the most famous and powerful fault-tolerant graph matching measures to determine the distance between graphs [38, 39, 40]. It is based on a kind of graph transformation called an edit operation. An edit operation is either an insertion, a suppression or a substitution of a node and/or an edge in the graph. A cost function associates a cost to each edit operation. The edit distance between two graphs is defined by the minimum costing sequence of edit operations that are necessary to transform one graph into an other [41]. This sequence is called an optimal edit path (See Chapter 2 for a formal definition). Tolerance to noise and distortion is one of the advantages of GED. Unfortunately, computing the exact value of the edit distance between two graphs is NP-Hard for general graphs and induces an exponential computational complexity [10]. This motivated the apparition of several heuristics giving rise to many inexact graph matching algorithms that approach the exact value of GED in polynomial time, using different methods such as bipartite assignment, dynamic programming and probability, etc.

Bipartite graph matching has been demonstrated to be one of the most efficient

algorithms to solve fault-tolerant graph matching [42]. In [9] and [10], the authors proposed an approach based on bipartite assignment in which they partition the compared graphs into smaller substructures and approximate GED by computing edit distance between substructures. A cost matrix between these substructures is defined and a mapping between them is realized using an algorithm of linear assignment, mainly, the Hungarian algorithm [43] or the Jonker-Volgenant algorithm [44]. These substructures are generally stars, *i.e.*, nodes with their direct neighbors and edges. However, they are called local descriptions in [9], stars in [10], b-stars in [45] and probe vectors in [46]. The edit distance between substructures is achieved in $\mathcal{O}((n+m)^3)$ time steps, where n and m are the number of nodes in the two compared graphs.

In addition to classic operators (insertion, suppression and substitution) used in GED, other ones are proposed. In [47], the authors proposed a novel solution of GED in which they introduce a new operator to support the node merging and splitting. They proposed also to apply edit operations in the both compared graphs until a common graph structure, instead of applying edit operations only to one graph in order to transform it to the other. They proposed to consider virtual nodes in the process of graph matching. Virtual nodes are the result of merging compatible nodes. Two nodes i, j are compatible nodes if they are adjacent and the distance $D_w(i,j)$ is less than a defined threshold.

Another approximation called BEAM is proposed in [48], where the authors present a fast suboptimal graph edit distance search which is a variant of a standard $A*$ *algorithm* reducing the search space. Rather than expanding all successor vertices in the search tree, only a fixed number of vertices to be processed are kept in the set of open vertices at all times. The search space is not completely explored, only the vertices belonging to the most promising partial matches are expanded.

Recent works are realized in order to speed up the runtime of the bipartite graph matching based approaches. In [49], the author proposed a new algorithm to compute the Graph Edit Distance in a sub-optimal way. The author demonstrated that the proposed algorithm ensure the same distance proposed in [9] with a reduced run time, which is $\mathcal{O}((\max(n,\ m))^3)$, where n and m are the number of nodes in the two compared graphs. However, the edit costs have to be defined in such way allowing that the GED to be defined as distance function, which means that the cost of insertion plus deletion of nodes (or arcs) have to be lower or equal than the cost of substitution of nodes (or arcs). The same author, in [50], proposed a new

fast algorithm, with $\mathcal{O}((\max(n,\ m))^3)$ time complexity, to compute the Graph Edit Distance in a sub-optimal way. The author used the Jonker-Volgenant linear solver [44] which is known to produce similar results than the Hungarian algorithm [43] but with an important run time reduction. However, the Jonker-Volgenant linear solver [44] has some convergence problems on some specific cost matrices. Hence, the author define a new cost matrix such that the Jonker-Volgenant linear solver [44] converges and the matching algorithm obtains the same distance value than the Bipartite algorithm [9]. In [51], the authors proposed a new distance called Hausdorff Edit Distance (HED). The proposed distance (HED) is an adaptation of the well-known Hausdorff distance between sets [52] for the Graph Edit Distance (GED). The proposed algorithm has a quadratic computational cost $\mathcal{O}(n*m)$, where n and m are the number of nodes in the two compared graphs. However, it does not obtain a bijective correspondence between the nodes of both graphs.
A comparison between the three algorithms Bipartite (BP) [9], Fast Bipartite (FBP) [49] and Square Fast Bipartite (SFBP) [50] was realized in [42]. The authors have shown that the performance and the optimality of FBP and SFBP were not affected by the violation of the theoretically restrictions imposed in FBP and SFBP. The authors have shown also that SFBP [50] with the Jonker-Volgenant solver [44] is the fastest algorithm. In [53], the authors proposed eight different options of local structures considered to construct the cost matrix in the Bipartite Graph Matching. The authors have shown also that the type of local structure and the distance defined between these structures is relevant for the runtime and classification ratio.
Other recent works have been proposed in order to speed up the runtime and/or improve the accuracy of Graph Edit Distance based approach. Among them we cite: [54], [55] and [56]. We refer the reader to [57] for a detailed GED survey.

3.2.2.2 Tree search based techniques

As in the exact graph matching, tree search based techniques with backtracking have been also utilized for inexact graph matching. The principe, in this case, is that both the cost of the current partial matching and the estimated cost of the rest of nodes using a heuristic, are used to guide the search process. The total cost is utilized either to cut unfruitful paths as in a branch and bound algorithm, or to specify the order of branches to be traversed in the search tree, as in the A^* algorithm [58]. The considered heuristics may not ensure to find the optimal solution, which means

that the matching is suboptimal. Various inexact graph matching algorithms based on tree search techniques have been proposed, for example: [59], [60], [61] and [62]. Many of the proposed algorithms are based on the well know A^* algorithm, such as: [63], [64], [65], [66] and [48].

3.2.2.3 Continuous optimization

Although graph matching is naturally a discrete optimization problem, and usually the methods proposed to solve it, use directly graphs. A thoroughly different method, is to solve the graph matching problem by solving an equivalent continuous one. This method is performed mainly on three steps: firstly, the graph matching problem is reformulated as a continuous problem. Secondly, the continuous problem is solved using an optimization algorithm. Finally, the continuous solution found is recast to the initial discrete domain. The inconvenient of this method is there is no guarantees to reach even the local optimality. Indeed, even the optimization algorithms used for the continuous problem in the second step ensure to find a local optimum (suboptimal solution). The final solution, resulting by the last approximation of the discretization step, may not guarantee to reach local optimality. However, continuous optimization based approach is very useful in many applications due to its very reduced computational cost which is usually polynomial [2].

Several inexact graph matching algorithms based on continuous optimization have been proposed, mainly organized into two families: probabilistic relaxation labeling and weighted graph matching problem (WGM). The first family (the probabilistic relaxation labeling) is an iterative process trying to find a correspondence between graphs by assigning a label to each node or substructure of one graph and this based on a set of constraints. Various algorithms have been proposed in this category, such as: [67], [68], [69], [70], [71], [72] and [73]. The second family is based on a formulation of the problem as a Weighted Graph Matching Problem (WGM). This approach is based on the use of a mapping matrix M containing real valued elements $[0, 1]$ in order to find a matching between two sets of nodes or substructures of the two compared graphs. A defined objective function which depends on the weights of the edges preserved by the match, must be optimised by the required matching. Due to the continuous values $[0, 1]$ of the elements of the matrix M, the WGM problem is usually and naturally transformed into a continuous problem. Hence, the WGM problem becomes a quadratic optimization problem. One of the important

disadvantages of the weighted graph matching WGM based approach is that only the weights of edges are accepted as attributes and the nodes cannot have. The authors of [74] were among the first, who linearized and solved the quadratic problem using the simplex algorithm. In [75], the authors propose an approach based on Lagrangian relaxation network for Graph Matching. The authors of [76], proposed a method to approximate the maximum common subgraph isomorphism problem by producing a weighted graph. The obtained weights indicate the probability that the associated link will be in the maximum common subgraph of the two graphs considered. Other methods for inexact graph matching based on continuous optimization have been proposed. Among them we can cite, the fuzzy graph matching ([77], [78] and [79]), Kernel Methods for graph matching such as Reproducing Kernel Hilbert Spaces based approaches ([80] and [81]).

3.2.2.4 Spectral methods

The basic idea considered in spectral methods is that the eigenvalues and the eigenvectors of the adjacency matrix of a graph are invariant to node permutations. Which means that, even if the rows and the columns of the adjacency matrix are permuted, its eigenvalues and eigenvectors stay unchanged. Consequently, the adjacency matrices of isomorphic graphs have the same eigenvalues and eigenvectors. However, the inverse is not true [82], which means that we cannot infer that two graphs are isomorphic, if their eigenvalues and/or eigenvectors are equal. Moreover, spectral methods are inexact in the sense that they do not ensure finding the optimal solution [3]. Furthermore, spectral methods cannot use the attributes of nodes or edges except some spectral methods which are able to exploit only real weights of edges. However, due to the polynomial time complexity of the spectral methods, they are widely used for solving the graph matching problem. Hence, various algorithms based on this approach have been proposed. We can cite, [83], [84], [85], [86], [87], [88], [89], [90], [73] and [91]. We refer the readers to [92], in which the authors present a survey of various graph based spectral approaches for comparing graphs and trees.

3.2.2.5 Other techniques

Other categories of inexact graph matching that are not mentioned previously, exist in the literature, where several algorithms have been proposed. We can cite: de-

composition methods ([93], [94], [95]), [96] and [97], neural networks ([98], [99], [100] and [101]), genetic algorithms ([102], [103], [104], [105] and [106]), methods based on local properties ([107], [108] and [109]) and methods based on tabu research ([110], [111]).

3.2.3 Other graph-based methods

In this section we briefly describe some graph-based methods in Pattern Recognition which are not exactly considered as forms of graph matching. However, these approaches can be affiliated to graph matching methods either because they present a manner of comparing two graphs, such as graph embedding and graph kernels, or because they use graphs to classify objects into classes, such as graph clustering and graph learning [3]. In the recent years, graph embedding and graph kernels have received a special attention and have gained popularity ([112] and [113]). The authors of [16] and [114] propose a survey related to graph kernels and graph embedding, in which they present these approaches as a manner to unify the statistical and structural techniques in Pattern Recognition. The authors show how to combine the complementary properties of the statistical and the structural techniques. We can also cite the Elastic Graph Matching problem (EGM) which is an image matching problem suing a graph structure. The idea is to superimpose a grid on the model image and define a set of attributes by computing some image features at the intersections of the grid lines. An isomorphic grid is also superimposed on the request image, and is then deformed (using the graph structure) in order to have the best matching between the features computed at the request grid points and the ones computed at the model grid. Several works dealing with the EGM problem have been realized, among them we cite: [115], [116], [117], [118], [119] and [120].

3.2.3.1 Graph embedding

Graph embedding approaches are based on a mapping onto a vector space. Graph embedding approaches cover mainly two slightly different categories [3]. The first one includes the methods that match the set of nodes (or substructures) of a graph onto a set of points in a vector space, where similar nodes (or substructures) will be mapped onto close points in the vector space. Several works have been realized in the literature, we can cite: [121], [122] and [89]. The second category includes approaches that match entire graphs onto points in a vector space, where similar

graphs will be mapped onto close points in the vector space. In this second category of graph embedding, according to the taxonomy proposed in [3], we distinguish four approaches. The first approach is the *isometric embedding*, in which the proposed methods use a similarity measure between graphs and aim to find a mapping to vectors conserving this measure. Among the proposed works in literature, we can cite: [101], [122] and [123]. The second approach is the *spectral embedding*, in which the proposed methods are based on the use of spectral properties of graphs such as the properties related to the eigenvalues and eigenvectors. Among the proposed works in literature, we can cite: [124], [125], [126] and [127]. The third approach is the *subpattern embedding*, in which the proposed methods aim to classify some specific kinds of subpatterns in the graphs to be embedded. Many works have been proposed in the literature, we can cite: [128], [129], [130], [131], [132], [133] and [134]. The fourth approach is the *prototype-based embedding*, in which the proposed methods aim to find a mapping of a graph onto a vector space, according to the distances of the graph from a set of supposed graph prototypes, using a defined distance function. Among the proposed works in the literature, we can cite: [128], [135], [136] and [137]. Several strategies for selecting the graph prototypes have been proposed such as: [138], [135], [139], [140], [141] and [142].

3.2.3.2 Graph kernels

A graph kernel is a symmetric and positive semi-definite function k that maps a couple of graphs from the space of all the graphs $\mathbb{G}$ onto a real number, $k : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{R}$. Every graph kernel k can be considered as a vector dot product, because they have similar properties. Instead of mapping graphs from $\mathbb{G}$ to a feature space and computing their dot product there, the value of kernel function k can simply be evaluated in $\mathbb{G}$ [143]. Kernel methods allow to extend basic linear algorithms to complex non-linear ones in a unified and elegant way [16]. Furthermore, kernel methods can replace the dot product in several vector-based algorithms and make standard algorithms, originally developed for vectors, applicable to more complex data structures such as graphs. Consequently, the concept of kernel machines can be extended from vectorial domains to structural domains [16]. In addition, using graph kernels methods allows to benefit from both the representational power of graphs and the huge number of vector-based algorithms. Hence, kernel methods are more appropriate for difficult Pattern Recognition tasks than traditional methods

under some conditions [144]. Several works have been realized in the literature, among them we cite: diffusion kernels ([145], [146] and [147]), convolution kernel ([148], [149] and [150]), walk kernel ([151], [152], [153], [154], [155] and [156]), based on GED ([157], [158], [159], [160]) and other methods ([161], [162] and [163]).

3.2.3.3 Graph clustering

Graph clustering approaches cover mainly two different categories. In the first one, a graph is used to represent each pattern and the clustering is realized on the set of graphs. Various works have been realized in this first category, among them we cite: [164], [165], [166] and [167]. In the second category, a graph is used to represent a set of patterns. Each node represent a pattern and edges are used to represent the relationships between couples of patterns. Usually, edges are weighted based on a similarity measure. In this category, the clustering is realized by partitioning the graph's nodes under some conditions. Many works have been realized in this second category, among them we cite: [168], [169], [170], [171], [172], [173], [174], [175], [176], [177], [178], [179], [180], [181], [182], [183], [184], [185] and [186].

3.2.3.4 Graph learning

The term *Graph learning* represent the learning approaches using graphs and refers mainly to two categories. In the first one, each pattern is represented by a graph. The class descriptions are also based on graph representation. Various works have been realized in this first category, among them we cite: [187], [188], [189], [190], [191], [192], [193], [194], [195] and [196]. In the second category, a graph is used to represent a set of objects. Each node represents an object and edges are used to represent the relationships between couples of objects. Usually, edges are weighted based on a similarity or distance measure. Many works have been realized in this second category, among them we cite: [197], [198], [199], [200], [201], [202], [203] and [204].

3.3 Applications of graph matching in Pattern Recognition

Several applications in Pattern Recognition using graph-based approaches have been proposed in the literature. Graph matching algorithms were the main build of

these approaches and were evaluated using various applications. Whatever are the considered graph-based applications, a step of graph representation and construction is needed. In this section we present some graph representations used in Pattern recognition. We present also some applications in Pattern recognition, mainly in the fields of 2D and 3D.

3.3.1 Graph representations

In the step of graph representation, nodes, edges and their attributes will be specified. Depending on the kind of the application, the nodes can represent points and/or regions of interest or any other components obtained by applying a specific process on the data such as segmentation. Edges represent connections between these nodes and define a specific relationship between nodes, such as proximity, adjacency, etc. Several graph databases are available in the literature, among them we can cite the well-known IAM Graph Database Repository [205], which contains several graph data sets. The first data set is the *Letter Graphs* data set which contains graphs that represent letter drawings (15 capital letters of the Roman alphabet that consist of straight lines only). Lines are represented by undirected edges and ending points of lines by nodes. Each node is labeled with a two-dimensional attribute giving its position relative to a reference coordinate system. Edges are unlabeled. The second data set is *Digit Graphs*, which consists of graphs representing handwritten digits. Nodes are inserted in regular intervals between the beginning and ending points of a line. Successive nodes are connected by undirected edges. Each node is labeled with a two-dimensional attribute giving its position relative to a reference coordinate system. Edges are attributed with an angle representing the orientation of the edge relative to the horizontal direction. *GREC Graphs* is the third data set consisting of graphs representing symbols from architectural and electronic drawings. Ending points, corners, intersections and circles are represented by nodes and labeled with a two-dimensional attribute giving their position. The nodes are connected by undirected edges which are labeled as line or arc. An additional attribute denotes the angle with respect to the horizontal direction or the diameter in case of arcs. The fourth data set is *Fingerprint Graphs* which consists of graphs representing fingerprints, which are converted into graphs by filtering the images and extracting important regions [206]. Then, a skeletonized representation of the extracted regions is obtained by applying a process of binarization and thinning.

Ending points and bifurcation points of the resulted skeleton are represented by nodes. Additional nodes are inserted in regular intervals between ending points and bifurcation points. Undirected edges link nodes that are directly connected through a ridge in the skeleton. Nodes are labeled with a two-dimensional attribute giving their positions. Edges are attributed with an angle giving the orientation of the edge with respect to the horizontal direction. Other data sets of the IAM Graph Database Repository (*COIL Graphs*, *Web-page Graphs*, *AIDS Graphs*, *Mutagenicity Graphs* and *Protein Graphs*) are detailed in [205].

In the field of 3D image or object analysis, several graph based methods have been proposed [207]. Among them, approaches based on skeletons ([208], [209] and [210]), in which the shapes are transformed into skeletons using a process of thinning. Then, the resulted skeletons are compared using graph matching methods. Approaches based on Reeb graphs which are constructed from functions defined on manifold objects. Several works have been proposed such as: [211], [212] and [213]. Other graph-based approaches have been proposed, such as methods using shape segmentation, in which the shapes are segmented into a set of components. Then, a graph of these components is constructed, where nodes represent the resulted segments which are linked by edges. Finally, the obtained graphs are compared using graph matching methods. Several works have been realized, among them, we cite: [214], [215] and [216]

3.3.2 Pattern Recognition examples using graphs

Various graph-based applications have been proposed in the literature, in many applications fields, among them, we can cite: **2D image analysis** ([62], [99], [217], [218], [9], [46], [45], [49], [51]) and **3D image analysis**, in which we can distinguish mainly two categories, rigid objects and non-rigid or deformable objects. In the first category, rigid objects are considered such as: a car, a chair, a bowl, ..., etc. Deformable objects can be also considered in the first category. However, each position of a deformable object is considered as a different object associated to a different class, for example, we associate each position of the human body (sitting, standing, hands raised, etc.) to a deferent class. In the second category, deformable objects are considered such as humans and animals. Unlike the first category, we associated one class to each deformable object, whatever the different positions. Various works have been realized in the field of **3D object analysis**, among them: [69], [95], [219],

[220], [220], [221], [222], [223], [224], [225], [226]. **Document analysis** including handwritten recognition (string, letters and digits) in different languages (Roman, Arabic, Chinese, etc.), symbol and graphics recognition. Several works have been realised in the field of **Document analysis**, among them, we cite: ([227], [228], [229], [230], [231], [16], [232], [233], [234], [51], [235]). **Biometric identification**, including face recognition, face mentions and expressions recognition, fingerprint recognition, etc. Various works have been realized in this field, among them, we cite: [236], [237], [238], [116], [239], [240], [241], [242], [243], [244], [245]. **Medical images analysis**, among them: [246], [247] and [248], etc. Other graph-based applications have been proposed in the literature in other fields, among them, we cite: Image database, Web-pages data, Video analysis, etc.

3.4 Conclusions

In this chapter, we presented and discussed a state of the art related to graph matching algorithms. The state of the art presented in this chapter only provides the necessary elements for understanding this thesis and placing it within the appropriate context. We surveyed graph matching algorithms available in the literature and especially the recent ones and we described the different classes of graph matching algorithms. We also presented some applications of graph-based approaches in Pattern Recognition.

Part I

Geometric graph matching: application to 2D Pattern Recognition

Introduction to Part I

This part addresses the issue of geometric graph matching and its applications on 2D Pattern Recognition. Kite recognition in satellite images is the main application considered in this part. We present a complete framework for Kite recognition on satellite images where Kites are modeled by graphs. Kites are huge archaeological structures of stone visible from satellite images. Because of their important number and their wide geographical distribution, automatic recognition of these structures on images is an important step towards understanding these enigmatic remnants. In this part, we present a complete identification tool relying on a graph representation of the Kites. As Kites are naturally represented by graphs, graph matching methods are thus the main building blocks in the Kite identification process. However, Kite graphs are disconnected geometric graphs for which traditional graph matching methods are useless.

This part contains two chapters, in the first one (Chapter 4), we present the process of Kite graph construction from real images, and the process of generating a synthetic data set of Kite graphs generated randomly. The two data sets (real and synthetic) are used to validate our algorithms. In the second one (Chapter 5), we propose a new graph similarity measure adapted to geometric graphs and consequently for Kite graphs. The proposed approach combines graph invariants with a geometric graph edit distance computation leading to an efficient Kite identification process. In Chapter 5, we analyze the time complexity of the proposed algorithms and conduct extensive experiments both on real and synthetic Kite graph data sets to attest the effectiveness of the approach. We also perform a set of experimentations on other data sets in order to show that the proposed approach is extensible and quite general.

The satellite images (ground truth data) used in the construction of the Kite database are provided by a team of archeologists expert on Kites. The archeologists have also checked and validated the experiment steps that we realized and the obtained results.

Chapter 4

Kite graph database construction

Contents

Kites are huge archaeological structures of stone visible from satellite images. Because of their important number and their wide geographical distribution, automatic recognition of these structures on satellite images is an important step towards understanding these enigmatic remnants. Kites are naturally represented by graphs. In this chapter, we present the process of Kite graph construction from real images, and the process of generating a synthetic data set of Kite graphs generated randomly. The two data sets (real and synthetic) are used to validate our algorithms.
This chapter is organized as follows: in Section 4.1, we give a short overview describing the archaeological structure called Kites. In Section 4.2, we explain the process of constructing and generating the real and synthetic Kite graph data sets. Section 4.3 concludes the chapter.

4.1 Kite

A Kite is an archaeological structure consisting of two long walls built of stones and arranged within a funnel shape opening onto an enclosure. The walls can reach a length of several kilometers and the enclosure can cover an area of several hectares. This yields huge constructions that are visible on satellite images as depicted in Figure 4.1(a). Kites were discovered in the Middle East in 1920. They were first

discovered by the British airmen who flew over the Jordanian desert during the period of the Mandate. They were thus called Kites due to the analogy of their shape with the shape of a Kite. Despite several studies, the issues related to their age and functions remain without satisfactory answers. Some rare dating attributes them to the Bronze Age but predated use of these structures is not excluded. The exact function of these structures has never been established. Many authors attribute a hunting function to the Kites, but the hypothesis of a pastoral use has not been refuted. These uncertainties are due to the extreme difficulty of obtaining reliable data during field investigations in contexts where archaeological material is most often absent [249, 250]. Recently, public access to high resolution satellite images (Google Earth, Bing) has significantly expanded the number of discovered Kites and also enlarged their geographical spread from the south of the Arabian Peninsula to the Aralo-Caspian region [251]. The massive use of Kites, judging by the density of these structures, probably had territorial implications and socioeconomic importance in a region that has seen the advent of agriculture and the birth of the urban phenomenon. Kites are thus an underestimated phenomenon. Establishing the duration of their utilization, outlining their use and functioning, and trying to identify the population responsible for these constructions are the challenges that would highlight the significance of this unknown phenomenon. However, these issues cannot be seriously addressed without an almost exhaustive inventory of these structures [252]. For this purpose, automatic recognition of Kites on satellite images offers archeologists valuable help in understanding this phenomenon. This will allow a systematic and homogeneous search in the entire distribution area of Kites and then in the peripheral regions.

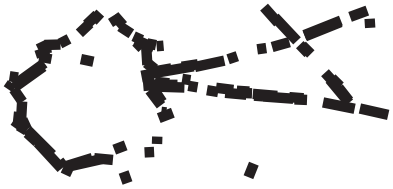
4.2 Kite graph data-set construction

In this section, we present the process of Kite graph construction from real images, and the process of generating a synthetic data set of Kite graphs generated randomly. The two data sets (real and synthetic) are used to evaluate the efficiency and the resilience of the proposed approach (described in Chapter 5).

4.2.1 Real data set construction

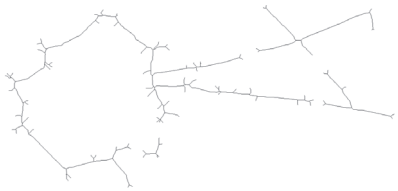
On satellite images, Kites appear as flat surfaces delimited by a set of lines as illustrated in (Figure 4.1(a)). To convert Kites' images into attributed graphs, the

(a)

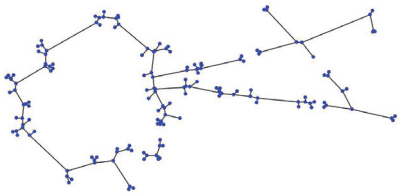


(b)

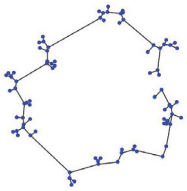
(c)



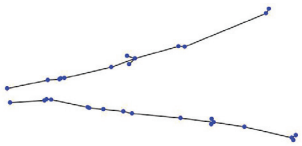
(d)



(e)



(f) Enclosure



(g) Antenna

Figure 4.1: Illustration of Kite detection

first step is to extract the Kite structures from the images by edge detection. Edge (segment or line) detection in images is an intensively studied topic in image analysis [253, 254]. Besides, several recent methods such as [255, 256, 257] give good results on satellite images. The main difficulty with such methods is to find the adequate settings to obtain an acceptable segment detection for a specific application. For Kites, we investigated several solutions with various settings and the LSD algorithm [257] gave us the most satisfactory set of segments (see Figure 4.1(b)). The LSD algorithm is followed by four steps to obtain the final Kite graphs:

- **Deleting isolated segments**: We consider that a segment is isolated if its length is less than a threshold $length_{min}$ and if it has no neighbors according to a minimum neighborhood distance $neighbor_{min}$.

- **Merging neighboring segments**: During this step, each pair of segments that are neighbors according to $neighbor_{min}$, do not cross each other and have the same angle with the horizontal line with a tolerance angle *delta*, are merged in one segment.
  $length_{min}$, $neighbor_{min}$ and *delta* are set during experimentations. Deleting isolated segments and merging neighboring ones are illustrated in Figure 4.1(c).

- **Thinning segments**: In this step, a skeleton is generated by reducing the width of all the segments to 1 pixel (see Figure 4.1(d)) using the Skeletonize "ImageJ" method, which is the implementation of the approach described in [258].

- **Graph construction**: Finally, we construct the graph from the skeleton by representing lines by edges and ending points of lines by vertices (Figure 4.1(e)). Each vertex is labeled with a two-dimensional attribute giving its position and an n-dimensional attribute containing the angles between every pair of consecutive incident edges. According to the state of preservation of the Kite, a graph obtained by this process can have a single connected component (i.e., the Kite is totally preserved) or it can be composed by two or more connected components (i.e., some parts of the Kite have been destroyed).

We executed our algorithm on 350 images (250 with Kites and 100 without Kites) with different states of preservation. We classified the obtained graphs into four preservation levels:

1. **State I**: The Kite is entire and well preserved. The Kite graph obtained is perfect and the few disconnections found are corrected manually with the help of the archeologists.

2. **State II**: The Kite is entire and well preserved. The Kite graph may be disconnected but the disconnections are neither frequent nor important.

3. **State III**: The Kite graph is very disconnected. Some parts of the Kite are not present.

4. **State IV**: The graph is not a Kite. These graphs are obtained by executing the algorithm on images that do not contain Kites. These images are extracted near (geographical positions) the images containing Kites, so these images have the same reliefs as the images containing the Kites, and the graphs obtained represent structures close to Kites.

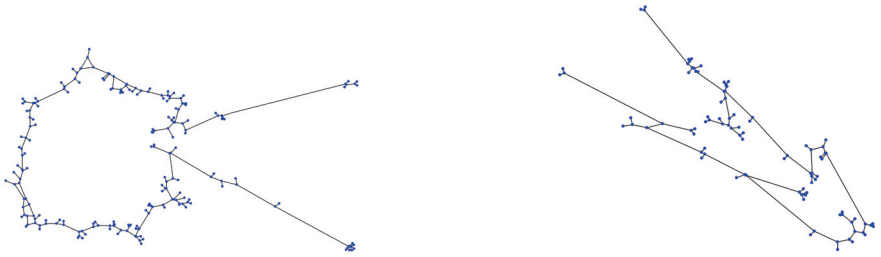
Figure 4.2 depicts some examples in each case. Figures (4.3, 4.4, 4.5 and 4.6) illustrate the process of extraction and transformation into graphs applied on twelve satellite images, three images per state of preservation.
The characteristics of the data set are summarized in Table 4.1.

	All the Data set	**State-I**	**State-II**	**State-III**	**State-IV**
$\#G$	4081	62	129	1581	2309
$\#Img$	350	50	100	100	100
$avg\ (V)$	26.14	110.84	113.74	30.09	15.59
$max\ (V)$	949	316	320	779	949
$avg\ (E)$	26.28	116.51	122.34	30.56	15.90
$max\ (E)$	1081	327	331	864	1081
$avgAng$	91.22	91.19	91.31	91.24	91.15
$maxAng$	180	180	180	180	180

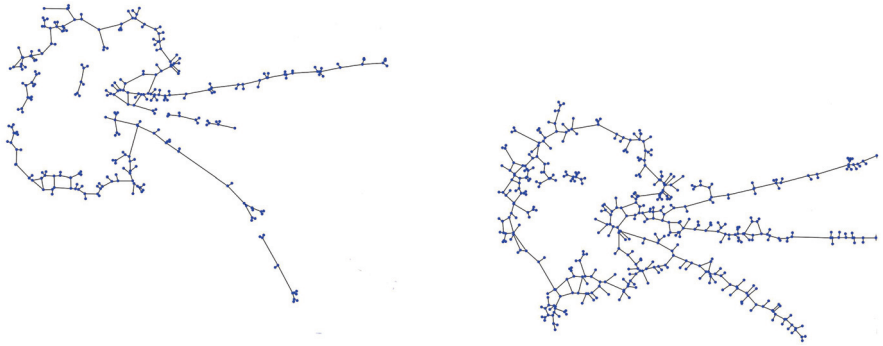
$\#G$: number of graphs. $\#Img$: number of images. $avg(V)$: average number of vertices. $avg(E)$: average number of edges. $max(V)$: maximum number of vertices. $max(E)$: maximum number of edges. $avgAng$: average value of the angles. $maxAng$ maximum angle value.

Table 4.1: Real Data set Characteristics

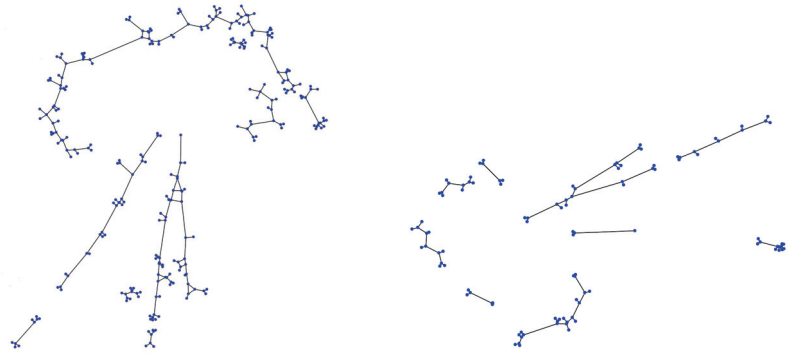
Kite graphs Prototype(Real) With the help of the archeologists, we selected from the graphs in **State-I**, the most preserved Kites as prototype Kite graphs. Also, to be able to deal with disconnected Kite graphs without adding significant computing costs, we constructed a prototype graph for each Kite component, namely:



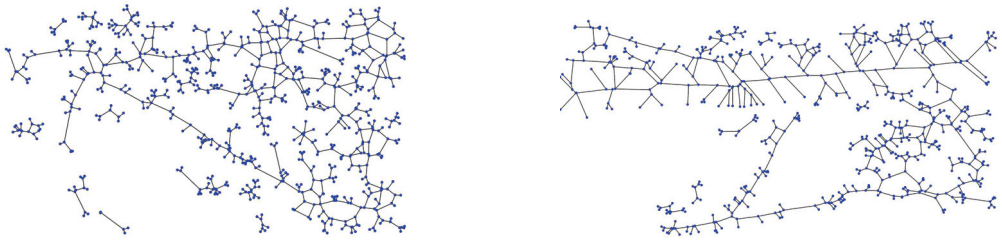
Preservation State I



Preservation State II



Preservation State III



Preservation State IV: non Kite graphs

Figure 4.2: Real data-set.

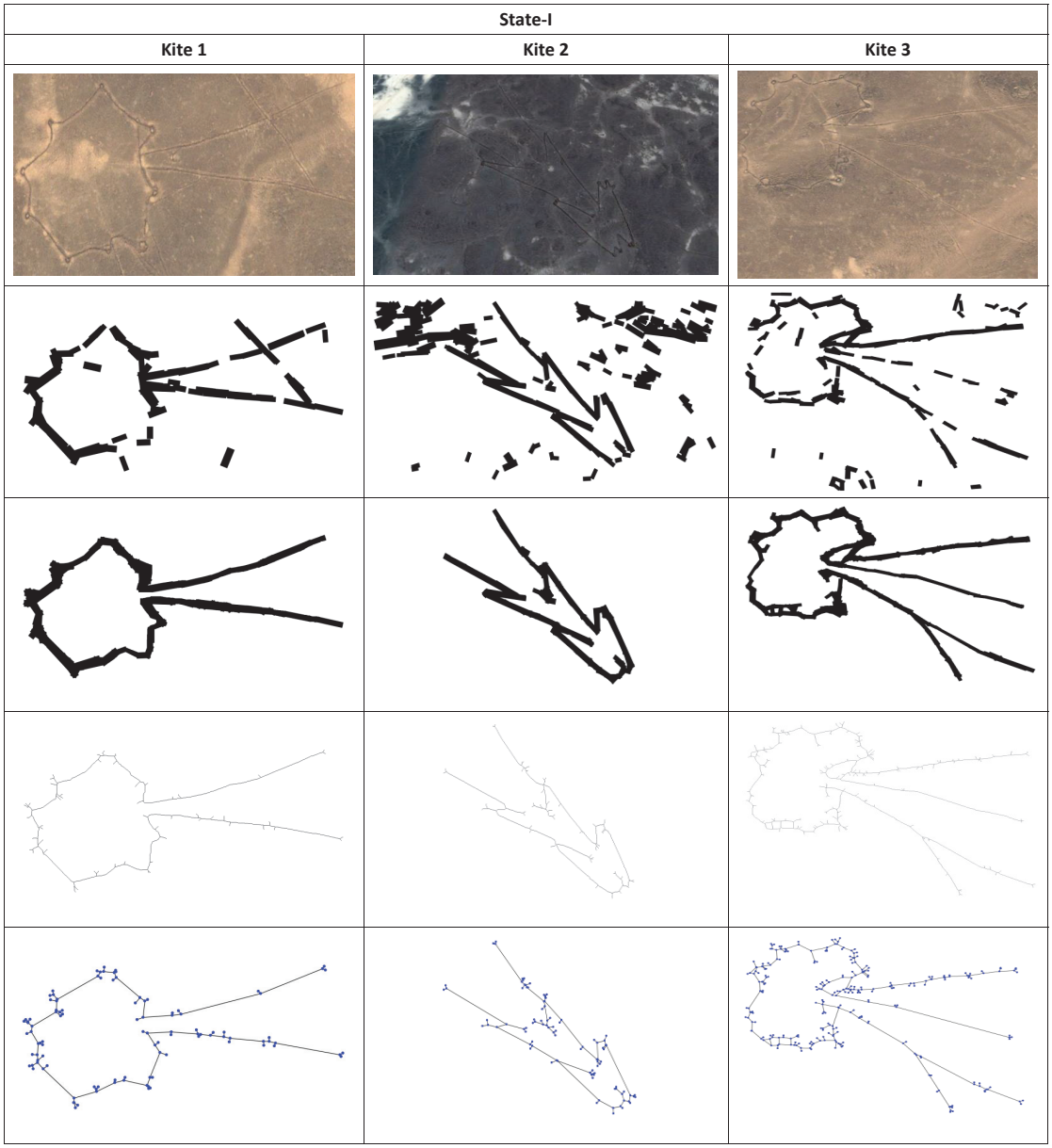


Figure 4.3: State-I: Example of extraction and transformation of images into graphs

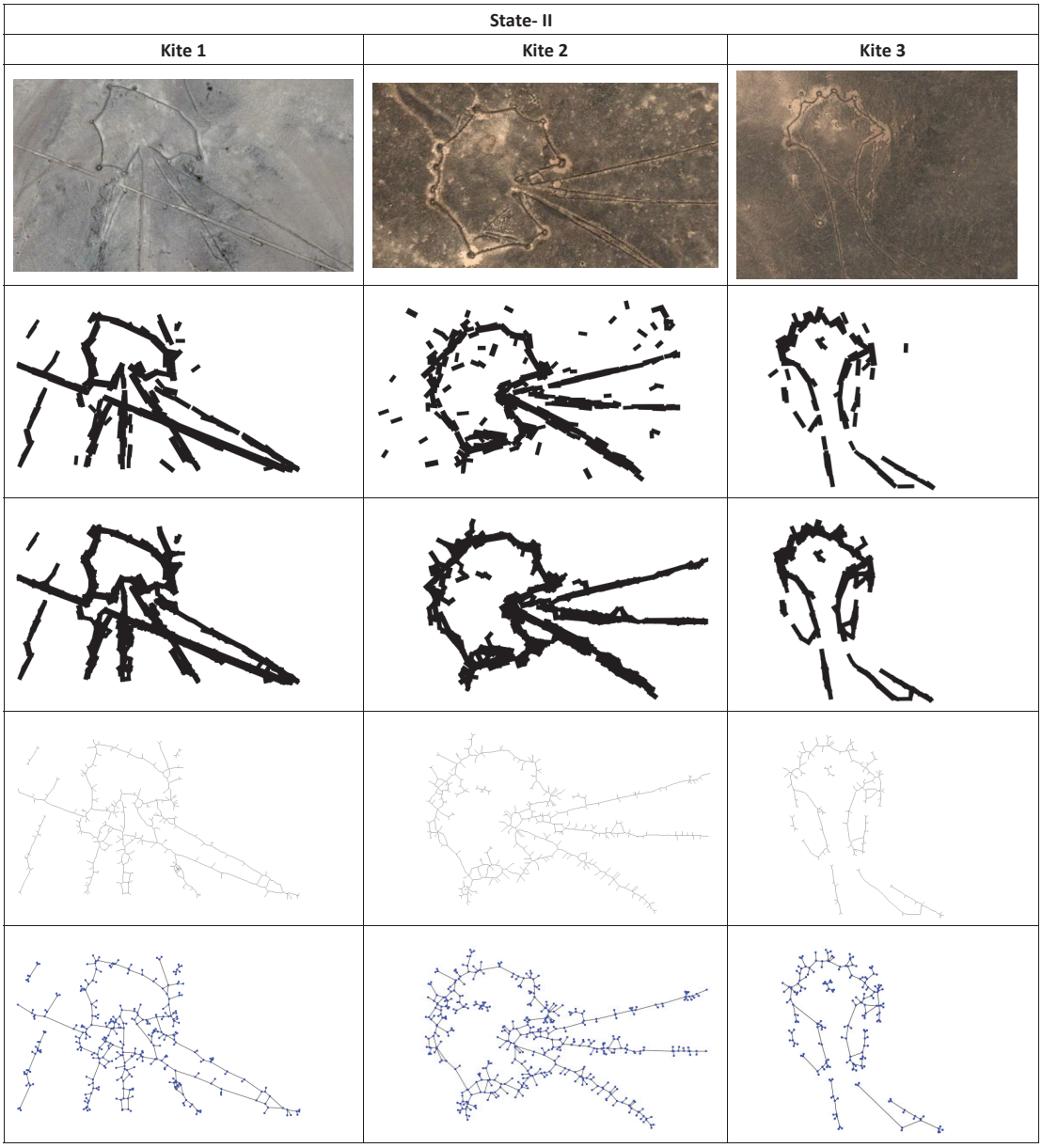


Figure 4.4: State-II: Example of extraction and transformation of images into graphs

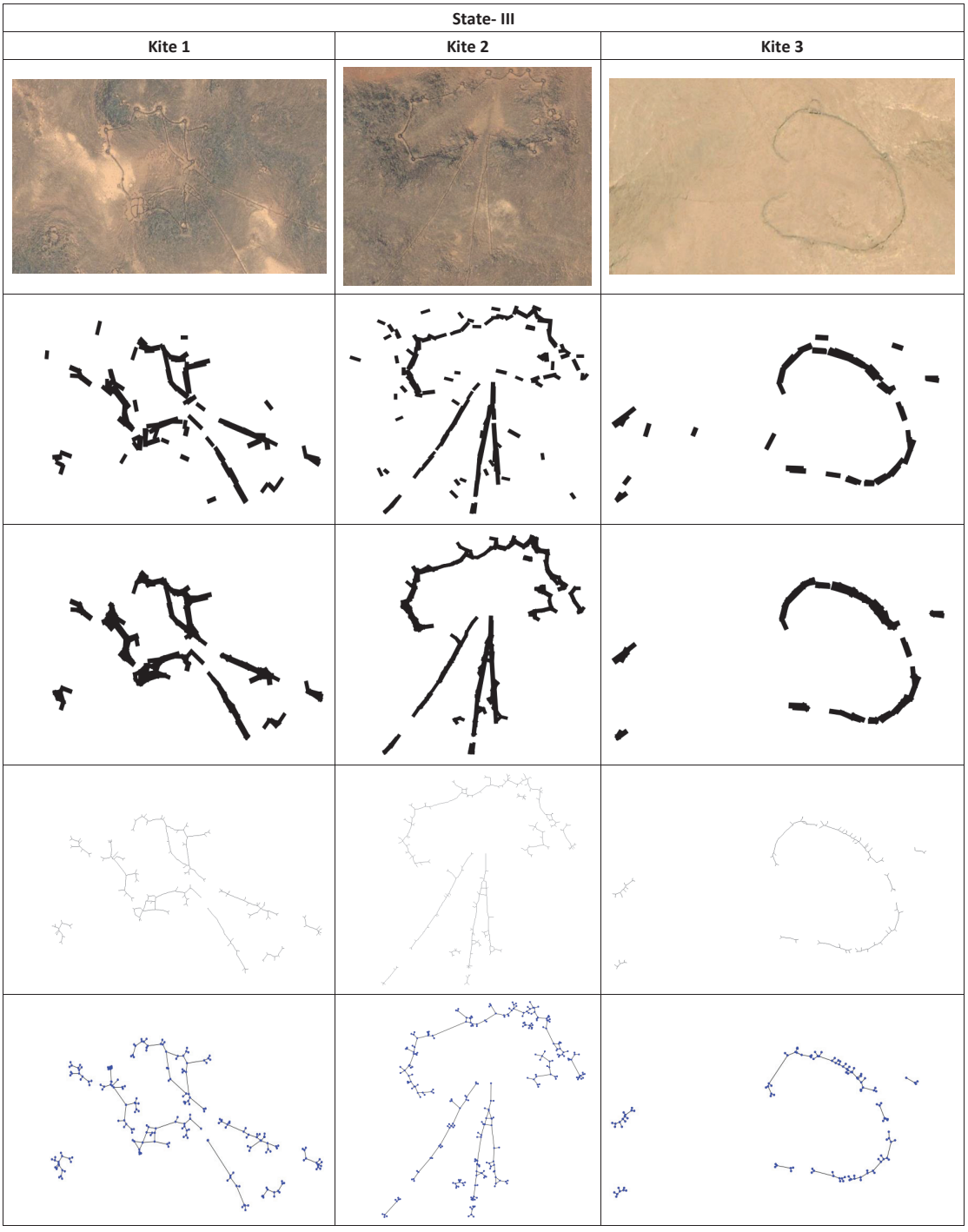


Figure 4.5: State-III: Example of extraction and transformation of images into graphs

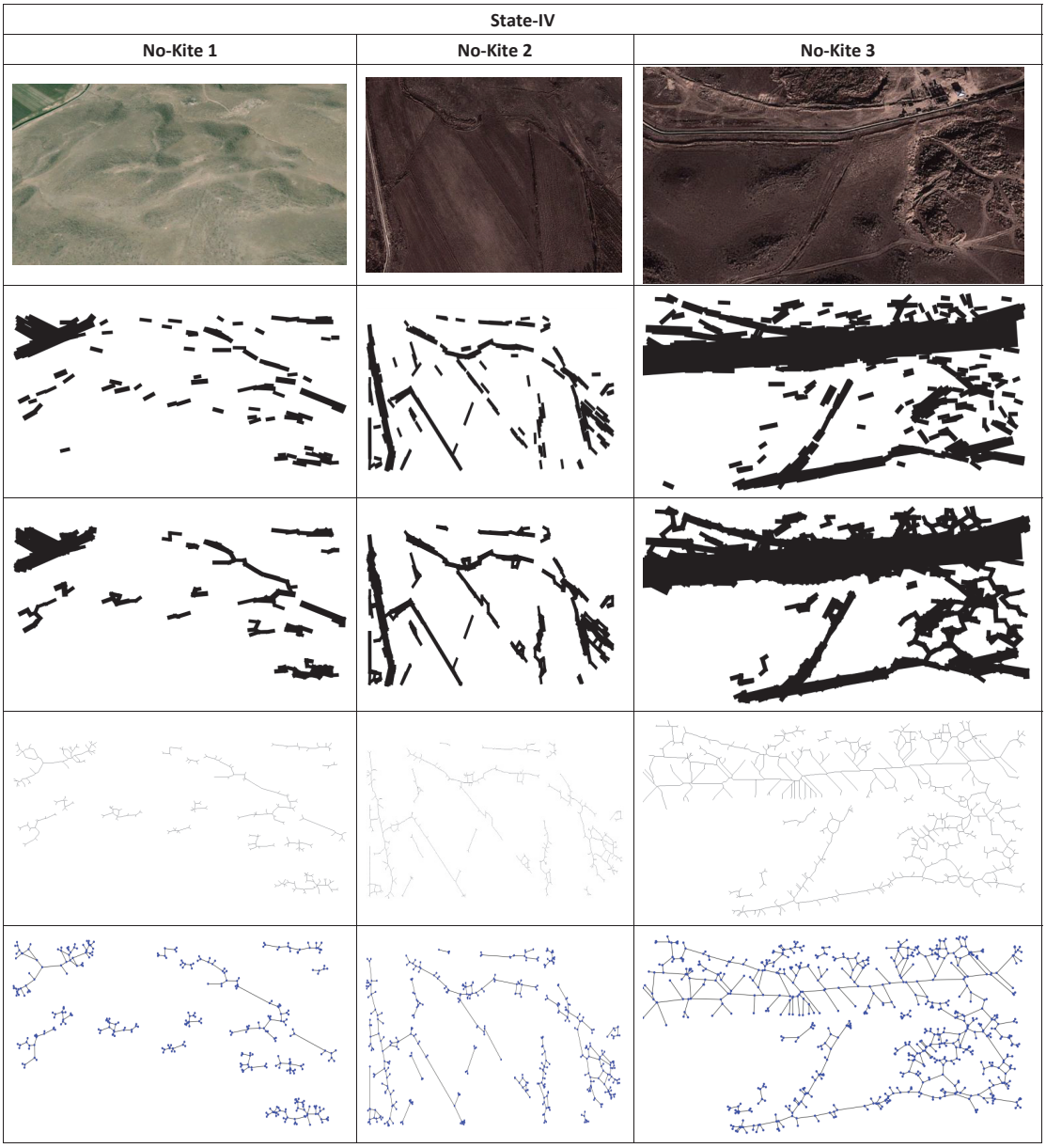


Figure 4.6: State-IV: Example of extraction and transformation of images into graphs

antenna and enclosure. Figures 4.1(f) and 4.1(g) give, respectively, an example of a Kite enclosure and a Kite antenna. In our experimentation, we consider a Graph Antenna, a Graph Enclosure and four different Graph Kites.

4.2.2 Synthetic data set generation

Random generation of a synthetic data set of Kite graphs offers us the possibility of: (i) obtaining Kite graphs in several possible preservation states. (ii) Obtaining Kite graphs with numerous deformations, which may correspond to the variations in form of Kite components or the absence of one or more of these components. (iii) Studying the scalability and resilience of our Kite recognition process.

In order to generate a graph representing a Kite (Figure 4.7 (d)), we generate the graphs of each component, namely the enclosure graphs and the antenna graphs. The different parameters used to generate the graphs of each Kite component are checked and controlled by a team of Kite expert archeologists.

Enclosure graph generation Due to the form of the Kite enclosure which is pseudo-convex, the generation of its graph is based on a circle equation. The center position c, the number of vertices N and the radius circle R are generated randomly according to a minimum and a maximum limit defined by the archaeologists. An angle Ang is generated according to the number of vertices in the Kite enclosure (see Figure 4.7 (a)). The coordinates $(x,\ y)$ of the vertices of the Kite enclosure are generated according to the circle equation. To obtain the *pseudo-convex* form of the enclosure, we vary the values of the radius ($R \pm \varepsilon_{R,\ i}$) and the angle ($Ang \pm \varepsilon_{Ang,\ i}$) for each generation of vertex coordinates (see Figure 4.7 (b)).

Antenna graph generation A Kite antenna is represented by a graph that is an open chain of edges (at least one edge). The number of edges, the distance between two vertices constituting an edge, and the inclination angle of an edge are generated randomly depending on a set of minimum and maximum values of the condescending parameters (see Figure 4.7(c)).

Using the described generation process, we obtain a synthetic data set containing 1000 graphs representing Kites. The characteristics of the synthetic data set are summarized in Table 4.2.

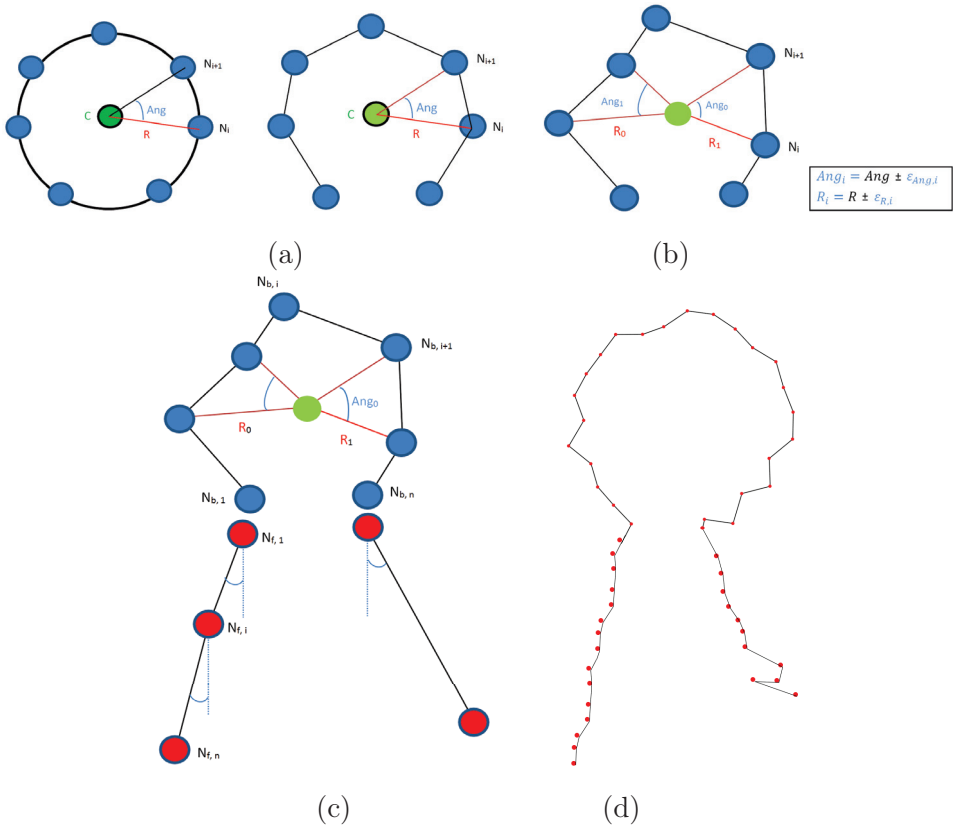


Figure 4.7: Kite graphs syntectic data-set generation process.

$\#G$	$avg\ (V)$	$max\ (V)$	$avg\ (E)$	$max\ (E)$	$avgAng$	$maxAng$
1000	58.44	90	57.41	89	108.89	180

$\#G$: number of graphs. $avg(V)$: average number of vertices. $avg(E)$: average number of edges. $max(V)$: maximum number of vertices. $max(E)$: maximum number of edges. $avgAng$: average value of the angles. $maxAng$ maximum angle value.

Table 4.2: Synthetic Graph Data set Characteristics

Kite graphs Prototype(Synthetic) Using the described process, we generate a set of prototype graphs representing: an antenna, an enclosure and four entire Kites.

4.3 Conclusion

In this chapter, we introduced a graph representation of Kites and we proposed an automatic process for extracting and transforming Kites from satellite images into a set of graphs. We also proposed a process of generating randomly a synthetic data

set of Kite graphs. Using the two proposed processes, we construct a benchmark of Kite graphs (real and synthetic) structured in different levels of deformations. This benchmark is used to validate our algorithms.

Chapter 5

Geometric Graph Matching

Contents

In this chapter, we propose a new graph similarity measure adapted to geometric graphs and consequently for Kite graphs. The proposed approach combines graph invariants with a geometric graph edit distance computation leading to an efficient Kite identification process. We analyze the time complexity of the proposed algorithms and conduct extensive experiments on both real and synthetic Kite graph data sets to attest the effectiveness of the proposed approach. We also perform a set of experimentations on other data sets in order to show that the proposed approach is extensible and quite general.

5.1 Introduction

Kite recognition as a graph matching problem is interesting because it raises several challenges not addressed by existing methods. In fact, Kite graphs are not connected and may contain several parts. They have specific geometric forms that distinguish them from other constructions. Furthermore, each processed image can involve a large number of graphs, thus implying the use of rapid recognition algorithms. To

tackle these challenges, we propose a multi-level recognition framework that first applies rapidly computed graph invariants to discard non-Kite graphs in the early stages of the recognition framework. Then, we use a local similarity measure that takes into account the geometric form of Kite graphs by considering the angles of the form. Finally, a reconstruction process allows us to consider disconnection within Kite graphs.

We compare our work with existing methods using the two data sets (real and synthetic) Kite graphs and we also apply it to other data sets mainly characterized by the geometric form of the graphs. These experimentations show that the proposed framework is a practical and efficient Kite recognition tool that applies directly to images. The realized experiments show also that the proposed approach is quite general.

The rest of the chapter is organized as follows: Section 5.2 describes the proposed similarity measure and presents its complexity analysis. Section 5.3 reports our experimental results and finally, Section 5.4 concludes the chapter.

5.2 Algorithm overview

In this section, we describe the proposed Kite recognition solution, which is a hierarchical graph-based approach consisting of: approaches measuring the distance between two graphs and a reconstruction process. Firstly we present the proposed approaches measuring the distance between two graphs: a global similarity measure denoted *Global*, a geometric local similarity measure denoted *GeoLocal* and two varieties of hierarchical measures that we call $Global_{GeoLocal}$ and $GeoLocal_{Global}$ which are the result of combining *Global* and *GeoLocal* depending on the defined order. The global similarity *Global* is a fast computable measure based on graph invariants. This similarity aims to rapidly discard the graphs that cannot be Kites and avoid unnecessary and more costly comparisons. The geometric local similarity *GeoLocal* is a more accurate measure based on the geometric form and the structured features extracted from the graphs. This similarity is based on graph edit distance GED to deal with the state of preservation of the Kites. Secondly, we present the reconstruction process, which aims to verify if the different connected components of the graph identified as Kite components (enclosure and antenna) constitute a Kite. Identification of the different connected components of the graph as Kite components is realized using one of the proposed approaches of graph similarity measure,

Symbol	Description
$G(V, E)$	undirected labeled graph where V is its vertex set and E its edge set. Both vertices and edges are labeled.
$V(G)$	vertex set of graph G.
$E(G)$	edge set of graph G.
$deg(v)$	degree of vertex v.
$\Delta(G)$	the greatest vertex degree in graph G.
$\ell(e)$	the label of edge e.
$\mathcal{A}(G)$	the greatest angle in graph G.
$\mathcal{L}(G)$	the greatest edge label in graph G.
$\|\|S\|\|$	size of the set S.
$(\angle v_1 v_2 v_3)_{G_i}$	the angle between the two edges $(v1, v2)$ and $(v2, v3)$ in the graph G_i.

Table 5.1: Notation

namely: *Global*, *GeoLocal* or one of two hierarchical measures $Global_{GeoLocal}$ or $GeoLocal_{Global}$. Finally, we present the computational complexity of the proposed algorithm.

Table 5.1 summarizes the notation that we use in the remainder of this chapter.

5.2.1 Global similarity

Global similarity computes graph invariants. We consider the number of vertices of the compared graphs, the labels of the edges, which correspond to the length of the Kite walls, and the angles between edges. So, the global similarity between two graphs G_1 and G_2 is given by:

$$\begin{aligned} Global(G_1, G_2) = & \, w_1 * d_{Vertices}(G_1, G_2) \\ & + w_2 * d_{Edges}(G_1, G_2) \\ & + w_3 * d_{Angles}(G_1, G_2) \\ & + w_4 * d_{Convex}(G_1, G_2) \end{aligned} \tag{5.1}$$

where w_i is a weighting coefficient with $\sum_{i=1}^{i=4} w_i = 1$, $d_{Vertices}(G_1, G_2)$ compares the order of the two graphs.

$$d_{Vertices}(G_1, G_2) = \frac{\big|\|V(G_1)\| - \|V(G_2)\|\big|}{Max\big(\|V(G_1)\|, \|V(G_2)\|\big)} \tag{5.2}$$

$d_{Edges}(G_1, G_2)$ compares the global size of the two Kites by comparing the distances reported on the edges of the corresponding graphs.

$$d_{Edges}(G_1, G_2) = \frac{\left|\sum_{i=1}^{\|E(G_1)\|} \ell(e_i) - \sum_{j=1}^{\|E(G_2)\|} \ell(e_j)\right|}{Max(\sum_{i=1}^{\|E(G_1)\|} \ell(e_i), \sum_{j=1}^{\|E(G_2)\|} \ell(e_j))} \tag{5.3}$$

$d_{Convex}(G_1, G_2)$ and $d_{Angles}(G_1, G_2)$ compare the global geometric forms of the two Kites based on the convexity of the angles and the total value of the angles, respectively:

$$\begin{aligned} d_{Convex}(G_1, G_2) = \Big| & \frac{\|a \in Angles_{G_1}, a < ConvexityTh\|}{\|Angles_{G_1}\|} \\ & - \frac{\|b \in Angles_{G_2}, b < ConvexityTh\|}{\|Angles_{G_2}\|} \Big| \end{aligned} \tag{5.4}$$

where $Angles_{G_i}$ denotes the set of angles of graph G_i and $ConvexityTh$ is an angle threshold at most equal to 180°. However, it will be defined according to the form of the Kites.

$$d_{Angles}(G_1, G_2) = \frac{|\sum Angle_{i,G_1} - \sum Angle_{j,G_2}|}{Max(\sum Angle_{i,G_1}, \sum Angle_{j,G_2})} \tag{5.5}$$

where $Angle_{i,G}$ denotes the i^{th} angle of graph G.

The algorithm takes as inputs a set of prototype graphs, which are: $G_{Antenna}$, $G_{Enclosure}$, four different G_{Kite} and a query graph. For each connected component of the query graph, the algorithm returns the most similar Kite component.

5.2.2 Geometric local similarity

The geometric local similarity measure *GeoLocal* is a distance based on the approximation of the graph edit distance that compares the graphs using local descriptions of substructures (see Figure 2.8). However, unlike the approaches proposed in [9, 46, 10], in our approach we extended local descriptions by considering angles in addition to degrees of vertices and labels of the edges. This allows us to distinguish between two isomorphic graphs with different geometry (Figure 5.1(a)). In fact, almost all existing graph similarity measures compare the structures of graphs in terms of vertices, edges and their labels, but they do not consider the geometric form of these graphs. Some authors even use the angle as an attribute or a label

associated with an edge [205]. This attribute represents the angle between the considered edge and a horizontal or a vertical line landmark (Figure 5.1(d, e)). The drawback of this representation is that the value of the angle may change if a rotation is applied. This is not a problem in some graph representations such as graphs representing letters, digits, etc. However, considering a model that resists rotation and other deformations is very important when the graphs represent objects with specific forms as is the case of Kite graphs. Thus, in our framework two graphs are isomorphic if they also have the same form according to the following definition.

Definition 1 (geometrical isomorphic) Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two graphs. G_1 and G_2 are *geometrical isomorphic* if they are *isomorphic* and have the same geometric form.

Example 1 Let $G_i(V, E)$ where $i = 1...5$, be five graphs, such that $V = \{v_1,\ v_2,\ v_3\}$ and $E = \{e_1,\ e_2,\ e_3\}$ (Figure 5.1(a, b)). We can easily find a mapping between the set of vertices of G_1 and G_2 ensuring edge preservation, thus G_1 and G_2 are isomorphic. However, they are not *geometrical isomorphic* because they do not have the same geometric form ($(\angle v_1v_2v_3)_{G_1} \neq (\angle v_1v_2v_3)_{G_2}$), (see Figure 5.1(a)). G_3, G_4 and G_5 (see Figure 5.1(b)) are *geometrical isomorphic*: because they are *isomorphic* and have the same geometric form ($(\angle v_1v_2v_3)_{G_3} = (\angle v_1v_2v_3)_{G_4} = (\angle v_1v_2v_3)_{G_5}$). However, if we consider angles between edges and the horizontal axis x, G_3 and G_5 in (Figure 5.1(d, e)) are *isomorphic* but not *geometrical isomorphic* : ($\alpha_{1,\ G_3} \neq \beta_{1,\ G_5}$ and $\alpha_{2,\ G_3} \neq \beta_{2,\ G_5}$).

In order to compute the geometric local similarity, each vertex v is represented by a signature *i.e.*, a vector defining its local structure as follows:
$s(v) = \{deg(v),\ \{\ Ang_i,\ \ell(e_{i,\ 1}),\ \ell(e_{i,\ 2})\ \}_1^{(deg(v)-1)}\}$, where:

- $deg(v)$ is the degree of the vertex v.
- $\ell(e_{i,\ 1})$ and $\ell(e_{i,\ 2})$ are the labels (weights) of the two edges constituting the angle Ang_i. $\ell(e_{i,\ 1})$ and $\ell(e_{i,\ 2})$ are ranked in descending order.
- The triplets $\{\ Ang_i,\ \ell(e_{i,\ 1}),\ \ell(e_{i,\ 2})\ \}$ are ranked according to the angle Ang_i in descending order.
- All the vertices are represented by signatures *i.e.* vectors which have the same size: $size = 1 + ((\Delta(G_1,\ G_2) - 1) * 3)$.

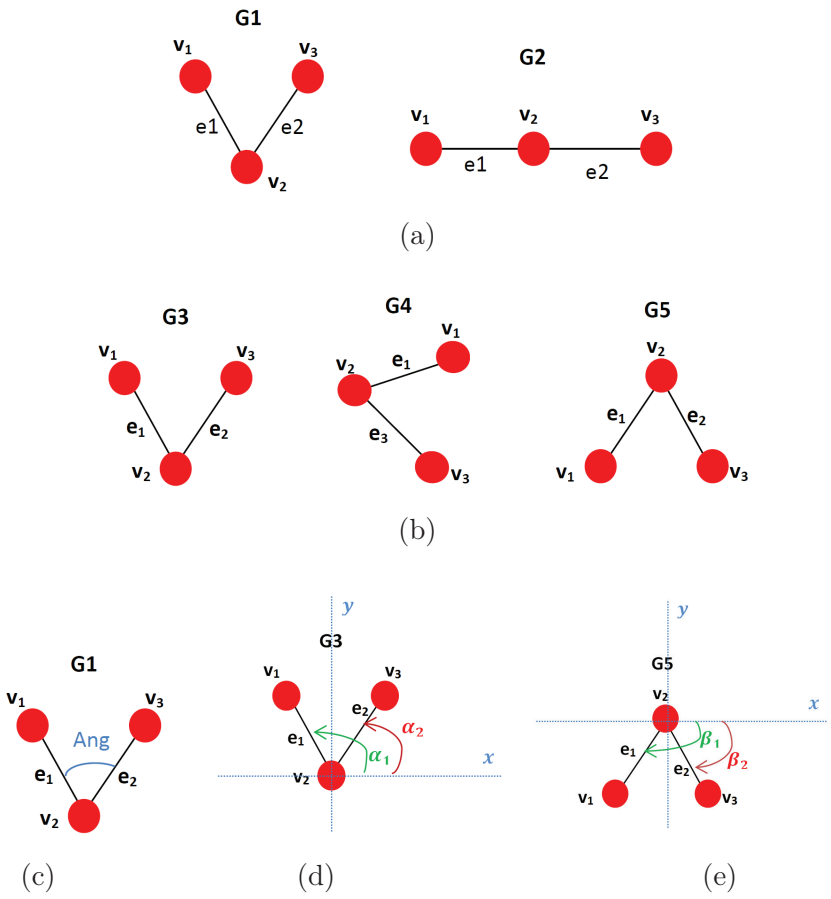


Figure 5.1: Example of isomorphic graphs

- $\Delta(G_1,\ G_2)$ is the greatest vertex degree in the compared graphs G_1 and G_2.
- If a vertex v has a degree less than $\Delta(G_1,\ G_2)$, the rest of the vector is completed with *zeros*.

The similarity measure d between two signatures s_1 and s_2 is computed as follows:

$$d(s_1,\ s_2) = 1 - \sum_{i=1}^{i=3} \omega_i * F_i \tag{5.6}$$

The functions F_i are defined as follows:

$$F_1 = \frac{|deg(v_1) - deg(v_2)|}{Max(\Delta(G_1), \Delta(G_2))} \tag{5.7}$$

$$F_2 = \sum_{k=1}^{Max(\Delta(G_1),\Delta(G_2))} \frac{|\ell(e_{1,k}) - \ell(e_{2,k})|}{Max(\Delta(G_1), \Delta(G_2)) * Max(\mathcal{L}(G_1), \mathcal{L}(G_2))} \tag{5.8}$$

$$F_3 = \sum_{k=1}^{Max(\Delta(G_1),\Delta(G_2))-1} \frac{|Ang_{1,k} - Ang_{2,k}|}{(Max(\Delta(G_1),\Delta(G_2)) - 1) * Max(\mathcal{A}(G_1),\mathcal{A}(G_2))} \tag{5.9}$$

where ω_i are weighting coefficients with $\sum_{i=1}^{i=3} \omega_i = 1$, $Ang_{i,\ k}$ is the k^{th} angle of vertex v_i. $F_{i=1...3}$ compares, respectively, the degree of the vertices, the labels of edges and the angles.

The Geometric Local Similarity *GeoLocal* aims to determine the best matching between the signatures (defining the local structure of each vertex) associated with the two compared graphs. Formally, let G_1 and G_2 be two graphs, S_1 and S_2 their corresponding sets of signatures, and M the set of all possible matching between S_1 and S_2. The similarity $GeoLocal(S_1,\ S_2)$ is formulated as follows:

$$GeoLocal(S_1, S_2) = 1 - \frac{\underset{m\ \in\ M}{Max} \sum_{s_i\ \in\ S_1,\ m(s_i)\ \in\ S_2} d(s_i\ ,\ m(s_i))}{Max(\|S_1\|,\ \|S_2\|)} \tag{5.10}$$

Computation of $GeoLocal(S_1,\ S_2)$ is equivalent to solving the assignment problem which is a fundamental combinatorial optimization problem that aims to find the minimum/maximum weight matching in a weighted bipartite graph. To solve this assignment problem, we define a $n \times n$ matrix D, where $n = \max(\|S_1\|,\ \|S_2\|)$. Each element $D_{i,j}$ of the matrix represents the similarity measure $d(s_i,\ s_j)$ between a signature s_i in S_1 and a signature s_j in S_2. In the case of $\|S_1\| \neq \|S_2\|$, the smallest set of signatures is completed by $(max(||S_1||, ||S_2||) - min(||S_1||, ||S_2||))$ empty signatures ε. The similarity between an empty signature ε and a signature s is computed by the formula (Eq. 5.6) and corresponds to the cost of adding s to the small graph (or of deleting s from the large graph).

We apply the Hungarian algorithm [43] on the matrix D in order to find the best assignment in $\mathcal{O}(n^3)$ time.

The resulting distance (dissimilarity) is compared to a threshold $th \in [0, 1]$ defined by an expert or by experimentation, in order to decide if the compared graphs are similar or not. Like the algorithm of *Global*, the algorithm of *GeoLocal* takes as inputs a set of prototype graphs, which are: $G_{Antenna}$, $G_{Enclosure}$, four different G_{Kite} and a query graph. For each connected component of the query graph, the algorithm returns the most similar Kite component.

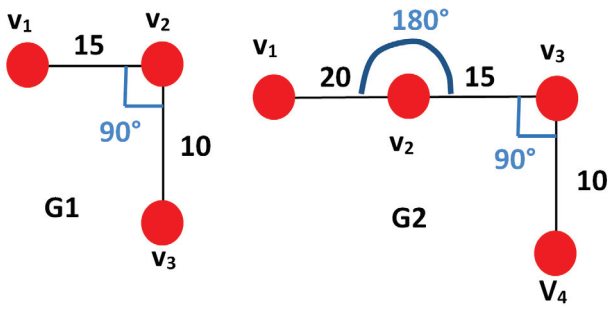


Figure 5.2: The graphs G1 and G2 in Example 2.

Example 2 Let G_1 and G_2 be two graphs (see Figure 5.2), S_1 and S_2 their corresponding sets of signatures, such that $\|S_1\| = 3$ and $\|S_2\| = 4$. Let D, be the matrix of similarities between S_1 and S_2. Where $D_{i,j} = d(s_i\ ,\ s_j)$. $\|S_1\| < \|S_2\|$, thus we add an ε signature to S_1 and we complete the matrix D by $d(\varepsilon\ ,\ s_{2,\ j=0...3})$.

$$
\begin{array}{c} \\ s_{1,0} \\ s_{1,1} \\ s_{1,2} \\ \varepsilon \end{array}
\begin{array}{c}
\begin{array}{cccc} s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \end{array} \\
\left(\begin{array}{cccc}
0.96 & \mathbf{0.33} & 0.58 & 0.96 \\
0.54 & 0.75 & \mathbf{1} & 0.54 \\
0.92 & 0.29 & 0.54 & \mathbf{1} \\
\mathbf{0.92} & 0.04 & 0.29 & 0.75
\end{array}\right)
\end{array}
$$

The max sum is 3.25. The normalized dissimilarity is $GeoLocal(S_1,\ S_2) = 1 - \frac{3.25}{4} = 0.1875$. The signature $s_{2,0}$ (of the node v_1 in G_2) is deleted ($s_{2,0} \to \varepsilon$).

5.2.3 Hierarchical similarity measure

In this section, we present two hierarchical measures that we call $Global_{GeoLocal}$ and $GeoLocal_{Global}$ which are the result of combining the global similarity measure *Global* and the geometric local similarity measure *GeoLocal* depending on the defined order.

Global geometric-Local similarity The Global geometric-Local similarity $Global_{GeoLocal}$ is a hierarchical similarity measure, which aims to measure the distance between two graphs using firstly the global similarity measure *Global*, then using the geometric local similarity measure *GeoLocal* if necessary. The main idea is to measure the distance between the two graphs using *Global*. If the distance obtained is less than a specific threshold, which means that the two graphs are similar accord-

ing to *Global*, then we check this result using *GeoLocal*. Otherwise, the two graphs are not similar, which means that we do not need to use *GeoLocal*. $Global_{GeoLocal}$ aims to enhance time processing of Kite graphs by first computing invariants on the graphs. Formally, let G_1 and G_2 be two graphs, S_1 and S_2 the set of signatures of G_1 and G_2 respectively, $th \in [0,1]$ a threshold and $d_1 = Global(G_1, G_2)$.

$$Global_{GeoLocal}(G_1, G_2) = \begin{cases} d_1, & if\ d_1 > th \\ GeoLocal(S_1,\ S_2), & otherwise \end{cases} \tag{5.11}$$

Geometric-Local Global similarity Like $Global_{GeoLocal}$, the Geometric-Local Global similarity $GeoLocal_{Global}$ is a hierarchical similarity measure, which aims to measure the distance between two graphs using firstly *GeoLocal*, then using *Global* if necessary. The main idea is to measure the distance between the two graphs using *GeoLocal*. If the distance obtained is less than a specific threshold, which means that the two graphs are similar according to *GeoLocal*, then we check this result using *Global*. Otherwise, the two graphs are not similar, which means we do not need to use *Global*. However, only the vertices assigned in the phase of *GeoLocal* will be considered in the second phase using *Global*. In the case where the two graphs have the same number of vertices, all the vertices will be considered in the second phase, *i.e.*, *Global*. $GeoLocal_{Global}$ aims to improve *Global*, based on the graph invariants, in the second level by only considering the assigned vertices in the first level using *GeoLocal*. Formally, let G_1 and G_2 be two graphs, G_2 is the graph prototype, S_1 and S_2 the sets of signatures of G_1 and G_2 respectively, $th \in [0,1]$ a threshold, $d_2 = GeoLocal(S_1, S_2)$ and G'_1 is the subgraph induced by the vertices of G_1 assigned in the first phase using *GeoLocal*.

$$GeoLocal_{Global}(G_1, G_2) = \begin{cases} d_2, & if\ d_2 > th \\ Global(G'_1,\ G_2), & otherwise \end{cases} \tag{5.12}$$

5.2.4 Reconstruction process

Each connected component from the whole graph representing the query image is compared to the set of prototype graphs ($G_{Antenna}$, $G_{Enclosure}$ and four different G_{Kite}), using the proposed similarity measure (*GeoLocal*, *Global*, $Global_{GeoLocal}$ or $GeoLocal_{Global}$). Consequently, each connected component (a query graph) is classified as Kite, a part of Kite or not a Kite nor a part of Kite. When a query graph passes the considered similarity measure (*GeoLocal*, *Global*, $Global_{GeoLocal}$

or $GeoLocal_{Global}$) with more than one connected component classified as a Kite part and at least one of them is classified as an enclosure, we need to know if these Kite parts are parts of the same Kite or belong to different Kites. This is the aim of the reconstruction step that uses the coordinates of the vertices to eventually reconstruct the entire Kite from different components: i.e, enclosure and antennas. The principle is to associate a subset of Kite parts classified as antennas to a Kite part classified as an enclosure, taking into account the distance between them and their orientations. The aggregated similarity Sim_{aggre} of the reconstructed Kite i, is given by:

$$Sim_{aggre}(Kite_i) = \psi * Sim(E_i) + \sum_{j=1}^{j=n} \mu * Sim(A_{i,j}) \tag{5.13}$$

Where: $\psi + \mu = 1$, $Sim(E_i)$ is the similarity attributed to the enclosure of the reconstructed Kite i, n is the number of antennas and $Sim(A_{i,j})$ is the similarity attributed to an antenna j of the reconstructed Kite i.

5.2.5 Complexity study

For the Geometric local similarity measure *GeoLocal*, the most important part, in term of complexity, is the one solving the assignment problem. We used the Hungarian algorithm [43] to find the best assignment in $\mathcal{O}(n^3)$ time, where n is the *maximum* number of vertices in the two compared graphs. Consequently, the time complexity of *GeoLocal* is $\mathcal{O}(n^3)$. The Global similarity measure *Global* is based on a graph invariant, which is linear in terms of computational complexity. Thus, the time complexity of *Global* is $\mathcal{O}(n)$, where n is the *maximum* number of vertices in the two compared graphs.

5.3 Experimental results

For evaluation, we used all the available graphs in the real data set described in Section 4.2.1 and the synthetic data set described in Section 4.2.2. We also used a well-known graph data set of symbols from architectural and electronic drawings named GREC [205], which is one of the data sets of the IAM graph database repository. The GREC data set is composed of 1100 undirected graphs distributed over 22 classes from the original GREC database [259]. The GREC data set is split into a training and a validation set, each of size 286, and a test set of size 528.

5.3. Experimental results

We conducted four series of experiments to evaluate the robustness and accuracy of our similarity measures. The first three series of experiments are realized on the real and synthetic Kite graph database, while the fourth experimentation is realized on the GREC data set. We compared our approach with two approaches from the state-of-the-art based on local structure comparison:

- $GED_{Bipartite}$: a GED based on a bipartite assignment of vertices and their local structures [9].
- $Beam_{GED}$: a simple and fast suboptimal GED based on beam search [48].

The proposed distances *GeoLocal* and *Global* are parameterized distances having a set of parameters α_k allowing different configurations. The default value is: $\sum \alpha_k = 1$ and $ConvexityTh = 150°$. In addition, we defined a threshold in order to improve classification accuracy. The parameters α_k and the threshold may be specified by inspection or by using machine learning techniques. In this chapter, for simplicity, we attribute to all the parameters of our methods and the methods with which we compare ($GED_{Bipartite}$ and $Beam_{GED}$) their default values. However, for each approach we choose the threshold giving the best accuracy. The default parameters of $GED_{Bipartite}$ and $Beam_{GED}$ are: the same cost for vertex/edge deletions/insertions which is 1, the weighting parameters per vertex/edge is the same 1, the same cost for vertices and edges ($vertexCost = edgeCost$) and for $Beam_{GED}$, the size of the OPEN set is 10.
In the first experiment, we show the impact of using the reconstruction process in the obtained accuracy. Table 5.2 depicts the results obtained depending on the use or not of the process of reconstruction ($Threshold = 0.28$). These experiments are conducted on the real Kite data set.

Methods	***Reconstruction***	**State-I**	**State-II**	**State-III**	**State-IV**
Global	Yes	93.87%	96%	83%	77%
	No	89.79%	95%	81%	80%
GeoLocal	Yes	100%	98%	91%	78%
	No	93.87%	93%	87%	78%
$Global_{GeoLocal}$	Yes	91.83%	94%	78%	78%
	No	87.75%	92%	76%	82%
$GeoLocal_{Global}$	Yes	93.87%	95%	85%	78%
	No	89.79%	93%	83%	77%

Table 5.2: The impact of the reconstruction process on the classification

We can see that the four methods are globally more accurate with considering the process of reconstruction. This shows the importance of using the reconstruction process.

In the second series of experiments, we evaluated the accuracy of the proposed approach by performing classification. These experiments are realized on both the real and the synthetic Kite data set. Table 5.3 depicts the results obtained by our approaches and the approaches with which we compare, using the adequate threshold.

Methods	***Threshold***	**State I**	**State II**	**State III**	**State IV**	**Synthetic data set**
Global	0.28	93.87%	96%	83%	77%	98%
GeoLocal	0.28	100%	98%	91%	77%	100%
$Global_{GeoLocal}$	0.28	91.83%	94%	78%	78%	98%
$GeoLocal_{Global}$	0.28	93.87%	95%	85%	78%	98%
$GED_{Bipartite}$	0.40	36.53%	41%	75%	11%	41.3%
$Beam_{GED}$	0.10	20.20%	28%	75%	44.44%	75%

Table 5.3: Classification on the Kite data set

We can see that our approaches *GeoLocal*, *Global*, $Global_{GeoLocal}$ and $GeoLocal_{Global}$ are more accurate than $GED_{Bipartite}$ and $Beam_{GED}$ at all the levels of the real and the synthetic Kite data set. This confirms that considering the geometric form (angles) has a high added value for Kite recognition. We can also see that *GeoLocal* is more accurate than *Global*, $Global_{GeoLocal}$ and $GeoLocal_{Global}$ at all the levels of the real and the synthetic Kite data set. However, $Global_{GeoLocal}$ and $GeoLocal_{Global}$ are slightly better in the negative data set (**State IV**) of the real data set. Although, *GeoLocal* achieves better classification accuracy compared to $Global_{GeoLocal}$ and $GeoLocal_{Global}$. However, use of the hierarchical measures $Global_{GeoLocal}$ and $GeoLocal_{Global}$ avoids unnecessary comparison in the second level, thus the general runtime on the data set is better. We note also that $GeoLocal_{Global}$ achieves better classification accuracy compared to $Global_{GeoLocal}$ at all the levels of the real and the synthetic Kite data sets. However, $Global_{GeoLocal}$ achieves a better general runtime on the data set, due to the fact that *Global* is faster than *GeoLocal*.

In the third series of experiments we evaluated the scalability of our approach over an increasing number of vertices in the query graphs. These experiments are realized on both the real Kite data set and the synthetic Kite data set. From the 4081 graphs of the real Kite data set and 1000 graphs of the synthetic Kite data set, we

constructed a set of query groups with the same number of vertices. The number of vertices vary from 2 vertices to 949 vertices in the real data set and from 30 vertices to 85 vertices in the synthetic data set.

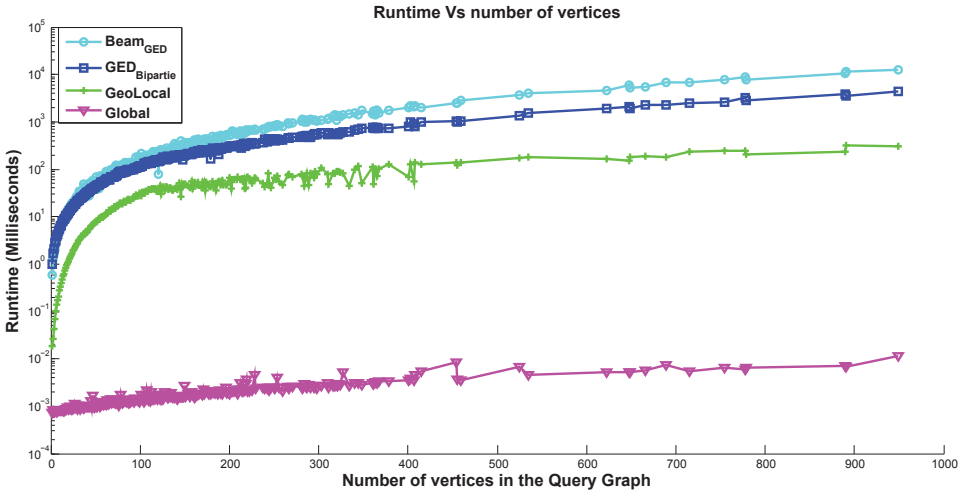
Figure 5.3 shows the average runtime performance of *Global*, *GeoLocal*, $GED_{Bipartie}$ and $Beam_{GED}$ in both the real Kite data set (Figure 5.3 (a)) and the synthetic Kite data set (Figure 5.3 (b)). The X-axis shows the number of vertices contained in the query graph and the Y-axis the average runtime, in log scale, obtained over the query group of the corresponding graph size when compared to the set of Kite prototype graphs. This figure clearly shows the interest of using the global similarity measure *Global*, which is largely faster than the geometric local similarity measure *GeoLocal*. Figure 5.3 also shows that *GeoLocal* is faster compared to $GED_{Bipartie}$ and $Beam_{GED}$. The approaches with which we compare ($GED_{Bipartie}$ and $Beam_{GED}$) are approximatively equivalent with a little difference making $GED_{Bipartie}$ slight faster than $Beam_{GED}$. The runtime performance shown in the figure confirms the theoretical time complexity, which is linear for *Global* and polynomial for *GeoLocal* and $GED_{Bipartie}$. However, *GeoLocal* has a better time complexity, which is $\mathcal{O}((\max(n\ ,\ m))^3)$ compared to $GED_{Bipartie}$ with $\mathcal{O}((n\ +\ m)^3)$, where n and m are the number of vertices of the two compared graphs.

Finally, we evaluated the accuracy of the proposed approach by performing classification on the GREC data set. We compare the results obtained by our approach *GeoLocal* with the results obtained by $GED_{Bipartite}$ and $Beam_{GED}$ in [9]. Table 5.4 depicts the results obtained by our approach *GeoLocal* using the adequate threshold (0.07) and $GED_{Bipartite}$ and $Beam_{GED}$.

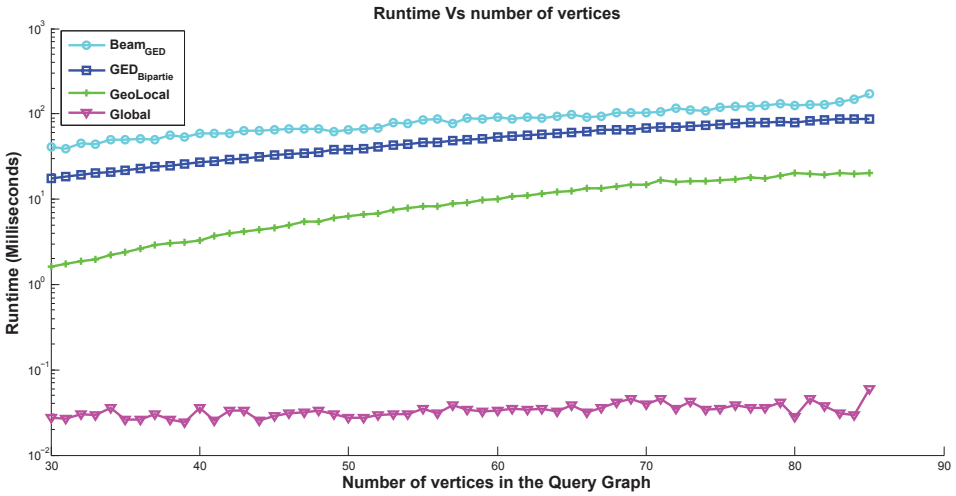
Methods	*GeoLocal*	$GED_{Bipartite}$	$Beam_{GED}$
GREC data set	96.19%	86.30%	76.70%

Table 5.4: Classification on the GREC data set

We can see that our method *GeoLocal* is more accurate than the two methods with which we compare $GED_{Bipartite}$ and $Beam_{GED}$ on the GREC data set. This confirms that considering the geometric form (angles) has a high added value for object recognition with specific geometric structures. This also shows that our method is extensible on other types of data and proves that the proposed approach is quite general.



(a) Real Kite data-set



(b) Synthetic Kite data-set

Figure 5.3: Runtime Vs number of vertices.

5.4 Conclusions

In this chapter, we proposed a novel geometric hierarchical graph matching method based on graph edit distance and graph invariants. The proposed method takes into account the geometric form of the graphs in addition to their structures. Both the theoretical time complexity and the experimental results on real and synthetic Kite data sets confirm the high performance of our approach. Furthermore, the experimentation performed on the GREC data set [205] proves that the proposed

approach is extensible and quite general.

Part II

Inexact graph matching for 3D object recognition

Introduction to Part II

In the field of 3D object recognition, it is often required to compare different 3D objects represented by graphs. Using triangular tessellations, 3D objects may be compared with graph matching techniques. This part addresses the problem of comparing deformable or non-rigid 3D objects (such as human and animal bodies). The shapes considered are represented by graphs, *i.e.*, triangular tessellations. We propose a new distance for comparing deformable 3D objects. This distance is based on the decomposition of triangular tessellations into a set of substructures that we call *triangle-stars*. A triangle-star is a connected component formed by the union of a triangle and its neighborhood. The proposed decomposition offers a parameterizable triangle-stars depending on the degree of the neighborhood considered. The number of triangle-stars obtained is much smaller than the number of nodes and the number of classic stars [9, 10] and, as a result, the computational complexity is reduced. The proposed graph edit distance is based on triangle-stars which is a local structure that covers a larger neighborhood than a classic star decomposition [9, 10]. Consequently, the proposed dissimilarity measure assures an optimal approximation. This is justified by the fact that optimal methods are based on graph's global structures and, consequently, a larger local structure allows to be closer to the global one. The distance that we propose uses a set of parameters which are either invariant or at least oblivious under most common deformations. We prove that the proposed distance is a pseudo-metric. We analyse its time complexity and we present a set of experimental results which confirm the high performance of our approach.

This part contains two chapters, in the first one (Chapter 6), we present the proposed decomposition of triangular tessellations into triangle-stars. We describe the proposed distance (dissimilarity measure). We prove that the proposed distance is a pseudo-metric and we analyse its time complexity. In the second chapter (Chapter 7), we describe the experimentations that we undertook to evaluate our approach. We present the different databases that we use in our experiments, some state of the art shape-matching algorithms to compare with, the evaluation criteria and the experimental results.

Chapter 6

Graph-based approach for non-rigid 3D Object Recognition

Contents

In the field of 3D object recognition, it is often required to compare different 3D objects represented by graphs, *i.e.*, triangular tessellations. Using triangular tessellations, 3D objects may be compared with graph matching techniques. In this chapter, we address the problem of comparing deformable or non-rigid shapes represented by graphs, *i.e.*, triangular tessellations. In order to deal with this problem, we propose a new distance (dissimilarity measure) based on new decomposition of triangular tessellations into a set of triangle-stars. We prove that the proposed distance is a pseudo-metric and we analyse its time complexity.

The remainder of the chapter is organized as follows. Firstly, in Section 6.1, we give an overview of the proposed approach. In Section 6.2, we introduce a set of concepts defining the triangle-stars substructures. In Section 6.3, we describe

the triangle order considered for the proposed decomposition method of triangular tessellations into a set of triangle-stars described in Section 6.4. In Section 6.5, we describe the proposed distance, we analyse its complexity in Section 6.7. Finally, Section 6.8 concludes the chapter.

6.1 Algorithm overview

We propose a new decomposition of triangular tessellations into connected components that we call triangle-stars. This decomposition aims to reduce the number of components while covering a larger number of neighbors. The neighborhood area covered by a triangle-star is parameterizable and depends on the N_k order neighborhood considered in the proposed decomposition. In addition, this decomposition allows obtaining a representation which is invariant or at least oblivious under most common deformations. Prior to the decomposition, a strict total order on the triangles must be established. This order aims to reduce the number of triangle-stars that is generated and guarantees the uniqueness of the resulting decomposition. Finally, we propose a distance (dissimilarity measure) between the triangle-stars of the two triangular tessellations and address their matching. We also prove that the proposed distance is a pseudo-metric. We present the computational complexity of the proposed algorithm.

6.2 Graph decomposition

We propose a decomposition of a triangular tessellation graph into a set of connected components that we call triangle-stars (TS). We define the concept of triangle-star as follows:

Definition 1 (Triangular tessellations): A triangular tessellation g_{Tr} is a graph defined by a set of nodes, edges and triangles. Formally, g_{Tr} is a graph defined by a six tuple $g_{Tr} = (N, E, T, \alpha, \beta, \theta)$, where T is a set of triangles and $\theta : T \rightarrow L_T$ is a labelling function and, L_T is the set of triangle labels.

Definition 2 (neighborhood of a triangle): two triangles are neighbors, if they share, at least, a common node. Let t_1 and t_2 two triangles and $N(t_1)$ and $N(t_2)$ their respective nodes. Then, t_1 and t_2 are *neighbors* $\Leftrightarrow \|N(t_1) \cap N(t_2)\| > 0$. In

other words, the neighbors of a triangle t are triangles sharing at least a common node with the triangle t.

Definition 3 (N_k-neighborhood of a triangle): two triangles t_0 and t_k are N_k-neighbors, if there is between t_0 and t_k a chain of $(k-1)$ distinct triangles, at most, which are pairwise consecutive neighbors. Formally, t_0 and t_k are N_k-*neighbors* $\Leftrightarrow \exists\ t_{i=1..k-1}$ *where* : $\forall\ i = 1...(k-1)$, t_i *and* t_{i+1} *are neighbors.* In other words, the N_k-neighbors of a triangle t are the k recursive triangle neighbors of t. In the case of $k = 1$, the N_k-neighborhood is equivalent to a simple neighborhood (Definition 2).

Definition 4 (triangle-star): A triangle-star ts is a labelled sub-graph, defined by a triangle and a set of its neighbors. Formally, a triangle-star ts is a three tuple $ts = (t_r, T', \theta)$, where: t_r is the root triangle, T' is the set of adjacent triangles and $\theta : T \rightarrow L_T$ is the triangle labelling function while L_T is a set of labels associated with the triangles.

Definition 5 (N_k-triangle-star): A N_k-triangle-star N_k-ts is triangle-star defined by a triangle and a set of its N_k-neighbors. In the case of $k = 1$, the N_k-triangle-star is a simple triangle-star (Definition 4). See Example 1.

Example 1. Let a graph-tessellation G_{tr} containing 17 triangles $t_{0...16}$. Table 6.1 shows the graph-tessellation G_{tr} and the N_k-triangle-stars of the triangle t_0 depending on the $N_{k=0...2}$ neighborhood considered. In the case of $N_{k=0}$, the graph-tessellation is decomposed into 17 triangle-stars (17 N_0-TS, exactly the same number of triangles). Each triangle-star is constituted only by one triangle without any triangle neighbors. Thus, the resulted triangle-star N_0-TS of the triangle t_0 is constituted only by the triangle t_0. In the case of $N_{k=1}$, only the direct triangle neighbors are considered which means that the graph-tessellation is decomposed to a set of triangle-stars constituted using a triangle with its neighbors (See Definition 2). Thus, the resulted triangle-star N_1-TS of the triangle t_0 is constituted by the triangle t_0 and its N_1-neighbors (direct neighbors), *i.e*, $T(N_1\text{-}TS) = \{t_{0...7}\}$. In the case of $N_{k=2}$, the N_2-neighbors are considered which means that the graph-tessellation is decomposed into a set of triangle-stars constituted using a triangle with its N_2-neighbors (See Definition 3). Thus, the resulted triangle-star N_2-TS

of the triangle t_0, is constituted by the triangle t_0 and its N_2-neighbors (second degree neighborhood). $T(N_2\text{-}TS) = \{t_{0...16}\}$. In this example, $T(N_2\text{-}TS)$ is the all graph-tessellation G_{tr}.

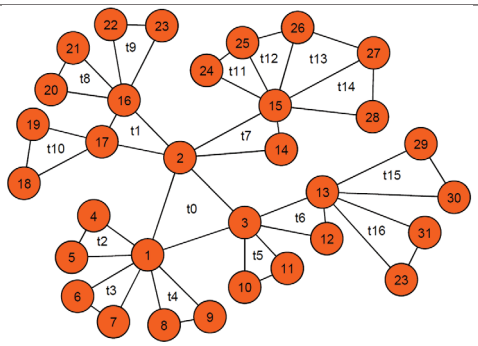
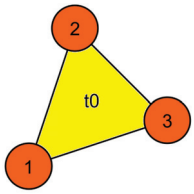
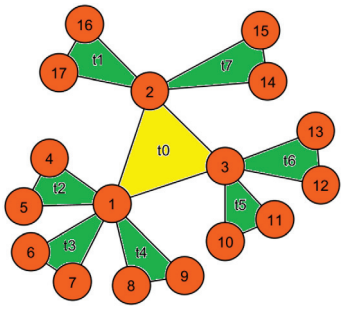
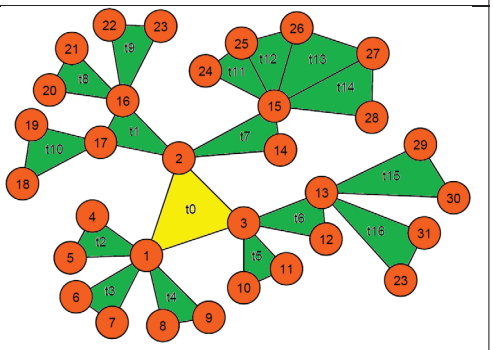
Graph G_{tr}	The N_0-TS of the triangle $t0$
	
The N_1-TS of the triangle $t0$	The N_2-TS of the triangle $t0$
	

Table 6.1: Example, N_k-triangle-stars.

Triangle-star features: Each triangle t_j is defined with six-tuple $t_j = (n_1, n_2, n_3, e_1, e_2, e_3)$. The nodes n_i are labelled by their Cartesian coordinates. In our case, the nodes n_i are labelled with three coordinates $n_i = (x, y, z)$ corresponding to the three dimensions. The edges $e_k = (n_p, n_w)$ are labelled (weighted) with the Euclidian distance between their associated nodes (n_p, n_w). The triangles are labelled by a three-tuple $t_j = (id, Area, Perimeter)$, where id is a number. Each triangle-star is characterised by a set of descriptors, allowing the evaluation of the dissimilarity between triangle-stars. We consider the following descriptors: Area of triangle-star, Perimeter of triangle-star, Area of the triangles forming the triangle-stars, their Perimeters, the Weights associated with their edges, and the Degrees of their nodes.

Our choice of the set of geometrical descriptors like the weights of edges and the area and the perimeter of the triangles and the triangle-stars, is justified by the fact that those geometrical descriptors are oblivious under different deformations and if necessary invariants (as required in some fields) by normalizing the weights of the edges and consequently the area and the perimeter. The normalization is realized according to the maximum edge's weight W_{MAX}.

Triangle-star Vector representation: A vector representation is associated to each triangle-star, the vector contains the global area AG of the triangle-star, the global perimeter PG of the triangle-star, the area A and the perimeter P of each triangles of the triangle-star, the weights of edges W of the triangle-star and the degrees of nodes deg of the triangle-star. The vector representation of triangle-star is given by (see Table 6.5 for symbols descriptions):
$\{AG(ts),\ PG(ts),\ \{A(t_i),\ P(t_i),\ W(t_{i,\ j=1...3}),\ \ deg(t_{i,\ j=1...3})\}_{i=1}^{i=\|T(ts)\|}\}$
where:

- The triangles of triangle-star ts are ranked according to their areas (descending order).
- The weights of edges are ranked by descending order.
- The degrees of nodes are ranked by descending order.
- All triangle-stars TS are represented by vectors which has the same size : $size = 2 + (\Gamma * 8)$.
- Γ is the maximum number of triangles in the two sets of triangle-stars in the two compared graphs.
- If a triangle-star ts has a number of triangle less than Γ , the rest of the vector is completed with $zeros$.

Definition 6 (Disjoint triangle-stars): Two triangle-stars ts_i and ts_j are $disjoints$, if they do not share a common triangle. let $i \neq j$, ts_i and ts_j are $disjoints \Rightarrow T(ts_i) \cap T(ts_j) = \varnothing$.

Definition 7 (Geodesic Distance): The geodesic distance is the length of the shortest path $\lambda \in \Lambda$ (with Λ is the set of all paths) between two points p_i and p_j.

The considered path must entirely lie within the object [260]. The geodesic distance GD is defined as follow:

$$GD_{\lambda}(p_i,\ p_j)\ =\ \underset{\lambda\ \in\ \Lambda}{\arg\min}\ \int_{\lambda} ds$$

6.3 Triangles ordering

The proposed method of decomposition, allows to have disjoint triangle-stars (Definition 6), which significantly reduces the number of components ($\|TS\| \ll \|N\| < \|T\|$, see Figure 6.1) while reducing the number of comparisons in between the triangle-stars associated with the two triangular tessellations. However, according to the order considered, the set of triangle-stars obtained may be different (see Example 2). Indeed, the same triangular tessellation may generate different sets of triangle-stars, both in terms of cardinality and in terms of triangle-stars obtained, if the ordering of the triangles is not the same.

In order to ensure the **uniqueness** of the decomposition and a further reduced number of triangle-stars, a descending strict total order must be established on the set of triangles prior to their decomposition into triangle-stars (see Example 3).

The Figure 6.2 presents the average number of TS (N_1-TS) obtained in the TOSCA database [261, 262] (See Section 7.1.1 for TOSCA database description) depending on the triangles order considered. This plot shows that considering a $\|Neighbours\ (Triangles)\|$ *descending order* give the minimum number of triangle-stars TS, which reduces the complexity of the proposed algorithm.

In order to establish a descending strict total order on the triangles set, each triangle is represented by a 10-tuple $< \|neighbors\|\ ;\ x_1, y_1, z_1\ ;\ x_2, y_2, z_2\ ;\ x_3, y_3, z_3 >$: the number of neighbors and the coordinates x, y, z, in the reference frame defined by the Eigen vectors of the tensor of inertia associated with the tessellation, of the three nodes associated with the triangle. The number of neighbors $\|neighbors\|$ is used in order to further reduce the number of triangle-stars. If two triangles have the same number of neighbors, the node's coordinates are utilised in order to ensure the uniqueness of the decomposition. The nodes of the triangle in the 10-tuple are ordered according to their coordinates, starting by the first coordinate x. In case of equality, the next coordinates are compared until an inequality is obtained. The coordinates of the nodes are solely considered in order to ensure the uniqueness the decomposition. The order of triangles may be affected by the pose changing due to the node's coordinates changing, consequently the obtained triangle-stars may

be different. However, the number of triangle-stars and their cardinality are the same, because the triangles ordering is based mainly on the number of neighbors. The node's coordinates are only used when two triangles have the same number of neighbors (See Example 4). A solution to avoid that the order of triangles be affected by the pose changing is to consider the coordinates x, y, z, in the reference frame defined by the Eigen vectors of the tensor of inertia associated with the tessellation.

Another solution, not affected by the object rotation, ensuring a descending strict total order on the set of triangles, is based on the selection of six nodes as a landmark reference and the use of geodesic distance (See **Definition 7**). The first step is to select six nodes $n_{1...6}$. The two nodes n_1 and n_2 are selected as the two nodes which have the largest geodesic distance $GD(n_1,\ n_2)$. The two nodes n_1 and n_2 represent the first axis that we call X. The nodes n_3 and n_4 are selected as the two nodes which have the largest geodesic distance $GD(n_3,\ n_4)$ in the perpendicular plane to the axis X. The two nodes n_3 and n_4 represent the second axis that we call Y. The nodes n_5 and n_6 are selected as the two nodes which have the largest geodesic distance $GD(n_5,\ n_6)$ in the perpendicular plane to the axis Y. The two nodes n_5 and n_6 represent the third axis that we call Z. The second step is to represent each node n_i using the geodesic distance regarding the six nodes references selected $n_{1...6}$ as a 6-tuple $R_i = < GD(n_i,\ n_1), GD(n_i,\ n_2), GD(n_i,\ n_3), GD(n_i,\ n_4), GD(n_i,\ n_5), GD(n_i,\ n_6) >$. The third step is to represent each triangles by a 19-tuple $< \|neighbors\| ; R_1 ; R_2 ; R_3 >$ constituted by the number of neighbors $\|neighbors\|$ and the three 6-tuples $R_{i=1...3}$ of the three nodes composing each triangle.

We can obtain from the same object a different tessellation depending on the tessellation algorithm used and its parametrisation. A different tessellation, even for the same object, may generate a different triangle-star decomposition. However, we suppose that the tessellation of objects is realized using the same algorithm with the same parameters. This ensures that same object have one tessellation representation, thus we obtain a unique triangle-star decomposition. In the worst case (the same object have a different tessellation), the proposed distance based on an approximation of Graph Edit Distance which is fault-tolerant to noise and distortion, allows to consider the two tessellations as the same (the distance between them is negligible).

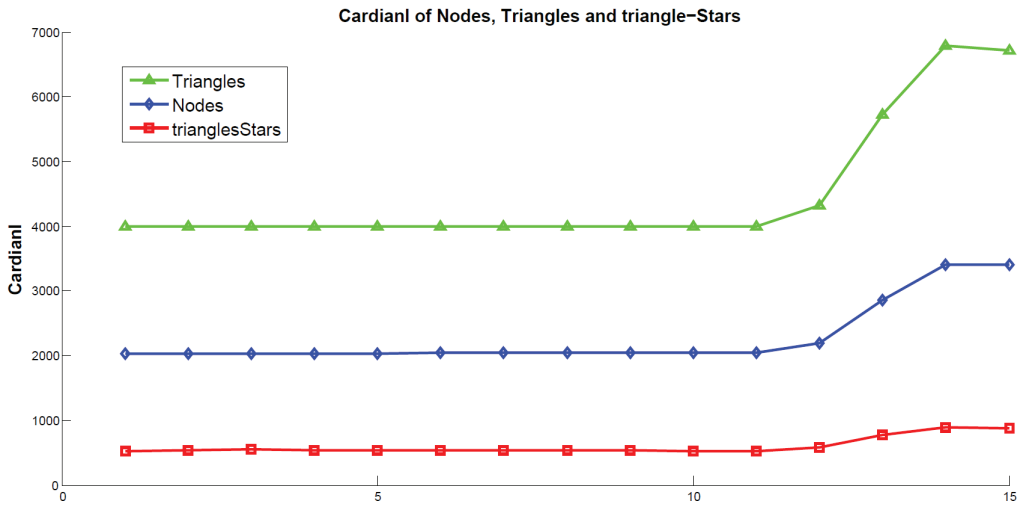


Figure 6.1: Comparison of the average number of nodes, triangles and triangle-stars in the TOSCA database.

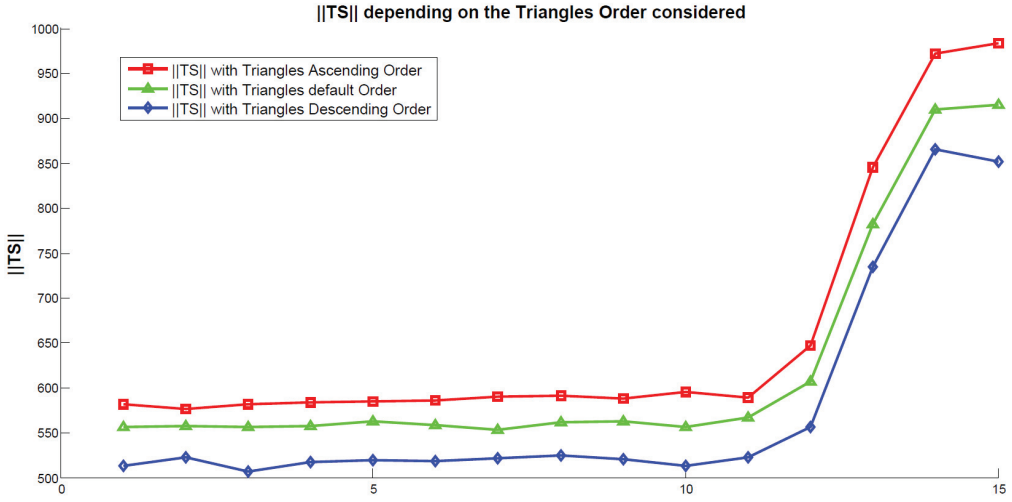


Figure 6.2: Comparison of the average of $\|TS\|$, depending on triangles order considered, in TOSCA database.

Example 2. Let a graph-tessellation G_{tr} containing 5 triangles $t_{1...5}$, with $\|N(t_1)\| = 1$, $\|N(t_2)\| = 3$, $\|N(t_3)\| = 2$, $\|N(t_4) = 1\|$ and $\|N(t_5)\| = 1$, where N is the triangle direct neighborhood (see Figure 6.3). By applying a descending order based on the number of neighbors $\|N(t_i)\|$, we obtained 2 triangle-stars (see Table 6.2). And by applying the ascending order based on the number of neighbors $\|N(t_i)\|$, we

obtained 3 triangle-stars (see Table 6.3).

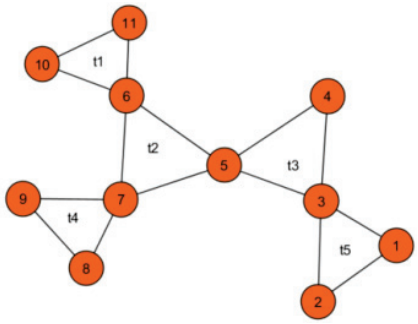


Figure 6.3: A graph-tessellation G_{tr}

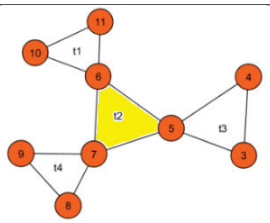
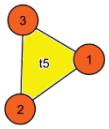
ST_1	ST_2
	

Table 6.2: The set of triangle-stars using a Descending order.

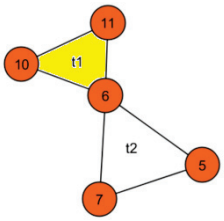
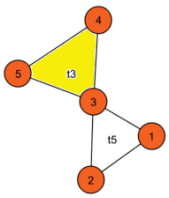
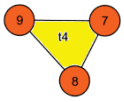
ST_1	ST_2	ST_3
		

Table 6.3: The set of triangle-stars using a Ascending order.

Example 3. Let a triangle t_0 with n triangle neighbors $t_{1...n}$ (see Figure 6.4). If we consider a triangle order based on the number of neighbors with descending order, we obtained only one triangle-star, otherwise, if the process of decomposition starts with any neighbors $t_{1...n}$ we obtained 3 triangle-stars.

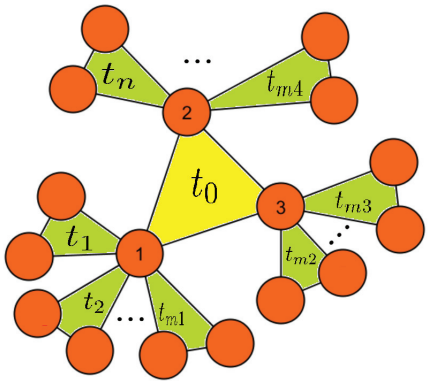


Figure 6.4: The triangle t_0 with n triangles neighbors.

Example 4. Let a graph-tessellation G containing 6 triangles $t_{0...5}$, with $\|N(t_0)\| = \|N(t_1)\| = 3$ and $\|N(t_{2...5})\| = 1$, where N is the triangle direct neighborhood (see Figure 6.5). When applying a descending order, we remark that t_0 and t_1 have the same number of neighbor which is the highest one. In this case, we use the coordinates of nodes to decide with which triangle we start t_0 or t_1. However, whatever starting with t_0 or t_1, we obtain the same number of triangle-stars with the same cardinality as shown in (Figure 6.5).

6.4 Triangle-stars decomposition

Once the strict total order of the triangles is established, we evaluate the decomposition of the graph into triangle-stars. The process of decomposition is presented in the following algorithm (Algo. 6.1).
We explore the list of triangles according to the defined order and we construct a N_k-triangle-star which is defined by the current triangle and its N_k-neighbors (Definition 4), according to the degree of neighborhood N_k considered. The process terminates when the list of triangles not yet explored is empty.

The proposed decomposition of triangular tessellations into triangle-stars offers a reduced number of triangle-stars ts as opposed to the number of nodes $\|TS\| \ll \|N\|$. The resulting triangle-stars are disjoints, $\forall\ ts_i,\ ts_j \in TS(G),\ i \neq j \ \Rightarrow\ T(ts_i) \cap T(ts_j) = \emptyset$. The triangle-star covers a larger local area than the classical star [9]. The proposed decomposition is also a parameterizable method depending on the degree of the neighborhood. Considering a high degree of neighborhood allows to

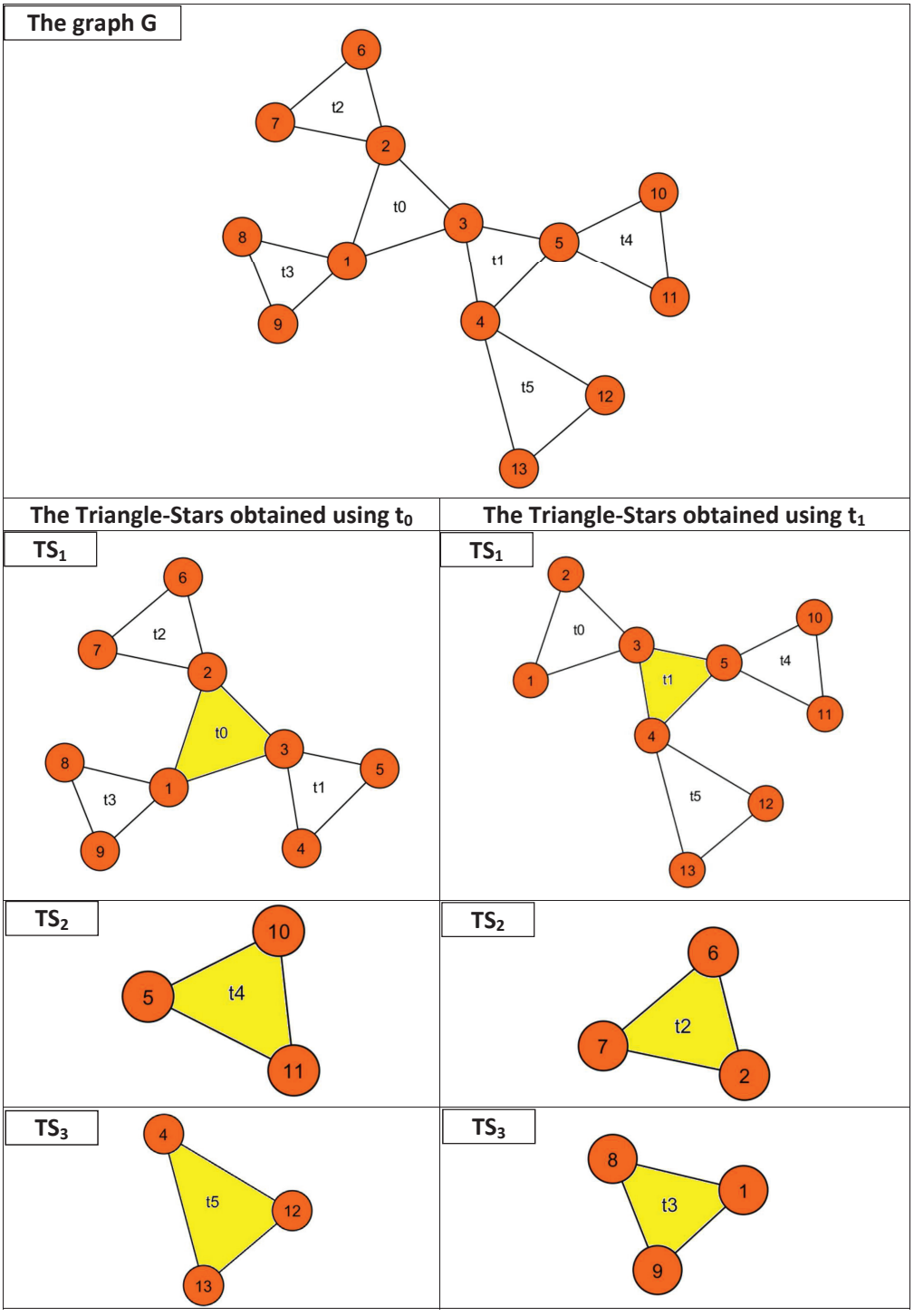


Figure 6.5: Example of graph decomposition with pose changing.

have a much smaller number of triangle-stars ($\|N_{k+1}\text{-}TS\| \leq \|N_k\text{-}TS\|$) covering a larger area of neighborhood ($\|T(N_{k+1}\text{-}TS)\| \geq \|T(N_k\text{-}TS)\|$). In addition, the proposed decomposition is unique.

Algorithm 6.1: Graph decomposition into N_k-triangle-stars.

Inputs: A graph g_{Tr} and the degree of neighborhood N_k.
Outputs: A set of N_k-triangle-stars (N_k-TS).
Begin
    Apply a **descending** strict total order on the set of triangles of g_{Tr};
    $N_k\text{-}TS = \varnothing$;
    while $(T(g_{Tr}) \neq \varnothing)$ **do**
        $t_i = T(g_{Tr})[0]$
        $T(N_k\text{-}ts_i) = t_i \cup N_k\text{-}\mathbf{neighbors}(t_i)$;
        $N_k\text{-}TS = N_k\text{-}TS \cup N_k\text{-}ts_i$;
        $T(g_{Tr}) = T(g_{Tr}) - T(N_k\text{-}ts_i)$;
    end while
    return TS ;
End

Example 5. Let us consider a triangular tessellation defined as follow: $g_{Tr} = \{$ 16 *nodes* , 20 *triangles* $t_{1...20}$ $\}$ (Table 6.4). The decomposition into triangles-Stars (N_1-TS) begins by constructing the first triangle-star TS_1 using the triangle t_{13} with the set of its triangle neighbors (N_1-neighbors). The triangle t_{13} is the first triangle chosen, since it is the one having the maximum number of neighbors, which is 12. In the remaining set of triangles not used in the construction of TS_1, the triangle t_1 that had 7 neighbors which is the maximum, is used to construct the second triangle-star TS_2. TS_2 is constructed using t_1 and its 3 neighbours (not 7, because $T(TS_1) \cap T(TS_2) = \varnothing$). The third triangle-star TS_3 is formed of t_{11} and its neighbors, t_{11} had 5 neighbors which is the maximum in the remaining set of triangles. TS_3 is constructed using t_{11} and its 2 neighbours (not 6 neighbours, because $\bigcup_{i=1}^{i=3} T(TS_i) = \varnothing$). In the case of N_2-neighborhood, we obtain one triangle-star N_2-TS_1 constituted by the triangle t_{13} and the set of its triangle N_2-neighbors which represent all the triangles of g_{Tr}. Consequently, N_2-TS_1 is exactly the triangular tessellation g_{Tr}.

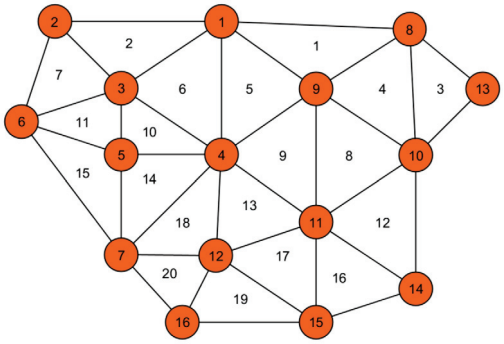
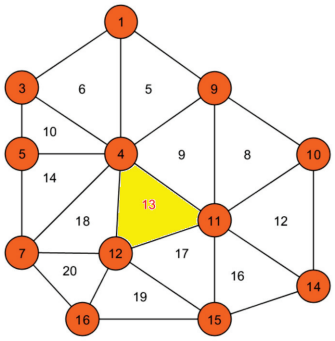
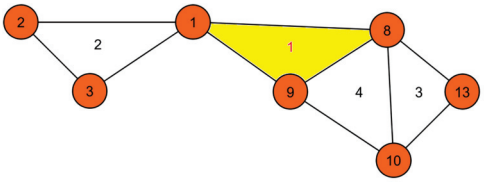
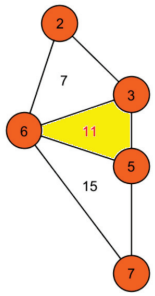
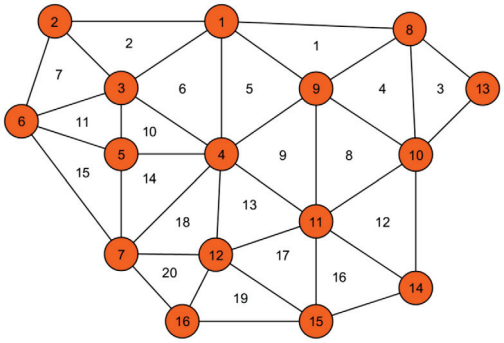
Graph g_{Tr}	TS_1
	
TS_2	TS_3
	
N_2-TS_1	
	

Table 6.4: Example, decomposition of a graph into a set of triangle-stars and N_2-triangle-stars.

6.5 Edit distance between triangle-stars and triangular tessellations

In this section, we show how to compute the graph edit distance between triangle-stars and between two triangular tessellations.

6.5.1 Edit distance between triangle-stars

The proposed similarity measure is intended for the comparison of deformable objects. Consequently, the set of descriptors must be invariant or at least oblivious under most common deformations. Indeed the proposed similarity measure is based on the following set of parameters: *Area* of **triangle-star**, *Perimeter* of **triangle-star**, *Area* of triangles, *Perimeter* of triangles, *Weights* of edges and *Degrees* of nodes. Formally a **triangle-star** is represented as follows: $\{AG(ts),\ PG(ts),\ \{A(t_i),\ P(t_i),\ W(t_{i,\ j=1...3}),\ deg(t_{i,\ j=1...3})\}_{i=1}^{i=\|T(ts)\|}\}$. The similarity measure d between two triangle-stars ts_i and ts_j is computed as:

$$d(ts_i, ts_j) = 1 - \frac{\sum_{k=1}^{k=6} \ sim_k\ (ts_i, ts_j)}{\sum_{k=1}^{k=6} \ \alpha_k} \tag{6.1}$$

The similarity measure d is a normalized value ($0 \leq d \leq 1$) based on the set of functions sim_k.

The functions sim_k are defined as follow:

$$sim_1(ts_i, ts_j) = \alpha_1 * \frac{|AG(ts_i)\ -\ AG(ts_j)|}{AG_{MAX}} \tag{6.2}$$

$sim_1(ts_i, ts_j)$ compares the Area AG of the two triangle-stars ts_i and ts_j.

$$sim_2(ts_i, ts_j) = \alpha_2 * \frac{|PG(ts_i)\ -\ PG(ts_j)|}{PG_{MAX}} \tag{6.3}$$

$sim_2(ts_i, ts_j)$ compares the Perimeter PG of the two triangle-stars ts_i and ts_j.

$$sim_3(ts_i, ts_j) = \alpha_3 * \frac{\sum_{l=1}^{l=\Gamma} |\ A(T(ts_i)_l)\ -\ A(T(ts_j)_l)\ |}{A_{MAX} * \Gamma} \tag{6.4}$$

$sim_3(ts_i, ts_j)$ compares the Area of each triangle A of the two triangle-stars ts_i and ts_j.

$$sim_4(ts_i, ts_j) = \alpha_4 * \frac{\sum_{l=1}^{l=\Gamma} |\ P(t_{i,l})\ -\ P(t_{j,l})\ |}{P_{MAX} * \Gamma} \tag{6.5}$$

$sim_4(ts_i, ts_j)$ compares the Perimeter of each triangle P of the two triangle-stars ts_i and ts_j.

$$sim_5(ts_i, ts_j) = \alpha_5 * \frac{\sum_{l=1}^{l=\Gamma} \sum_{k=1}^{k=3} |\ W_{i,l,k}\ -\ W_{j,l,k}\ |}{3 * W_{MAX} * \Gamma} \tag{6.6}$$

$sim_5(ts_i, ts_j)$ compares the weights of each edge W of the two triangle-stars ts_i and ts_j.

$$sim_6(ts_i, ts_j) = \alpha_6 * \frac{\sum_{l=1}^{l=\Gamma} \sum_{k=1}^{k=3} \mid Deg_{i,l,k} \; - \; Deg_{j,l,k} \mid}{3 * Deg_{MAX} * \Gamma} \tag{6.7}$$

$sim_6(ts_i, ts_j)$ compares the Degree of each Node Deg of the two triangle-stars ts_i and ts_j.

The symbols associated with the similarity measure are described in Table 6.5.

Symbol	Description
$t_{i,l}$	The triangle t_l in the triangle-star $ts_i : t_l \in ts_i$
$W_{i,l,k}$	The weight (Euclidian distance) of the edge e_k of the triangle $t_l \in ts_i$
$Deg_{i,l,k}$	The degree of node n_k of the triangle $t_l \in ts_i$
Γ	Max number of triangles in the set triangle-stars of the two graphs g_1 and g_2.
$\alpha_{k=1...6}$	Parameters associated with the descriptors $\alpha_k \in \mathbb{N}$ and $\sum_{k=1}^{k=6} \quad \alpha_k > 0$
$A(t_i)$	Area of the triangle i.
$P(t_i)$	Perimeter of the triangle i.
$AG(ts_i)$	Area of the triangle-star i. $AG(ts_i) = \sum_{j=1}^{j=\|T(ts_j)\|} A(t_j)$
$PG(ts_i)$	Perimeter of the triangle-star i. $DG(ts_i) = \sum_{j=1}^{j=\|T(ts_j)\|} D(t_j)$

Table 6.5: Symbols associated with the similarity measure and theirs description.

Scale invariance. The proposed similarity measure d (Eq. 6.1) is sensitive to the scale changing, this means that two triangle-stars ts_i and ts_j having the same structure with different scale, are not similar $d(ts_i, ts_j) < 1$. In the case that we want to ignore the scale changing (as required in some fields), the weights W of the edges of the compared graphs are normalized relativizing to the maximum edge's weight W_{MAX}. Formally $\forall i, W_i = \frac{W_i}{W_{MAX}}$.

Example 6. Let consider two triangle tessellations gT_{r1} and gT_{r2} having the same structure and composed by one triangle. Consequently, the triangle-stars ts_1 and ts_2 obtained are constituted by one triangle $\|T(ts_1)\| = \|T(ts_2)\| = 1$ (See Table 6.6). Even ts_1 and ts_2 have the same structure, the weights of edges are different $W(ts_1) = \{10, 15, 20\}$ and $W(ts_2) = \{30, 45, 60\}$. Consequently,

$d(ts_i, ts_j) < 1$, which means that ts_1 and ts_2 don't represent the same triangle-star. In the case where the weights of edges are normalized: $W_{MAX1} = 20$ and $W_{MAX2} = 60$ thus $W'(ts_1) = \{10/20, 15/20, 20/20\} = \{0.5, 0.75, 1\}$ and $W'(ts_2) = \{30/60, 45/60, 60/60\} = \{0.5, 0.75, 1\}$. The normalized weights of edges are the same $W'(ts_1) = W'(ts_2)$ and ts_1 and ts_2 have the same structure (See Table 6.6), consequently $d(ts_i, ts_j) = 1$, which means that ts_1 and ts_2 (with normalized weights) represent the same triangle-star.

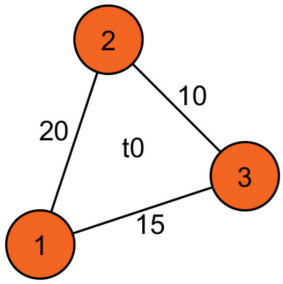
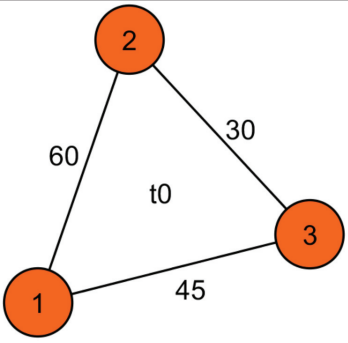
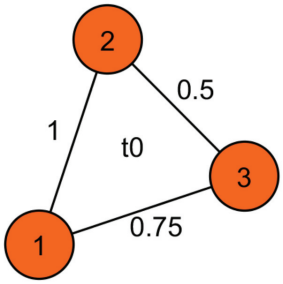
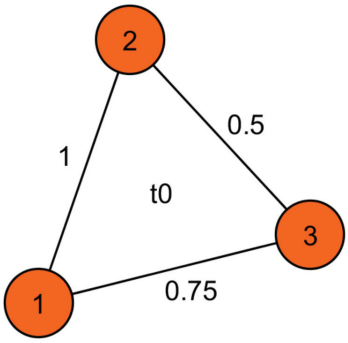
ST_1	ST_2
	
ST_1 with normalized weights	ST_2 with normalized weights
	

Table 6.6: Example of triangle-stars with and without normalized weights.

Deformable objects. The aim of the proposed similarity measure d (Eq. 6.1) is to measure the similarity between **non-rigid** (**deformable**) objects. However, it can be adapted for the comparison of **rigid** objects by adding *Angles* to the set of descriptors. We associate an angle to each pair of adjacent edges. Consequently, a set of modifications are realized as follow:

- The vector representation of triangle-star is modified as follow (see Table 6.5 for

symbols descriptions): $\{AG(ts),\ PG(ts),\ \{A(t_i),\ P(t_i),\ W(t_{i,\ j=1...3}),\ deg(t_{i,\ j=1...3}),\ Ang(t_{i,\ j=1...3})\}_{i=1}^{i=\|T(ts)\|}\}$. Where the angles Ang are ranked by descending order. The size of the vector is increased $size = 2 + (\Gamma * 11)$.

- A new similarity $sim_7(ts_i, ts_j)$ comparing the Angles of the two triangle-stars ts_i and ts_j is defined as follow:

$$sim_7(ts_i, ts_j) = \alpha_7 * \frac{\sum_{l=1}^{l=\Gamma} \sum_{k=1}^{k=3} \mid Ang_{i,l,k} \ - \ Ang_{j,l,k} \mid}{3 * Ang_{MAX} * \Gamma} \tag{6.8}$$

  With Ang_{MAX} is the maximum angle (See Table 6.5 for the rest of symbols).

- The proposed similarity measure d (Eq. 6.1) is modified as follow:

$$d(ts_i, ts_j) = 1 - \frac{\sum_{k=1}^{k=7} \ sim_k\ (ts_i, ts_j)}{\sum_{k=1}^{k=7} \ \alpha_k} \tag{6.9}$$

6.5.2 Edit distance between two triangular tessellations

The dissimilarity between two graphs represented by **triangle-stars** is addressed in the last part of the algorithm. We call this dissimilarity measure *triangle-star Measure* TSM which aims to determine the best matching between the **triangle-stars** associated with two graphs. The dissimilarity between two sets of **triangle-stars** is defined as follows:

Definition 7 (TSM) Let g_{Tr1} and g_{Tr2} be two triangular tessellations, TS_1 and TS_2 their corresponding sets of triangle-stars and M the set of all possible matching between TS_1 and TS_2. The similarity $TSM(TS_1, TS_2)$ (normalised similarity) is formulated as follow (Eq. 6.10):

$$TSM(TS_1,\ TS_2) = 1 - \frac{\max_{m \in M} \sum_{ts_i\ \in\ TS_1,\ m(ts_i)\ \in\ TS_2} d(ts_i,\ m(ts_i))}{\max(\|TS_1\|,\ \|TS_2\|)} \tag{6.10}$$

The computation of $TSM(TS_1, TS_2)$ is equivalent to solving the assignment problem which is one of the fundamental combinatorial optimization problems that aim to find the minimum/maximum weight matching in a weighted bipartite graph. To solve this assignment problem, we define a $n \times n$ matrix D, with $n = \max(\|TS_1\|,\ \|TS_2\|)$. Each element $D_{i,j}$ of the matrix represents the dissimilarity measure $d(ts_i,\ ts_j)$ between a triangle-star ts_i in TS_1 and a triangle-star ts_j in TS_2. In the case of $\|TS_1\| \neq \|TS_2\|$, the smallest set of triangle-stars is completed by $(max(||TS_1||, ||TS_2||)-$

$min(||TS_1||, ||TS_2||))$ empty triangle-stars ε. The similarity between an empty triangle-star ε and a triangle-star ts is computed by the equation (Eq. 6.1) and correspond to the cost of adding ts to the small set of triangle-stars (or to delete ts from the large set of triangle-stars).

We apply the Hungarian algorithm [43] on the matrix D in order to find the best assignment in $\mathcal{O}(n^3)$ time.

The resulting distance (dissimilarity) is compared to a threshold $th \in [0, 1]$ defined by an expert or by experimentation (depending on the database), in order to decide if the compared triangular tessellations are similar or not.

The general process of computing the distance between two graphs (triangle tessellations), is summarized in the following algorithm (Algo. 6.2).

Example 7. Let TS_1 and TS_2 two set of triangle-stars, $\|TS_1\| = \|TS_2\| = 4$. Let D the matrix of similarities between TS_1 and TS_2.

$$\begin{array}{c} \\ ts_{1,0} \\ ts_{1,1} \\ ts_{1,2} \\ ts_{1,3} \end{array} \begin{array}{c} \begin{array}{cccc} ts_{2,0} & ts_{2,1} & ts_{2,2} & ts_{2,3} \end{array} \\ \begin{pmatrix} 0.11 & \mathbf{0.90} & 0.25 & 0.21 \\ 0.10 & 0.15 & \mathbf{0.65} & 0.89 \\ \mathbf{0.67} & 0.03 & 0.51 & 0.17 \\ 0.66 & 0.88 & 0.33 & \mathbf{0.99} \end{pmatrix} \end{array}$$

The max sum, $similarity = 3.21$. The normalised dissimilarity (edit distance) is $TSM(TS_1\ ,\ TS_2) = 1 - \frac{3.21}{4} = 0.1975$

Algorithm 6.2: The distance between two graphs using TSM.

Inputs: Two graphs g_1 and g_2.
Outputs: The distance between g_1 and g_2.
Begin
    Decomposition of g_1 into a set of triangle-stars TS_1, (Algo. 6.1).
    Decomposition of g_2 into a set of triangle-stars TS_2, (Algo. 6.1).
    Construct a matrix of distance D.
    For each $ts_i \in TS_1$ and $ts_j \in TS_2$ do
        $D_{i,j} = d(ts_i,\ ts_j)$ (Eq. 6.1);
    end For each
    Solving (Eq. 6.10), by applying the Hungarian algorithm [43] on the matrix D.
    return the distance $TSM(TS_1,\ TS_2)$;
End

6.6 The pseudo metric

In this section we prove that the proposed distance is a **pseudo-metric**.

Definition : Let X a set of objects and $x, y, z \in X$. Let f be a function defined as follow $f : X \times X \longrightarrow \mathbb{R}$.
Let the following set of properties:

1. **non-negativity**: $f(x, y) \geq 0$
2. **symmetry**: $f(x, y) = f(y, x)$
3. **triangle inequality**: $f(x, y) \leq f(x, z) + f(z, y)$
4. **uniqueness**: $f(x, y) = 0 \Rightarrow x = y$

The function f is a metric if f satisfies the four mentioned properties and f is a pseudo-metric if f satisfies only the first three properties (1, 2 and 3).

Since f is a pseudo metric, a distance function may be defined between each pair of graphs. As a result, the similarity of the objects associated with these graphs may be efficiently determined. Concretely, f is pseudo metric means that f return a correct value ($f(X, Y) \geq 0$ and $f(X, Y) = f(Y, X)$) and f makes searching more efficient in the database using the triangle inequality [263, 264].

Lemma The proposed similarity measure TSM (Eq. 6.10) between two sets of triangles-stars TS_1 and TS_2 is a **pseudo-metric**.

Proof: From (Eq. 6.10) it may be concluded that if TSM is a **pseudo-metric** then d (Eq. 6.1) is a **pseudo-metric** which implies that sim_k (Eq. 6.2) is a **pseudo-metric**. Consequently, we shall prove that sim_k (Eq. 6.2) is a **pseudo-metric**. Proving that sim_k (Eq. 6.2) is a **pseudo-metric** is equivalent to check the first three properties in sim_k (Eq. 6.2). The functions sim_k are defined as follows:

$sim_k = \alpha_k * \frac{|x_1 - x_2|}{\beta}$ with $x_1, x_2, \in R_{\geq 0},\ \alpha_k, \beta \in R_{>0}$.

1. **non-negativity**: $TSM(TS_1, TS_2) \geq 0$. We have $sim_k \geq 0 \Rightarrow TSM \geq 0$ Thus TSM is non-negative.

2. **symmetry**: $TSM(TS_1, TS_2) = TSM(TS_2, TS_1)$. The proposed decomposition is unique and the TSM is only based on symmetrical operations (addition and subtraction in absolute value). Consequently TSM is symmetric.

3. **triangle inequality**: $TSM(TS_1, TS_2) \leq TSM(TS_1, TS_3) + TSM(TS_3, TS_2)$. We have the triangle inequality verified in: $|x_1 - x_2| \leq |x_1 - x_3| + |x_3 - x_2|$. Thus the triangle inequality is verified in sim_k therefore, we have: $TSM(TS_1, TS_2) \leq TSM(TS_1, TS_3) + TSM(TS_3, TS_2)$. Consequently the triangle inequality is verified in TSM.

6.7 Complexity of the proposed algorithm

The most important part, in term of complexity, is the one solving the assignment problem. We used the Hungarian algorithm [43] to find the best assignment in $\mathcal{O}(n^3)$ time, where n is the *maximum* number of components in the two graphs compared. Let $n = \max(\|N_1\|, \|N_2\|)$ and $n' = \max(\|TS_1\|, \|TS_2\|)$, where N_i is the set of nodes and TS_i is the set of **triangle-stars** in $g_{tr\ i}$. In the proposed decomposition, any **triangle-star** has at least one triangle. Consequently, in the worst case, we have $n' = \frac{n}{3} = 0.33 * n$, which means that the complexity is $\mathcal{O}(0.037\ \ n^3)$. However the number of **triangle-stars** depends on the structure of the underlying graph and the degree of neighborhood N_k considered in the process of decomposition. The number of **triangle-stars** is decreasing with the increasing of the degree of neighborhood N_k as shown in Figure 6.6, in which we can see also that the number of **triangle-stars** is less than the number of nodes and triangles $\|N_{i+1}\text{-}TS\| \leq \|N_i\text{-}TS\| \leq \|Nodes\| \leq \|Triangles\|$, with $i = 1...5$.
In the following, we present the time complexity depending on the degree of neighborhood N_k in the **TOSCA** database [261, 262] (See Section 7.1.1 for TOSCA database description), which is one of the databases that we use in our experiments. In the case the degree of Neighborhood N_1 (N_1-TS), we have on average $n' = \frac{n}{3.8510} = 0.2596 * n$ which means that the complexity is: $\mathcal{O}(0.0174\ \ n^3)$. Since $\frac{\|N\|}{\|TS\|} \cong 1.1029 * \log(\|N\|)$, the complexity is of the order of $\mathcal{O}(0.74 * [\frac{n}{\log(n)}]^3)$. In the case the degree of Neighborhood N_2 (N_2-TS), we have on average $n' = \frac{n}{68.6054} = 0.0145 * n$ which means that the complexity is $\mathcal{O}(3.0968 * 10^{-6}\ \ n^3)$. Since $\frac{\|N\|}{\|TS\|} \cong 19.6480 * \log(\|N\|)$, the complexity is of the order of $\mathcal{O}(0.0001 * [\frac{n}{\log(n)}]^3)$. For the degree of Neighborhood N_3 (N_3-TS), we have on average $n' = \frac{n}{196.4914} =$

$0.0050 * n$ which means that the complexity is $\mathcal{O}(1.3181 * 10^{-7} \ n^3)$. Since $\frac{\|N\|}{\|TS\|} \cong 56.2736 * \log(\|N\|)$, the complexity is of the order of $\mathcal{O}(5.6115 * 10^{-6} * [\frac{n}{\log(n)}]^3)$. In the case the degree of Neighborhood N_4 (N_4-TS), we have on average $n' = \frac{n}{350.9977} = 0.0028 * n$ which means that the complexity is $\mathcal{O}(2.3125 * 10^{-8} \ n^3)$. Since $\frac{\|N\|}{\|TS\|} \cong 100.5231 * \log(\|N\|)$, the complexity is of the order of $\mathcal{O}(9.84 * 10^{-7} * [\frac{n}{\log(n)}]^3)$. For the degree of Neighborhood N_5 (N_5-TS), we have on average $n' = \frac{n}{491.0660} = 0.0020 * n$ which means that the complexity is $\mathcal{O}(8.4446 * 10^{-9} \ n^3)$. Since $\frac{\|N\|}{\|TS\|} \cong 140.6376 * \log(\|N\|)$, the complexity is of the order of $\mathcal{O}(3.59 * 10^{-7} * [\frac{n}{\log(n)}]^3)$. In the case the degree of Neighborhood N_6 (N_6-TS): we have on average $n' = \frac{n}{617.3547} = 0.0016 * n$ which means that the complexity is: $\mathcal{O}(4.25 * 10^{-9} \ n^3)$. Since $\frac{\|N\|}{\|TS\|} \cong 176.8057 * \log(\|N\|)$, the complexity is of the order of $\mathcal{O}(1.80 * 10^{-7} * [\frac{n}{\log(n)}]^3)$. Table 6.7 summarizes the obtained time complexity depending on the degree of neighborhood $N_{k=1...6}$, in the TOSCA database. Table 6.7 shows that the obtained complexity is improved with the increasing of the degree of neighborhood K_k.

N_K	Complexity
N_1	$\mathcal{O}(0.74 * [\frac{n}{\log(n)}]^3)$
N_2	$\mathcal{O}(0.0001 * [\frac{n}{\log(n)}]^3)$
N_3	$\mathcal{O}(5.6115 * 10^{-6} * [\frac{n}{\log(n)}]^3)$
N_4	$\mathcal{O}(9.84 * 10^{-7} * [\frac{n}{\log(n)}]^3)$
N_5	$\mathcal{O}(3.59 * 10^{-7} * [\frac{n}{\log(n)}]^3)$
N_6	$\mathcal{O}(1.80 * 10^{-7} * [\frac{n}{\log(n)}]^3)$

Table 6.7: The time complexity depending on the degree of neighborhood N_k, in the TOSCA database.

6.8 Conclusion

In this chapter, we presented a new matching algorithm for addressing the problem of comparing deformable 3D objects represented by graphs (triangular tessellations). The proposed approach is based on a new decomposition of triangular tessellations into triangle-stars depending on the degree of the considered neighborhood. The resulting triangle-stars are used to determine the distance between triangular tessel-

lation using the Hungarian algorithm. The proposed algorithm assures a minimum number of disjoints triangle-stars, offers a better dissimilarity by covering a larger area of neighbors in triangle-stars and uses a set of descriptors which are invariant or at least oblivious under most common deformations. The proposed approach is based on an approximation of Graph Edit Distance which is fault-tolerant to noise and distortion, making our approach very appropriate for comparing deformable objects. We proved that the proposed distance TSM is a pseudo-metric. The analysis of the time complexity confirm the high performance of our algorithm. In the next chapter (Chapter 7), we evaluate our approach by performing a set of experimentations and comparisons in different databases.

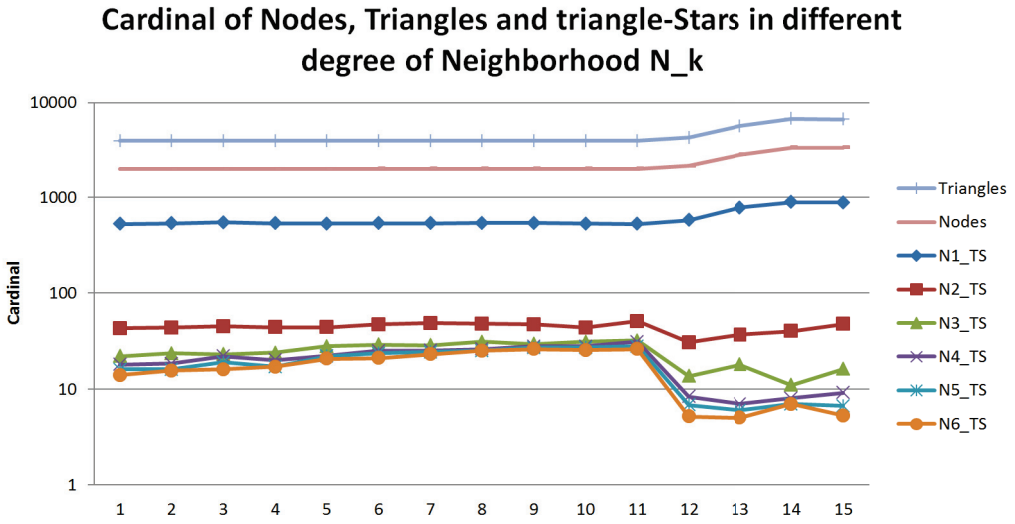


Figure 6.6: The number of Nodes, Triangles and triangle-stars in different N_k neighborhood in the TOSCA Database.

Chapter 7

Experimental results of the Graph-based approach for non-rigid 3D Object Recognition

Contents

In order to evaluate the proposed approach (Chapter 6), we undertook a set of experimentations and we compare our approach with some state of the art shape-matching algorithms on three databases: TOSCA database [261, 262], SHREC11 watertight [4] and SHREC09 database [5]. In this chapter, we present the three databases that we use in our experiments, some state of the art shape-matching algorithms to compare with, the evaluation criteria and the experimental results.
The remainder of the chapter is organized as follows. Firstly, in Section 7.1, we describe the three databases that we use in our experiments. In Section 7.2, we

describe the state of the art shape-matching algorithms with which we compare our approach. In Section 7.3, we detail the performance measures that we consider to evaluate our approach. In Section 7.4, we give and discuss the obtained results and we compare them with some state of the art shape-matching algorithms. Finally, Section 7.5 concludes the chapter.

7.1 Databases description

In this section, we describe the three databases that we use in our experiments.

7.1.1 TOSCA database

The TOSCA database [261, 262] is an interesting database for non-rigid object similarity measures. it consists of 148 three-dimensional objects. Each object is represented by a triangular tessellation. The database is categorized into 12 classes. Each class contains an object with different poses (deformations). The cardinality of the classes is not the same, between 1 and 24 poses for each class (see Table 7.1). On average, each triangular tessellation has 3154 nodes, 6220 triangles. Table 7.2 shows some 3D objects of the TOSCA Database.

Class	Number of poses
cat	9
centaur	6
david	15
dog	11
gorilla	21
horse	17
lioness	15
michael	20
seahorse	6
shark	1
victoria	24
wolf	3

Table 7.1: Number of poses per class in the TOSCA Database.

Class	Pose 1	Pose 2	Pose 3
Centaur			
Gorilla			

Table 7.2: Some objects from the TOSCA Database.

7.1.2 SHREC11 watertight

The SHREC11 watertight [4] is a large and diverse non-rigid 3D shape database, recreated and modified from several publicly available databases such as the McGill database [265], TOSCA database [261, 262] and the Princeton Shape Benchmark [266]. The SHREC11 watertight [4] contains 600 non-rigid objects represented by triangular tessellations. The data-set is equally classified into 30 classes, with 20 poses per class. Figure 7.1 shows two objects of each class in the SHREC11 watertight data-set [4].

7.1.3 SHREC09 database

The SHREC09 database [5] is Partial 3D Models with the objective to retrieve the models which have parts similar to the query. It consists on two data-sets:

- Target data-set: The target database contains 720 complete 3D models is equally classified into 40 classes, with 18 models per class. The classes are defined with respect to their semantic categories. Table 7.3 shows the different classes in the target database.
- Query data-set: The query data-set consists of 20 3D partial models which are obtained by cutting parts from complete models (see Figure 7.2).

7.2 Some state of the art shape-matching algorithms

In order to evaluate and show the efficient of our approach, we compare it with a state of the art set of shape-matching algorithms.

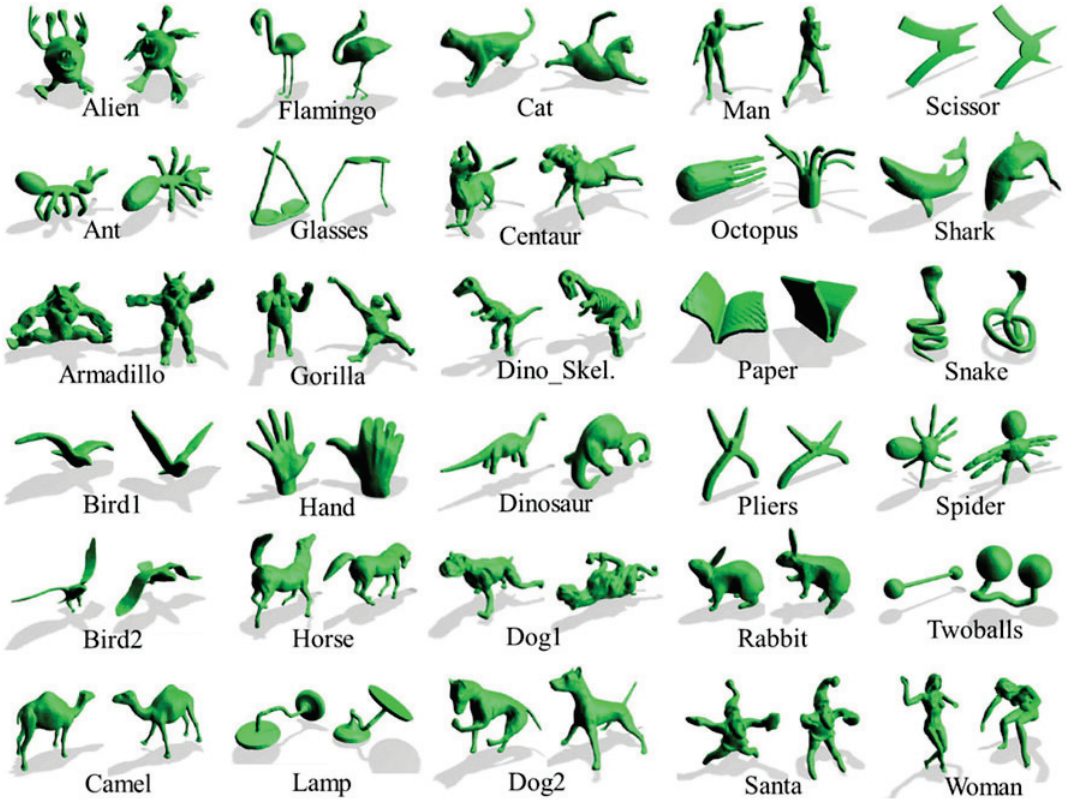


Figure 7.1: Example of two objects in each class of SHREC11 watertight data-set [4]

Bird	Tree	Fish	NonFlyingInsect
FlyingInsect	Monoplane	Biped	Quadruped
ApartmentHouse	Motorcycle	Skyscraper	SingleHouse
Bottle	Bicycle	Cup	Glasses
HandGun	Biplane	SubmachineGun	MusicalInstrument
Mug	Rocket	FloorLamp	DeskLamp
Sword	Car	Cellphone	DeskPhone
Monitor	Helicopter	Bed	NonWheelChair
WheelChair	Ship	Sofa	RectangleTable
RoundTable	MilitaryVehicle	Bookshelf	HomePlant

Table 7.3: The 40 classes of the target database in SHREC09.

Figure 7.2: The 20 3D partial models of the query data-set in SHREC09 [5].

The set of algorithms with which we compare are:

- **CAM**: 3D-Matching method using curve analysis [267].
- **GeodesicD2**: An extension of the Euclidean D2 [268], computed as a global distribution of geodesic distances in 3D shapes.
- **DSR**: The Hybrid Feature Vector, which is a combination of two view-based descriptors: the depth buffer and the silhouette and extent radialized function

descriptor [269].

- **RSH**: The Ray-Based Approach with Spherical Harmonic Representation in which the authors of [270] align the models into the canonical position, extract the maximal extents and apply spherical harmonic.

- **TD**: The temperature distribution (TD) descriptor [271] is shape descriptor based on geometric features. TD descriptor is driven by heat kernel and it is represented as one dimensional histogram. The L2 norm is used as matching method to compute the distance between two TD descriptors.

- **Shape-DNA**: The Shape-DNA [272] is a numerical fingerprint or signature, of any 2d or 3d manifold by taking the eigenvalues of its Laplace-Beltrami operator.

- **SRCP-TD**: The SRCP-TD [273] method is based on sparse representation of scale-invariant heat kernel. Laplace-Beltrami eigenfunctions are used, a shape descriptor is formed based on the heat kernels and Sparse representation is used.

- **Modal Repr**: An isometric deformation model is used based on the geodesic distance matrix as an isometry-invariant shape representation. The method proposed does not need explicit point correspondences for the comparison of 3D shapes [274].

- **CMVD-Binary**: The method CMVD-Binary ([275], [5]) CMVD-Binary is based on the Compact Multi-View Descriptor using the silhouettes of 3D objects.

- **CMVD-Depth**: The method CMVD-Depth ([275], [5]) is based Compact Multi-View Descriptor Depth (CMVD-Depth) and processes the depth maps.

- **Merged**: The method Merged corresponds to the fusion of the CMVD-Binary and CMVD-Depth methods ([275], [5]).

- **BF-SIFT**: The method BF-SIFT is based on the ideas of Bag of Features (BF) and the Scale Invariant Feature Transform (SIFT) ([276], [5]). The method compares shapes of 3D models visually by using a set of local features extracted from multiple view 2D depth images of the model.

- **BF-GridSIFT**: The method BF-GridSIFT is based on the ideas of Bag of Features (BF) and the Scale Invariant Feature Transform (SIFT) ([276], [5]).

7.3 Evaluation criteria

Several performance measures are used in the literature to evaluate retrieval methods and 3D similarity measures ([277], [278]). In this thesis, we evaluate our approach using the following performance measures:

- **True positive** (TP): The set of objects correctly identified.
  False positive (FP): The set of objects incorrectly identified.
  True Negative (TN): The set of objects correctly not identified or rejected.
  False Negative (FN): The set of objects incorrectly not identified or rejected. (See Figure 7.3).

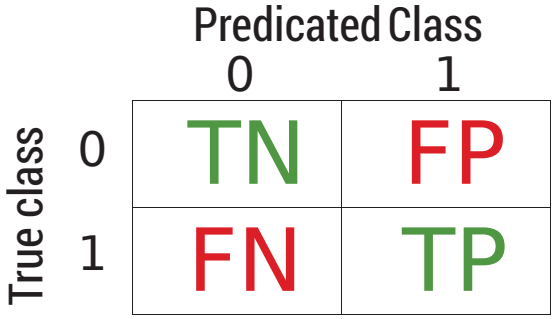


Figure 7.3: Confusion matrix of metric performances.

- **Precision and Recall**: We used the following equations to compute the precision and recall of an object from the class i.
  $Precision = \frac{\|objects_{found} \in C_i\|}{\|objects_{found}\|}$ $Recall = \frac{\|objects_{found} \in C_i\|}{\|C_i\|}$
  Precision and Recall can be also computed as follow:
  $Precision = \frac{TP}{TP + FP}$ $Recall = \frac{TP}{TP + FN}$.

- **Accuracy**: The percentage of elements correctly identified and those which are correctly rejected. It is computed as follow: $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$.
  True Positive Rate (TPR): The percentage of elements correctly classified within their own class. It is computed as follow: $TPR = \frac{TP}{TP + FN}$.
  True Negative Rate (TNR): The percentage of elements which are not

attributed to classes in which they are not part (correctly rejected). It is computed as follow: $TNR = \frac{TN}{TN + FP}$.

- **Run time**: We measure the average run time of computing the distance between two graphs and the global time needed for all the database.

- **E_Measure**: is based on the combination of precision and recall and it is defined using F_Measure. F_Measure is defined as follow: $F\_Measure = \frac{2 * Precision * Recall}{Precision + Recall}$. And E_Measure is defined as follow: $E\_Measure = 1 - F\_Measure$ ($F\_Measure \in [0, 1]$) which is equivalent to: $E\_Measure = 1 - \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$. With $E\_Measure \in [0, 1]$ and the higher value indicate better results.

7.4 Results

In this section, we discuss and compare the obtained results on the three databases (TOSCA [261, 262], SHREC11 watertight [4] and SHREC09 [5]).
The proposed distance TSM is a parameterized distance having a set of parameters α_k allowing different configurations, the default value is: $\alpha_k = 1, \forall\ k$. In addition, we defined a threshold in order to improve the classification accuracy. Considering the set of parameters α_k and the threshold in our approach offer a error-tolerant distance and make the proposed approach invariant to different deformations. The parameters α_k and the threshold may be specified by inspection or by using machine learning techniques.
In our experiments we consider: a descending strict total order on the triangles set of each object. Different degree of neighborhood N_K. The default value of parameters $\forall\ k,\ \alpha_k = 1$. The best *threshlod* for each degree of neighborhood N_K and for each database.

7.4.1 Results on the TOSCA database

We compute the distance between each pair of triangular tessellations in the TOSCA database [261, 262] using the proposed similarity measure TSM. Two triangular tessellations are considered similar if their distance is less than a specific *threshold*. Depending on the parameters α_k, the *threshlod* and the degree of neighborhood N_K, the results may be different. Tables (7.4, 7.5, 7.6, 7.7, 7.8 and 7.9) show

respectively some typical results depending on the degree of neighborhood $N_{K=1...6}$ with the following settings: $\forall\ k,\ \alpha_k = 1$, $threshlod \in [0.04, 0.273]$.

Threshold	Accuracy	TPR	TNR
0.06	83.24 %	82.29 %	83.35 %
0.07	78.3 %	91.1 %	76.79 %
0.08	74.6 %	95.46 %	72.14 %
0.05	88.03 %	64.92 %	90.75 %
0.04	90.54 %	40.75 %	96.40 %
0.036	91.02 %	30.27 %	98.18 %
0	89.48 %	0.17 %	100 %

Table 7.4: The accuracy with the degree of neighborhood $N_{K=1}$ and according to the classification threshold, in the TOSCA database.

Threshold	Accuracy	TPR	TNR
0.13	60.56 %	61.34 %	60.47 %
0.11	68.45 %	52.00 %	70.40 %
0.06	86.04 %	20.42 %	93.76 %
0.08	79.90 %	32.90 %	85.44 %
0	89.48 %	0.17 %	100 %

Table 7.5: The accuracy with the degree of neighborhood $N_{K=2}$ and according to the classification threshold, in the TOSCA database.

Table 7.10 shows the Accuracy, TPR and TNR results obtained by TSM in the TOSCA database, using the *thresholds* giving us the highest Accuracy, TPR and TNR for each degree of neighborhood $N_{K=1...6}$ and $\forall\ k,\ \alpha_k = 1$.
We obtained excellent results which are between 60% and 83% in the Accuracy, TPR and TNR, depending on the degree of neighborhood N_K. The direct neighborhood $N_{K=1}$ with $threshold = 0.06$, give us the best accuracy (in average 83%), followed by the accuracy obtained with $N_{K=4}$ (in average 68%) then $N_{K=3}$ and $N_{K=5}$ (in average 66%) and finally the accuracy obtained with $N_{K=6}$ and $N_{K=2}$ (in average 62%).

Threshold	Accuracy	TPR	TNR
0.19	65.78 %	66.93 %	65.65 %
0.15	73.87 %	53.66 %	76.25 %
0.11	80.70 %	36.47 %	85.91 %
0	89.48 %	0.17 %	100 %

Table 7.6: The accuracy with the degree of neighborhood $N_{K=3}$ and according to the classification threshold, in the TOSCA database.

Threshold	Accuracy	TPR	TNR
0.22	68.65 %	67.89 %	68.74 %
0.18	73.18 %	54.01 %	75.43 %
0.14	82.04 %	34.99 %	87.58 %
0.2	70.30 %	63.44 %	71.11 %
0	89.48 %	0.17 %	100 %

Table 7.7: The accuracy with the degree of neighborhood $N_{K=4}$ and according to the classification threshold, in the TOSCA database.

Threshold	Accuracy	TPR	TNR
0.264	65.01 %	64.31 %	65.09 %
0.22	71.27 %	51.92 %	73.55 %
0.16	80.34 %	32.29 %	86 %
0	89.48 %	0.17 %	100 %

Table 7.8: The accuracy with the degree of neighborhood $N_{K=5}$ and according to the classification threshold, in the TOSCA database.

We have computed the confusion matrix for all the 3D-objects belonging to the TOSCA database [261, 262]. Each element of the confusion matrix is associated with the dissimilarity between objects i and j. **Dark** colours are associated to dissimilarity close or equal to zero, **Light** colours are associated to dissimilarity close or equal to one. Objects are similar if their dissimilarity is close or equal to

zero. Using TOSCA database, we generate a $n \times n$ matrix, with $n = 148$ (the number of 3D objects) for each degree of neighborhood $N_{K=1...6}$ using the adequate *thresholds* (see Table 7.10).

Threshold	Accuracy	TPR	TNR
0.273	62.11 %	61.52 %	62.18 %
0.24	66.95 %	50.17 %	68.93 %
0.1	80.84 %	22.69 %	87.69 %
0	89.48 %	0.17 %	100 %

Table 7.9: The accuracy with the degree of neighborhood $N_{K=6}$ and according to the classification threshold, in the TOSCA database.

N_K	Threshold	Accuracy	TPR	TNR
1	0.06	83.24 %	82.29 %	83.35 %
2	0.13	60.56 %	61.34 %	60.47 %
3	0.19	65.78 %	66.93 %	65.65 %
4	0.22	68.65 %	67.89 %	68.74 %
5	0.264	65.01 %	64.31 %	65.09 %
6	0.273	62.11 %	61.52 %	62.18 %

Table 7.10: TSM Best Accuracy, TPR and TNR results in TOSCA database for the degree of neighborhood $N_{K=1...6}$.

Figure 7.4 shows the confusion matrix associated with its dissimilarity for the degrees of neighborhood $N_{K=1}$. The darkest regions correspond to the block-diagonals of the confusion matrix which are associated with the intra-class dissimilarity. In the Figure 7.4, we observe that objects from the same classes are similar, for instance, the following classes: gorilla, centaur, horse ... etc. and we observe also that objects from different classes are dissimilar, for example: (cat, gorilla), (cat, seahorse), (gorilla, lioness) ... etc. In a few cases, there is some interclass similarity: the dog and the wolf, David and Victoria and, David and Michael. This is not surprising considering that their shape is relatively similar. All these observations demonstrate the efficiency of the proposed algorithm.

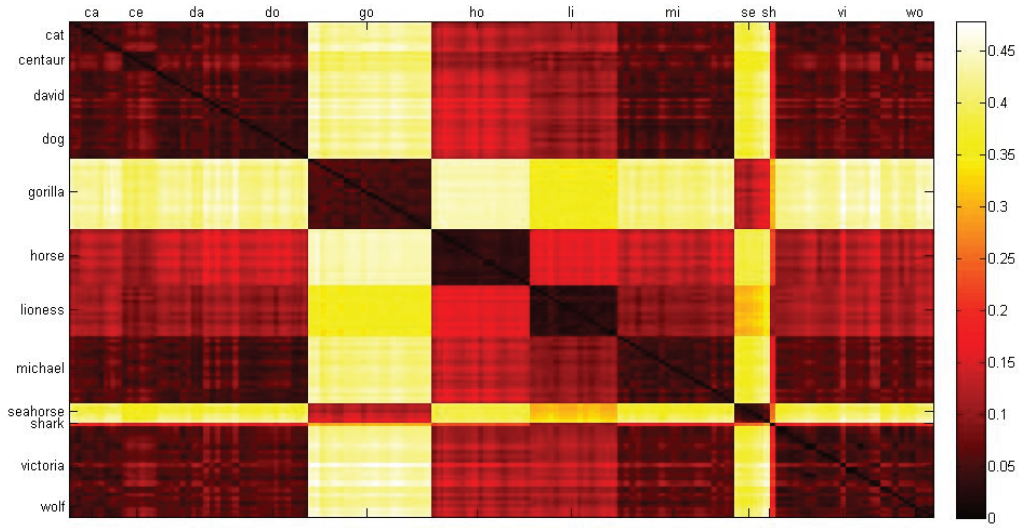


Figure 7.4: The $N_{K=1}$ Confusion matrix associated with the TOSCA Database.

Even the obtained results using the direct neighborhood $N_{K=1}$ are highly better than the results obtained with other degree of neighborhood $N_K = 2...6$. However, we obtain a high performance in term of the run time using a high degree of neighborhood.

Figure 7.5 shows the average run time performance for each degree of neighborhood $N_{K=1...6}$ in the TOSCA database. The X-axis shows the number of nodes in the two compared graph tessellations represented by the product $\|G1XG2\|$ and the Y-axis the average run time (Seconds), in log scale. This figure 7.5 shows clearly that the proposed dissimilarity TSM is much faster using a higher degree of neighborhood, *i.e.*, TSM with $N_{K=i+1}$ is more faster than TSM with $N_{K=i}$. Figure 7.5 shows also that TSM with $N_{K=2...6}$ is largely more faster comparing to TSM with $N_{K=1}$. The runtime performance shown in Figure 7.5 confirms the theoretical time complexity (see Table 6.7).

We have also measured the total run time on the TOSCA database for each degree of neighborhood $N_{k=1...6}$ (see Figure 7.6). The total run time performance shown in Figure 7.6 confirm the performances shown in Figure 7.5 and the theoretical time complexity. Figure 7.6 show clearly that the time needed to process all the database using TSM with $N_{K=i}$ is longer than TSM with $N_{K=i+1}$, *i.e.*, TSM with $N_{K=i}$ need more time to process all the database than TSM with $N_{K=i+1}$. We remark also a big difference between the time needed to process all the database using TSM

with $N_{K=1}$ (approximately 9 days) and TSM with $N_{K=2...6}$ (5 minutes for $N_{K=2}$ and between 1 to 3 minutes for $N_{K=3...6}$).

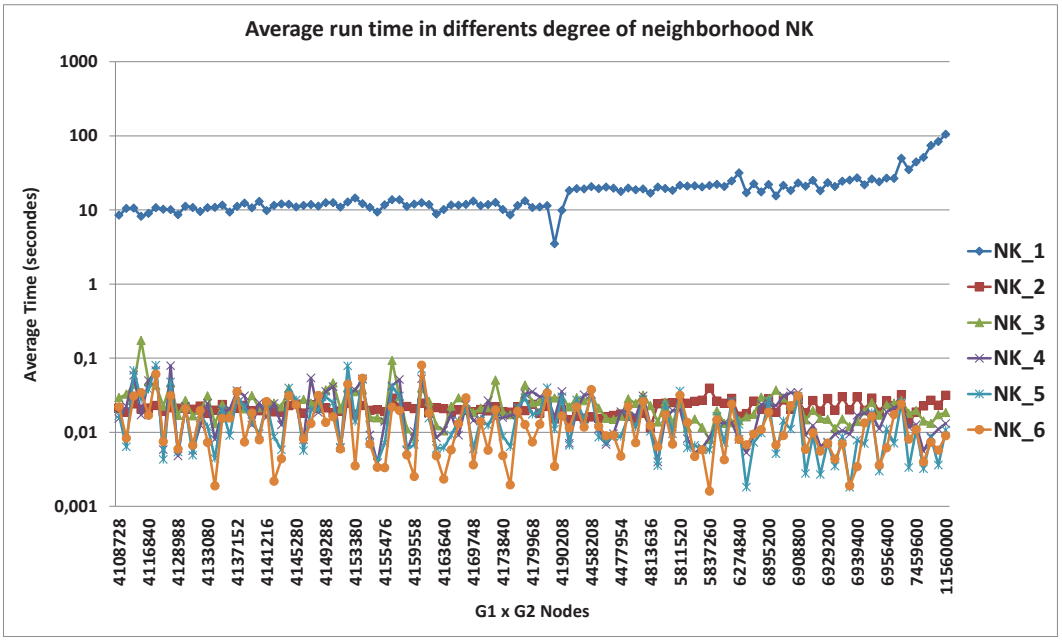


Figure 7.5: The average run time (Seconds) for each degree of neighborhood $N_{K=1...6}$, on the TOSCA database.

In addition to the visual results represented as confusion matrix (Figure 7.4), other visual results are shown in Figure 7.7. The first column contains six 3D-object query, while the rest of columns contain the four retrieved 3D-objects in the TOSCA database for each 3D-object query using the $N_{K=1}$ neighborhood. Figure 7.7 shows excellent results in which we can see that the 3D-object queries and their corresponding four retrieved 3D-objects belong to the same class (for example, the first query is a centaur and the four retrieved 3D-objects are centaurs also). An interesting result for the last 3D-object query "david11" which has as retrieved 3D-objects, three objects from the same class: "david7", "david10", "david4" and an object from an other class: "michael9". However, the two classes "david" and "michael" are semantically equivalent (belong to the class "man").

Figure 7.8 shows six precision-recall curves of the six 3D-object queries presented in Figure 7.7, from the TOSCA database using the $N_{K=1}$ neighborhood.

Our method TSM uses a threshold ($threshold = 0.06$ for $N_{K=1}$ neighborhood), which means that only the objects which present a $dissimilarity \leq threshold$

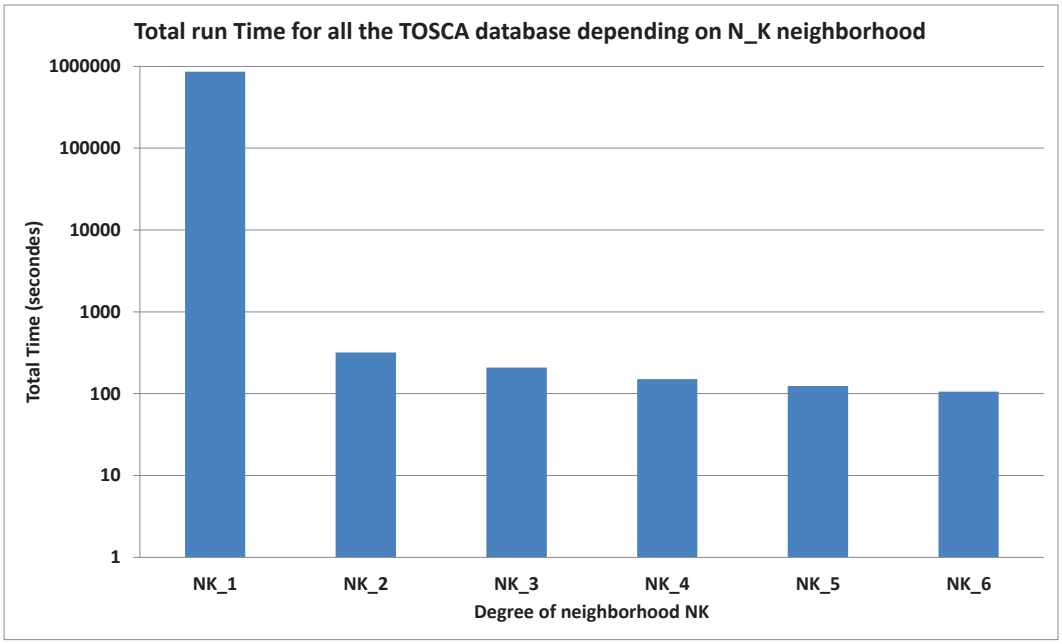


Figure 7.6: The total run time (Seconds) of the TOSCA database, for degree of neighborhood $N_{K=1...6}$.

are considered in the process of computing the recall and precision since, otherwise, they are automatically classified as dissimilar by our algorithm TSM. As showed in Figure 7.8, we have obtained excellent precision-recall curves which confirm the visual results shown in Figure 7.7. For instance, $gorilla_0$, $horse_0$, $lioness_0$ and $seahorse_0$ have a precision of 100% for a recall that goes from 86% to 100%.

We performed also a comparison in term of precision and recall in which we compare our method TSM using the $N_{K=1}$ neighborhood with the four methods (CAM, GeodesicD2, DSR and RSH) in the TOSCA database. Figure 7.9 shows the comparison of the precision and pecall plot of our approach TSM using the $N_{K=1}$ neighborhood with the four methods (CAM, GeodesicD2, DSR and RSH) in the TOSCA database. As the curve of our approach is higher than the four approaches to which it was compared, we conclude that our method performs better than the others.(in [267], CAM was compared to GeodesicD2, DSR and RSH).

We performed also an other comparison in term of E_Measure in which we compare our method TSM using the $N_{K=1...6}$ neighborhood with the three methods (TD, Shape-DNA and SRCP-TD) in the TOSCA database. Table 7.11 shows the obtained results (in term of E_Measure) of our approach TSM using the $N_{K=1...6}$

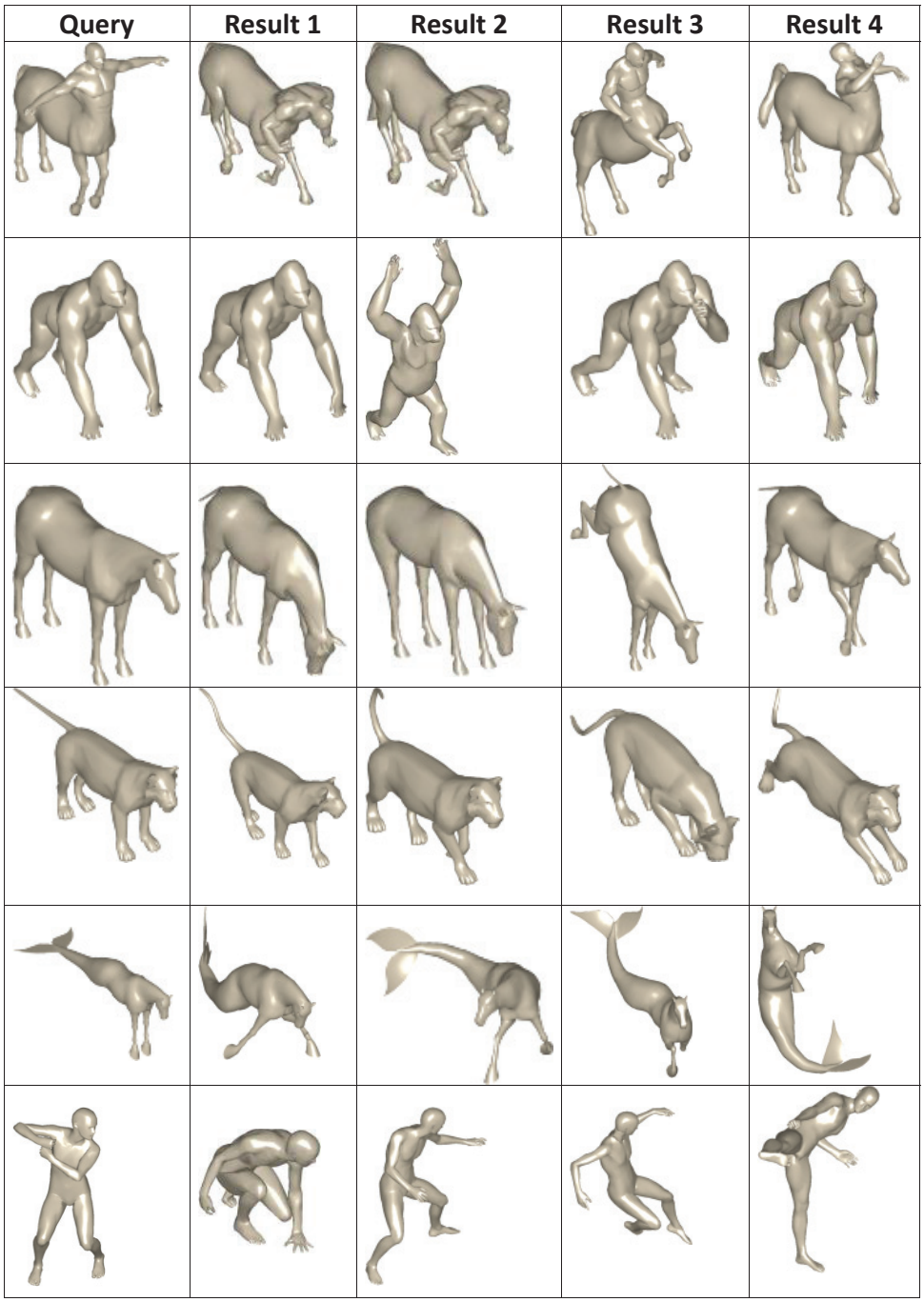


Figure 7.7: The first four retrieved results in the TOSCA database, with $N_{K=1}$ neighborhood.

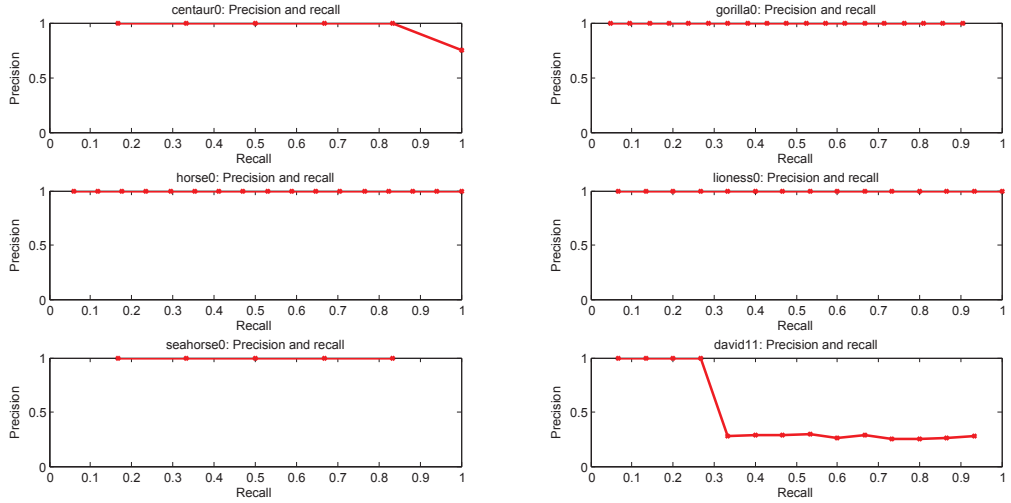


Figure 7.8: Precision-recall curves for six distinct 3D-objects of the TOSCA database with $N_{K=1}$ neighborhood.

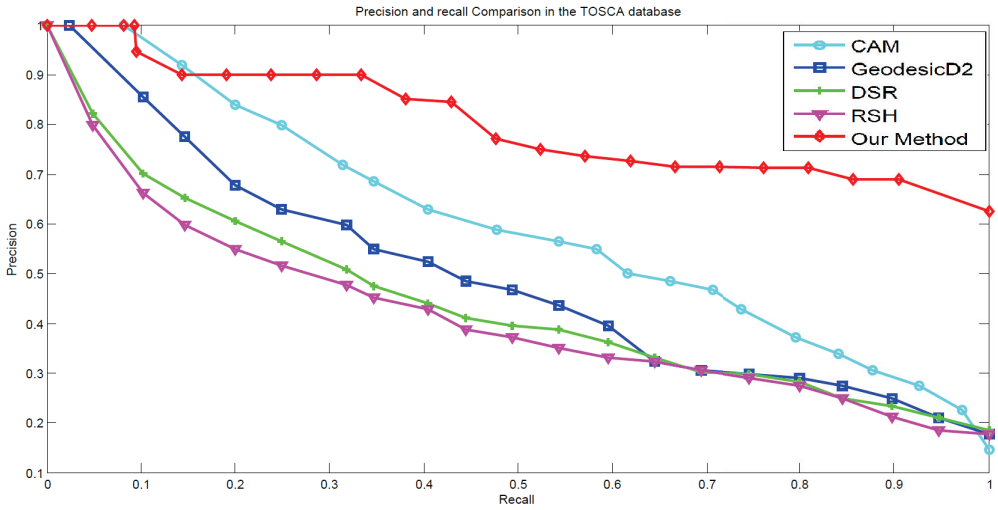


Figure 7.9: Precision and Recall plots comparing our approach to the CAM, GeodesicD2, DSR and RSH approaches on the TOSCA database.

neighborhood and the three methods (TD, Shape-DNA and SRCP-TD). Table 7.11 shows that our method TSM provide excellent results in terms of E_Measure, and using TSM with the $N_{K=1}$ neighborhood, give us the highest results. Table 7.11 shows also that our method TSM provides an E_Measure highly better than the

three methods with which we compare.

Methods	E_Measure
Our method TSM with $N_{K=1}$	0.9965
Our method TSM with $N_{K=2}$	0.9965
Our method TSM with $N_{K=3}$	0.9965
Our method TSM with $N_{K=4}$	0.9965
Our method TSM with $N_{K=5}$	0.9965
Our method TSM with $N_{K=6}$	0.9965
TD	0.33
Shape-DNA	0.55
SRCP-TD	0.56

Table 7.11: E_Measure results of our method TSM using $N_{K=1...6}$ neighborhood and other methods of the state of the art in TOSCA database.

7.4.2 Results on the SHREC11 watertight

We compute the distance between each pair of triangular tessellations in the SHREC11 watertight database [4] using the proposed similarity measure TSM. Two triangular tessellations are considered similar if their distance is less than a specific *threshold*. Depending on the parameters α_k, the *threshlod* and the degree of neighborhood N_K, the results may be different. Tables (7.12 and 7.13) show respectively some typical results depending on the degree of neighborhood $N_{K=2}$ and $N_{K=6}$ with the following settings: $\forall\ k,\ \alpha_k = 1$, $threshlod \in [0.05, 0.198]$. Using $N_{K=1}$ neighborhood gives us naturally better results accuracy than the ones obtained using $N_{K=2}$ and $N_{K=6}$ neighborhood. However, we are limited by the time and as shown in Figure 7.6, the time needed to process all the database using TSM with $N_{K=i}$ is longer than TSM with $N_{K=i+1}$ and especially when using $N_{K=1}$ neighborhood.
Table 7.14 shows the Accuracy, TPR and TNR results obtained by TSM in the SHREC11 watertight database, using the *thresholds* giving us the highest Accuracy, TPR and TNR for the two degrees of neighborhood ($N_{K=2}$ and $N_{K=6}$) and $\forall\ k,\ \alpha_k = 1$.
We obtained excellent results which are between 61% and 67% in the Accuracy,

Threshold	Accuracy	TPR	TNR
0.1076	67.08 %	67.04 %	67.08 %
0.09	73.86 %	55.95 %	74.45 %
0.05	89.36 %	23.77 %	91.51 %
0.01	96.82 %	0.05 %	99.99 %

Table 7.12: The Accuracy, TPR and TNR results with the degree of neighborhood $N_{K=2}$ and according to the classification threshold, in the SHREC11 watertight database.

Threshold	Accuracy	TPR	TNR
0.198	61.46 %	62.14 %	61.44 %
0.165	70.39 %	50.42 %	71.05 %
0.1	84.28 %	24.40 %	86.24 %
0.05	92.58 %	14.42 %	95.14 %
0.0101	96.83 %	0.07 %	99.99 %

Table 7.13: The Accuracy, TPR and TNR results with the degree of neighborhood $N_{K=6}$ and according to the classification threshold, in the SHREC11 watertight database.

TPR and TNR results, depending on the degree of neighborhood $N_{K=2}$ and $N_{K=6}$. Using the $N_{K=2}$ neighborhood with $threshold = 0.1076$, give us the best accuracy (in average 67%) which is better than the accuracy obtained using the $N_{K=6}$ neighborhood with $threshold = 0.198$ (in average 61%).

We performed also an other comparison in term of E_Measure in which we compare our method TSM using $N_{K=2}$ and $N_{K=6}$ neighborhood with different methods of the state of art in the SHREC11 watertight database. Table 7.15 shows the obtained results (in term of E_Measure) of our approach TSM using $N_{K=2}$ and $N_{K=6}$ neighborhood compared to the other methods. Table 7.15 shows that our method TSM provides an E_Measure highly better than the other methods with which we compare.

N_K	Threshold	Accuracy	TPR	TNR
2	0.1076	67.08 %	67.04 %	67.08 %
6	0.198	61.46 %	62.14 %	61.44 %

Table 7.14: TSM Best Accuracy, TPR and TNR results in SHREC11 watertight database for the degree of neighborhood $N_{K=2}$ and $N_{K=6}$.

Methods	E_Measure
Our method TSM with $N_{K=2}$	0.9961
Our method TSM with $N_{K=6}$	0.9966
Modal-repr	0.731
TD	0.3369
Shape-DNA	0.6797
SRCP-TD	0.69

Table 7.15: E_Measure results of our method TSM using $N_{K=2}$ and $N_{K=6}$ neighborhood and other methods of the state of the art in SHREC11 watertight database.

7.4.3 Results on the SHREC09 database

We compute the distance between each query and target triangular tessellations in the SHREC09 database [5] using the proposed similarity measure TSM. Two triangular tessellations are considered similar if their distance is less than a specific *threshold*. Depending on the parameters α_k, the *threshlod* and the degree of neighborhood N_K, the results may be different. Tables (7.16 and 7.17) show respectively some typical results depending on the degree of neighborhood $N_{K=2}$ and $N_{K=6}$ with the following settings: $\forall\ k,\ \alpha_k = 1$, $threshlod \in [0.09, 0.0401]$. We don't present the results with $N_{K=1}$ neighborhood because we are limited by the time and as shown in Figure 7.6, the time needed to process all the database using TSM with $N_{K=i}$ is longer than TSM with $N_{K=i+1}$ and especially when using $N_{K=1}$ neighborhood.
Table 7.18 shows the Accuracy, TPR and TNR results obtained by TSM in the SHREC09 database, using the *thresholds* giving us the highest Accuracy, TPR and TNR for the two degrees of neighborhood ($N_{K=2}$ and $N_{K=6}$) and $\forall\ k,\ \alpha_k = 1$.
We obtained excellent results which are between 55% and 61% in the Accuracy, TPR

Threshold	Accuracy	TPR	TNR
0.026	55.41 %	55 %	55.42 %
0.023	60.07 %	50.28 %	60.32 %
0.01	84.62 %	15.83 %	86.38 %
0.009	86.49 %	14.17 %	88.34 %

Table 7.16: The Accuracy, TPR and TNR results with the degree of neighborhood $N_{K=2}$ and according to the classification threshold, in the SHREC09 database.

Threshold	Accuracy	TPR	TNR
0.0401	61.19 %	59.17 %	61.25 %
0.034	65.85 %	51.39 %	66.22 %
0.02	78.13 %	30 %	79.37 %
0.01	88.47 %	13.89 %	90.38 %

Table 7.17: The Accuracy, TPR and TNR results with the degree of neighborhood $N_{K=6}$ and according to the classification threshold, in the SHREC09 database.

and TNR, depending on the degree of neighborhood $N_{K=2}$ and $N_{K=6}$. Using the $N_{K=6}$ neighborhood with $threshold = 0.0401$, give us the best accuracy (in average 61%) which is better than the accuracy obtained using the $N_{K=2}$ neighborhood with $threshold = 0.026$ (in average 55%).

We performed also an other comparison in term of E_Measure in which we compare our method TSM using the $N_{K=2}$ and $N_{K=6}$ neighborhood with the five methods (CMVD-Binary, CMVD-Depth, Merged, BF-SIFT and BF-GridSIFT) in the SHREC09 database. Table 7.19 shows the obtained results (in term of E_Measure) of our approach TSM using the $N_{K=2}$ and $N_{K=6}$ neighborhood compared to the five methods (CMVD-Binary, CMVD-Depth, Merged, BF-SIFT and BF-GridSIFT). Table 7.19 shows that our method TSM provides excellent results in terms of E_Measure. Table 7.19 shows also that our method TSM provides an E_Measure highly better than the other methods with which we compare.

N_K	Threshold	Accuracy	TPR	TNR
2	0.026	55.41 %	55 %	55.42 %
6	0.0401	61.19 %	59.17 %	61.25 %

Table 7.18: TSM Best Accuracy, TPR and TNR results in SHREC09 database for the degree of neighborhood $N_{K=2}$ and $N_{K=6}$.

Methods	E_Measure
Our method TSM with $N_{K=2}$	0.94
Our method TSM with $N_{K=6}$	0.93
CMVD-Binary	0.2
CMVD-Depth	0.174
Merged	0.192
BF-SIFT	0.116
BF-GridSIFT	0.204

Table 7.19: E_Measure results of our method TSM using $N_{K=2}$ and $N_{K=6}$ neighborhood and other methods of the state of the art in SHREC09 database.

7.4.4 Results discussion

The set of experimentations and the comparisons with some state of the art shape-matching algorithms, on the three databases: TOSCA database [261, 262], SHREC11 watertight [4] and SHREC09 database [5], show the high performance of our approach.

Indeed, we obtained excellent results in term of Accuracy, TPR and TNR on the three databases (TOSCA database [261, 262], SHREC11 watertight [4] and SHREC09 database [5]). The visual results represented as a confusion matrix (Figure 7.4) and as the retrieved results in the TOSCA database [261, 262], confirm the high performance of our approach. Our approach also realize a high performance in term of the run time (average and total run time) depending on the considered neighborhood degree, which confirms the theoretical time complexity. The excellent precision-recall curves obtained and the comparison performed with the four methods (CAM, GeodesicD2, DSR and RSH) in the TOSCA database, show the high

performance of our approach in term of precision and recall and that our method is better. We also performed a comparison in term of E_Measure in which we compare our method TSM using the different degree of neighborhood with a set of methods from the state of the art shape-matching algorithms, on the three databases: TOSCA database [261, 262], SHREC11 watertight [4] and SHREC09 database [5]. The obtained results show that our method performs excellent results in term of E_Measure and show that our approach is highly better in term of E_Measure than methods with which we compare.

The results shown in this chapter are obtained using the default parameters and the adequate threshold. Even the obtained results are excellent, an improvement in term of performance can be realized by specifying the adequate parameters, threshold and degree of neighborhood depending on the database considered and this by using machine learning techniques. The set of descriptors can be also enriched, which can be allows an improvement of the obtained results.

7.5 Conclusion

We proposed a new graph matching distance for addressing the problem of comparing deformable 3D objects represented by graphs (triangular tessellations). The proposed approach is based on a new decomposition of triangular tessellations into triangle-Stars. In order to evaluate our approach, we undertook a set of experiments in different well know databases for entire and partial deformable shape comparison. We compared the obtained results with some state of the art shape-matching algorithms. We used in our experiments different evaluation criteria. In this chapter, we described and discussed the experimentations that we performed to evaluate our approach. We presented the different databases that we used in our experiments, some state of the art shape-matching algorithms to compare with, the evaluation criteria and the experimental results. The analysis of the time complexity (Section 6.7) and our experimental results on three standard databases (TOSCA, SHREC09 and SHREC11) under different evaluation criteria confirm the high performance and accuracy of our algorithm.

Chapter 8

Conclusions and future works

Contents

In this chapter, we first concludes the thesis with a summary of our contributions, in Section 8.1. We then describe some suggestions for further research and future works, in Section 8.2.

8.1 Conclusions

In this thesis, graph based approaches for Pattern Recognition and the associated applications, namely 2D and 3D, are investigated. Graph based techniques for Pattern Recognition aim to solve mainly two major problems. The first one is to find an optimal way to represent the considered patterns by graphs. The second problem is to find the adequate method to compare the objects represented by graphs. In this context, finding solutions to the problem of graph modeling and graph matching that ensure optimality in terms of accuracy and time complexity is a difficult research challenge and a topical issue. Graph matching and more generally graph comparison is the aim operation in the process of pattern recognition using graph-based approaches. Graph matching solutions are classified into two wide categories: exact approaches and inexact approaches. In this thesis, we focused on inexact graph matching approaches and them applications on 2D and 3D Pattern Recognition.

In the first part of this thesis, we addressed the issue of geometric graph matching and its applications on 2D Pattern Recognition. Kite recognition in satellite images is the main application considered in this part. The visibility of Kites in satellite

images (due to their huge size), their important number and their wide geographical distribution, make automatic recognition of Kites an important step towards understanding these enigmatic remnants. We presented a complete identification tool relying on a graph representation of the Kites. In this context, we realized mainly two major contributions. The first one is the introducing of a graph representation of Kites and the proposition of an automatic process for extracting and transforming Kites from real images into graphs. We also proposed a process of generating randomly a synthetic data set of Kite graphs. Using the two proposed processes, we constructed a benchmark of Kite graphs (real and synthetic) structured in different level of deformations. This benchmark is used to validate our algorithms. The second contribution in this part, is the proposition of a new graph similarity measure adapted to geometric graphs and consequently for Kite graphs. The proposed approach combines graph invariants with a geometric graph edit distance computation leading to an efficient Kite identification process. We analyzed the time complexity of the proposed algorithms and conducted extensive experiments both on real and synthetic Kite graph data sets which attested the effectiveness of the proposed approach. We also performed a set of experimentations on other data sets which showed that the proposed approach is extensible and quite general. The satellite images (ground truth data) used to the construction of the Kite database are provided by a team of archeologists expert on Kites. The archeologists have also checked and validated the experiments steps that we realized and the obtained results.

In the second part of this thesis, we addressed the problem of comparing deformable or non-rigid 3D objects. The shapes considered are represented by graphs, *i.e.*, triangular tessellations. In the field of 3D object recognition, it is often required to compare different 3D objects represented by graphs. Using triangular tessellations, 3D objects may be compared with graph matching techniques. We proposed a new graph based distance for comparing deformable 3D objects. This distance is based on a new decomposition of triangular tessellations into a set of substructures that we called *triangle-Stars*. A triangle-Star is a connected component formed by the union of a triangle and its neighborhood. The proposed decomposition offered a parameterizable triangle-Stars depending on the degree of the neighborhood considered. The number of triangle-Stars obtained is much smaller than the number of nodes and the number of classic stars [9, 10] and, as a result, the computational

complexity is reduced. The proposed graph edit distance is based on triangle-Stars which is a local structure that covers a larger neighborhood than a classic star decomposition [9, 10]. Consequently, the proposed dissimilarity measure assures an optimal approximation. The resulting triangle-Stars are used to determine the distance between triangular tessellation using the Hungarian algorithm. The proposed approach assured a minimum number of disjoints triangle-Stars, offered a better dissimilarity by covering a larger area of neighbors in triangle-Stars and used a set of descriptors which are invariant or at least oblivious under most common deformations. The proposed approach is based on an approximation of Graph Edit Distance which is fault-tolerant to noise and distortion, making our approach very appropriate for comparing deformable objects. We proved that the proposed distance is a pseudo-metric and we analysed its time complexity. We realized a set of experiments including comparisons, on different well known databases for entire and partial deformable shape comparison, and under various evaluation criteria. The analysis of the time complexity and our experimental results on three standard databases (TOSCA, SHREC09 and SHREC11) confirmed the high performance and accuracy of our algorithm. The set of experimentations and the obtained results on SHREC09 database showed that the proposed approach is efficient also for the 3D objects sub-matching, which proved that our method is extensible and quite general.

8.2 Further works

In this section, we describe some suggestions for further research and future works.

The approaches that we proposed are parameterizable. Indeed, many parameters are considered: parameters for methods extracting and transforming Kites from real images to graphs, parameters for the process of generating randomly the synthetic data set of Kite graphs, parameters for the proposed geometric graph similarity measure, the degree of neighborhood to be considered in the proposed decomposition of triangular tessellations into *triangle-Stars*, parameters associated to the proposed graph based distance for deformable 3D objects and the thresholds associated with the proposed similarity measures. Even, by using default values of these parameters, we obtained excellent results in our experiments, using machine learning techniques will allow us to use the adequate parameters depending on the databases considered which should naturally further improve the obtained results.

The proposed approach dealing with the geometric graphs matching was evaluated on the proposed Kite benchmark (real and synthetic) and on the well known database GREC [205]. We project to realize other experiments on other well known graph databases, namely on the rest of data sets of the well-known IAM Graph Database Repository [205], in order to further evaluate the performances of the proposed approach and specialty for confirming that the proposed approach is quite general. We project also to improve the proposed approach to be a general solution for structural Pattern Recognition in satellite images.

The images used for the construction of the prosed Kite database are acquired manually, as a technical improvement, we project to automate the process of image acquisition.

In this thesis, two graph based approaches for 2D and 3D Pattern Recognition are proposed. Their time complexity is excellent, $\mathcal{O}(n^3)$. However, a more general graph matching approach in term of applicability ($2D$ and $3D$) with a reduced time complexity, can be proposed. We can meet this objective using a new formalization based on the stable marriage problem [11]. This approach is based on graph decomposition into a set of substructures, and then followed by a matching of these substructures based on stable marriage algorithm. The choice of the graph decomposition method depends on the kind of the considered graphs: a triangle-Stars decomposition for the triangular tessellations (graphs of 3D shapes) and a star decomposition for other kinds of graphs (graphs of $2D$ shapes).

The stable marriage problem is the problem of finding a stable matching between two sets of elements with equal size, based on an ordering of preferences associated to each element regarding other elements. Stability means that every matched couple of elements prefers to stay together rather than be mapped to an other one [11]. Originally, stable marriage problem is introduced by Gale and Shapley in 1962 in order to find a matching between men and women, based on preference lists in which each person (man and woman) presents his or her preferences regarding the other persons with opposite gender. The problem has then adapted in several research areas, such as mathematics, economics, game theory, computer science, etc. We refer readers to [279] for more details on stable marriage problem and its variants.

The basic idea of our new approach is firstly to decompose the two graphs G_1 and G_2 that we want to compare into two sets of substructures S_1 and S_2. Depending on the kind of the graphs (2D or 3D), the resulted substructures are either stars or triangle-Stars. A triangle-Stars decomposition (introduced in Chapter 6) for the triangular tessellations (graphs of 3D shapes) and a star decomposition for other kinds of graphs (graphs of $2D$ shapes). The second step is to associate to each substructure $s_{1,i} \in S_1$ from one graph G_1, a vector of preferences regarding to the other substructures $s_{2,j} \in S_2$ of the other graph G_2. The vectors associated to $s_{1,i} \in S_1$ (respectively $s_{2,j} \in S_2$) contain the set of substructures S_2 (respectively S_1) odored following the preferences with a descending order. The preference $P(s_{1,i},\ s_{2,j})$ between $s_{1,i} \in S_1$ and $s_{2,j} \in S_2$ is measured either using the proposed distance defined by the formula (Eq. 5.6, Chapter 5) in the case of stars or using the proposed distance defined by the formula (Eq. 6.1, Chapter 6) in the case of triangle-Stars. All the vectors have the same size t, which is equal to the maximum size of the two sets of substructures $t = \max(||S_1||,\ ||S_2||)$. In the case of $||S_1|| \neq ||S_2||$, the vectors with small size will completed by $(\max(||S_1||, ||S_2||) - \min(||S_1||, ||S_2||))$ empty substructures (ε), with the corresponding preference $P(s_{k,i},\ \varepsilon)$. In the final, we obtain a set of vectors constituting a square matrix of preferences $D_{t,t}$ in which each case $D_{i,j}$ contain a substructure (or ε) with the corresponding preferences $P_{i,j}$. The third step consists to use the algorithm of stable marriage in order to find the best match of the different substructures S_1 and S_2 based on their preferences vectors, in quadratic time $\mathcal{O}(n^2)$. The last step is to compute the score of the matching and this by the sum of preference values of each couple of substructures $Score = \sum_{i,j=1}^{i,j=t} P(s_{1,i},\ s_{2,j})$. An illustration of the the proposed approach is giving in **Example 1**.

Example 1. Let S_1 and S_2 two set of substructures (stars or triangle-Stars) of two graphs G_1 and G_2. $\|S_1\| = \|S_2\| = 4$. Let D the matrix of similarities between S_1 and S_2 calculated using one of the two formulas: Eq. 5.6 in the case of stars or Eq. 6.1 in the case of triangle-Stars.

$$
\begin{array}{c} \\ s_{1,0} \\ s_{1,1} \\ s_{1,2} \\ s_{1,3} \end{array}
\begin{array}{c}
\begin{array}{cccc} s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \end{array} \\
\begin{pmatrix}
0.11 & \mathbf{0.90} & 0.25 & 0.21 \\
0.10 & 0.15 & \mathbf{0.65} & 0.89 \\
\mathbf{0.67} & 0.03 & 0.51 & 0.17 \\
0.66 & 0.88 & 0.33 & \mathbf{0.99}
\end{pmatrix}
\end{array}
$$

S1,0	S2,1	S2,2	S2,3	S2,0
	0,90	0,25	0,21	0,11

S1,1	S2,3	S2,2	S2,1	S2,0
	0,89	0,65	0,15	0,10

S1,2	S2,0	S2,2	S2,3	S2,1
	0,67	0,51	0,17	0,03

S1,3	S2,3	S2,1	S2,0	S2,2
	0,99	0,88	0,66	0,33

(a) $S1$ preferences

S1,0	S2,1	S2,3	S2,0	S2,2
	0,92	0,99	0,55	0,50

S1,2	S2,0	S2,3	S2,2	S2,1
	0,10	0,88	0,56	0,07

S1,1	S2,2	S2,1	S2,3	S2,0
	0,96	0,65	0,77	0,36

S1,3	S2,3	S2,2	S2,0	S2,1
	0,11	0,81	0,35	0,52

(b) $S2$ preferences

Figure 8.1: Vectors of Preferences of $s_{1,i=0..3} \in S_1$ regarding to $s_{2,j=0..3} \in S_2$ and vectors of preferences of $s_{2,j=0..3} \in S_2$ regarding to $s_{1,i=0..3} \in S_1$, in descending order.

Firstly, we associate to each substructure $s_{1,i=0..3} \in S_1$ and $s_{2,j=0..3} \in S_2$, a vector of preferences regarding to the other set of substructures $s_{2,j=0..3} \in S_2$ and $s_{1,i=0..3} \in S_1$ respectively, with a descending order. Figure 8.1(a) shows the four vectors representing the preferences of $s_{1,i=0..3} \in S_1$ regarding to $s_{2,j=0..3} \in S_2$.
Figure 8.1(b) shows four vectors representing the preferences of $s_{2,j=0..3} \in S_2$ regarding to $s_{1,i=0..3} \in S_1$. Secondly, we use the algorithm of stable marriage in order to find the best match of the different substructures S_1 and S_2 based on their preferences vectors. We obtain the following couples: $\{(s_{1,0}, s_{2,1})$; $(s_{1,1}, s_{2,2})$; $(s_{1,2}, s_{2,0})$; $(s_{1,3}, s_{2,3})\}$. Finally, we calculate the distance corresponding the couples selected. The score obtained is $Score = 3.21$ and the normalised dissimilarity is $Score = 1 - \frac{3.21}{4} = 0.1975$.

As future work, we project to develop this new approach based on stable marriage and realize a set of experiments, including comparisons, on different well known databases (real and synthetic) and under various evaluation criteria, in order to evaluate the performance of our new approach based on Stable Marriage formulation.

Appendix A

Academic Achievements

International journals:

1. **Kamel Madi**, Hamida Seba, Hamamache Kheddouci, Olivier Barge. *A Graph-based approach for Kites recognition.* Pattern Recognition Letters. 2016. 10.1016/j.patrec.2016.05.005

2. Remy Crassard, Olivier Barge, Charles-Edmond Bichot, Jacques Elie Brochier, Jwana Chahoud, Marie-Laure Chambrade, Christine Chataigner, **Kamel Madi**, Emmanuelle Regagnon, Hamida Seba, Emmanuelle Vila. *Addressing the Desert Kites Phenomenon and Its Global Range Through a Multi-proxy Approach.* Journal of Archaeological Method and Theory. 2014, 1-29. 10.1007/s10816-014-9218-7

International conferences:

3. **Kamel Madi**, Eric Paquet, Hamida Seba, Hamamache Kheddouci. *Graph Edit Distance based on Triangle-Stars Decomposition for Deformable 3D Objects Recognition.* International Conference on 3D Vision (3DV 2015), Lyon (France) 19-22 October 2015. 55-63.

4. **Kamel Madi**, Hamida Seba, Hamamache Kheddouci, Charles-Edmont Bichot, Olivier Barge, Christine Chataigner, Remy Crassard, Emmanuelle Reganon, and Emmanuelle Vila. *Kite Recognition by means of Graph Matching.* Graph-based Representations in Pattern Recognition (GbR2015), Beijing (China) 13-15 may 2015. 118-127.

Papers in preparation (for international journals):

5. **Kamel Madi**, Eric Paquet, Hamida Seba, Hamamache Kheddouci. *New Graph matching and Subgraph Matching methods for Deformable 3D Objects Recognition based on Triangle-Stars Decomposition.* Pattern Recognition. (In preparation).

Bibliography

[1] "The seven bridges of königsberg." http://world.mathigon.org/Graph_Theory. (Cited on pages xix and 2.)

[2] D. CONTE, P. FOGGIA, C. SANSONE, and M. VENTO, "Thirty years of graph matching in pattern recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 03, pp. 265–298, 2004. (Cited on pages xix, 2, 15, 17, 18, 22 and 25.)

[3] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 01, p. 1450001, 2014. (Cited on pages xix, 15, 16, 18, 26, 27 and 28.)

[4] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen, "SHREC '11 track: Shape retrieval on non-rigid 3d watertight meshes," pp. 79–88, 2011. (Cited on pages xx, 93, 95, 96, 100, 109, 113 and 114.)

[5] A. Axenopoulos, P. Daras, H. Dutagaci, T. Furuya, A. Godil, and R. Ohbuchi, "Shrec 2009-shape retrieval contest of partial 3d models," 2009. (Cited on pages xx, 93, 95, 97, 98, 99, 100, 111, 113 and 114.)

[6] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Academiae Sci. I. Petropolitanae*, vol. 8, pp. 128–140, 1741. (Cited on page 1.)

[7] K. Riesen, *Structural Pattern Recognition with Graph Edit Distance*. Springer, 2016. (Cited on page 1.)

[8] S. Ullman, *High-Level Vision: Object Recognition and Visual Cognition*. (Cited on page 4.)

[9] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image Vision Comput.*, vol. 27, no. 7, pp. 950–959, 2009. (Cited on pages 5, 13, 23, 24, 31, 54, 61, 63, 69, 80, 116 and 117.)

[10] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou, "Comparing stars: On approximating graph edit distance," *PVLDB*, vol. 2, no. 1, pp. 25–36, 2009. (Cited on pages 5, 10, 22, 23, 54, 69, 116 and 117.)

[11] D. Gusfield and R. W. Irving, *The stable marriage problem: structure and algorithms*. MIT press, 1989. (Cited on pages 5 and 118.)

[12] J. A. Bondy, "Graph theory with applications," 1976. (Cited on page 7.)

[13] F. Harary, *Graph theory*. Addison-Wesley Series in Mathematics, Perseus Books, 1994. (Cited on page 7.)

[14] D. West, *An Introduction to Graph Theory*. Prentice-Hall, 1996. (Cited on page 7.)

[15] R. Diestel, *Graph Theory. 4th Edition*, vol. 173. Graduate Texts in Mathematics. Heidelberg: Springer-Verlag, 2010. (Cited on page 7.)

[16] H. Bunke and K. Riesen, "Recent advances in graph-based pattern recognition with applications in document analysis," *Pattern Recognition*, vol. 44, pp. 1057–1067, 2011. (Cited on pages 15, 27, 28 and 32.)

[17] M. Vento, "A long trip in the charming world of graphs for pattern recognition," *Pattern Recognition*, vol. 48, no. 2, pp. 291–301, 2015. (Cited on pages 15 and 16.)

[18] J. Yan, X.-C. Yin, W. Lin, C. Deng, H. Zha, and X. Yang, "A short survey of recent advances in graph matching," pp. 167–174, 2016. (Cited on page 15.)

[19] L. Babai, "Graph isomorphism in quasipolynomial time [extended abstract]," pp. 684–697, 2016. (Cited on page 17.)

[20] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The design and analysis of computer algorithms. series in computer science and information processing," *Addison-Wesley Publishing, Reading, Ma, USA*, vol. 2, p. 6, 1974. (Cited on page 19.)

[21] J. E. Hopcroft and J.-K. Wong, "Linear time algorithm for isomorphism of planar graphs (preliminary report)," pp. 172–184, 1974. (Cited on page 19.)

[22] J. R. Ullmann, "An Algorithm for Subgraph Isomorphism," *J. ACM*, vol. 23, pp. 31–42, Jan. 1976. (Cited on pages 19 and 20.)

[23] L. P. Cordella, P. Foggia, C. Sansone, F. Tortorella, and M. Vento, "Graph matching: a fast algorithm and its evaluation," pp. 1582–1584, 1998. (Cited on page 19.)

[24] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004. (Cited on page 19.)

[25] V. Carletti, P. Foggia, and M. Vento, "VF2 plus: An improved version of VF2 for biological graphs," pp. 168–177, 2015. (Cited on page 19.)

[26] J. Konc and D. Janezic, "An improved branch and bound algorithm for the maximum clique problem," *proteins*, vol. 4, no. 5, 2007. (Cited on page 20.)

[27] J. R. Ullmann, "Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism," *Journal of Experimental Algorithmics (JEA)*, vol. 15, pp. 1–6, 2010. (Cited on page 20.)

[28] J. Larrosa and G. Valiente, "Constraint satisfaction algorithms for graph pattern matching," *Mathematical structures in computer science*, vol. 12, no. 04, pp. 403–422, 2002. (Cited on page 20.)

[29] S. Zampelli, Y. Deville, and C. Solnon, "Solving subgraph isomorphism problems with constraint programming," *Constraints*, vol. 15, no. 3, pp. 327–353, 2010. (Cited on page 20.)

[30] C. Solnon, "Alldifferent-based filtering for subgraph isomorphism," *Artificial Intelligence*, vol. 174, no. 12, pp. 850–864, 2010. (Cited on page 20.)

[31] P. San Segundo, A. Lopez, and P. M. Pardalos, "A new exact maximum clique algorithm for large and massive sparse graphs," *Computers & Operations Research*, vol. 66, pp. 81–94, 2016. (Cited on page 20.)

[32] B. McKay, "Practical graph isomorphism," *Congress Numerantium*, vol. 87, pp. 30–45, 1981. (Cited on page 20.)

[33] D. P. Lopresti and G. T. Wilfong, "Comparing Semi-Structured Documents via Graph Probing," pp. 41–50, 2001. (Cited on page 20.)

[34] Y. Xiao, H. Dong, W. Wu, M. Xiong, W. Wang, and B. Shi, "Structure-based graph distance measures of high degree of precision," *Pattern Recognition*, vol. 41, pp. 3547–3561, 2008. (Cited on page 21.)

[35] M. Gori, M. Maggini, and L. Sarti, "Exact and approximate graph matching using random walks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 1100–1111, July 2005. (Cited on page 21.)

[36] P. J. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl, "Matching graphs with unique node labels," *Pattern Analysis and Applications*, vol. 7, no. 3, pp. 243–254, 2004. (Cited on page 21.)

[37] N. Dahm, H. Bunke, T. Caelli, and Y. Gao, "Topological features and iterative node elimination for speeding up subgraph isomorphism detection," pp. 1164–1167, 2012. (Cited on page 21.)

[38] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recogn. Lett.*, vol. 19, pp. 255–259, Mar. 1998. (Cited on page 22.)

[39] A. N. Papadopoulos and Y. Manolopoulos, "Structure-based similarity search with graph histograms," pp. 174–, 1999. (Cited on page 22.)

[40] S. Sorlin, C. Solnon, and J.-M. Jolion, "A generic graph distance measure based on multivalent matchings.," vol. 52, pp. 151–181, 2007. (Cited on page 22.)

[41] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-13, pp. 353–362, May 1983. (Cited on page 22.)

[42] F. Serratosa, "Computation of graph edit distance: Reasoning about optimality and speed-up," *Image Vision Comput.*, vol. 40, pp. 38–48, 2015. (Cited on pages 23 and 24.)

[43] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. (Cited on pages 23, 24, 57, 60, 88 and 90.)

[44] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987. (Cited on pages 23 and 24.)

[45] S. Yahiaoui, M. Haddad, B. Effantin, and H. Kheddouci, "Coloring based approach for matching unrooted and/or unordered trees," *Pattern Recognition Letters*, vol. 34, no. 6, pp. 686–695, 2013. (Cited on pages 23 and 31.)

[46] R. Raveaux, J.-C. Burie, and J.-M. Ogier, "A graph matching method and a graph matching distance based on subgraph assignments," *Pattern Recognition Letters*, vol. 31, pp. 394–406, 2010. (Cited on pages 23, 31 and 54.)

[47] S. Berretti, A. Del Bimbo, and P. Pala, "A graph edit distance based on node merging," vol. 3115, pp. 464–472, 2004. (Cited on page 23.)

[48] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the computation of graph edit distance," pp. 163–172, 2006. (Cited on pages 23, 25 and 61.)

[49] F. Serratosa, "Fast computation of bipartite graph matching," *Pattern Recognition Letters*, vol. 45, pp. 244–250, 2014. (Cited on pages 23, 24 and 31.)

[50] F. Serratosa, "Speeding up fast bipartite graph matching through a new cost matrix," *IJPRAI*, vol. 29, no. 2, 2015. (Cited on pages 23 and 24.)

[51] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, and H. Bunke, "Approximation of graph edit distance based on hausdorff matching," *Pattern Recognition*, vol. 48, no. 2, pp. 331–343, 2015. (Cited on pages 24, 31 and 32.)

[52] D. P. Huttenlocher, G. A. Klanderman, and W. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, 1993. (Cited on page 24.)

[53] F. Serratosa and X. Cortés, "Graph edit distance: Moving from global to local structure to solve the graph-matching problem," *Pattern Recognition Letters*, vol. 65, pp. 204–210, 2015. (Cited on page 24.)

[54] K. Riesen, M. Ferrer, and H. Bunke, "Approximate graph edit distance in quadratic time," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. PP, no. 99, pp. 1–1, 2015. (Cited on page 24.)

[55] K. Riesen, "Improving the distance accuracy of bipartite graph edit distance," pp. 69–99, 2015. (Cited on page 24.)

[56] K. Riesen, A. Fischer, and H. Bunke, "Approximation of graph edit distance by means of a utility matrix," pp. 185–194, 2016. (Cited on page 24.)

[57] X. Gao, B. Xiao, D. Tao, and X. Li., "A survey of graph edit distance," *Pattern Analysis Applications*, no. 13, pp. 113–129, 2010. (Cited on page 24.)

[58] P. E. Hart, N. J. Nilsson, and B. Raphael, "Correction to "a formal basis for the heuristic determination of minimum cost paths"," *SIGART Newsletter*, vol. 37, pp. 28–29, 1972. (Cited on page 24.)

[59] W. Tsai and K. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 9, no. 12, pp. 757–768, 1979. (Cited on page 25.)

[60] W. Tsai and K. Fu, "Subgraph error-correcting isomorphisms for syntactic pattern recognition," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 13, no. 1, pp. 48–62, 1983. (Cited on page 25.)

[61] M. Eshera and K. Fu, "A similarity measure between attributed relational graphs for image analysis," pp. 75–77, 1984. (Cited on page 25.)

[62] A. K. Wong, M. You, and S. Chan, "An algorithm for graph optimal monomorphism," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 3, pp. 628–638, 1990. (Cited on pages 25 and 31.)

[63] A. C. Dumay, R. J. van der Geest, J. J. Gerbrands, E. Jansen, and J. H. Reiber, "Consistent inexact graph matching applied to labelling coronary segments in arteriograms," pp. 439–442, 1992. (Cited on page 25.)

[64] S. Berretti, A. Del Bimbo, and E. Vicario, "A look-ahead strategy for graph matching in retrieval by spatial arrangement," vol. 3, pp. 1721–1724, 2000. (Cited on page 25.)

[65] H. Bunke, "Graph matching: Theoretical foundations, algorithms, and applications," vol. 2000, pp. 82–88, 2000. (Cited on page 25.)

[66] S. Berretti, A. Del Bimbo, and E. Vicario, "Efficient matching and indexing of graph models in content-based retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1089–1105, 2001. (Cited on page 25.)

[67] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on computers*, vol. 22, no. 1, pp. 67–92, 1973. (Cited on page 25.)

[68] J. Kittler and E. R. Hancock, "Combining evidence in probabilistic relaxation," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 3, no. 01, pp. 29–51, 1989. (Cited on page 25.)

[69] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 17, no. 8, pp. 749–764, 1995. (Cited on pages 25 and 31.)

[70] R. C. Wilson and E. R. Hancock, "Structural matching by discrete relaxation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 634–648, 1997. (Cited on page 25.)

[71] B. Huet and E. R. Hancock, "Shape recognition from large image libraries by inexact graph matching," *Pattern Recognition Letters*, vol. 20, no. 11, pp. 1259–1269, 1999. (Cited on page 25.)

[72] A. Torsello and E. R. Hancock, "Computing approximate tree edit distance using relaxation labeling," *Pattern Recognition Letters*, vol. 24, no. 8, pp. 1089–1097, 2003. (Cited on page 25.)

[73] Y. Aflalo, A. Bronstein, and R. Kimmel, "On convex relaxation of graph isomorphism," *Proceedings of the National Academy of Sciences*, vol. 112, no. 10, pp. 2942–2947, 2015. (Cited on pages 25 and 26.)

[74] H. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 5, pp. 522–525, 1993. (Cited on page 26.)

[75] A. Rangarajan and E. Mjolsness, "A lagrangian relaxation network for graph matching," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1365–1381, 1996. (Cited on page 26.)

[76] A. C.-L. Chen, A. Elhajj, S. Gao, A. Sarhan, S. Afra, A. Kassem, and R. Alhajj, "Approximating the maximum common subgraph isomorphism problem

with a weighted graph," *Knowledge-Based Systems*, vol. 85, pp. 265–276, 2015. (Cited on page 26.)

[77] S. Medasani and R. Krishnapuram, "A fuzzy approach to content-based image retrieval," vol. 3, pp. 1251–1260, 1999. (Cited on page 26.)

[78] S. Medasani, R. Krishnapuram, and Y. Choi, "Graph matching by relaxation of fuzzy assignments," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 1, pp. 173–182, 2001. (Cited on page 26.)

[79] G. Gross, R. Nagi, and K. Sambhoos, "A fuzzy graph matching approach in intelligence analysis and maintenance of continuous situational awareness," *Information Fusion*, vol. 18, pp. 43 – 61, 2014. (Cited on page 26.)

[80] B. J. van Wyk and M. A. van Wyk, "Non-bayesian graph matching without explicit compatibility calculations," pp. 74–83, 2002. (Cited on page 26.)

[81] B. J. van Wyk, M. A. van Wyk, and H. E. Hanrahan, "Successive projection graph matching," pp. 263–271, 2002. (Cited on page 26.)

[82] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 4, pp. 515–519, 2004. (Cited on page 26.)

[83] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 10, no. 5, pp. 695–703, 1988. (Cited on page 26.)

[84] L. S. Shapiro and J. M. Brady, "Feature-based correspondence: an eigenvector approach," *Image and vision computing*, vol. 10, no. 5, pp. 283–288, 1992. (Cited on page 26.)

[85] M. Carcassoni and E. R. Hancock, "Spectral correspondence for point pattern matching," *Pattern Recognition*, vol. 36, no. 1, pp. 193–204, 2003. (Cited on page 26.)

[86] H. Wang and E. R. Hancock, "A kernel view of spectral point pattern matching," pp. 361–369, 2004. (Cited on page 26.)

[87] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," vol. 2, pp. 1482–1489, 2005. (Cited on page 26.)

[88] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," *Advances in Neural Information Processing Systems*, vol. 19, p. 313, 2007. (Cited on page 26.)

[89] F. Escolano, B. Bonev, and M. A. Lozano, "Information-geometric graph indexing from bags of partial node coverages.," pp. 52–61, 2011. (Cited on pages 26 and 27.)

[90] C. Leng, W. Xu, I. Cheng, and A. Basu, "Graph matching based on stochastic perturbation," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4862–4875, 2015. (Cited on page 26.)

[91] V. Lyzinski, D. L. Sussman, D. E. Fishkind, H. Pao, L. Chen, J. T. Vogelstein, Y. Park, and C. E. Priebe, "Spectral clustering for divide-and-conquer graph matching," *Parallel Computing*, vol. 47, pp. 70–87, 2015. (Cited on page 26.)

[92] R. C. Wilson and P. Zhu, "A study of graph spectra for comparing graphs and trees," *Pattern Recognition*, vol. 41, no. 9, pp. 2833–2841, 2008. (Cited on page 26.)

[93] B. T. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 493–504, 1998. (Cited on page 27.)

[94] F. Fuchs and H. Le Men, "Building reconstruction on aerial images through multi-primitive graph matching," pp. 21–30, 1999. (Cited on page 27.)

[95] F. Fuchs and H. Le-Men, "Efficient subgraph isomorphism with 'a priori'knowledge," pp. 427–436, 2000. (Cited on pages 27 and 31.)

[96] J. Yan, H. Xu, H. Zha, X. Yang, H. Liu, and S. Chu, "A matrix decomposition perspective to multiple graph matching," pp. 199–207, 2015. (Cited on page 27.)

[97] F. Zhou and F. D. la Torre, "Factorized graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 1774–1789, Sept 2016. (Cited on page 27.)

[98] A. ShouXuy and M. Aboutabl, "Neural network approach for solving the maximal common subgraph problem," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B: CYBERNETICS*, vol. 26, no. 5, p. 785, 1996. (Cited on page 27.)

[99] P. N. Suganthan and H. Yan, "Recognition of handprinted chinese characters by constrained graph matching," *Image and Vision Computing*, vol. 16, no. 3, pp. 191–201, 1998. (Cited on pages 27 and 31.)

[100] P. Suganthan, "Attributed relational graph matching by neural-gas networks," vol. 1, pp. 366–374, 2000. (Cited on page 27.)

[101] C. de Mauro, M. Diligenti, M. Gori, and M. Maggini, "Similarity learning for graph-based image representations," *Pattern Recognition Letters*, vol. 24, no. 8, pp. 1115–1122, 2003. (Cited on pages 27 and 28.)

[102] C.-W. Liu, K.-C. Fan, J.-T. Horng, and Y.-K. Wang, "Solving weighted graph matching problem by modified microgenetic algorithm," vol. 1, pp. 638–643, 1995. (Cited on page 27.)

[103] Y.-K. Wang, K.-C. Fan, and J.-T. Horng, "Genetic-based search for error-correcting graph isomorphism," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 4, pp. 588–597, 1997. (Cited on page 27.)

[104] A. Perchant, C. Boeres, I. Bloch, M. Roux, and C. Ribeiro, "Model-based scene recognition using graph fuzzy homomorphism solved by genetic algorithm," *GbR*, vol. 99, pp. 61–70, 1999. (Cited on page 27.)

[105] K. Khoo and P. Suganthan, "Multiple relational graphs mapping using genetic algorithms," vol. 2, pp. 727–733, 2001. (Cited on page 27.)

[106] K. Riesen, A. Fischer, and H. Bunke, "Improving approximate graph edit distance using genetic algorithms," pp. 63–72, 2014. (Cited on page 27.)

[107] F. Depiero, M. Trivedi, and S. Serbin, "Graph matching using a direct classification of node attendance," *Pattern Recognition*, vol. 29, no. 6, pp. 1031–1048, 1996. (Cited on page 27.)

[108] B. Ozer, W. Wolf, and A. Akansui, "A graph based object description for information retrieval in digital image and video libraries," pp. 79–83, 1999. (Cited on page 27.)

[109] A. Hlaoui and S. Wang, "A new algorithm for graph matching with application to content-based image retrieval," pp. 291–300, 2002. (Cited on page 27.)

[110] S. Sorlin and C. Solnon, "Reactive tabu search for measuring graph similarity," pp. 172–182, 2005. (Cited on page 27.)

[111] K. Adamczewski, Y. Suh, and K. Mu Lee, "Discrete tabu search for graph matching," pp. 109–117, 2015. (Cited on page 27.)

[112] T. Gärtner, J. W. Lloyd, and P. A. Flach, "Kernels for structured data," pp. 66–83, 2002. (Cited on page 27.)

[113] H. Bunke, C. Irniger, and M. Neuhaus, "Graph matching–challenges and potential solutions," pp. 1–10, 2005. (Cited on page 27.)

[114] H. Bunke and K. Riesen, "Towards the unification of structural and statistical pattern recognition," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 811–825, 2012. (Cited on page 27.)

[115] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. Von Der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 775–779, 1997. (Cited on page 27.)

[116] B. Duc, S. Fischer, and J. Bigun, "Face authentication with gabor information on deformable graphs," *IEEE Transactions on Image Processing*, vol. 8, no. 4, pp. 504–516, 1999. (Cited on pages 27 and 32.)

[117] Y.-T. Li and J. P. Wachs, "Recognizing hand gestures using the weighted elastic graph matching (wegm) method," *Image and Vision Computing*, vol. 31, no. 9, pp. 649–657, 2013. (Cited on page 27.)

[118] Y. D. Sato and Y. Kuriya, "Multi-scale elastic graph matching for face detection," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 1, pp. 1–12, 2013. (Cited on page 27.)

[119] H. Proença and J. C. Briceño, "Periocular biometrics: constraining the elastic graph matching algorithm to biologically plausible distortions," *IET Biometrics*, vol. 3, no. 4, pp. 167–175, 2014. (Cited on page 27.)

[120] Y.-T. Li and J. P. Wachs, "Hegm: A hierarchical elastic graph matching for hand gesture recognition," *Pattern Recognition*, vol. 47, no. 1, pp. 80–88, 2014. (Cited on page 27.)

[121] T. Caelli and S. Kosinov, "Inexact graph matching using eigen-subspace projection clustering," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 03, pp. 329–354, 2004. (Cited on page 27.)

[122] A. Robles-Kelly and E. R. Hancock, "A riemannian approach to graph embedding," *Pattern Recognition*, vol. 40, no. 3, pp. 1042–1056, 2007. (Cited on pages 27 and 28.)

[123] S. Jouili and S. Tabbone, "Graph embedding using constant shift embedding," pp. 83–92, 2010. (Cited on page 28.)

[124] B. Luo, R. C. Wilson, and E. R. Hancock, "Spectral embedding of graphs," *Pattern recognition*, vol. 36, no. 10, pp. 2213–2230, 2003. (Cited on page 28.)

[125] R. C. Wilson, E. R. Hancock, and B. Luo, "Pattern vectors from algebraic graph theory," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1112–1124, 2005. (Cited on page 28.)

[126] B. Xiao, E. R. Hancock, and R. C. Wilson, "Graph characteristics from the heat kernel trace," *Pattern Recognition*, vol. 42, no. 11, pp. 2589–2606, 2009. (Cited on page 28.)

[127] B. Xiao, S. Yi-Zhe, and P. Hall, "Learning invariant structure for object identification by using graph methods," *Computer Vision and Image Understanding*, vol. 115, no. 7, pp. 1023–1031, 2011. (Cited on page 28.)

[128] A. Torsello and E. R. Hancock, "Graph embedding using tree edit-union," *Pattern recognition*, vol. 40, no. 5, pp. 1393–1405, 2007. (Cited on page 28.)

[129] J. Richiardi, D. Van De Ville, K. Riesen, and H. Bunke, "Vector space embedding of undirected graphs with fixed-cardinality vertex sequences for classification," pp. 902–905, 2010. (Cited on page 28.)

[130] W. Czech, "Graph descriptors from b-matrix representation," pp. 12–21, 2011. (Cited on page 28.)

[131] J. Gibert, E. Valveny, and H. Bunke, "Dimensionality reduction for graph of words embedding," pp. 22–31, 2011. (Cited on page 28.)

[132] J. Gibert, E. Valveny, and H. Bunke, "Feature selection on node statistics based embedding of graphs," *Pattern Recognition Letters*, vol. 33, no. 15, pp. 1980–1990, 2012. (Cited on page 28.)

[133] J. Gibert, E. Valveny, and H. Bunke, "Graph embedding in vector spaces by node attribute statistics," *Pattern Recognition*, vol. 45, no. 9, pp. 3072–3083, 2012. (Cited on page 28.)

[134] J. Gibert, E. Valveny, and H. Bunke, "Embedding of graphs with discrete attributes via label frequencies," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 27, no. 03, p. 1360002, 2013. (Cited on page 28.)

[135] K. Riesen and H. Bunke, "Classifier ensembles for vector space embedding of graphs," pp. 220–230, 2007. (Cited on page 28.)

[136] W.-J. Lee and R. P. Duin, "A labelled graph based multiple classifier system," pp. 201–210, 2009. (Cited on page 28.)

[137] W.-J. Lee, R. P. Duin, and H. Bunke, "Selecting structural base classifiers for graph-based multiple classifier systems," pp. 155–164, 2010. (Cited on page 28.)

[138] K. Riesen, M. Neuhaus, and H. Bunke, "Graph embedding in vector spaces by means of prototype selection," pp. 383–393, 2007. (Cited on page 28.)

[139] K. Riesen and H. Bunke, "Non-linear transformations of vector space embedded graphs," pp. 173–183, 2008. (Cited on page 28.)

[140] K. Riesen and H. Bunke, "Graph classification based on vector space embedding," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 06, pp. 1053–1081, 2009. (Cited on page 28.)

[141] K. Riesen and H. Bunke, "Reducing the dimensionality of dissimilarity space embedding graph kernels," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 1, pp. 48–56, 2009. (Cited on page 28.)

[142] E. Z. Borzeshi, M. Piccardi, K. Riesen, and H. Bunke, "Discriminative prototype selection methods for graph embedding," *Pattern Recognition*, vol. 46, no. 6, pp. 1648–1657, 2013. (Cited on page 28.)

[143] M. Neuhaus and H. Bunke, *Bridging the gap between graph edit distance and kernel machines*. World Scientific Publishing Co., Inc., 2007. (Cited on page 28.)

[144] V. N. Vapnik and V. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998. (Cited on page 29.)

[145] R. Kondor and J. D. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," pp. 315–322, 2002. (Cited on page 29.)

[146] G. Lebanon and J. D. Lafferty, "Information diffusion kernels," vol. 15, p. 391, 2003. (Cited on page 29.)

[147] J. Lafferty and G. Lebanon, "Diffusion kernels on statistical manifolds," *Journal of Machine Learning Research*, vol. 6, no. Jan, pp. 129–163, 2005. (Cited on page 29.)

[148] D. Haussler, "Convolution kernels on discrete structures," tech. rep., Citeseer, 1999. (Cited on page 29.)

[149] C. Watkins, "Dynamic alignment kernels," *Advances in neural information processing systems*, pp. 39–50, 1999. (Cited on page 29.)

[150] C. Watkins, "Kernels from matching operations," tech. rep., Technical report, Department of Computer Science, Royal Holloway, University of London, 1999. (Cited on page 29.)

[151] H. Kashima and A. Inokuchi, "Kernels for graph classification," vol. 2002, 2002. (Cited on page 29.)

[152] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," pp. 129–143, 2003. (Cited on page 29.)

[153] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi, "Graph kernels for chemical informatics," *Neural networks*, vol. 18, no. 8, pp. 1093–1110, 2005. (Cited on page 29.)

[154] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl 1, pp. i47–i56, 2005. (Cited on page 29.)

[155] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," pp. 8–pp, 2005. (Cited on page 29.)

[156] L. Rossi, A. Torsello, and E. R. Hancock, "A continuous-time quantum walk kernel for unattributed graphs," pp. 101–110, 2013. (Cited on page 29.)

[157] M. Neuhaus, K. Riesen, and H. Bunke, "Novel kernels for error-tolerant graph classification," *Spatial vision*, vol. 22, no. 5, pp. 425–441, 2009. (Cited on page 29.)

[158] B. Gaüzere, L. Brun, and D. Villemin, "Two new graphs kernels in chemoinformatics," *Pattern Recognition Letters*, vol. 33, no. 15, pp. 2038–2047, 2012. (Cited on page 29.)

[159] B. Gaüzere, L. Brun, and D. Villemin, "Graph kernels: crossing information from different patterns using graph edit distance," pp. 42–50, 2012. (Cited on page 29.)

[160] P.-A. Grenier, L. Brun, and D. Villemin, "Treelet kernel incorporating chiral information," pp. 132–141, 2013. (Cited on page 29.)

[161] R. Kondor, N. Shervashidze, and K. M. Borgwardt, "The graphlet spectrum," pp. 529–536, 2009. (Cited on page 29.)

[162] B. Strug, "Using kernels on hierarchical graphs in automatic classification of designs," pp. 335–344, 2011. (Cited on page 29.)

[163] L. Bai and E. R. Hancock, "Graph kernels from the jensen-shannon divergence," *Journal of mathematical imaging and vision*, vol. 47, no. 1-2, pp. 60–69, 2013. (Cited on page 29.)

[164] S. Günter and H. Bunke, "Self-organizing map for clustering in the graph domain," *Pattern Recognition Letters*, vol. 23, no. 4, pp. 405–417, 2002. (Cited on page 29.)

[165] F. Serratosa, R. Alquézar, and A. Sanfeliu, "Synthesis of function-described graphs and clustering of attributed graphs," *International journal of pattern*

recognition and artificial intelligence, vol. 16, no. 06, pp. 621–655, 2002. (Cited on page 29.)

[166] S. Günter and H. Bunke, "Validation indices for graph clustering," *Pattern Recognition Letters*, vol. 24, no. 8, pp. 1107–1113, 2003. (Cited on page 29.)

[167] B. J. Jain and K. Obermayer, "Graph quantization," *Computer Vision and Image Understanding*, vol. 115, no. 7, pp. 946–961, 2011. (Cited on page 29.)

[168] L. Guigues, H. Le Men, and J.-P. Cocquerez, "The hierarchy of the cocoons of a graph and its application to image segmentation," *Pattern Recognition Letters*, vol. 24, no. 8, pp. 1059–1066, 2003. (Cited on page 29.)

[169] H. B. Silva, P. Brito, and J. P. da Costa, "A partitional clustering algorithm validated by a clustering tendency index based on graph theory," *Pattern Recognition*, vol. 39, no. 5, pp. 776–788, 2006. (Cited on page 29.)

[170] L. Grady and E. L. Schwartz, "Isoperimetric graph partitioning for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 3, pp. 469–475, 2006. (Cited on page 29.)

[171] P. Franti, O. Virmajoki, and V. Hautamaki, "Fast agglomerative clustering using a k-nearest neighbor graph," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1875–1881, 2006. (Cited on page 29.)

[172] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007. (Cited on page 29.)

[173] P. Foggia, G. Percannella, C. Sansone, and M. Vento, "A graph-based algorithm for cluster detection," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 22, no. 05, pp. 843–860, 2008. (Cited on page 29.)

[174] N. A. Laskaris and S. P. Zafeiriou, "Beyond fcm: Graph-theoretic post-processing algorithms for learning and representing the data structure," *Pattern Recognition*, vol. 41, no. 8, pp. 2630–2644, 2008. (Cited on page 29.)

[175] H. Zanghi, C. Ambroise, and V. Miele, "Fast online graph clustering via erdős–rényi mixture," *Pattern Recognition*, vol. 41, no. 12, pp. 3592–3599, 2008. (Cited on page 29.)

[176] J. K. Kim and S. Choi, "Clustering with r-regular graphs," *Pattern Recognition*, vol. 42, no. 9, pp. 2020–2028, 2009. (Cited on page 29.)

[177] F. Wang, C. H. Ding, and T. Li, "Integrated kl (k-means-laplacian) clustering: A new clustering approach by combining attribute data and pairwise relations.," pp. 38–48, 2009. (Cited on page 29.)

[178] H. Zanghi, S. Volant, and C. Ambroise, "Clustering based on random graph model embedding vertex features," *Pattern Recognition Letters*, vol. 31, no. 9, pp. 830–836, 2010. (Cited on page 29.)

[179] D. Conte, P. Foggia, G. Percannella, and M. Vento, "A method based on the indirect approach for counting people in crowded scenes," 2010. (Cited on page 29.)

[180] S. Mimaroglu and E. Erdil, "Combining multiple clusterings using similarity graph," *Pattern Recognition*, vol. 44, no. 3, pp. 694–703, 2011. (Cited on page 29.)

[181] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watershed: A unifying graph-based optimization framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1384–1399, 2011. (Cited on page 29.)

[182] F. Nie, H. Wang, H. Huang, and C. Ding, "Unsupervised and semi-supervised learning via l 1-norm graph," pp. 2268–2273, 2011. (Cited on page 29.)

[183] S. S. Tabatabaei, M. Coates, and M. Rabbat, "Ganc: Greedy agglomerative normalized cut for graph clustering," *Pattern Recognition*, vol. 45, no. 2, pp. 831–843, 2012. (Cited on page 29.)

[184] A. Ducournau, A. Bretto, S. Rital, and B. Laget, "A reductive approach to hypergraph clustering: An application to image segmentation," *Pattern Recognition*, vol. 45, no. 7, pp. 2788–2803, 2012. (Cited on page 29.)

[185] F. Shang, L. Jiao, and F. Wang, "Graph dual regularization non-negative matrix factorization for co-clustering," *Pattern Recognition*, vol. 45, no. 6, pp. 2237–2250, 2012. (Cited on page 29.)

[186] D. LaSalle and G. Karypis, "Multi-threaded modularity based graph clustering using the multilevel paradigm," *Journal of Parallel and Distributed Computing*, vol. 76, pp. 66–80, 2015. (Cited on page 29.)

[187] M. Neuhaus and H. Bunke, "Self-organizing maps for learning the edit costs in graph matching," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 3, pp. 503–514, 2005. (Cited on page 29.)

[188] M. Neuhaus and H. Bunke, "Automatic learning of cost functions for graph edit distance," *Information Sciences*, vol. 177, no. 1, pp. 239–247, 2007. (Cited on page 29.)

[189] U. Maulik, "Hierarchical pattern discovery in graphs," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 6, pp. 867–872, 2008. (Cited on page 29.)

[190] M. Ferrer, E. Valveny, and F. Serratosa, "Median graph: A new exact algorithm using a distance based on the maximum common subgraph," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 579–588, 2009. (Cited on page 29.)

[191] M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, and H. Bunke, "Generalized median graph computation by means of graph embedding in vector spaces," *Pattern Recognition*, vol. 43, no. 4, pp. 1642–1655, 2010. (Cited on page 29.)

[192] F. Serratosa, A. Solé-Ribalta, and X. Cortés, "Automatic learning of edit costs based on interactive and adaptive graph recognition," pp. 152–163, 2011. (Cited on page 29.)

[193] A. Solé-Ribalta and F. Serratosa, "Exploration of the labelling space given graph edit distance costs," pp. 164–174, 2011. (Cited on page 29.)

[194] M. Ferrer, D. Karatzas, E. Valveny, I. Bardají, and H. Bunke, "A generic framework for median graph computation based on a recursive embedding approach," *Computer Vision and Image Understanding*, vol. 115, no. 7, pp. 919–928, 2011. (Cited on page 29.)

[195] R. Raveaux, S. Adam, P. Héroux, and É. Trupin, "Learning graph prototypes for shape recognition," *Computer Vision and Image Understanding*, vol. 115, no. 7, pp. 905–918, 2011. (Cited on page 29.)

[196] M. Leordeanu, R. Sukthankar, and M. Hebert, "Unsupervised learning for graph matching," *International journal of computer vision*, vol. 96, no. 1, pp. 28–45, 2012. (Cited on page 29.)

[197] M. Culp and G. Michailidis, "Graph-based semisupervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 174–179, 2008. (Cited on page 29.)

[198] A. Elmoataz, O. Lezoray, and S. Bougleux, "Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing," *IEEE transactions on Image Processing*, vol. 17, no. 7, pp. 1047–1060, 2008. (Cited on page 29.)

[199] W. Hu, W. Hu, N. Xie, and S. Maybank, "Unsupervised active learning based on hierarchical graph-theoretic clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 5, pp. 1147–1161, 2009. (Cited on page 29.)

[200] M. H. Rohban and H. R. Rabiee, "Supervised neighborhood graph construction for semi-supervised classification," *Pattern Recognition*, vol. 45, no. 4, pp. 1363–1372, 2012. (Cited on page 29.)

[201] B. Wang, F. Pan, K.-M. Hu, and J.-C. Paul, "Manifold-ranking based retrieval using k-regular nearest neighbor graph," *Pattern Recognition*, vol. 45, no. 4, pp. 1569–1577, 2012. (Cited on page 29.)

[202] M. Shiga and H. Mamitsuka, "Efficient semi-supervised learning on locally informative multiple graphs," *Pattern Recognition*, vol. 45, no. 3, pp. 1035–1049, 2012. (Cited on page 29.)

[203] L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, and N. Yu, "Non-negative low rank and sparse graph for semi-supervised learning," pp. 2328–2335, 2012. (Cited on page 29.)

[204] L. Gao, J. Song, F. Nie, Y. Yan, N. Sebe, and H. Tao Shen, "Optimal graph learning with partial tags and multiple features for image and video annotation," pp. 4371–4379, 2015. (Cited on page 29.)

[205] K. Riesen and H. Bunke, "IAM graph database repository for graph based pattern recognition and machine learning," pp. 287–297, 2008. (Cited on pages 30, 31, 55, 60, 64 and 118.)

[206] M. Neuhaus and H. Bunke, "A graph matching based approach to fingerprint classification using directional variance," pp. 191–200, 2005. (Cited on page 30.)

[207] J. W. Tangelder and R. C. Veltkamp, "A survey of content based 3d shape retrieval methods," *Multimedia tools and applications*, vol. 39, no. 3, pp. 441–471, 2008. (Cited on page 31.)

[208] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, "Skeleton based shape matching and retrieval," pp. 130–139, 2003. (Cited on page 31.)

[209] S. Biasotti, S. Marini, M. Mortara, and G. Patané, "An overview on properties and efficacy of topological skeletons in shape modeling," pp. 245–254, 2003. (Cited on page 31.)

[210] J. Czajkowska, C. Feinen, M. Grzegorzek, M. Raspe, and R. WickenhÃ¶fer, "Skeleton graph matching vs. maximum weight cliques aorta registration techniques," *Computerized Medical Imaging and Graphics*, vol. 46, Part 2, pp. 142 – 152, 2015. Information Technologies in Biomedicine. (Cited on page 31.)

[211] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, "Topology matching for fully automatic similarity estimation of 3d shapes," pp. 203–212, 2001. (Cited on page 31.)

[212] S. Biasotti, S. Marini, M. Mortara, G. Patane, M. Spagnuolo, and B. Falcidieno, "3d shape matching through topological structures," pp. 194–203, 2003. (Cited on page 31.)

[213] V. Barra and S. Biasotti, "3d shape retrieval using kernels on extended reeb graphs," *Pattern Recognition*, vol. 46, no. 11, pp. 2985–2999, 2013. (Cited on page 31.)

[214] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun, "A probabilistic model for component-based shape synthesis," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 55, 2012. (Cited on page 31.)

[215] H. Laga, M. Mortara, and M. Spagnuolo, "Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 5, p. 150, 2013. (Cited on page 31.)

[216] Y. Kleiman, O. van Kaick, O. Sorkine-Hornung, and D. Cohen-Or, "SHED: shape edit distance for fine-grained shape similarity," *ACM Trans. Graph.*, vol. 34, no. 6, p. 235, 2015. (Cited on page 31.)

[217] M. Pelillo, K. Siddiqi, and S. W. Zucker, "Matching hierarchical structures using association graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1105–1120, 1999. (Cited on page 31.)

[218] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento, "A comparison of algorithms for maximum common subgraph on randomly connected graphs," pp. 123–132, 2002. (Cited on page 31.)

[219] R. Myers, R. C. Wilson, and E. R. Hancock, "Bayesian graph edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 628–635, 2000. (Cited on page 31.)

[220] A. Shokoufandeh and S. Dickinson, "A unified framework for indexing and matching hierarchical shape structures," pp. 67–84, 2001. (Cited on page 32.)

[221] B. Luo and E. R. Hancock, "Structural graph matching using the em algorithm and singular value decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1120–1136, 2001. (Cited on page 32.)

[222] F. DiMaio and J. Shavlik, "Belief propagation in large, highly connected graphs for 3d part-based object recognition," pp. 845–850, 2006. (Cited on page 32.)

[223] T.-M. Su, C.-C. Lin, P.-C. Lin, and J.-S. Hu, "Shape memorization and recognition of 3d objects using a similarity-based aspect-graph approach," vol. 6, pp. 4920–4925, 2006. (Cited on page 32.)

[224] C. M. Cyr and B. B. Kimia, "3d object recognition using shape similiarity-based aspect graph," vol. 1, pp. 254–261, 2001. (Cited on page 32.)

[225] C. Bauckhage, S. Wachsmuth, and G. Sagerer, "3d assembly recognition by matching functional subparts," 2001. (Cited on page 32.)

[226] V. Garro and A. Giachetti, "Scale space graph representation and kernel matching for non rigid and textured 3d shape retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 6, pp. 1258–1271, 2016. (Cited on page 32.)

[227] A. Filatov, A. Gitis, and I. Kil, "Graph-based handwritten digit string recognition," vol. 2, pp. 845–848, 1995. (Cited on page 32.)

[228] P. Foggia, C. Sansone, F. Tortorella, and M. Vento, "Definition and validation of a distance measure between structural primitives," *Pattern Analysis & Applications*, vol. 2, no. 3, pp. 215–227, 1999. (Cited on page 32.)

[229] J. Liu, K. Ma, W. Cham, and M. Chang, "Two-layer assignment method for online chinese character recognition," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 147, no. 1, pp. 47–54, 2000. (Cited on page 32.)

[230] P. Foggia, R. Genna, and M. Vento, "Symbolic vs. connectionist learning: an experimental comparison in a structured domain," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 2, pp. 176–195, 2001. (Cited on page 32.)

[231] J. Lladós, E. Martí, and J. J. Villanueva, "Symbol recognition by error-tolerant subgraph matching between region adjacency graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1137–1143, 2001. (Cited on page 32.)

[232] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 4, pp. 331–357, 2012. (Cited on page 32.)

[233] F. Serratosa, X. CortéS, and A. Solé-Ribalta, "Component retrieval based on a database of graphs for hand-written electronic-scheme digitalisation," *Expert Systems With Applications*, vol. 40, no. 7, pp. 2493–2502, 2013. (Cited on page 32.)

[234] K. Riesen, R. Dornberger, and H. Bunke, "Iterative bipartite graph edit distance approximation," pp. 61–65, 2014. (Cited on page 32.)

[235] P. Riba, J. Lladãs, and A. Fornes, "Handwritten word spotting by inexact matching of grapheme graphs," pp. 781–785, 2015. (Cited on page 32.)

[236] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Wurtz, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture," *IEEE Transactions on computers*, vol. 42, no. 3, pp. 300–311, 1993. (Cited on page 32.)

[237] D. Maio and D. Maltoni, "A structural approach to fingerprint classification," vol. 3, pp. 578–585, 1996. (Cited on page 32.)

[238] E. Elagin, J. Steffens, and H. Neven, "Automatic pose estimation system for human faces based on bunch graph matching technology," pp. 136–141, 1998. (Cited on page 32.)

[239] C. Kotropoulos, A. Tefas, and I. Pitas, "Frontal face authentication using morphological elastic graph matching," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 555–560, 2000. (Cited on page 32.)

[240] P. Hong, R. Wang, and T. Huang, "Learning patterns from images by combining soft decisions and hard decisions," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, pp. 78–83, IEEE, 2000. (Cited on page 32.)

[241] R. Lim, "Facial landmarks localization based on fuzzy and gabor wavelet graph matching," vol. 2, pp. 683–686, 2001. (Cited on page 32.)

[242] M. F. A. Abdullah, M. S. Sayeed, K. S. Muthu, H. K. Bashier, A. Azman, and S. Z. Ibrahim, "Face recognition with symmetric local graph structure (slgs)," *Expert Systems with Applications*, vol. 41, no. 14, pp. 6131–6137, 2014. (Cited on page 32.)

[243] Z. Jin, G. Bok-Min, A. B. J. Teoh, and Y. H. Tay, "Non-invertible analysis on graph-based hamming embedding transform for protecting fingerprint minutiae," pp. 1–2, 2014. (Cited on page 32.)

[244] J. Cui, J. Wen, Z. Li, and L. Bin, "Discriminant non-negative graph embedding for face recognition," *Neurocomputing*, vol. 149, pp. 1451–1460, 2015. (Cited on page 32.)

[245] V. Pawar and M. Zaveri, "K-means graph database clustering and matching for fingerprint recognition," *Intelligent Information Management*, vol. 7, no. 04, p. 242, 2015. (Cited on page 32.)

[246] L. Gao, H. Pan, Q. Han, X. Xie, Z. Zhang, X. Zhai, and P. Li, "Finding frequent approximate subgraphs in medical image database," pp. 1004–1007, 2015. (Cited on page 32.)

[247] M. Hernandez, A. Zaribafiyan, M. Aramon, and M. Naghibi, "A novel graph-based approach for determining molecular similarity," *arXiv preprint arXiv:1601.06693*, 2016. (Cited on page 32.)

[248] L. Gao, H. Pan, X. Xie, Z. Zhang, Q. Li, and Q. Han, "Graph modeling and mining methods for brain images," *Multimedia Tools and Applications*, pp. 1–37, 2016. (Cited on page 32.)

[249] J.-C. Échallier and F. Braemer, "Nature and function of 'desert kites': new datta and hypothesis," *Paléorient*, vol. 21, no. 1, pp. 35–63, 1995. (Cited on page 38.)

[250] S. Helms and A. Betts, "The desert 'kites' of the badiyat esh-sham and north arabia," *Paléorient*, vol. 13, no. 1, pp. 41–67, 1987. (Cited on page 38.)

[251] O. Barge and J. Brochier, "Visible from space, understood during the field-work: the example of desert kites in Armenia," 2011. (Cited on page 38.)

[252] R. Crassard, O. Barge, C.-E. Bichot, J. Brochier, J. Chahoud, M.-L. Chambrade, C. Chataigner, K. Madi, E. Régagnon, H. Seba, and E. Vila, "Addressing the desert kites phenomenon and its global range through a multi-proxy approach," *Journal of Archaeological Method and Theory*, pp. 1–29, 2014. (Cited on page 38.)

[253] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988. (Cited on page 40.)

[254] A. Desolneux, L. Moisan, and J.-M. Morel, "Meaningful alignments," *Int. J. Comput. Vision*, vol. 40, pp. 7–23, Oct. 2000. (Cited on page 40.)

[255] C. Lacoste, X. Descombes, and J. Zerubia, "Point processes for unsupervised line network extraction in remote sensing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 1568–1579, Oct 2005. (Cited on page 40.)

[256] M. Rochery, I. H. Jermyn, and J. Zerubia, "Higher order active contours," *International Journal of Computer Vision*, vol. 69, no. 1, pp. 27–42, 2006. (Cited on page 40.)

[257] R. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 722–732, April 2010. (Cited on page 40.)

[258] T. Lee, R. L. Kashyap, and C. Chu, "Building skeleton models via 3-d medial surface/axis thinning algorithms," *CVGIP: Graphical Model and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994. (Cited on page 40.)

[259] P. Dosch and E. Valveny, "Report on the second symbol recognition contest," pp. 381–397, 2005. (Cited on page 60.)

[260] E. Paquet, D. De Youngster, H. Viktor, and E. Petriu, "Physics-based measurements, reflective measurements and meta-measurements for nonrigid and deformable shapes with application to structural proteomics and macromolecular docking," pp. 112–117, 2014. (Cited on page 76.)

[261] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Efficient computation of isometry-invariant distances between surfaces," *SIAM J. Scientific Computing*, vol. 28, no. 5, pp. 1812–1836, 2006. (Cited on pages 76, 90, 93, 94, 95, 100, 102, 113 and 114.)

[262] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Calculus of nonrigid surfaces for geometry and texture manipulation," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 5, pp. 902–913, 2007. (Cited on pages 76, 90, 93, 94, 95, 100, 102, 113 and 114.)

[263] J. Vleugels and R. C. Veltkamp, "Efficient image retrieval through vantage objects," *Pattern Recognition*, vol. 35, no. 1, pp. 69–80, 2002. (Cited on page 89.)

[264] J. E. Barros, J. C. French, W. N. Martin, P. M. Kelly, and T. M. Cannon, "Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval," pp. 392–403, 1996. (Cited on page 89.)

[265] K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. J. Dickinson, "Retrieving articulated 3-d models using medial surfaces," *Mach. Vis. Appl.*, vol. 19, no. 4, pp. 261–275, 2008. (Cited on page 95.)

[266] P. Shilane, P. Min, M. M. Kazhdan, and T. A. Funkhouser, "The princeton shape benchmark," pp. 167–178, 2004. (Cited on page 95.)

[267] H. Tabia, M. Daoudi, J. Vandeborre, and O. Colot, "A new 3d-matching method of nonrigid and partially similar models using curve analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 4, pp. 852–858, 2011. (Cited on pages 97 and 106.)

[268] R. Osada, T. A. Funkhouser, B. Chazelle, and D. P. Dobkin, "Shape distributions," *ACM Trans. Graph.*, vol. 21, no. 4, pp. 807–832, 2002. (Cited on page 97.)

[269] D. Vranic, *3D Model Retrieval*. PhD thesis, University of of Leipzig, may 2004. (Cited on page 98.)

[270] D. Saupe and D. V. Vranic, "3d model retrieval with spherical harmonics and moments," pp. 392–397, 2001. (Cited on page 98.)

[271] Y. Fang, M. Sun, and K. Ramani, "Temperature distribution descriptor for robust 3d shape retrieval," pp. 9–16, 2011. (Cited on page 98.)

[272] M. Reuter, F. Wolter, and N. Peinecke, "Laplace-beltrami spectra as 'shape-dna' of surfaces and solids," *Computer-Aided Design*, vol. 38, no. 4, pp. 342–366, 2006. (Cited on page 98.)

[273] M. Abdelrahman, M. T. El-Melegy, and A. A. Farag, "Heat kernels for non-rigid shape retrieval: Sparse representation and efficient classification," pp. 153–160, 2012. (Cited on page 98.)

[274] D. Smeets, J. Hermans, D. Vandermeulen, and P. Suetens, "Isometric deformation invariant 3d shape recognition," *Pattern Recognition*, vol. 45, no. 7, pp. 2817–2831, 2012. (Cited on page 98.)

[275] P. Daras and A. Axenopoulos, "A compact multi-view descriptor for 3d object retrieval," pp. 115–119, 2009. (Cited on page 98.)

[276] H. Dutagaci, A. Godil, A. Axenopoulos, P. Daras, T. Furuya, and R. Ohbuchi, "Shrec'09 track: Querying with partial models," pp. 69–76, 2009. (Cited on pages 98 and 99.)

[277] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. (Cited on page 99.)

[278] D. M. W. Powers, "Evaluation: from precision, recall and f-factor to roc, informedness, markedness and correlation," *School of Informatics and Engineering Technical Reports*, no. SIE-07-001, 2007. (Cited on page 99.)

[279] K. Iwama and S. Miyazaki, "A survey of the stable marriage problem and its variants," pp. 131–136, 2008. (Cited on page 118.)