FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A Graph Centrality Approach to Computer Vision

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Samuel Felix de Sousa Junior
Registration Number 1228089

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: o. Univ.-Prof. Dipl. Ing. Dr. Walter G. Kropatsch

External reviewers:
Dr. Dennis Strelow. Google Inc., Mountain View, California, United States of America.
Prof. Francesc Serratosa. Universitat Rovira i Virgili, Tarragona, Catalonia, Spain.

Vienna, 17th August, 2015

Samuel Felix de Sousa Junior | Walter G. Kropatsch

Declaration of Authorship

Samuel Felix de Sousa Junior
Favoritenstraße 9/2/186-3, 1040, Wien, Österreich

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 17th August, 2015

Samuel Felix de Sousa Junior

Acknowledgements

All doctors, from the newest graduate until the tenured professor cultivate the dream of impacting the world with their research. Their research becomes part of themselves and as such it would lead them to find the hidden treasures in science. This is the ambition of every new pupil trying to master its way through the scientific world. Throughout time, they realize that they are nothing but a small piece of a gigantic jigsaw puzzle, a piece which would not make sense if it was not by the other pieces giving meaning to the whole. My doctoral thesis was only possible because others supported me in this journey and I thank them for coming along this whole time.

I dedicate this work to my family, none of this could ever have existed without them. I thank my parents Silsa and Samuel and my sister Samantha for teaching me valuable lessons which I carry with me every day. My father has always been the dreamer of the family and my mother the curbstone which kept us united. My family supported my dreams and helped me fixing my hope on a better future.

Education makes it impossible to go back to the primitive ways, when we would be satisfied with *panem et circenses*. I thank all the ones who taught me something, from the most complex and deepest theorem to the simplest and most humble tip. I thank my *Doktorvater*, prof. Walter Kropatsch for welcoming me at PRIP during these years in Austria, for being always present whenever I needed, for all the discussions and paper reviews. It is truly a privilege for a student to be able to talk to his supervisor whenever he needs. I also thank all the friends I made at PRIP, Ines and Michi, who always taught me the Austrian way of doing things, the dialects, and constant motivation to teach me *Wienerisch*, *danke sehr*! I also would like to thank my first supervisor prof. Cleidson de Souza, who introduced me to the field of Social Networks almost ten years ago, even though I diverted to Computer Vision, it turns out I found a way of combining both fields.

My internships at Google and Siemens gave me real industrial experience, but they gave me much more than that. I owe Jan Ernst a great deal, since it was him who showed me how to structure myself better in order to tackle complicated problems, he also introduced me to the beautiful field of machine learning. I learned many life lessons with Dennis Strelow and I am deeply grateful for his guidance, advice, and friendship. He always paid attention to my ideas, even the ones which did not make much sense, showing me how one can exercise patience in real life.

Finally, there are friends who are closer than brothers. I remember how happy Joanna Zysiak was when I first told her that I would be back in Europe for my PhD. I have

no words to express the gratitude for her friendship, I am just lucky to have found her. Marina Oikawa, Luana Leite, and Séverine Habert shared many moments of my life and supported me before, during, and I believe after my PhD. They are friends who stick for life even though we have been far away from each other during all these years. Luana had to personally validate all my documents in the Brazilian Ministry of Foreign Affairs so I could start my PhD.

I met many great people in Austria, which made my life here more pleasant than I could ever have imagined, Magnus Köhler and Paul Wandl are my closest friends here and they were always there for me whenever I needed or struggled with the German language. When I think of Christina Lengauer, the first thing that comes to my mind is her beautiful smile and delicious cakes which made me very happy throughout my PhD. Finally, at last but not least, I owe many thanks to Lucas Gerrand, who just like me, came to Austria from a very sunny place in the opposite hemisphere. He proof read many of my papers and was always happy to do so. During my studies, I was supported by the PhD School of Informatics and I thank the organizers of the program for giving me this opportunity. Special thanks go to Clarissa Schmid and Mamen for being so kind and answering all my questions, and there were many of them.

Kurzfassung

Das maschinelles Sehen hat die Art und Weise, wie wir mit Computern interagieren, von Grund auf verändert; von der ersten automatischen Gesichtserkennung bis hin zu den neuesten Augmented-Reality Anwendungen hat es eine komplett neue Ära eingeläutet. Die Analyse von sozialen Netzwerken, die in den letzten Jahren populär geworden sind, basiert auf Graph-basierten Konzepten. Die Zentralität ist eines davon. Die Zentralität bestimmt, wie wichtig ein bestimmter Knoten innerhalb eines Netzwerks ist. In dieser Dissertation zeigen wir Anwendungen des Konzepts der Zentralität für die Lösung mehrerer Probleme, die im Bereich der Computer Vision auftreten und untersuchen, wie gut die Zentralität in dieser neuen Domäne angewendet werden kann.

Das erste Problem, das wir untersuchen, ist die Registrierung einer Punktwolke, welches immer dann auftritt, wenn Feature-Punkte, Orientierungspunkte oder Punkte, die die Oberfläche eines Objekts repräsentieren, zwischen zwei oder mehreren Bildern bestmöglich in Übereinstimmung gebracht werden müssen. Dazu führen wir in einem ersten Schritt die Zentralität in den Coherent Point-Drift Algorithmus ein und analysieren, wie sich dieser unter verschiedenen Szenarios verhält.

Im weiteren Verlauf dieser Dissertation stellen wir unseren eigenen, auf der Zentralität basierenden Registrierungs-Algorithmus vor, dessen primäre Aufgabe es ist, einen unterscheidbaren Graphen zu erzeugen, dessen Verteilung der Zentralität möglichst gleichmäßig ist, um die Registrierung zu erleichtern. Auch das Erstellen von Diagrammen ist der Registrierung einer Punktwolke sehr ähnlich: in beiden Fällen soll ein Graph aus unstrukturierten Daten erstellt werden. Wir untersuchen die aktuellen Techniken für das Lösen dieses Problem unter besonderer Beachtung der Delaunay-Triangulation, einem gebräuchlichen Verfahren, um aus einer Punktemenge ein Dreiecksnetz zu erstellen. Wir präsentieren unsere stabile Graphen-Generierung, die versucht, mit Hilfe eines Optimierungsverfahrens einen minimalen Spannbaum von maximaler Entropie zu erhalten. Wir evaluieren die Robustheit der generierten Graphen unter Berücksichtigung von unterschiedlichen Mengen von Rauschen in der Nachbarschaft von Punkten und vergleichen die Stabilität der Ergebnisse mit denen der Delaunay-Triangulation.

Der letzte Aspekt, den wir in dieser Dissertation untersuchen, ist der Einfluss der Zentralität für Verfahren des maschinellen Lernens. Dazu beschreiben wir zunächst die verschiedenen Möglichkeiten, wie mit unserem Verfahren Merkmalsvektoren modelliert werden können, die statt der Pixel des Bildes die Zentralität der entsprechenden Knoten verwenden. Danach trainieren wir Algorithmen des maschinellen Lernens auf der Grundlage dieser Zentralität-basierten Merkmalsvektoren, um eine Klassifizierung von

Formen vornehmen zu können. Wir stören die Topologie des Graphen mit unterschiedlichen Mengen an Rauschen und bewerten, wie gut unsere trainierten Klassifikatoren abschneiden.

Diese Dissertation behandelt erstmalig die Verwendung der Zentralität eines Graphen zur Lösung einer Vielzahl von Problemen im Bereich des maschinellen Sehens. Das Ziel dabei soll nicht nur sein, die Zentralität als neues Verfahren zu etablieren, sondern auch zu untersuchen, wie effektiv sie bei der Lösung der Probleme im Bereich des maschinellen Sehens ist und abzuschätzen, wann und wo ein bestimmtes Zentralitäts-Maß am besten abschneidet.

Abstract

Computer vision has changed the way we deal with computers, since the first automatic face detection algorithm to the latest augmented reality application, brought us into a new era of computing. The study of social networks models human relations using graph theory and it has been emerging in the recent years. This field designs the concept of centrality by defining how important a certain node is within a network. In this dissertation, we study the centrality concept applied to computer vision and show how this information can be used in such a different domain.

We start by analyzing the point-set registration problem. This problem often arises in computer vision whenever one needs to match information available in images. The kind of information to be matched consists of feature locations, landmarks, or points representing a surface of an object. We first introduce various centrality measures into the Coherent Point-Drift algorithm and analyze their behavior under different scenarios. Later, we propose our own centrality-oriented matching algorithm, focusing on the generation of a distinguishable data-graph. We aim at obtaining a distribution of centrality as uniform as possible, thus, alleviating the matching process.

The construction of the data-graph is also related to the registration of point-sets, whose goal is to build a graph from unstructured points. We review the current techniques for this problem, paying important attention to the Delaunay algorithm, an off-the-shelf solution to build a triangulation of points. We pose our stable data-graph generation using an optimization formulation whose goal is to obtain the minimum spanning tree of maximum entropy. We evaluate the robustness of the generated data-graph in the presence of different amounts of noise in the local neighborhood of points and compare the stability of the results with Delaunay.

The last aspect studied in this dissertation is the impact of centrality in a machine learning scenario. First, we describe different approaches to model feature vectors using the centrality of nodes instead of pixels. Later, we train machine learning algorithms using those centrality-based features focusing on the classification of shapes. We evaluate our approach by perturbing the topology of the graph using a different amount of noise and evaluate how well each trained classifier performs.

This dissertation introduces graph centralities in a variety of computer vision problems for the first time. Our goal is not only to apply those measures, but to understand them, study their behavior in the different vision problems, and to estimate when a certain centrality measure can perform better than another.

Resumo

A visão computacional mudou a forma que lidamos com a computação, desde a primeira detecção automática de faces até o algoritmo de realidade aumentada mais avançado. O estudo das redes sociais modela as relações humanas usando teoria de grafos e tem recebido grande atenção nos últimos anos. Neste campo, existe uma medida chamada *centralidade* que define a importância de um nó dentro de um grafo. Nesta tese, centralidades são aplicadas pela primeira vez em uma grande diversidade de problemas de visão computacional e demonstraremos como utilizar esta medida em um domínio tão diferente.

Estudamos o registro de nuvem de pontos, que surge frequentemente em visão quando é necessário encontrar a correspondência de informações disponíveis em imagens, tais como locais de saliência ou outros pontos representando a superfície de um objeto. Introduzimos as diversas medidas de centralidade no algoritmo *Coherent Point Drift* e analisamos os seus comportamentos em diferentes cenários. Propomos, também, um novo algoritmo para a correspondência de pontos baseado nas centralidades que foca na geração de um grafo distinto. O objetivo é obter uma distribuição de centralidade que seja próxima de uma distribuição uniforme, de forma a amenizar o problema da correspondência.

A construção do grafo espacial também está relacionada ao registro de nuvens de pontos, cujo objetivo é construir um grafo a partir de pontos não estruturados. Revisamos as técnicas atuais para este problema, focando especialmente no algoritmo de Delaunay, que é uma solução padrão para construir uma triangulação de pontos. Nós modelamos a geração de um grafo espacial usando uma formulação de otimização, cujo objetivo é obter uma árvore geradora mínima de entropia máxima.

Avaliamos a robustez dos grafos gerados na presença de diversos níveis de perturbações nas vizinhanças dos pontos e comparamos os resultados com o algoritmo de Delaunay. O impacto da centralidade de grafos em aprendizado de máquina é o último aspecto abordado nesta tese. Primeiramente, descrevemos várias maneiras para modelar os vetores de características usando as centralidades ao invés dos *pixels* de imagens. Posteriormente, treinamos algoritmos de aprendizado baseados nos vetores características obtidos pelas centralidades focando no problema de classificação de formas. Perturbamos as topologias dos grafos usando vários níveis de ruído e avaliamos a performance dos algoritmos treinados. Esta tese introduz as centralidades de grafo em uma variedade de problemas de visão computacional pela primeira vez. O objetivo não é apenas aplicar tais medidas, mas entender os seus comportamentos e estimar quando uma medida de centralidade poderá obter melhor performance em um cenário específico.

Contents

List of Figures

List of Tables

List of Algorithms

List of Theorems

CHAPTER 1

Introduction

> "Who can tell that a machine can not learn? Just because it does it in a different way than you do, it does not mean it is not learning."
>
> – Unknown

Many fields in science were inspired by the desire of building machines capable of thinking and perceiving the world as we do. Alan Turing himself envisioned that computers could go beyond simple arithmetic. He was puzzled by the question if machines could ever think when he proposed the Imitation Game [121]. We are not going to dwell on such a question here, but it is worth-mentioning that Artificial Intelligence (AI) was born to ultimately develop machines as intelligent as humans, to create the artificial brain. In the early 1960s, researchers jumped on modeling an artificial neural network (the perceptron algorithm [100]) and later (1974) the back-propagation [127] algorithm in order to emulate brain activity. One could say that it was the birth of the Machine Learning (ML) field.

At some point in the 1990s, machine learning decoupled from artificial intelligence as it aimed at solving less ambitious problems compared with the artificial brain, such as recognizing digits from images. Some of those problems could actually be solved given a sufficient amount of data and a suitable learning algorithm. This new direction focused at looking at the data hoping that the data itself would help the researchers solve the problem. The goal was not only to find out what was happening in the problem but how one could extrapolate beyond what has been seen and make inference about unpredicted behavior. At this point in time, many techniques were borrowed from statistics and probability theory [67] and incorporated into machine learning. A milestone occurred when the Support Vector Machines (SVM) [23] algorithm was proposed by finding a linear separator between classes (the support vector) and later allowing a non-linear separation by using kernel methods. Machine learning continued evolving and more recent research indicates that ML is coming back towards its original artificial intelligence roots with the

return of neural networks in deep architectures (Deep Learning) which is breaking brand new records not only in digit [49], but also in facial [116] and speech [54] recognition at an impressive rate like never seen before.

The artificial brain was not the only dream that scientists had, but also envisioned the development of artificial eyes and thus, giving the machines the ability to see and think. This was the time when the first digital images were produced and the ground was set for the beginning of new fields of digital images. Due to the newspaper industry, digital images started being transmitted via a submarine cable[1] in the early 1920s between London and New York. We could consider this as the birth of the image processing field since it was important to start developing methods to improve the printing procedures. For instance, the number of gray tones displayed increased from 5 (in the initial Bartlane transmission) to 15 gray tones in 1929 [48]. The goal of image processing was to develop algorithms that would "process" an input image in order to generate output images enhancing or modifying certain aspects. Research in Computer Vision (CV) started flourishing at this time, and simple operators (such as the Sobel operator to compute derivatives) were proposed to extract information from images. Images started not only to display a moment for the viewer's appreciation, but they were telling us things about it, e.g. curvatures, high gradient information, salient points.

As time passed and technology evolved, we started to consolidate fields which would work together to build a comprehensive understanding of the scene. Figure 1.1 shows the relationship between different fields related to digital images. A scene is imaged by a sensor (by a process called acquisition), which could happen in many ways, by an ordinary camera, or using more sophisticated depth devices, range sensors, etc. When the image passes through this low-level processing (denoising, filtering, etc.), we could say it belongs to the Digital Image Processing (DIP) field, whose input and output are composed of images. The output would be an image with a specific property enhanced. This includes operators such as sharpening, brightening, edge detection, etc. We are entering the CV field when the goal is to estimate the models projected onto that image. There is a vast literature on reconstruction of objects from one, two (stereo), or multiple views (trifocal tensor, bundle adjustment, etc). Once models are acquired, they could be rendered for other purposes (movies, games, augmented reality) which is the responsibility of Computer Graphics (CG), as they are the ones rendering realistic scenes. Finally, we close the gap of image understanding when given all the knowledge we gathered from the image, we are able to tell what is happening in the scene, i.e. there are two trees and a country house. This is considered high level vision when semantics are involved. We can achieve such results by bringing the ML field into the game with its powerful algorithms.

Computer vision is the field we are mostly interested in, as it is the one which combines both learning and image processing in order to emulate the human vision, whose ultimate goal is not only to observe or reconstruct a scene but also to understand it. One could argue that there are no clear boundaries, or consensus among researchers, on where image processing ends and computer vision starts. In the geometrical aspect of computer vision,

[1]The Bartlane picture transmission system transmitted around 500 pictures of important events between London and New York before the outbreak of the second world war in Europe [81].

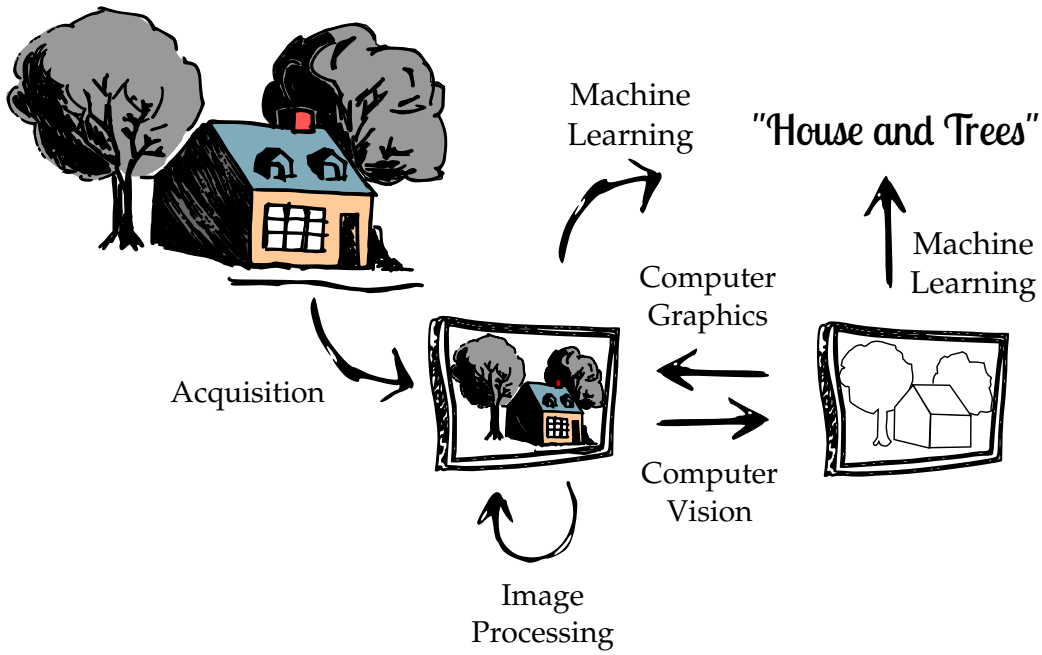


Figure 1.1: Scene Understanding. Low level vision: an image is acquired using a sensor and processed. Mid-level vision: using information enhanced by processing the image, one can estimate the geometry of the scene. High level vision: by combining machine learning, one can identify semantics of the scene.

it could also be interpreted as the opposite field of computer graphics, where the former tries to the estimate the model given an image, and the latter is interested in rendering the image given a certain model with as much realism as possible.

1.1 On the Computer Vision Road

From the problems in images and learning, we will start narrowing towards computer vision problems. CV could be decomposed into many other fields which work together to grasp an understanding of the whole scene. As aforementioned, CV can tackle a wide range of problems varying from acquisition (the process in which an image is formed) to more complex ones such as reconstruction, localization, and understanding.

Since we can not cover all topics relevant to CV here in this dissertation, we would like to briefly talk about some of the problems that will be addressed later. The goal in this section is build the intuition and importance behind each problem. Later, we will provide a formal definition of the individual problems, revisit the state of the art and

explain how we are going to address such a problem by bringing graph centralities into computer vision. Graph centralities in computer vision are the main novelty introduced in this dissertation.

1.1.1 A glimpse on registration

We will spend a considerable amount of time dealing with the registration problem. Registration is a widely explored topic in Computer Vision and an essential piece in feature matching [73], stereo vision [107], tracking [60], medical imaging [52] and many other applications. Intuitively, the problem consists of retrieving the transformation and computing the correspondence between two or more point-sets on their own coordinate systems. Approaches for registration can be classified into either rigid or non-rigid methods. Rigid registration only considers rigid transformations while non-rigid registration allows for more involved tasks once it can account for scaling, skews, and complex *deformations* (e.g. articulations, morphing). We could exemplify this problem by looking at two objects in different images and tracing their correspondences.

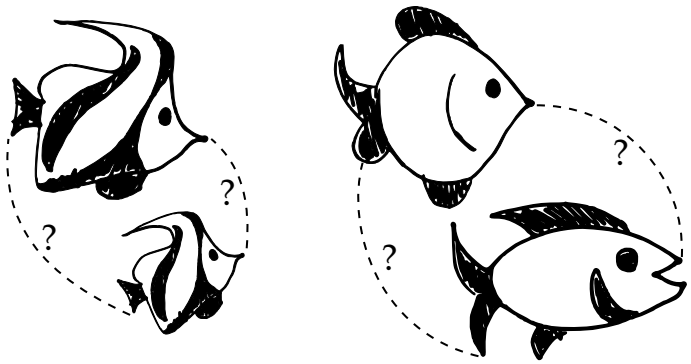


Figure 1.2: Example of Image Registration. In Figure a) two fishes are related by a similarity transformation. In Figure b) two fishes are related by a non rigid transformation.

Figure 1.2 shows two scenarios for image registration. As seen above, Figure 1.2a shows two fishes related by a similarity transformation, which is composed of a rotation, translation, and isotropic scaling, i.e. the object is scaled equally in all dimensions, note that we will refer again to the different types of transformations in Chapter 3. Figure 1.2b, in contrast, shows two fishes related by a non-rigid registration. In fact, they are not the same type of fish. It is a trivial task for humans to match the different parts of two distinctive fishes although. Nevertheless, it is still a challenging task for a computer to find the right correspondence.

There are several surveys reviewing algorithms for both rigid and non-rigid registration techniques. Papers differ in their focus from medical imagery [6] to range images [99], 3D shapes, and point clouds [93, 106, 124, 117]. Despite the extensive number of approaches

available in the literature, two popular iterative techniques are worth mentioning: The Iterative Closest Point (ICP) [12] and the Coherent Point Drift (CPD) [86]. They have been investigated and extended by several authors [102] and we will cover both algorithms in Chapter 3. We focus our attention on CPD, since in Chapter 5 we propose an extension of CPD through the addition of graph centralities. However, this is not our only contribution to centralities for registration. In addition, we also propose a centrality-based algorithm for registration which is independent of CPD in Chapter 6.

1.1.2 Breaking an image into pieces

Image segmentation is a prevalent problem in CV whose goal is to decompose an image into its parts. This problem has been broadly studied [15, 16]. It is somehow similar to the clustering problem when points in a certain d-dimensional space are given and we would like to separate them in clusters. When segmenting an image, one could consider only the color information in a gray scale (256 colors for a 8-bits unsigned integer channel), or 256^3 colors for an Red, Green, and Blue (RGB) image. Nevertheless, in the computer vision task, we have not only the intensity information (color), but also spatial information associated with the points. Each point in our space could be seen as a quintuple $< r, g, b, x, y >$. The way points are segmented (or clustered) vary according to the algorithm used.

a) b)

Figure 1.3: Image Segmentation. Image a) shows a man with a hat and the segmentation attempt on image b) decomposes the image into several parts. Reprinted from de Sousa et al. [32], with kind permission of Springer Science and Business Media.

One popular approach for clustering is the K-Means [78] algorithm which given a number of clusters to be found in the image, assigns a centroid for each cluster. Those centroids move towards the most suitable location until the algorithm outputs the final label

for the points. A more powerful approach for image segmentation consists of minimizing the Mumford-Shah functional [85] which models the problem as an optimization task. It is a variational approach which establishes an optimality criteria for segmenting an image into regions. Many approaches were proposed in order to minimize the Mumford-Shah functional via convex graph-cuts relaxation and discrete optimizations [50, 92].

An image can also be segmented via graph-cuts technique [15, 16, 51], which associates a node to each pixel of the image. In such a formulation, there is the definition of source (s) and sink (t) nodes. The solution consists of finding the maximum flow between s and t, which is equivalent to finding the minimum cut in the graph. The result obtained by the min cut is a segmentation of the image. Alternatively, the segmentation problem has also been possible via Normalized Cuts approach by Shi and Malik [113]. We also addressed such a problem, but instead of the minimum or normalized cut, we modeled the segmentation problem with the maximum cut [32]. A clear difference between min-cut and max-cut is that min-cut can be solved in polynomial time, whereas the max-cut belongs to the $\mathcal{NP}$-Hard class, which means unless $\mathcal{P} = \mathcal{NP}$, Max-Cut cannot be solved in polynomial time.

1.1.3 Separating the wheat from the chaff

One of the most important tasks in vision consists of separating objects into classes. You could assess the importance of such a problem to trying to navigate within a scene in complete darkness holding a device which could tell you where and which objects exist in the scene. This could change the lives of blind people as it could assist them with household tasks, e.g. to identify a corn can from tomato sauce. We could go further and state that it is an essential task for self driving cars to be able to recognize a pedestrian from a tree.

Applications for classification problems are countless, with each day more and more devices will be produced incorporating some sort of trained algorithm to assist humans. In essence, this goes beyond computer vision, as this is a machine learning problem which is not aware that we are training an image, plain text, or numbers. Humans have the innate skill of learning the representation of an object and we are able to classify a new object into an already learned class. In order to train the computer to be able to recognize an object, we need an algorithm which is able to split the data according to an objective function. In order to obtain good classification performance, we also need to feed considerable amount of training data.

We deal with classification related problems on a daily basis. Figure 1.4 shows a classification problem that we could face in real life. We have a box of tea and many bags of teas to separate, whenever we have to refill the box, we need to find the correct class for the new tea bag. In this box, the teas were manually classified and organized according to their type (feature). One could choose the tea bag color as a feature for classification, although it would be a meaningful criteria, there are two bags with the same color and yet different types. This example illustrates how hard it is to obtain automatic classification.

Figure 1.4: Tea bags organized according to their classes.

There are many complex tasks involving the classification problem, for instance, the differences in the learning architecture to be used. One could call a shallow architecture when we create a feature vector representation for our object in a high-dimensional feature space and try to split the feature spaces into the desired classes. The goal is to design a representation which would hopefully allow the features to be separable in a high dimensional feature space, i.e. features of the same class would end up in the same location in the feature space. Algorithms can also be divided by the type of separations they make, some algorithms require those classes to be linearly separable (e.g. SVM), while others can adapt towards more complex separations (e.g. decision forests), but at the risk of overfitting the training data. Regarding SVM, there are many approaches which allow non-linear separations, such as the ones using kernel methods [66]. In Chapter 4, we will perform a centrality-based learning. We will model the shape of an object using the centrality information of a graph. This is a novel approach and to the best of our knowledge, we were the first ones to address the problem in this fashion.

1.1.4 Playing the game with graphs

We have briefly spoken about graphs and will later expand on the concept (Chapter 2). Graphs are a useful, yet powerful, data structure in computer science. Most of the problems we mentioned can be modeled as a graph problem. In Chapter 2, we will formally introduce graphs, but for now, it suffices to look at an example in Figure 1.5. We show the image of a fish and a possible graph representation. We call a graph created out of a point-set as the data-graph. In this example, the data-graph is the output of a triangulation. We will discuss many ways that graphs could be represented. A popular representation is the Delaunay triangulation [33], which is the dual of the Voronoi diagram.

In Chapter 6, we create our own type of data-graph in which the generated graph has some important properties. In Chapter 7, we propose another type of graph based on a particular type of Minimum Spanning Tree (MST) built upon a multi-objective cost function. We compare our approach with Delaunay regarding stability of the data-graph to small variations in the neighborhood of the points.

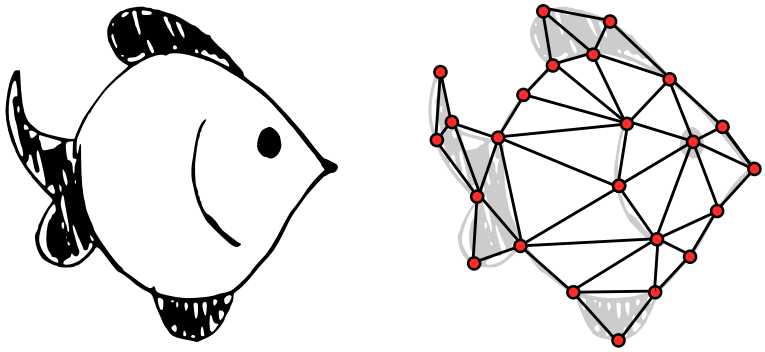


Figure 1.5: Graph Representation of a Shape. Figure a) depicts a fish, b) Graph triangulation created to represent the original shape.

Previously, we introduced the registration problem, but now we could cite some approaches which model this problem using graph theory [122, 25, 77, 75, 18, 130, 131]. Graph-based approaches provide structural information in which edges could describe relationships between entities (e.g. components of an object). On the one hand, when there is no need to recover the spatial adjacency matrix or such information is not available, the correspondence problem is reduced to the Graph Matching (GM) problem [21]. On the other hand, there are approaches estimating the spatial transformation without finding the correspondence between points [41].

The approaches for image segmentation using graph-cuts is one of the most successful usages of graphs in computer vision. This motivates why it is important to learn and study graphs. Probabilistic Graphical Models (PGM) aim at using graph theory together with probability theory in order to tackle more complicated problems such as the Markov Random Field (MRF) which we mentioned in the segmentation problem. MRF are based on undirected graphs, but there are graphical models based on directed graphs such as the Bayesian Networks. As discussed, we pointed out some useful techniques using graphs, therefore, they are essential for computer vision.

1.2 About Relations and Importance

We have briefly discussed the problems that will be addressed later using graph centralities. We will give an intuition about centralities and how they were conceived. Computer vision

is not the only field using graph theory, in fact, many fields in science take advantage of graph representations. In particular, the field of Social Network Analysis (SNA) has employed the concept of centrality for many years [43, 1, 8, 90, 30, 7]. The notion of centrality tries to capture the measure of importance within a network (graph). Although well-known for sociologists, such a concept has not been broadly explored outside SNA. By using centralities, one can state that a certain node is more relevant than another or that it is possible to rank the nodes according to their importance.

In this dissertation, we bring the idea of graph centralities into computer vision problems, such as registration of point-sets, the centrality-driven data graph creation, and even image classification. For instance, when we extend the CPD algorithm, we propose to integrate the centrality measures during the computation of correspondence of points. Thus, we consider not only the spatial information but also the importance of nodes according to each centrality measure.

The flavor of centrality has already been applied in some problems in computer vision before, especially on the approaches which emerged from the spectral graph theory [122, 108, 111, 19, 69]. Nonetheless, to the best of our knowledge, our approach is the first one to explicitly adopt centralities for the problems previously mentioned. The techniques which use similar concepts often explore eigenvectors of some proximity matrix encoding the location [108, 111] or the orientation of points. Some centralities (e.g. eigenvector) work directly on the eigenvectors of the adjacency matrix. We aim at showing that graph centralities can be used as good prior information in computer vision. Also, exploiting topological information could bring us more benefits instead of solely relying on the spatial information for points. Thus, one could extend those ideas to other vision tasks as well.

1.3 Goals and Contributions

The ultimate goal of this dissertation is to perform an in-depth analysis of graph centralities in Computer Vision. Starting from their definition, creation and historical usage until practical applications in vision with experimental evaluation. To the best of our knowledge, we were the first to introduce centralities in many tasks. Our contributions can be listed as follows:

1. We introduce the centrality concept originated in the Social Sciences into many computer vision problems and provide a literature review on their applications in Chapter 2.

2. We propose to model a shape in a machine learning context by using the centrality information of its graph representation (Chapter 4).

3. We introduce the centrality measure into the CPD algorithm, thus, integrating topological information into the registration problem (Chapter 5).

4. We propose to register point-sets by generating a data-graph which is driven by centralities (Chapter 6).

5. We propose a MST formulation also driven by the centrality distribution of the generated tree (Chapter 7).

6. We show how one can integrate topological information by those centralities in a diversity of problems (Chapters 2,4,5,6,7).

1.3.1 Publications

The results obtained in this dissertation were published in proceedings of conferences, book chapters, and journals. We have obtained permission to reprint the material from Springer and Elsevier and the licenses are available in Appendix A.

1. Samuel de Sousa, Walter G. Kropatsch. *Data Graph Formulation as the Minimum-Weight Maximum-Entropy Problem*, Graph-Based Representations in Pattern Recognition, Lecture Notes in Computer Science, Springer Verlag, Volume 9069, 2015, pp. 13–22, May 2015. [2]

2. Samuel de Sousa, Walter G. Kropatsch. *The Minimum Spanning Tree of Maximum Entropy*, 39th annual workshop of Austrian Association for Pattern Recognition (OAGM/AAPR), Salzburg, Austria, 2015[3].

3. Samuel de Sousa, Walter G. Kropatsch. *Graph-based point drift: Graph centrality on the registration of point-sets*, Elsevier Pattern Recognition, Volume 48, Issue 2, February 2015, Pages 368-379, ISSN 0031-3203[4].

4. Samuel de Sousa, Nicole M. Artner, Walter G. Kropatsch. *On the Evaluation of Graph Centrality for Shape Matching*, Graph-Based Representations in Pattern Recognition, Lecture Notes in Computer Science, Springer Verlag, Volume 7877, 2013, pp 204-213[2].

5. Samuel de Sousa, Yll Haxhimusa, Walter G. Kropatsch. *Estimation of Distribution Algorithm for the Max-Cut Problem*, Graph-Based Representations in Pattern Recognition, Lecture Notes in Computer Science, Springer Verlag, Volume 7877, 2013, pp 244-253[2].

1.4 Roadmap of this Dissertation

Chapter 2 brings the theory necessary to understand the concept of centrality. It provides a brief introduction of graph theory and discrete geometry. It is important to define concepts such as metrics and distances in graphs as soon as possible in order to understand the remainder of this dissertation. We explain how one can model those concepts in

[2]Reprinted with kind permission of Springer Science and Business Media.

[3]I own the copyright of this work and it is publicly available at arXiv, no additional permission from any publisher is necessary.

[4]Reprinted with permission of Elsevier.

graphs and how they are applied in our algorithms. We also overview earlier usages of centrality ideas in computer vision. As our main contributions consist of centrality measures, it is essential to understand these concepts.

Chapter 3 formally defines the registration problem and provides a literature review on related algorithms focusing on those whose methodologies are closely related to ours. ICP and CPD algorithms are detailed in this chapter. Moreover, in this chapter we define the different types of transformation between points which are addressed later by the different registration algorithms.

Chapter 4 uses of graph centralities in the classification of shapes. We create a graph-based representation of a shape and describe this graph by using different centrality measures. We briefly talk about some learning algorithms, general training vs. testing scenarios and we analyze the performance of the different centrality measures based on a Naïve Bayes classifier.

Chapter 5 explains our algorithm for point-set registration as an extension of the CPD by integrating the degree, betweenness, closeness, and eigenvectorcentralities. The centrality values bring topological information used during the computation of correspondence between points. We analyze the performance on several datasets and compared with the traditional CPD algorithm.

Chapter 6 formulates the registration problem in a very unusual way. We are interested in finding a graph that would represent s point-set according to some properties. Such a representation would allow us to match two objects (graphs) by exploiting topological properties instead of solely relying on geometrical properties. Our desired graph is (i) as unique as possible and (ii) and discriminative as possible regarding the degree distribution. We pose a combinatorial optimization problem, the Minimum-Weight Maximum-Entropy (MWME) to build such a graph by minimizing the total weight cost of the edges and at the same time maximizing the entropy of the degree distribution. Our optimization approach is based on Dynamic Programming (DP) and yields a polynomial time algorithm.

Chapter 7 deals with a similar problem in nature to the MWME. We pose the problem of the Minimum Spanning Tree of Maximum Entropy (MSTME). The goal in this chapter is not to perform a full registration, but to create a data-graph construction technique which can be more stable than the Delaunay triangulation regarding variations in the neighborhood of points. Although we are not tackling the registration problem in this chapter, the type of data-graph we create is still useful for easing the registration process. Considering that we are maximizing the entropy of the degree distribution, we will still have a higher variability than the standard minimum spanning tree and we can take advantage of such property to reduce the search space when matching points. We performed experiments and compared with Delaunay's in a number of different data-sets in order to evaluate the stability of the data-graph created.

Chapter 8 presents our conclusions regarding graph centrality in computer vision. We discussed the general results obtained in each problem, the limitations, the outcomes, and we present our future research directions, as well as open questions.

CHAPTER 2

Graph Centrality and Discrete Geometry

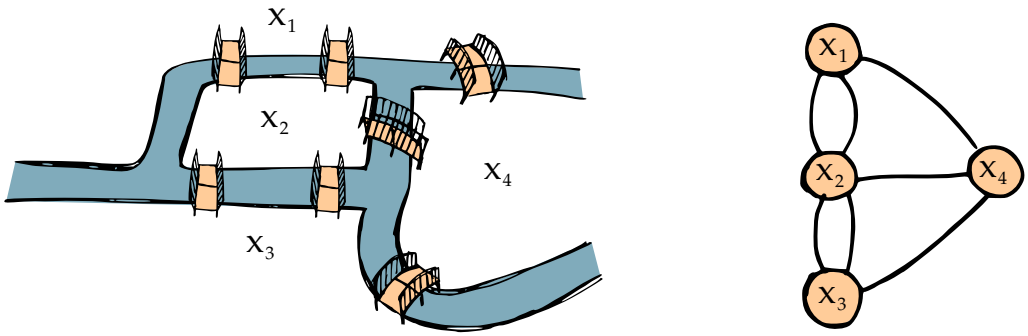
> "Our experience of the world depends on the actual structure of the networks in which we are residing and on all the kinds of things that ripple and flow through the network."
>
> – Nicholas Christakis

People are connected. There are several interconnected clusters being created and modified as you read this dissertation. Networks are created by the natural human relations happening at every moment. We evolve those live networks when we communicate at work, or when we share activities such as sports in which a common interest led us to develop and grow our bonds with others. The connections we build and destroy define this complex and live organism, often referred to *the social network*. As one can imagine, there is not only one type of network, but you are most likely living within many different types of networks. It is easy to associate a connection as a friendship, but the connections do not necessarily imply such a fact. In fact, people might be connected without knowing it, as when doctors analyze the growth of an epidemic to predict the contamination ratio or when they need to find, in real time, new areas which are spreading the disease [47]. Many processes can be modeled as networks, from alcohol consumption behaviour [101] to scene categorization in images [71]. The main contribution of this dissertation is to model computer vision problems by using measures of centrality which were born in social networks. In this chapter, we cover graph theory concepts which are necessary to understand centralities. We also briefly overview some relevant concepts in geometry for bridging the gap between computer vision and social networks.

A network is commonly represented using a graph, which is a formal way to describe a world and its relations in discrete mathematics. Before diving into the centrality world,

it is convenient to define some ground knowledge on discrete geometry which will allow us to pose problems associated with centralities later on.

The study of graphs can be dated to Euler [38] who posed the problem of the Königsberg bridges. Figure 2.1 shows an schematic representation of the problem faced by Euler: to determine whether or not it was be possible to cross each bridge exactly once. Euler found out that the desired tour through the Königsberg bridges was not possible. This is what we call an *Euler tour* or *Eulerian cycle*: a cycle using each edge exactly once, or coincidentally for connected graphs, having all nodes with even degree. It was the birth of the so-called graph theory and we will use this powerful abstraction to model many of our problems.



a) Königsberg b) Graph representation

Figure 2.1: A representation of the seven bridges of Königsberg. The problem consisted of finding a path crossing each bridge only once. Figure a) shows an schematic representation of the bridges and figure b) shows a graph representation in which each bridge counts as an edge between two nodes.

2.1 Graph

We need a representation to encode the problems we are going to address in this dissertation. A graph (Definition 1) is a model which allows us to associate an object with a node and to define a certain types of relationships by adding edges.

Definition 1 (Graph)**.** *A graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$ *is a representation in which* $\mathcal{V} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ *is the set of nodes and* $\mathcal{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K\}$ *the set of edges.*

An edge in a graph is a link between two nodes: $\mathbf{e}_k = \{\mathbf{x}_i, \mathbf{x}_j\}, 1 \leq i, j \leq N, k \leq K$. The nodes might represent entities such as cities, people, countries, web pages, etc. The edges describe relations between those entities, and those relations might be different in their nature, i.e. directed, undirected, weighted, etc. In the example of the Könisberg's bridges, there are several land regions surrounded by the Pregel river. The nodes

$\mathcal{V} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ are just an abstraction of those regions. Each bridge connects two land regions and can be represented by an edge $\mathbf{e}_k = \{\mathbf{x}_i, \mathbf{x}_j\}$ in a our graph.

A great advantage of such a model is the fact that many problems have already been formulated using graph theory. Thus, we can solve a problem by addressing its graph formulation instead. Examples of graph problems include the Travelling Salesman Problem [5], Minimum-Cut and Maximum-Flow [14], and many others.

Definition 2 (Simple Graph)**.** *A simple graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a graph with no parallel edges or self loops.*

There are many types of graphs, for instance, the graph used in the Euler problem contained parallel edges between nodes (as there are two bridges connecting the same island). Another type of graph is the one which does not allow multiple edges between nodes (Definition 2). They can also differ regarding the type of edges. A graph with unordered edges is called undirected graph, whereas when there is an order, the graph becomes directed (Definition 3).

Definition 3 (Directed Graph)**.** *A directed graph is a graph whose edges are an ordered pair of nodes implying in a direction of flow.*

Figure 2.2 displays a directed graph with contains 5 nodes and 9 edges. A graph also has the concept of degree, which is the number of neighbors connected to a node. This concept will be covered later in details when we talk about graph centralities (Section 2.3.1), but it is worth mentioning that in directed graphs, there is also the difference between in-degree (the number of edges incident in a node) and out-degree (the number of edges exiting a node).

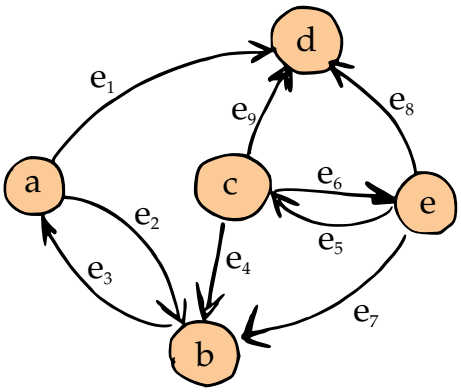


Figure 2.2: A directed graph. In this graph, information might not flow in all directions, one node can only reach another node in case there is a path of directed edges connecting the former to the latter. All nodes can reach node d either directly (a, c, e) or indirectly (b). Nevertheless, node d cannot reach any other node in the graph.

A graph containing directed edges does not allow the flow in both directions unless there are two edges of opposite directions connecting two nodes, such as the circumstance displayed in Fig. 2.2. as it happens between nodes a and b and between c and e.

Definition 4 (Weighted Graph)**.** *A weighted graph is a graph which associates a real number (weight) with each edge of the graph.*

A graph can also carry weights in the edges (Definition 4) and those weights might represent many things, such as the distance between cities as in the Travel Salesman Problem (TSP) or the capacity of pipes in a maximum flow problem. Weighted graphs are really important in computer science. Our graph in Chapter 6 contains weighted edges according to a distance function. The graph we modeled for segmentation tasks [32] associates a weight for each edge using a distance function representing the color similarity between regions. We are going to cover distances later in Section 2.2.2.

Graphs can also be created out of other graphs. We focus on two types of graphs, one is induced by nodes (Definition 5) and the other is induced by edges (Definition 6). Figure 2.3 shows two induced subgraphs of the graph in Figure 2.2. The first graph (Figure 2.3a) is a vertex-induced subgraph, which contains some of the nodes of the original graph. In the second example (Figure 2.3b), all nodes from the original graph are now present, but only a subset of its edges are available.

Definition 5 (Vertex-induced subgraph)**.** *Given a vector $S = \{0,1\}^N$ and a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, a node-induced subgraph $\mathcal{G}\langle S\rangle(\mathcal{V}\langle S\rangle,\mathcal{E}\langle S\rangle)$ is a graph induced by S in which a node $v_i \in S$ will also exist in $\mathcal{G}\langle S\rangle$ if and only if $S_i = 1$. In case v_i is also present in $\mathcal{V}_S$, so it will be all of its edges whose endpoints are also present in $\mathcal{V}_S$.*

Definition 6 (Edge-induced subgraph)**.** *Given a vector $U = \{0,1\}^M$ and a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, an edge-induced subgraph $\mathcal{G}[S](\mathcal{V},E[U])$ is a graph whose node set is the same as the original graph $\mathcal{V}$, but only a subset of edges $e \in \mathcal{E}$ will be available in $\mathcal{E}[U]$.*

Induced subgraphs will be particularly important for us in Chapter 6 in which we pose a combinatorial optimization problem in the pursuit of an edge-induced subgraph. This graph which is obtained by the optimization procedure should have the with maximum entropy in the degree distribution of the nodes and the minimum cost in the total weight of the edges.

2.1.1 The Minimum Spanning Tree

An important type of edge-induced subgraph is the MST. Considering a weighted graph such as the one in Figure 7.1a, one might be interested in finding a tree whose total edges' weight is minimum. The MST is an edge-induced subgraph since it contains all the nodes of the original graph but only a subset of its edges. There is an important information about such an induced graph, The MST is, in fact, a tree, meaning there are exactly $N-1$ edges, and those edges have the minimum total weight[1].

[1] Assuming there are no negative weighted edges.

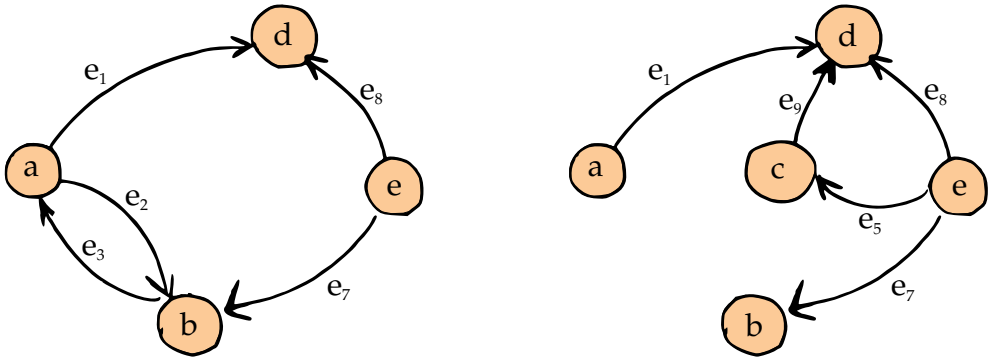


Figure 2.3: Induced subgraphs. Graph a) displays a vertex-induced subgraph and graph b) displays an edge-induced subgraph.

Definition 7 (Minimum Spanning Tree)**.** *A minimum spanning tree of an undirected graph is a tree connecting all nodes whose total weight is minimum.*

In Chapter 7, we propose a special type of MST, which has the minimum total weight, but also the maximum entropy in the degree distribution of the nodes. The classical algorithms to calculate the MST are the Prim [94] and Kruskal [65] algorithms. Both are greedy algorithms that compute the MST in polynomial time. The complexity depends on the actual data structure chosen for the graph (i.e. Adjacency Matrix, Heap, Fibonacci Heap, etc.).

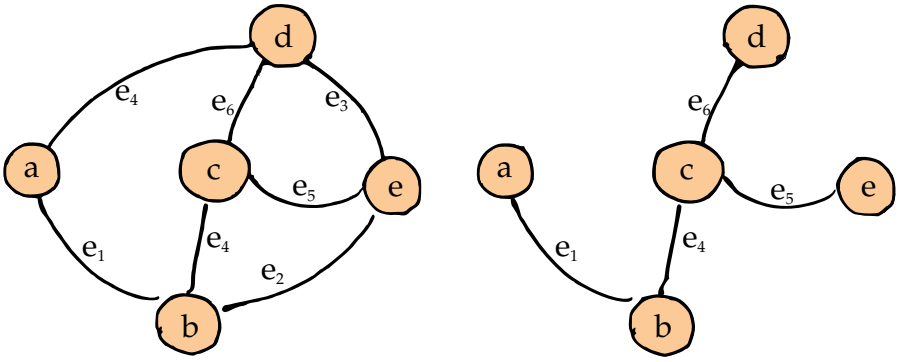


Figure 2.4: The Minimum Spanning Tree. Graph a) has $|V| = 5$ nodes and its MST has $|V| - 1 = 4$ edges, whose total weight is minimum.

2.1.2 Paths and Cycles

We faced a problem which uses paths and cycles back in the beginning of this chapter when we first discussed about the bridges of Königsberg. We mentioned we needed to find an Eulerian cycle, which was the problem of finding a path in the graph which would pass through each bridge only once and *return* to the original starting point. We will provide a more formal definition of both concepts.

Definition 8 (Path)**.** *A path in graph is an ordered sequence of edges connecting a sequence of distinct vertices.*

By simply having a sequence of distinct nodes, we call it a path. Nevertheless, if one has the constrain that we should start and end in the same node, we call this a cycle.

Definition 9 (Cycle)**.** *A cycle in graph is a path in which the starting and ending nodes are the same.*

In the graph of Figure 7.1a, there is a path $p_1(\mathbf{a}, \mathbf{d}) = \{\mathbf{e}_1, \mathbf{e}_4, \mathbf{e}_6\}$ between points $\mathbf{a}$ and $\mathbf{d}$ and another path $p_2(\mathbf{a}, \mathbf{d}) = \{\mathbf{e}_4\}$. One could compare paths by evaluating their lengths or the sum of their weights.

Definition 10 (Length of a Path)**.** *The length of a path is the sum of the weights of the edges in the path:* $|p| = \sum_{e \in p} w(e)$.

In an unweighted graph, the length of a path is equal to the number of edges considered in the path, for instance, in the previous example of paths $p_1(\mathbf{a}, \mathbf{d}) = \{\mathbf{e}_1, \mathbf{e}_4, \mathbf{e}_6\}$ and $p_2(\mathbf{a}, \mathbf{d}) = \{\mathbf{e}_4\}$, the lengths are $|p_1| = 3$ and $|p_2| = 1$. One could state that p_1 is a shorter path than p_2. When weights are now assigned to the edges, this might not be true since the weights in p_2 might be higher than in p_1. We will discuss more about distances in Section 2.2.

In the paper of Euler[38] about the bridges of Königsberg, he proves the so-called Handshake Lemma (Definition 11). This lemma states that an undirected graph has an even number of vertices with odd degree. Imagine you find yourself in a group of people which are being introduced to each other, an even number of people must shake the hands of an odd number of people. We will take advantage of the handshake lemma later in Chapter 6 to prove one of our theorems (Theorem 4).

Definition 11 (Handshake Lemma)**.** *The handshake lemma establishes the connection between the node degrees in a graph as* $\sum_{\mathbf{x} \in \mathcal{X}} deg(\mathbf{x}) = 2|\mathcal{E}|$.

2.1.3 Representation

A graph can be represented in a computer by using different data structures. We will exemplify it with three possibilities: the (i) adjacency matrix, the incidence matrix (ii), and the (iii) incidence list. All representations have their own advantages and

disadvantages. Figure 2.5 shows the graph of the graph Figure 2.2 using the different representations.

An incidence list (Figure 2.5a) only stores the nodes which are connected to a certain node and therefore, it is a very compact and good representation for sparse graphs. It relies on a data structure containing the value of the node (e.g. an integer) and a pointer to the next element, we shall refer to this structure as a *node*. The space complexity of such representation is $O(N + |\mathcal{E}|)$ since we only allocate $N + |\mathcal{E}|$ nodes. There is a vector indexing all nodes and therefore the initial space for the nodes is $O(N)$, and whenever there is an edge connecting two nodes, we allocate a new *node* and append it to the end of the list. As there are $|\mathcal{E}|$ edges, the total complexity is $O(N + |\mathcal{E}|)$. For instance, the edge $\mathbf{e}_2$ of the graph in Figure 2.2 starts from node **a** to node **b**.

In an incidence list representation, we would access node **a** with time $O(1)$, create a new node and append it to the end of the list, whose time complexity is bounded by $O(deg(\mathbf{a}))$. A *node* is added to the list of a node only if there is an edge connecting that node to the former. Nevertheless, when there is a considerable number of edges, the access to an element will be slow since all the information we have is accessible via the head of the list, therefore, we will always iterate through the whole list until the last element is accessed. Whenever a node has no more outgoing edges, the pointer of the end of the list will contain the *NULL* value indicating the list has no more elements.

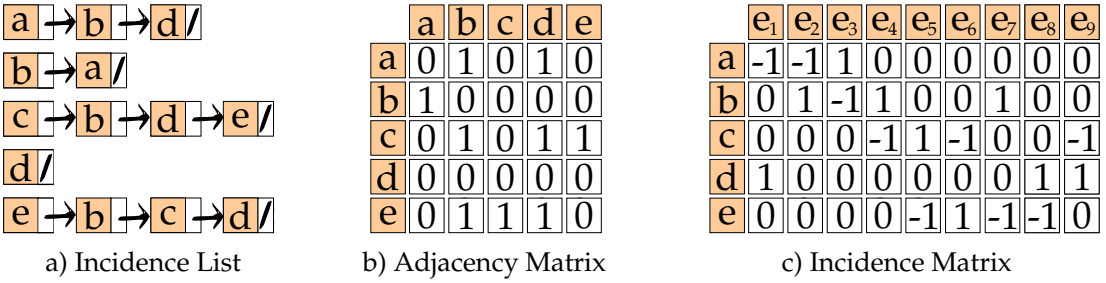


Figure 2.5: Data structures for representing the graph of Figure 2.2. An incidence list (a), an adjacency matrix (b), and an incidence matrix.

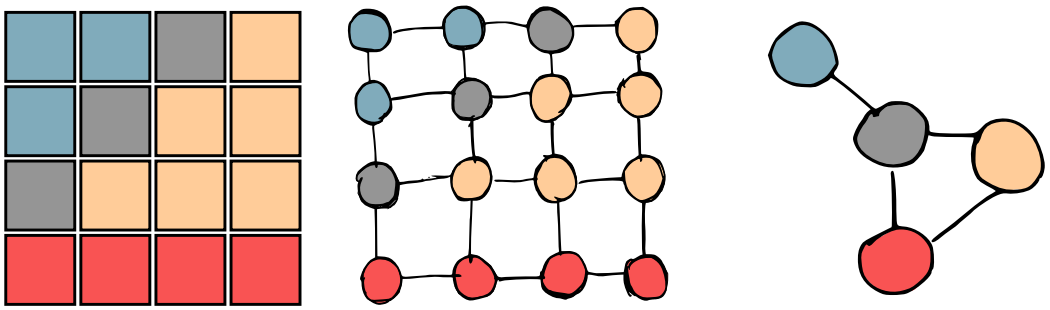
A graph can have a matrix representation, such as the adjacency matrix **A** (Figure 2.5b), in which the space complexity is bounded by $O(N^2)$. We represent the existence of an edge by the element 1 in the matrix (or by the weight of that edge). For instance, if the element $\mathbf{A}[b][a] = 1$, it means that node **b** has an edge to node **a**. The graph of Figure 2.2 is directed, but for undirected graphs, this matrix is symmetric. The advantage of an adjacency matrix is the indexed access, which makes it fast ($O(1)$) to find out if two nodes are connected. In this example, the matrix was filled with ones and zeros. Nevertheless, in case there are weighted edges, this information would not increase the space in memory. On the other hand, when the graph is sparse, i.e. $|\mathcal{E}|$ is small ($|\mathcal{E}| << N$), there would be space not being used in the memory.

A third type of graph representation is the incidence matrix **M** (Figure 2.5c). The

incidence matrix is a $N \times |\mathcal{E}|$ matrix, in which there is one column for each edge. The incidence matrix allows multiple edges between nodes, which could not described by the adjacency matrix. In this representation, we are flagging the nodes which participate in each edge. As there are always two nodes per edge (source and target), this is a very computationally expensive data structure. For instance, we could define a non-zero real number as the weight of the edge and its sign would represent if this is a source node or a target node. In this, example, $M[a][\mathbf{e}_1] = -1$ indicating that there is an outgoing edge from node **a**. The target of such an edge would be in the same column with the opposite sign, i.e. $M[d][\mathbf{e}_1] = 1$.

2.1.4 Images as Graphs

We can also represent images using graphs in many different ways. One possibility is to assume, for instance, that each pixel corresponds to a node in the graph. The edges could be added according to a 4-connected neighborhood as in a Manhattan distance (Section 2.2.2) or 8-connected neighborhood.



a) Image b) A node per pixel c) A node per region

Figure 2.6: Representing images as graphs. Figure a) shows a synthetic image. A graph representation using a node per pixel (b) and a node per region (c).

Max-flow algorithms [14] in CV have become a powerful choice for energy minimization. They often start by modelling each pixel of the image as node in the graph in order to search for the minimum cut. When each pixel in the image is used as a node, the number of nodes in the graph will be $N = w \times h$, for w, h the width and height of the image respectively. This can be, for some applications, a considerably high number. Therefore, there are other possibilities to represent the image as graph, including the super pixels representation [2], or watershed based algorithms which are solutions used often to decrease the number of the nodes in the graph. The choice of contraction[2] varies according to the algorithm.

[2]Merging two or more nodes

2.2 Metric Spaces

Since we have spoken about many properties of graphs, it is now time to move towards geometry. On a daily basis, we deal with the concept of distances and paths in a very intuitive fashion, such as how far our house is from work, or how close two objects are on an Euclidean space. The concept of distance is intuitive and we can safely use it without the need to formally define it. In our study of centrality for vision, it becomes essential to understand some basic concepts related to geometry and topology. Many centrality measures establish the importance of a node in a graph based on the distance to another node or to a group of nodes in the graph. The Euclidean distance (Section 2.2.2) is, for instance, just one instantiation of the more general concept called distance.

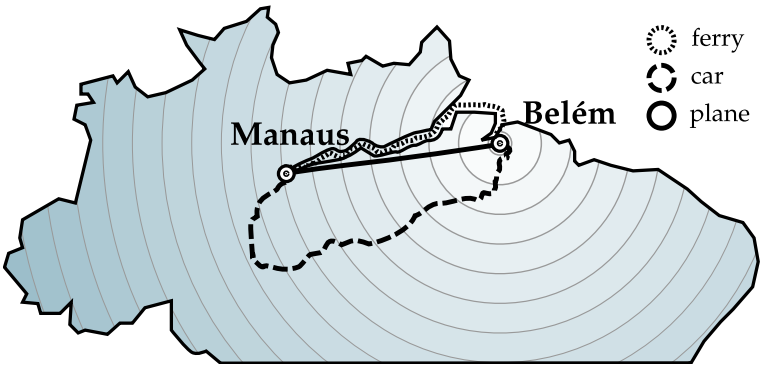


Figure 2.7: The distance between two cities in Brazil. Paths taken can vary between 2 hours and five days.

Figure 2.7 shows, intuitively, how the concept of distance can vary. Although one can measure the Euclidean distance between cities, it does not mean that one can take that path. For instance, by traveling between the two Brazilian cities Belém and Manaus, we can take approximately 2 hours in a non-stop flight. Traveling by car without stopping would mean to travel at least for 2 days, and there is a part of the road that needs to be crossed by ship. If, for instance, we decide to perform the whole trip by ferry, it could take between 5 and 7 days. In real life, there are many different ways to measure distance between two points, specially because we need to account for obstacles, traffic jam, and so forth. The abstraction of distance is extremely important since most of problems in computer vision encode some type of metric or distance function between points, colors, or shapes. In this section, we will spend some time discussing about metrics and its variations.

Definition 12 (Metric)**.** *Given a set* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$*, a metric is a function* $\mathrm{d} : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$ *if it fulfills the metric axioms:*

- **Non-negativity**: $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) \geq 0$, for any two points $\mathbf{x}_i$, $\mathbf{x}_j \in \mathbf{X}$. This axiom states

that there are no negative distances. For any two points, the distance will be at least zero, e.g. when two points have are located in the same place.

- **Coincidence**: $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) = 0 \iff \mathbf{x}_i = \mathbf{x}_j$. This axiom states the distance between two points $\mathbf{x}_i$, $\mathbf{x}_j \in \mathbf{X}$ is zero if and only if they are the same.

- **Symmetry**: $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{d}(\mathbf{x}_j, \mathbf{x}_i)$. This axiom states that the distance is symmetric, the distance from $\mathbf{x}_i$ to $\mathbf{x}_j$ is the same as the distance from $\mathbf{x}_j$ to $\mathbf{x}_i$. Notice that this might not be true in many cases (e.g. a directed graph), therefore later we will specify the extensions of the metric concept which allows relaxation of those axioms.

- **Triangle Inequality**: $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) \leq \mathrm{d}(\mathbf{x}_i, \mathbf{x}_k) + \mathrm{d}(\mathbf{x}_k, \mathbf{x}_j)$ for any three points $\mathbf{x}_i$, $\mathbf{x}_k$, $\mathbf{x}_j \in \mathbf{X}$. Consider the non-degenerate triangle of Figure 2.8a, as the sum of all angles in triangle add up to π, we know that $\gamma < \pi$. Now we stretch γ in such a way that all points are colinear (Figure 2.8b). We can see that, when the triangle is degenerate, $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{d}(\mathbf{x}_i, \mathbf{x}_k) + \mathrm{d}(\mathbf{x}_k, \mathbf{x}_j)$. Therefore, for any $\gamma < \pi$, the triangle inequality will hold.

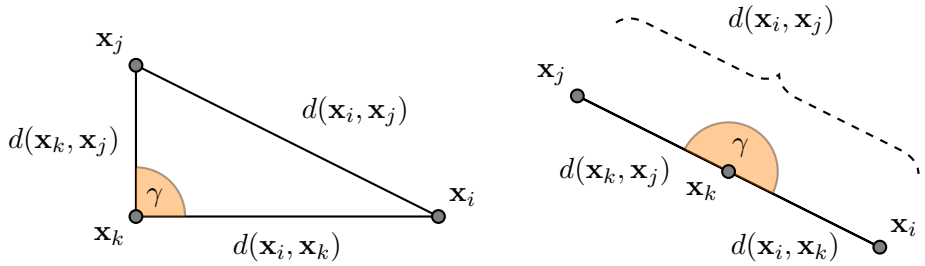


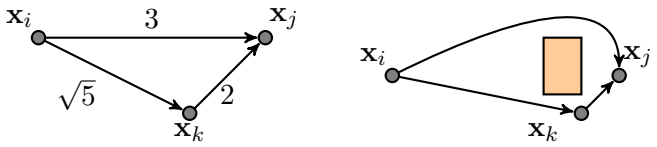
Figure 2.8: Triangle Inequality. When the $\gamma = \pi$, (i.e. there is a degenerate triangle), then the inequality turns into an equality: $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{d}(\mathbf{x}_i, \mathbf{x}_k) + \mathrm{d}(\mathbf{x}_k, \mathbf{x}_j)$. The inequality will hold for any $\gamma < 180$.

2.2.1 Generalized Metric

Some of the aforementioned axioms might not be fulfilled, therefore, some metric variants allow one or more axioms to be infringed.

Definition 13 (Quasimetric)**.** *A quasimetric is a function which fulfills all metric axioms but the symmetry.*

The quasimetric is an important metric relaxation since it is commonly seeing on the daily life, for instance when traveling in one way streets or in a directed graph formulation. In Figure 2.9, the distance $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) = 3$ while $\mathrm{d}(\mathbf{x}_j, \mathbf{x}_i) = \infty$ since $\mathbf{x}_i$ can not be reached.



(a) Quasimetric (b) Semimetric

Figure 2.9: Relaxation of the metric axioms.

Definition 14 (Semimetric)**.** *A semimetric is a function fulfilling all metric axioms but the triangle inequality.*

In a real life, it is not hard to understand that when we calculate the distance between two places, either by estimating with time via Global Positioning System (GPS), or calculating the kilometers until reaching a destination, we are not really calculating the Euclidean distance (Section 2.2.2). We are not travelling to the place in the straight line from our origin. We need to account for the *feasible path* in order to reach that destination. Thus, as our world has obstacles, the triangle inequality might not always be guaranteed. There is an active research area in mobile robotics regarding path planning which takes that into account. Figure 2.9 shows an example of semimetric in which $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) > \mathrm{d}(\mathbf{x}_i, \mathbf{x}_k) + \mathrm{d}(\mathbf{x}_k, \mathbf{x}_j)$.

Definition 15 (Pseudometric)**.** *A pseudometric is a generalized metric that does not fulfill the coincidence axiom.*

Two points might be distinct even though they have the same location in a pseudometric: $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_k) = 0$ and $\mathbf{x}_i \neq \mathbf{x}_k$.

2.2.2 Distances and Norms

Metric spaces and distances define the ground basis for understanding centralities. The first type of metric we will discuss is the Euclidean, which is the natural concept of distance. We will use this metric later in Chapter 6 when we define the weights of our graph as the Euclidean distance which will be used for the DP optimization.

Definition 16 (Euclidean Distance)**.** *The Euclidean distance* ($\mathrm{d_e}$) *between two points* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as the length of the straight line segment between* $\mathbf{x}$ *and* $\mathbf{y}$ *in* $\mathbb{R}^n$ *(Eq. 2.1).*

$$\mathrm{d_e}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{y}_i)^2}. \tag{2.1}$$

Eq. 2.1 is also known as the Pythagorean formula, for $\mathbf{x}$ and $\mathbf{y} \in \mathbb{R}^n$. The Euclidean distance describes in a really concise way how far two points are by estimating a straight line between the two points. It is a metric as it satisfies all metric axioms. Nevertheless,

it might not be always a feasible way to travel between two points, as when one is driving a car, one needs to make turns and it is, therefore, not possible to travel through a straight line. There is another type of distance called Manhattan which is a metric used when one walks through blocks.

Definition 17 (Manhattan Distance)**.** *The Manhattan distance* ($\mathrm{d_m}$) *is distance computed as the absolute differences based on a rectangular grid (Eq 2.2).*

$$\mathrm{d_m}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i|. \tag{2.2}$$

This metric is also called 4-connected neighborhood in an image, since a pixel is connected to $(x \pm 1, y)$ or $(x, y \pm 1)$.

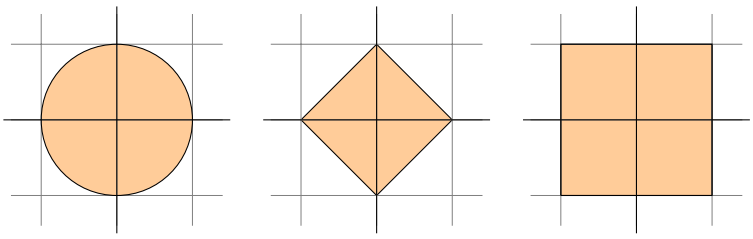
Definition 18 (Chevycheb distance)**.** *The Chevycheb distance* ($\mathrm{d_c}$) *is a distance computed as the sum absolute of differences based on a rectangular grid (Eq 2.3).*

$$\mathrm{d_c}(\mathbf{x}, \mathbf{y}) = \max_i(|\mathbf{x}_i - \mathbf{y}_i|). \tag{2.3}$$

In an image, this metric is also called 8-connected neighborhood, a pixel is now connected to $(x \pm 1, y \pm 1)$. Another important related concept is the norm, which is a function assigning a length to a vector in a vector space. For $p \geq 1$, a L_p-norm is defined as:

$$\|x\|_p = (\mathbf{x}_1^p + \mathbf{x}_2^p + \cdots + x_n^p)^{\frac{1}{p}} \tag{2.4}$$

Figure 2.10 shows the metric unit balls for the Euclidean, Manhattan, and Chevycheb distances. The norm mapped for those distances are the L_2, L_1, and L_∞ respectively.



(a) L_2 Norm (b) L_1 Norm (c) L_∞ Norm

Figure 2.10: Metric Balls.

2.2.3 Distances in Graphs

In graphs, we are often concerned about the shortest paths between nodes. In fact, many centrality values described in the next section rely on the shortest paths, and lengths of those shortest path. Consider the graph on Figure 2.11. Each edge has the unit weight.

Definition 19 (Geodesic)**.** *A geodesic between two nodes* $\mathbf{x}$ *and* $\mathbf{y} \in \mathcal{X}$ *is the path with shortest distance.*

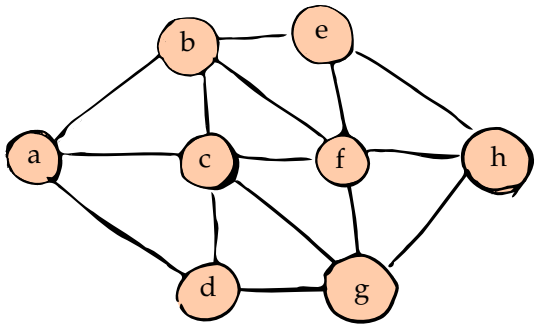


Figure 2.11: Geodesics in a Graph. The geodesic distance between nodes **a** and **h** is 3.

The geodesic between nodes **a** and **h** is a path whose distance is the shortest one. There are many possible geodesics between those two points, for instance $g_1(\mathbf{a}, \mathbf{h}) = \{\mathbf{a}, \mathbf{c}, \mathbf{f}, \mathbf{h}\}$, $g_2(\mathbf{a}, \mathbf{h}) = \{\mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{h}\}$, $g_3(\mathbf{a}, \mathbf{h}) = \{\mathbf{a}, \mathbf{d}, \mathbf{g}, \mathbf{h}\}$ are all geodesics between **a** and **h**.

Considering a path $\{\mathbf{x}_k, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_k$ is the source and $\mathbf{x}_n$ target nodes, the geodesic could be formulated in Eq. 2.5:

$$g(\mathbf{x}_k, \mathbf{x}_n) = \text{minimize} \sum_{i=k}^{n-1} w(\mathbf{x}_i, \mathbf{x}_{i+1}) \tag{2.5}$$

where $w(\mathbf{x}_i, \mathbf{x}_{i+1})$ is the weight of the edge between the two points $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ in the shortest path to be optimized.

Definition 20 (Geodesic Distance)**.** *The geodesic distance is the length of the geodesic.*

The geodesic distance between points **a** and **h** is $|\{\mathbf{a} \rightarrow \mathbf{d} \rightarrow \mathbf{g} \rightarrow \mathbf{h}\}| = 3$. There are many possible geodesics in a graph, i.e. paths with the same length, but there is only one geodesic distance.

Definition 21 (Eccentricity of a node)**.** *The eccentricity of a node is the greatest geodesic distance between this node and any other node in the graph.*

The shortest path between point **a** and any other node is 1 (e.g. $g(\mathbf{a}, \mathbf{b})$,$g(\mathbf{a}, \mathbf{c})$,$g(\mathbf{a}, \mathbf{d})$). The eccentricity of a node considers the shortest path between that node and all the

other nodes. The diameter of the graph is the maximum eccentricity of any node in the graph.

$$\text{ecc}(\mathbf{x}_n) = \underset{\mathbf{y}_n}{\text{maximize}}\ g(\mathbf{x}_n, \mathbf{y}_n),\ \forall\ \mathbf{y}_n \in \mathcal{X} \setminus \{\mathbf{x}_n\}\,. \tag{2.6}$$

Definition 22 (Eccentricity of a graph)**.** *The eccentricity is the greatest geodesic distance in a graph.*

Finally, given the eccentricity of all points in the graph, we can define the eccentricity of the graph as the maximum of all nodes.

$$\text{ecc}(\mathcal{G}) = \max\{ecc(\mathbf{x}_i);\ \forall\ \mathbf{x}_i \in \mathcal{X}\}\,. \tag{2.7}$$

Mukherjee et al. [83] used the eccentricity of graph as a measure of importance for modeling human action recognition. This concept could be considered as a centrality and it was also used by Kropatsch et al. [64] in the analysis and matching of binary shapes.

2.3 Centrality

We will start discussing now about graph centrality, which is the main focus of this work. The idea of centrality was born in the Social Sciences when Bavelas [9, 10] analyzed group processes in human communication. It was later when Freeman [43] examined the studies of different researchers that the concept of centrality was in fact clarified. Freeman [43] illustrates some distinct conceptual properties using a star graph[3] to explain the notion behind centrality. For instance, the central node of such graph not only possesses the highest *degree* among the others, but it also lies on the geodesic path *between* any two nodes[4]. Given the fact that it has the minimum distance to any node, it is considered the *closest* to them. Those ideas compose the core of the degree, betweenness, and closeness centralities.

Definition 23 (Centrality)**.** *A centrality is a measure of importance of a node within a network.*

Concepts initially applied in social networks are being brought into other fields that are not necessarily human-related such as virtual network [125], power-grid [87], image saliency [91], resistor network [62, 17], and so forth. Nevertheless, we did not find many approaches in computer vision taking advantage of centralities. In this section, we review four centrality measurements: Degree, Betweenness, Closeness, and Eigenvector. Some of those centralities have been applied in CV before, and we will be discussing those cases accordingly.

[3] A star graph $\mathbf{S}_t$ is a tree of depth one where the central node is the root of the tree.

[4] The geodesic of any pair of nodes $\mathbf{s}_i, \mathbf{s}_k \in \mathbf{S}_t$ is the shortest path between them. In a star graph, the shortest path between two non adjacent nodes include the central node.

2.3.1 The Degree Centrality

The node degree is one of the most well known centrality measurements, due to its simplicity but also due to its widespread usage in many graph theoretical problems. The degree of a point $\mathrm{d}(\mathbf{x}_n)$ measures how many nodes are connected to $\mathbf{x}_n$, in other words, how many edges are incident to this point. This concept is further broaden into *in-degree* and *out-degree* for a directed graph related to the edges that arrive in $\mathbf{x}_n$ and edges that exit from it.

Definition 24 (Degree Centrality)**.** *The degree is a measure of centrality of a node which associates importance with the number of nodes connected to it.*

Let $\mathbf{A}$ be the adjacency matrix of a graph $\mathcal{X}$. The degree $\mathbf{x}_n$ can be computed as the sum of elements in the row (or column) of $\mathbf{A}$:

$$\mathrm{d}(\mathbf{x}_n) = \sum_{k=1}^{N} \mathbf{A}_{n,k} = \sum_{k=1}^{N} \mathbf{A}_{k,n}. \tag{2.8}$$

Although we can easily quantify the degree of a node, its importance to the rest of the graph is not really clear as it depends on the average degree of the graph. For instance, a degree 8 might be considered high in a graph whose average degree is 2, but it is low in a graph whose average degree is 30. Hence, to overcome this, one could simply normalize $\mathrm{d}(\mathbf{x}_n)$:

$$\hat{\mathrm{d}}(\mathbf{x}_n) = \frac{\mathrm{d}(\mathbf{x}_n)}{\max\{\mathrm{d}(\mathbf{x}_k) : 1 \leq k \leq N\}}, \tag{2.9}$$

where the maximum degree value of the graph would now be equal to 1. Alternatively, Freeman [43] suggests that this normalization should occur with respect to the highest difference between centrality values within the graph.

In CV, the node degree has been explored in several manners such as the abstraction of rectangular lattice defining a 4-neighborhood until more specialized usages such as in segmentation. However, the work of Pal et al. [91] treated the degree of a node as centrality, i.e. measuring the importance of the nodes in a graph. In their work, they aimed at extracting visual saliency of images. They were motivated by studies reporting that objects whose locations differ significantly from their surrounding are considered as salient by the means of drawing attention. Thus, they decided to design networks that model such saliency (called Visual Saliency Networks) by encoding salient regions as central nodes using the degree centrality.

2.3.2 The Betweenness Centrality

Imagine the situation in which you reside on a city **a** and you need to drive to a city **c**. On the road, you must cross city **b** since **a** and **c** are not directly connected. Therefore, one could say that city **b** has some sort of control over the path between **a** and **c**. For instance, if **b** decides to take tolls on those roads or if it blocks the roads, it would limit

the traffic between cities. This situation illustrates the concept behind the betweenness centrality, where the communication between two non-adjacent nodes depends totally on the path between them [126]. The main idea of betweenness centrality defined by Freeman [42] is that vertices that lie on the geodesic path of many other vertices will possess great control over the information flow, due to the fact that they reside *between* others.

The betweenness centrality $b(\mathbf{x}_n)$ of a node $\mathbf{x}_n$ is calculated as follows:

$$b(\mathbf{x}_n) = \sum_{s \neq n \neq t \in V} \frac{\gamma_n(\mathbf{x}_s, \mathbf{x}_t)}{\gamma(\mathbf{x}_s, \mathbf{x}_t)}, \tag{2.10}$$

where $\gamma(\mathbf{x}_s, \mathbf{x}_t)$ is the number of geodesic paths between nodes $\mathbf{x}_s$ and $\mathbf{x}_t$ and $\gamma_n(\mathbf{x}_s, \mathbf{x}_t)$ is the number of those geodesics passing through $\mathbf{x}_n$. The computation of the betweenness centrality requires the calculation of the geodesic paths to all vertices in the graph, this could be achieved by the Floyd-Warshall algorithm. It is important to mention that there is another measure of betweenness centrality based on random walks defined by Newman [88].

Definition 25 (Betweenness Centrality)**.** *Betweenness is a measure of centrality which associates importance based on how often a node is found on the geodesic path of other nodes in the graph.*

Mantrach et al. [79] proposed a covariance measure called Sum-over-Path (SoP) which measures the similarity between nodes in a graph. Two nodes are considered highly correlated if they co-occur frequently on the same (shortest) paths between nodes in the graph. Thus, the SoP estimates the expected number of times a node occurs on a path and they compare their betweenness measure with both Freeman's and Newman's. A SoP formulation for string edit distance has been proposed by Garcia-Diez et al. [45] and the Sum-over-Forest (SoF) index was later proposed by Senelle et al. [110] with the same inspiration that large forests would occur with lower probability and short forests would occur with high probability in a graph.

In CV, Li et al. [71] used betweenness centrality for scene categorization. They build a social network of images of one class (e.g. kitchen) and calculate the betweenness of a certain image of this class with respect to the network. In this way, this is not an image-to-image measure, but an image-to-class measure estimating the overall connectivity to the class to which this image belongs. Their results were superior to the methods in the state of the art on three datasets.

Mukherjee et al. [83] present an application of centrality in human action recognition. They employ centrality to create a compact codebook out of a large vocabulary of poses (bag-of-word approach). Cukierski and Foran [26] use centrality to solve an open problem in the ISOMAP algorithm [118], which is a non-linear dimensionality reduction method. The ISOMAP algorithm computes geodesic distances between data points with the help of a neighborhood graph. Unfortunately, these graphs sometimes contain unwanted edges, which connect disparate regions of one or more manifolds and this leads to a

distortion of the calculated geodesic distance matrix. This problem is where [26] propose an edge-removal method based on graph betweenness centrality, which can robustly identify manifold-shorting edges. Correa and Ma [22] use derivatives of centrality in the visualization of social networks. The derivative of centrality informs how much a given node influences the importance of another node, even if they are not directly connected. They found out that derivatives of centrality are tool for analyzing social networks, they help to simplify the layout of complex networks and to visually measure the centralization degree of a network. Furthermore, they provide information to estimate other metrics like structural balance and uncertainty.

2.3.3 The Closeness Centrality

The closeness centrality is a measure that evaluates how close a certain node is to all the other nodes in the graph. It was initially proposed in [10, 55, 11, 103]. As defined by Sabidussi [103], the closeness centrality $c(\mathbf{x}_n)$ of a node $\mathbf{x}_n$ can be computed as follows:

$$c(\mathbf{x}_n) = \sum_{\mathbf{x}_k \in \mathcal{X} \setminus \{\mathbf{x}_n\}} \frac{1}{\mathrm{d}(\mathbf{x}_n, \mathbf{x}_k)}, \tag{2.11}$$

where $\mathrm{d}(\mathbf{x}_n, \mathbf{x}_k)$ is the shortest distance between vertices $\mathbf{x}_n$ and $\mathbf{x}_k$. Roughly speaking, when computing the distance between two nodes, we are able to define how far those nodes are. Therefore, by inverting it, we intuitively obtain the measure of how close they are.

Definition 26 (The Closeness Centrality)**.** *Closeness is a measure of centrality which associates importance based on how close a certain node is to all the other nodes in the network.*

When analyzing shapes, de Sousa et al. [31] described a shape by the histogram of the centrality values using an 8-connected neighborhood graph. A desired behaviour would expect that similar shapes should have similar histograms with respect to their centrality values. They have evaluated the robustness of each individual centrality by randomly removing nodes of the graph and calculating the new histogram after this topological change in the graph. The results indicated that the closeness centrality was the most robust centrality against that type of noise for the shape matching task.

2.3.4 The Eigenvector Centrality

The eigenvector centrality of graph $\mathcal{X}$ corresponds to the eigenvector $\mathbf{w} = (w_1, w_2, \ldots, w_N)^T$ associated with the highest eigenvalue λ of the adjacency matrix $\mathbf{A}$ [90]. The study of eigenvalues and eigenvectors of the adjacency matrix and Laplacian matrix of a graph has received a vast contribution from many authors [20, 82, 35] in mathematics and spectral graph theory. In CV, it has important results in image segmentation such as in the work of graph-cuts by Greig et al. [51] and other segmentation methods exploring the Laplacian matrix [95].

Definition 27 (Eigenvector Centrality)**.** *The eigenvector centrality is a measure of importance in a graph based on the first eigenvector of the graph adjacency matrix.*

The idea of *eigenvectors* of some matrix describing the relationship between points was already exploited in point-set registration. In fact, one of the earlier works employing eigenvector analysis (although not in a graph) was conducted by Scott and Higgins [108] and later refined by Shapiro and Brady [111]. Both approaches apply a Singular Value Decomposition (SVD) of a matrix $\mathbf{H}_{i,j} = exp(-r_{ij}^2/2\sigma^2)$ which uses the squared distance (r_{ij}^2) between nodes i and j to define the proximity of nodes. Shapiro and Brady [111] handled some limitations of the original work such as handling large rotations and translations in the image plane. The eigenvector analysis employed by them occurred over the matrix $\mathbf{H}$ (proximity) while the so called eigenvector centrality occurs directly on the adjacency matrix of the graph. However, for point-set registration, it is not uncommon to define a proximity matrix between points. The recent work of Zhou and De la Torre [131, 130] (described in Section 3.6) factorizes a proximity matrix for the graph matching problem.

Leordeanu and Hebert [69] propose a solution for the registration problem by exploring the principal eigenvector of an association matrix $\mathbf{M}$. The interpretation of the eigenvector by the authors relates to the confidence of a certain assignment. For each correspondence element $a = (i, i') \in \mathbf{M}$, the eigenvector $x^*(a)$ stands to the confidence of a.

2.3.5 Example

We shall demonstrate all those measures using a more concrete application. Observe the graph of Figure 2.12. We have color coded the nodes according to the degree, node **a** is the lowest degree node (1) and **b** is the highest one (4). Although it is intuitive to visualize the degree, the other centralities are not so trivial to reason. Table 2.1 shows the four centrality measures we have discussed and their values for each node.

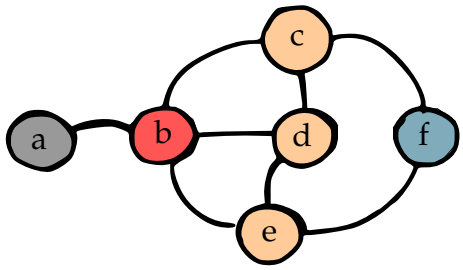


Figure 2.12: A simple graph color code according to the degree distritbution

When looking at Table 2.1, we notice some interesting facts about the importance of the nodes in our graph. The degree is a discrete measure, we have a very clear distinction of the nodes according to their degree, but when looking at the closeness centrality, we see that nodes a and e have similar values, they are the nodes which are most distance

from all the other nodes. When looking at the distribution, it is a more smooth measure than the degree.

Centrality	**a**	**b**	**c**	**d**	**e**	**f**
Degree	1	4	3	3	3	2
Closeness	0.50	0.83	0.71	0.71	0.71	0.56
Betweenness	0.00	4.33	1.50	0.33	1.50	0.33
Eigenvector	0.34	1.00	0.85	0.91	0.85	0.57

Table 2.1: Centrality values obtained for graph of Figure 2.12

The betweenness centrality decided to assign zero to node **a** since it has no control over the information flow of other vertices. Node **b** becomes the most important node in the graph, since any information who wishes to reach **a** needs to go through **b**. Therefore, the importance of **b** according to betweeness is much higher than all the other nodes. Eigenvector centrality also considered **b** as the most important node and **a** as the least important one.

2.4 Conclusions

There are still many other meaningful properties of graph which we did not cover in this chapter, such as the Laplacian matrix which is commonly defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix with the degree of the nodes and $\mathbf{A}$ is the adjacency matrix. This Laplacian matrix has important results in spectral graph theory and it is closely related to the commute time [104]. In fact, it can be computed using the pseudoinverse of the Laplacian matrix of the graph. The commute time is a distance measure between nodes in a graph using the idea of average number of steps a random walker would take to reach a node **j** starting from a node **i** and to return back to the origin. It has a relation to the resistance distance proposed by Klein and Randić [62] in their study of resistive electrical network. This concept has been explored in computer vision by Qiu and Hancock [95, 96] on the problems of image segmentation and clustering. Recently, Cortés et al. [24] proposed the usage of centralities on the problem of Graph-Edit Distance.

There has been some work on Fuzzy Social Network Analysis (FSNA) [58] and many of those centralities (degree, betweenness, and closeness) were also reformulated as Fuzzy centralities [57, 74]. We could also evaluate how those centrality values behave in CV. There is still many vision applications which could take advantage of centralities.

CHAPTER 3

Correspondence and Registration

> "I have six locks on my door all in a row. When I go out, I lock every other one. I figure no matter how long somebody stands there picking the locks, they are always locking three."
>
> – Elayne Boosler

Imagine you are in a dark room with several keys in your hand and there is a locked door which leads to the exit of that place. You are desperate to go out and you need to find the correct key which opens the door. When you try the first key, it does not enter the lock hole. The second key gives you hope since it partially enters the lock, but the door knob does not turn. The problem you are facing is called matching, in which given $|\mathbf{K}|$ keys and a lock $\mathbf{l}$, you are interested in finding a key $\mathbf{k} \in \mathbf{K}$, such that the distance between the silhouette of the key and the lock $\mathrm{d}(\mathbf{k}, \mathbf{l}) = 0$. We could also call this problem registration, since you are not only interested in finding the right key, but you are aligning both objects by applying a rigid transformation to the key until you can perfectly align each ridge. A partial match might also occur due to similarities, as occurred with the second key. This problem could also be thought as belonging to the Special Orthogonal Group SO(3), as the key is an object of the $\mathbb{R}^3$ and it cannot be bent without breaking since it is rigid. Therefore, given the correct correspondence (the right key), we only need to find a rotation and translation to register the key and the lock.

Definition 28 (Matching)**.** *Given two point-sets* $\mathbf{X}$ *and* $\mathbf{Y}$*, a point matching consists of finding a bijection* $\mathbf{x} \leftrightarrow \mathbf{y}$*, for all* $\mathbf{x} \in \mathbf{X}, \mathbf{y} \in \mathbf{Y}$*.*

Definition 29 (Registration)**.** *Registration is the problem of finding a spatial transformation which best aligns two point-sets.*

In this chapter, we will overview the problem of correspondence (as in a graph theory framework) and the registration problem when assuming that the points belong to a

certain space (e.g. Euclidean space). We will decompose it into several parts given the nature of the transformation (rigid, similarity, affine, etc.). Finally, we will review the main algorithms used in the literature to deal with registration problems. Chapter 5 aims at integrating centralities into the CPD algorithm reviewed later in this chapter. Chapter 6 proposes a centrality-oriented approach focusing solely on the correspondence problem between point sets.

3.1 Transformations

Given two point-sets $\mathcal{X}$ and $\mathcal{Y}$, our task is to retrieve the transformation that best maps $\mathcal{Y}$ onto $\mathcal{X}$ and to estimate the correspondence between points in both sets.

There are several types of transformations that will appear throughout this dissertation and each transformation has special properties which are relevant for us. Some of them are linear transformations (rigid, similarity, affine) and some are non-linear (tapering, bending, twisting). We will focus on linear transformations through this dissertation.

3.1.1 Rigid Transformation

The simplest type of transformation between two object is the rigid one:

Definition 30 (Rigid Transformation)**.** *A rigid transformation between two points* $\mathbf{x}$ *and* $\mathbf{y}$ *is a transformation which can be described by a rotation matrix* $\mathbf{R}$ *and a translation vector* $\mathbf{t}$*, such that* $\mathbf{y} = \mathbf{R}\mathbf{x} + \mathbf{t}$*.*

All rigid transformations can be described a rotation $\mathbf{R} \in SO(3)$ [1] and a translation $\mathbf{t}$, in which the three angles α, β, γ consist of the rotation around the axes. A rotation matrix is defined as $\mathbf{R} = \mathbf{R}_\alpha \mathbf{R}_\beta \mathbf{R}_\gamma$:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\ \alpha & -sin\ \alpha \\ 0 & sin\ \alpha & cos\ \alpha \end{bmatrix} \begin{bmatrix} cos\ \beta & 0 & sin\ \beta \\ 0 & 1 & 0 \\ -sin\ \beta & 0 & cos\ \beta \end{bmatrix} \begin{bmatrix} cos\ \gamma & -sin\ \gamma & 0 \\ sin\ \gamma & cos\ \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

A point $\mathbf{x}$ transformed by a rigid transformation might not only have been rotated by $\mathbf{R}$, but it also could have been translated by $\mathbf{t}^\top = (\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z)$ in such a way that the new transformed point $\mathbf{y} = \mathbf{R}\mathbf{x} + \mathbf{t}$. We could describe the transformation of a point $\mathbf{x}$ by both rotation and translation in a single matrix using homogeneous coordinates:

$$\begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{3.2}$$

When using homogeneous coordinates, there is the advantage of performing the rotation and translation as a single matrix multiplication. This matrix has six degrees of freedom, three for the rotation and three for translation. Figure 3.1 shows examples of rigid transformation.

[1] $SO(3)$ stands for the Special Orthogonal Group.

a) Translation b) Translation + Rotation

Figure 3.1: Rigid transformation. Figure a) is only transformed by a translation vector $\mathbf{t}$, whereas Figure b) also has a 90 degree rotation around z axis.

3.1.2 Similarity Transformation

A similarity transformation adds one degree of freedom more to the rigid transformation which is the isotropic scaling (s). We could say that an object $\mathcal{Y}$ scaled by s in all dimensions, rotated by $\mathbf{R}$ and translated by $\mathbf{t}$ is be related to another object $\mathcal{X}$ by a similarity transformation and its matrix representation is:

$$\begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{3.3}$$

Figure 3.2 shows four ducks related by similarity transformation. All of them have a different scale and a rotation.

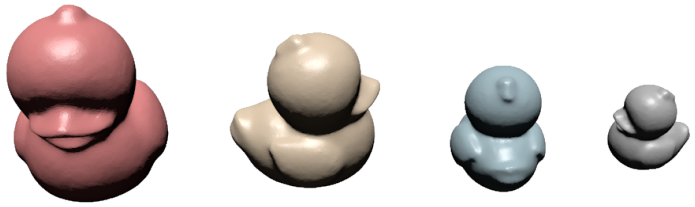


Figure 3.2: Similarity Transformation. The ducks are related by rotation, translation, and isotropic scaling.

Definition 31 (Similarity Transformation)**.** *A similarity transformation between two points* $\mathbf{x}$ *and* $\mathbf{y}$ *is a transformation which can be described by a rotation matrix* $\mathbf{R}$*, a translation vector* $\mathbf{t}$*, and an isotropic scaling factor* s*, such that* $\mathbf{y} = s\mathbf{R}\mathbf{x} + \mathbf{t}$*.*

Chapter 6 introduces a point-set correspondence algorithm which can cope up to a similarity transformation.

3.1.3 Affine Transformation

The previous rigid and similarity transformations could be considered as simplified affine transformations, since affine transformation describes the most general linear transformations between two objects. Linear transformations are the ones we can describe by a matrix multiplication. The affine matrix is described as follows:

$$\begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{3.4}$$

Definition 32 (Affine Transformation)**.** *A similarity transformation between two points* $\mathbf{x}$ *and* $\mathbf{y}$ *is a transformation which can be described by an affine matrix* $\mathbf{A}$*.*

We can exemplify the more general affine transformation with a shearing transformation not covered in rigid and similarity, but it is affine. When we shear along one direction (e.g. z), we fix that dimension and change the others by, for instance, the following matrix:

$$\begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{3.5}$$

a) Anisotropic scaling b) Shearing

Figure 3.3: Affine transformation. Figure a) is only transformed by an anisotropic scaling and Figure b) by shearing along the z axis.

Anisotropic scaling occurs when we scale the objects with different scaling factors in each dimension, which differs from the similarity transformation. Figure 3.3a shows an anisotropic scaling (the object has been scaled in the dimension x). Figure 3.3b shows an example of shearing.

3.2 Iterative Closest Point Algorithm (ICP)

The Iterative Closest Point (ICP) algorithm is one of the most famous algorithms on registration of point-sets, it was proposed by Besl and McKay [12]. It is an iterative

algorithm whose idea is to find, for each point in one set, the closest point in the corresponding set. Once the closest points are computed, the algorithm performs a least-square minimization in order to estimate the rigid transformation between both point sets. After the transformation is estimated, the points are transformed and the algorithm iterates again until the obtained error is below a certain threshold.

The first step is to compute for each reference point $\mathbf{x}_i \in \mathcal{X}$ the closest data point $\mathbf{y}_i$:

$$\mathbf{y}_i(\mathbf{x}_i) = \arg\min ||\mathbf{x}_i - \mathbf{y}||^2, \forall \mathbf{y} \in \mathcal{Y} \tag{3.6}$$

Once the correspondence is obtained (i.e. the closest points), the algorithm estimates the registration assuming a rigid transformation. At every iteration, the algorithm minimizes the distance between the transformed point cloud ($\mathbf{R} * \mathcal{Y} + \mathbf{t}$) and the reference point could $\mathcal{X}$:

$$\text{minimize} \sum_{i=1}^{N} d(\mathbf{R} * \mathbf{y}_i + \mathbf{t}, \mathcal{X}) \tag{3.7}$$

where $d(.,.)$ is a metric function (Section 2.2) which determines the error between the point clouds. Pottmann et al. [93] provides a study on the convergence of the ICP algorithm. Ezra et al. [39] also analyze the performance of ICP.

ICP is known to suffer from local minima issues which makes it highly dependant on the initialization. Several ICP-variants have been proposed to address that problem. One approach worth mentioning is the go-ICP [128] which performs the registration using a Branch and Bound optimization.

3.3 Coherent Point Drift (CPD)

The CPD algorithm proposed by Myronenko and Song [86] also addresses the point-set registration problem with an iterative approach that aligns one point-set towards the other until convergence. However, it is a more probabilistic approach since they address the registration as a density estimation problem. They consider the points in one set (e.g. $\mathcal{Y}$) as Gaussian Mixture Models (GMM) centroids and $\mathcal{X}$ as the data points generated by the GMM, in which the correspondence and transformation are estimated via Expectation Maximization (EM). For instance, points in $\mathcal{Y}$ are considered as GMM centroids with the following probability density function:

$$p(\mathbf{x}) = \sum_{m=1}^{M+1} P(m) p(\mathbf{x}|m) \tag{3.8}$$

where $P(m) = 1/M$ is a uniform probability uniform for the GMM centroids and the authors introduced $p(x|M+1) = 1/N$ to account for outliers. Expanding Eq. 3.8 with a linear combination using parameter $0 \leq w \leq 1$, they end up in the form:

$$p(\mathbf{x}) = w\frac{1}{N} + (1-w)\sum_{m=1}^{M} P(m) p(\mathbf{x}|m). \tag{3.9}$$

N is the number of points in $\mathcal{X}$ and M the number of points in $\mathcal{Y}$ (i.e. the number of centroids). CPD considers the spatial coordinates associated with the points during the computation of probabilities $p(\mathbf{x}|m)$ and $P(m|\mathbf{x}_n)$. $P_c(m|\mathbf{x}_n)$ stand for the posterior which calculates the probability of a point $\mathbf{x}n$ being generated by the centroid m. The subscript "c" stands for CPD since later we will introduce our probability estimation with the subscript "g" (Chapter 5). We will consider the centrality information of the graph along with the spatial clues. Finally, the original CPD probability function is defined in Eq. 3.10:

$$P_c(m|\mathbf{x}_n) = \frac{\exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_m, \theta)}{\sigma}\right\|^2\right)}{\sum_{k=1}^{M} \exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_k, \theta)}{\sigma}\right\|^2\right) + c} \tag{3.10}$$

where $c = (2\pi\sigma^2)^{D/2}\frac{w}{1-w}\frac{M}{N}$ and θ and σ are the parameters estimated by EM, $\mathcal{T}(\mathbf{y}_m, \theta)$ stands for the transformation applied to $\mathbf{y}_m$ given the set of parameters θ, D is the dimension of the point-sets. This transformation can be rigid, affine, or non-rigid as discussed in the previous chapter. For instance, let us take a look at the similarity transformation where there is a rotation, translation, and a scale factor. The transformation $\mathcal{T}(\mathbf{y}_m, \theta) = s\mathbf{R}\mathbf{y}_m + t$. The point $\mathbf{y}_m$ is being rotated by $\mathbf{R}$, scaled by s and translated by t, which is known as the absolute orientation problem. If the difference between points $\mathbf{x}_n$ and the transformed point $\mathbf{y}_n$ is zero, it means they are perfectly aligned.

CPD reparametrizes the GMM centroids using parameters θ and σ^2 which are estimated by minimizing the negative log-likelihood. This is equivalent to maximizing the likelihood function:

$$E(\theta, \sigma^2) = -\sum_{n=1}^{M} \log \sum_{m=1}^{M+1} P(m)p(\mathbf{x}|m) \tag{3.11}$$

and they solve the transformation $\mathcal{T}$ for the rigid, affine, and non-rigid cases separately. For details on the final estimation of those parameters, refer to [86].

3.4 Expectation Maximization Methods

Many researchers decouple the registration problem into smaller problems (transformation and correspondence). It is often referred to an alternate optimization scheme, i.e. the transformation between point-sets is fixed while the correspondence between points is estimated. Later, the correspondence is fixed and the transformation is obtained [86, 25, 77, 75, 19, 105, 80]. Combining solutions for the small problems will lead to the solution to the bigger problem.

Cross and Hancock [25] developed a framework based on EM algorithm as a dual step optimization. Previous works [123, 46] estimated the affine and Euclidean parameters of point-sets and they could be considered as one of the first to introduce structural

constraints into correspondence. Authors employed the Delaunay [33] triangulation to build the graph and to constrain the correspondences matches of the point-sets based on the edges of the graph. They define a structure called the *superclique* as the sets of nodes connecting to a central node. This structure is later used in the computation of the probability associated with the matching. It is required to have a dictionary of possible mappings and this is clearly the bottleneck of their approach. The matching algorithm estimates the geometric transformations and correspondences by jointly maximizing the likelihood over the space of matches and transformations using the EM algorithm.

3.5 Spectrum-based Methods

One of the first usages of spectral methods in GM was conducted by Umeyama [122] who performed an eigendecomposition of the graph adjacency matrix to obtain the permutation matrix $\mathbf{P}$ representing the correspondence between graphs. The solution created by Scott and Higgins [108] and later refined by Shapiro and Brady [111] was based on a proximity matrix between points (Section 2.3.4). Carcassoni and Hancock [19] also defined a point-proximity matrix $\mathbf{H}$ which can be constructed in different ways, it could be based on a Gaussian weighting function similar to [111, 108], sigmoid, or Euclidean function. They use the eigenvectors of this matrix $\mathbf{H}$ to compute the probability of assignment between points along with the EM algorithm. In their experiments, they outperformed Shapiro and Brady [111]'s approach.

Jain and Zhang [59] dealt with the shape correspondence problem by embedding the shapes in the spectral domain. They encode the spatial coordinates of each point-set using a Gaussian kernel with the geodesic distances to create two affinity matrices. Once they find the spectral embedding of those matrices, they perform an iterative alignment and obtain the correspondence by finding the best matching according to the ℓ-2 distance. In their experiments, they outperformed the previous approaches [122, 111, 19].

Other approaches using spectral theory include the work of Mateus et al. [80] that tackled the shape matching problem by exploiting the Laplacian matrix (Section 2.3.4).

3.6 Conclusions

Many algorithms have been proposed to address the point-set registration problem. More Recently, Zhou and De la Torre [131, 130] exploited the problem of point correspondence via graph matching. They first proposed the Factorized Graph Matching (FGM) [130] and later the Deformable Graph Matching (DGM) [131]. The latter creates an affinity matrix $\mathbf{K}$ encoding the similarity of both nodes and edges as well as the pairwise geometry of the points. They factorize the matrix $\mathbf{K}$ into matrices that preserve local structure of each graph. This factorization decouples the structures of the graph by replacing $\mathbf{K}$ with six smaller matrices. They alternate the optimization of the correspondence (using the path-following algorithm) and the geometric transformation (with an optimization scheme similar to ICP).

Torresani et al. [119] also propose a dual decomposition scheme for the problem of correspondence. Their scenario considers the correspondence of sparse features extracted from images. The proposed energy function is based on several terms: feature appearance, compatibility of correspondence, and spatial coherence.

In this chapter we provided a short summary of the most popular algorithms such as ICP and CPD and briefly described more approaches based on EM, which is the optimization framework behind CPD and many other algorithms. Later we will extend CPD by incorporating centralities and propose our own centrality-based registration.

CHAPTER 4

On Centrality-based Learning

> "*Siempre hago lo que no puedo hacer, para aprender cómo hacerlo.* - I am always doing that which I cannot do, in order that I may learn how to do it."
>
> – Pablo Picasso

We started the first chapter by motivating that computers could do more than just arithmetics, that they could "learn" things. The concept of learning is not necessarily the same as our understanding of learning for human beings. In the machine learning field, one could say that a computer is learning if it is able to maximize or minimize an objective function. The algorithm has an input data and based on the cost function, it tries to reach the minimum (or equivalently the maximum) of that function. Such a solution could be used to separate objects between classes, or to estimate regression coefficients, learn parameters, and many other applications. We will be providing different objective functions in this dissertation. We provide different algorithms that make decisions based on the cost of a current solution. For instance, in a classification task, the algorithm could decide to which class an object belongs, or which word is being pronounced in a speech recognition task.

We perform a review on related work and to the best of our knowledge, we were the first ones to employ centrality measures for 2D shape matching. The results of this chapter were published in the proceedings of the Graph-based Representation in Pattern Recognition in Vienna, Austria, 2013 [31]. We have obtained permission from Springer and Business Media to reuse the contents of the paper in this dissertation, in both written and online forms. The license is attached in Appendix A.

4.1 Shape Matching

We focus on the problem of shape matching and we address it by using machine learning techniques. The shape matching problem could be defined as follows:

Definition 33 (Shape Matching)**.** *A shape s_k matches another shape s_j if $i(s_k) = i(s_j)$, for $i(.)$ being an arbitrary description function that receives a shape and outputs a descriptor.*

We can illustrate the concept of learning with the following toy example. Imagine you have a camera which is able to find circles accurately. It is not important at the moment how it does it, but you can obtain the circle in each image you acquire. Based on this information, one poses the question if it is possible to distinguish between the eye of a cat from a human eye given only the radius of the circle. Hence, you build your first *feature* for your human-vs-cat classification problem. When you acquire a considerable number of images of both cats and humans, you could plot the data according to the radius vs the classes and see if those two classes would overlap or not. Let us assume for now that they do not overlap. Since we have a one-dimensional simple feature (the radius) which lie in distinctive regions in our feature space, we could find a linear function that separates both classes. This linear function only has one parameter τ to be found. Therefore, our cost function would try to estimate τ to separate both classes.

In the real world, a simple feature such as the radius would not be enough to separate the classes. For instance, the closer the cat is to the camera, the bigger the radius of the circle will be. You could make your feature more complex, by taking, for instance, color or gradient information. The design of a good feature could be considered a research problem by itself. In Section 4.3, we model a shape using centrality information, which is another application of centrality in CV.

4.2 Graph-based Shape Matching

Let us narrow and focus on the problem of matching shapes using graph representations. Notice that we are not trying to match points of a shape as we performed in the previous chapter, but are matching shapes of objects according to their classes. Let us look at some shapes on Figure 4.1 which are used in this chapter. We chose Kimia's Shape 99 database [112] which contains 11 images of 9 classes. It consists of simple binary shapes, which can easily be turned into graphs. We chose this dataset because it captures some properties which help us analyze the performance of the centralities. For instance, it contains elongated objects (wrench and shark), circular shapes (rabbit), shapes with high curvatures (hand and airplane), and so on. Instead of just randomly choosing databases, we would rather focus on properties of shapes which can help us obtain an idea about the behaviour of the algorithm according to topological properties.

In the pattern recognition literature, there are many approaches that were used throughout the years to match shapes. Shock graphs [115] were frequently employed for such a task and they are closely related to the medial axis transform [13]. A node in a shock graph is labeled according to the shock type and the edges depend on the shock formation time, therefore, having a more descriptive power than the medial axis. Siddiqi et al. [115] address the shape matching problem based on acyclic shock graphs. The task

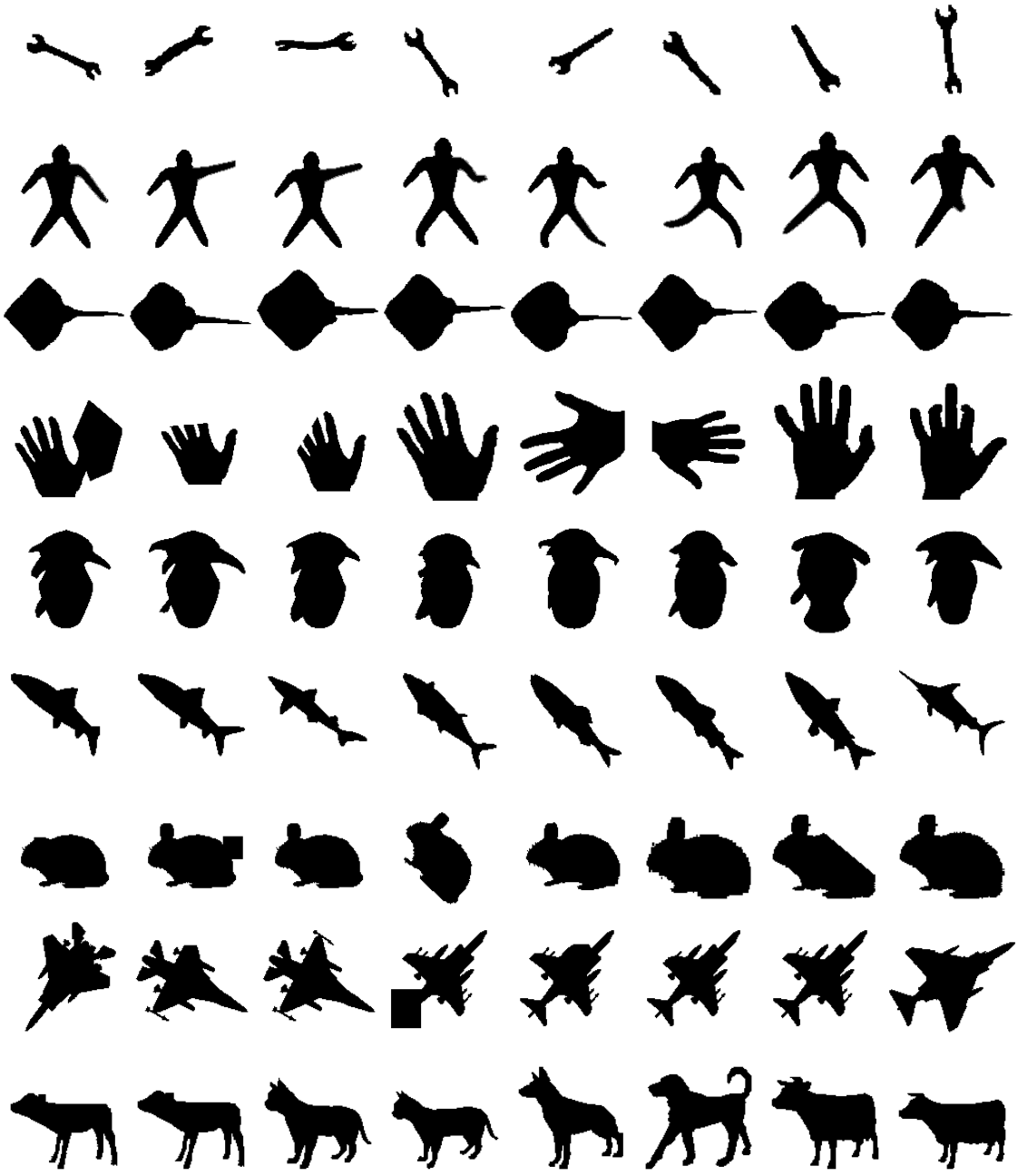


Figure 4.1: Images from the Kimia's 99 Database. This database is composed of 11 images of 9 different classes: wrench, dude, stingray, hand, scribble, fish, bunny, airplane, quadrupedes.

of matching shock graphs can be reduced to matching shock trees, using a shock graph grammar, in polynomial time.

Sebastian et al. [109] performed 2D shape outline matching based on the edit distance between their corresponding shock graphs. Their framework partitions the shape space based on the shock graph topology and discretizes the space of deformations with the help of their shock graph transitions. Finally, they find the globally optimal sequence of transitions by employing a graph edit distance algorithm. Torsello and Hancock [120] present a geometric measure to compute the similarity between skeletons of shapes.

Felzenszwalb and Schwartz [40] propose a representation called shape-tree by describing the boundary of a shape using multiple levels of resolution. In this way, it is possible to determine the similarity between two shapes. Zhu et al. [132] describe an object using a hierarchical graph by defining a deformable template which considers parent-child relationships. In the top vertex the pose (position, orientation, and scale) of the center of the object is stored and in the child vertices the poses of points on the object boundary are described.

4.3 Centrality-based Feature

In this section, we deal with the problem of describing shapes in a machine learning fashion. A good description is essential for a successful training/testing task.

Definition 34 (Feature)**.** *A feature is a description or representation of an observable phenomenon.*

Let us reason about the actual feature representation for an image class. Consider we have many images whose dimensions consists of 32 pixels in height and 32 pixels in width with only a single channel. Those image contain the handwritten digits from "0" to "9". Our task consists of recognizing the class depicted in the image. We try to find a representation that would allow us to encode that information, i.e. images of that same class would ideally have similar representation. A simple approach is to create a vector of size 32^2 using all pixel values. In this way, we can represent each image as a single point in this 2^{10}-dimensional space. We would build feature vectors for all images in the same manner and expect that those points would end up close to each other in the feature space.

It is a much deeper issue to decide how the actual training process will be performed. There are unsupervised approaches which try to cluster points which belong to the same without any prior knowledge. Some perform dimensionality reduction techniques to bring the problem to a lower dimensional space. Other approaches require the label (e.g. the class) for each image and use supervised algorithms to split that space based on the labels of each point. The feature representation has a big impact on the results, since a poorly designed feature might generate the same representation for images of different classes. Thus, the learning algorithm will not be able to split the space accordingly.

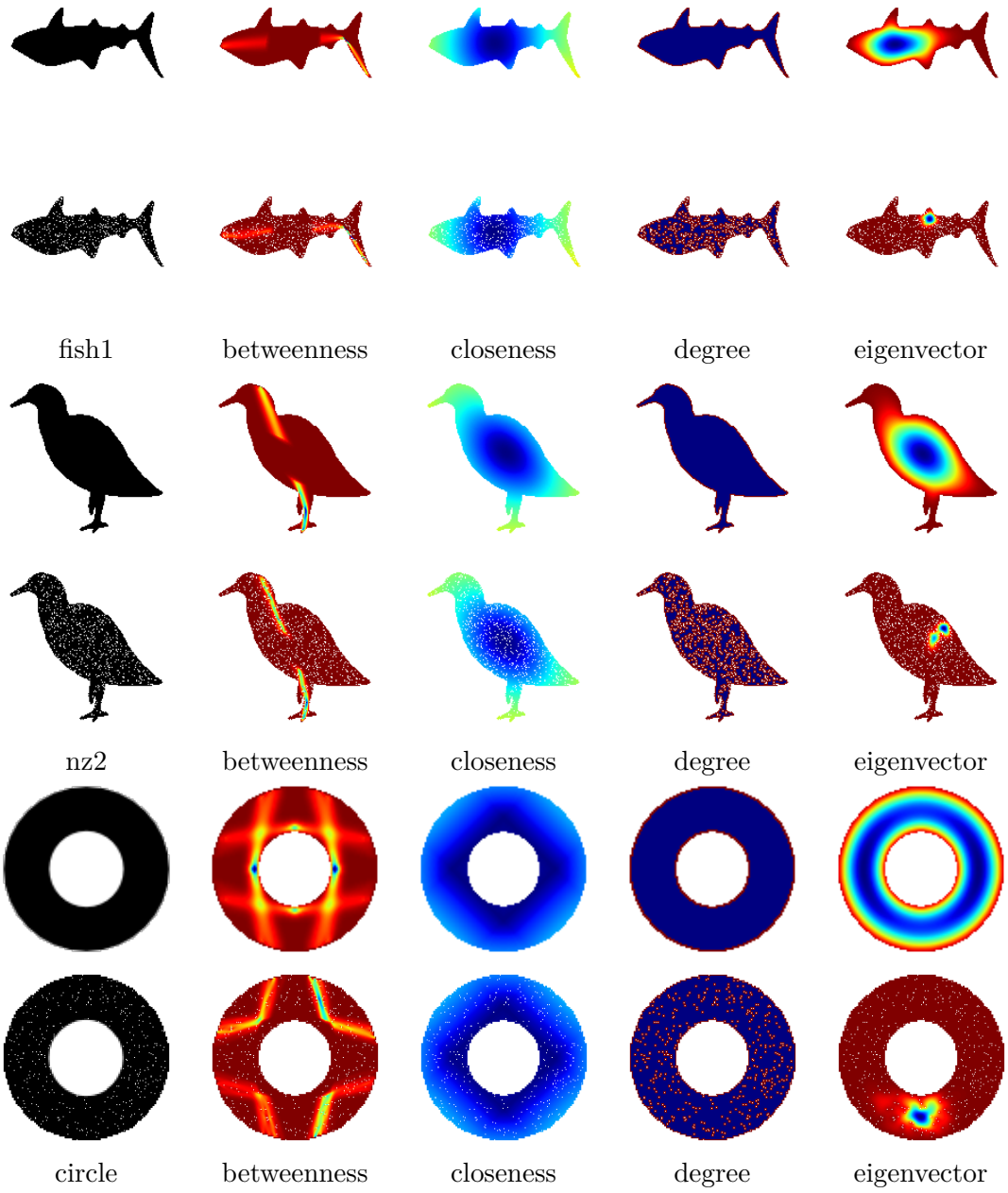


Figure 4.2: Centrality distribution of graphs (fish1, nz2, circle1) using an 8-connected neighborhood and their noisy versions.

4.3.1 Centrality of Shapes

Figure 4.2 displays centrality values of some images of Kimia's database which we are going to use. The values are color coded using a rainbow palette. For each image, we display the centralities introduced in Chapter 2. The idea is to visualize how the centrality varies according to different shapes to evaluate if the shape has an impact on the centrality values. Our graph is built using an 8-connected neighborhood (as discussed in Section 2.2.2). We encode topological information of a graph (centrality) and not image properties such as gradients, borders, or moments. The question to be asked is if such properties could be useful as features for learning, i.e. if an algorithm is able to separate the classes in the feature space based on that representation. Figure 4.2 gives us an intuition about the behaviour of centrality of nodes across the graph, such as the eigenvector centrality.

We should also focus on the noisy scenario. This would let us analyze how robust those features are and how they react in the presence of noise. We simulate some salt noise in the image as depicted in Figure 4.2. Notice that although such a noise could be easily removed by using basic image processing techniques such as the median filter, what we are doing here is changing the topology of the graph by removing nodes. Remember that as soon as we build our graph, we are no longer working on the image domain, but on the graph one, and in this domain a node removal causes a topological change. Therefore, we would like to see how robust each centrality is when the topology of the graph changes.

4.3.2 Histogram feature

A feature vector could be built using the histogram of centrality values. A histogram accumulates, in a predefined number of bins, how many values arise within a certain range according to the bins to be grouped. The salt noise added to the graph causes a topological change in which nodes are removed. We would like to evaluate the impact of that change on the histogram of centrality values.

Figure 4.3 shows the histogram of the fish1 shape according to the *betweenness* centrality. This image shows the histograms of the original images (displayed in Figure 4.2) and the noisy ones. The betweenness centrality has a high peak in the zero centrality value, which means that majority of points have the lowest importance in the graph. When the noise was added, this peak slightly reduced, but the distribution remained similar to the original image. An interesting aspect is that for many shapes, this centrality value will have similar distribution. The location of the high betweenness values is the most discriminant feature in the shape, which cannot be encoded in the simple histogram representation. The betweenness centrality has apparently the same behaviour with or without noise. We can see that the betweenness centrality has a long-tail distribution, there is a spike in the zero-betweenness value (the red region in Figure 4.2) and a fast decay for higher values, this means that there will be few points with high betweenness values. When we look at the noisy image, we see small changes on high betweenness nodes when compared to the noiseless case. The location of the "holes" in the graph

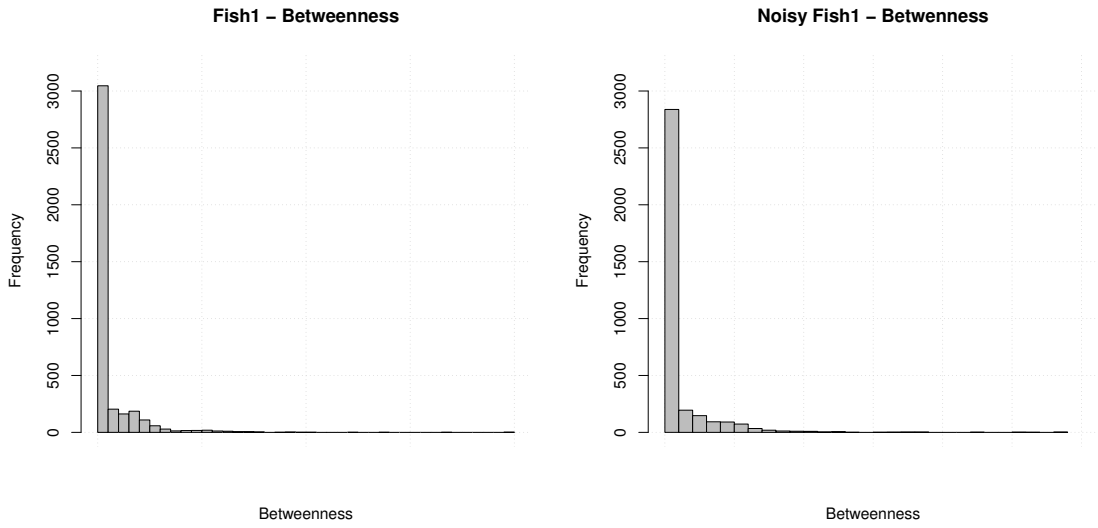


Figure 4.3: Comparison of betweenness centrality histograms

might affect the overall betweenness distribution, especially if those nodes used to lie on the geodesic paths of other points. This centrality seems to be stable to noise, although not very discriminative using this simple histogram modeling for the feature.

Figure 4.4 shows the histogram of the fish1 shape according to the *closeness* centrality. We perform a density estimation of the histogram to help visualize the distribution before and after the noise was added (the red line). We notice that the distribution remains very similar. In fact, in the experimental section, we will see that the closeness centrality was the most robust measure against the type of noise we added. The closeness centrality shows promising results and it seems to be robust to the presence of noise. Although we can see some differences in both histograms, the distribution remained very similar with respect to the noiseless case. We will show later quantitative experiments to support this observation. Remember that this centrality shows how close a node is to all the other nodes in the graph. When a single node is missing on a shortest path, there will probably be other shortest paths with the same length as alternative routes, and the distribution will remain stable over this type of noise.

The addition of noise in the graph had a strong impact on the distribution of the *degree* node. Figure 4.5 shows first the histogram of the original degree distribution. As we have closed shapes in the images, all points inside the shape will have the same degree value, i.e. the number of neighbors used in the graph construction. When the noise was added, points inside the shape will have smaller degree values, and the high peak visualized in the first image is spread across several other bins. We first notice a significant change in the distribution of the degree centrality. The degree will be defined based on the type of connections using during the creation of the graph (an 8-connected

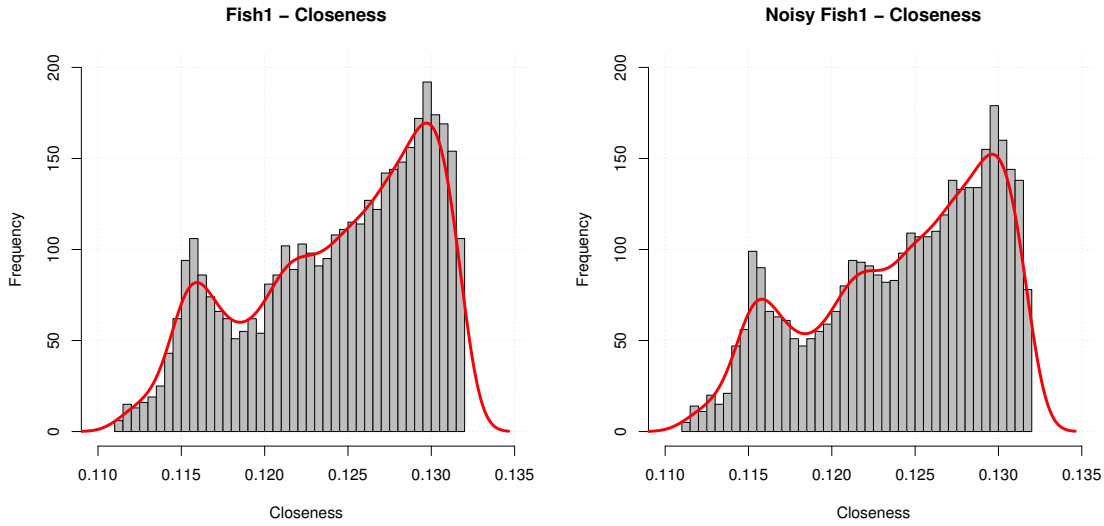


Figure 4.4: Comparison of closeness centrality histograms

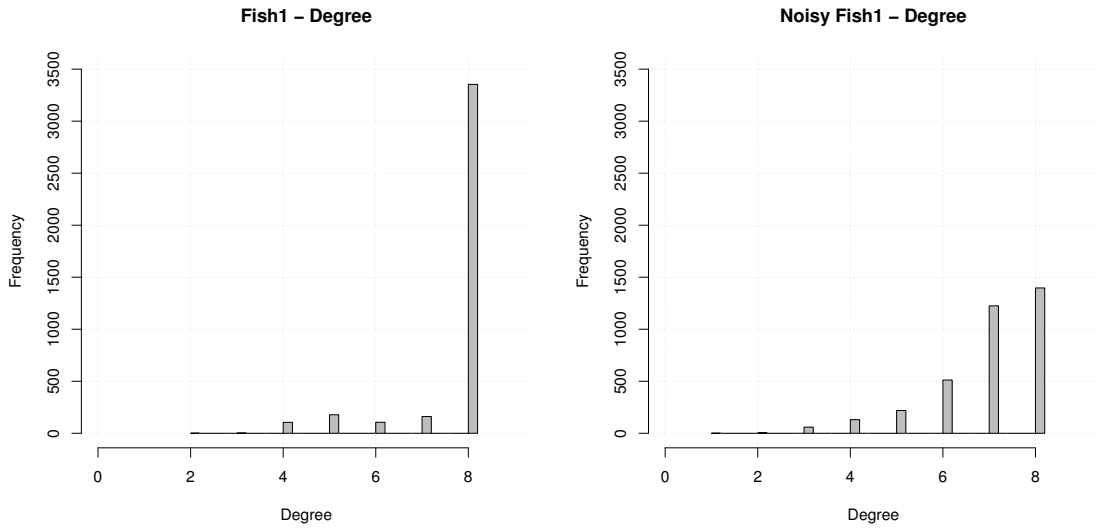


Figure 4.5: Comparison of degree centrality histograms

neighborhood in this example). In such a representation pixels might lie inside the shape or in the border, the degree distribution would have a spike on the number of neighbors used, e.g. all pixels inside the shape will have degree 8. A pixel would only have a value different than 8 when it lies in the border. Notice also that since there are no disconnected nodes, no pixel will have a 0 degree. When the salt noise is added, the degree distribution changes significantly, since a single "hole" inside the shape will affect directly 8 nodes. Therefore, trying to match shapes using the degree centrality might end up not being an easy task in the presence of noise.

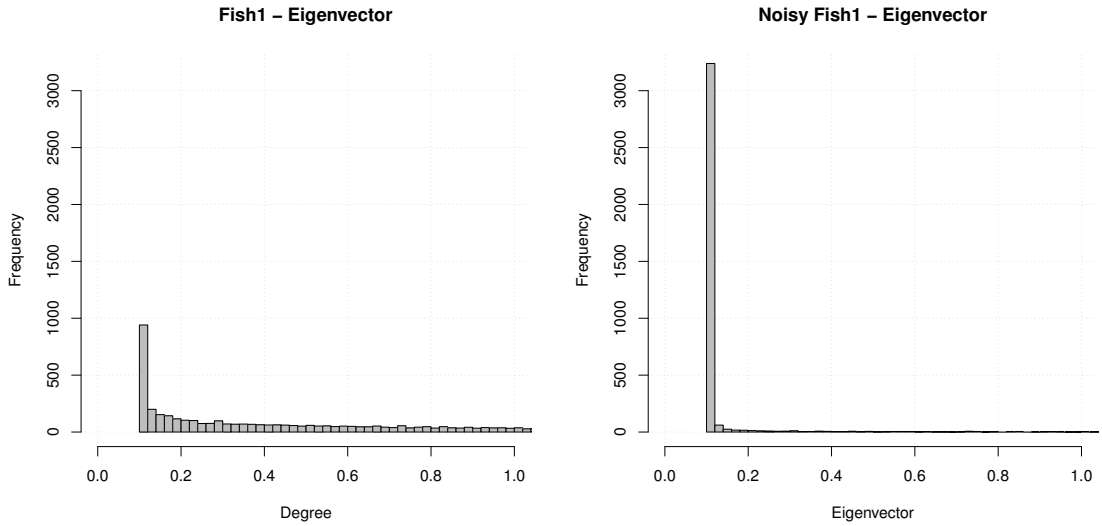


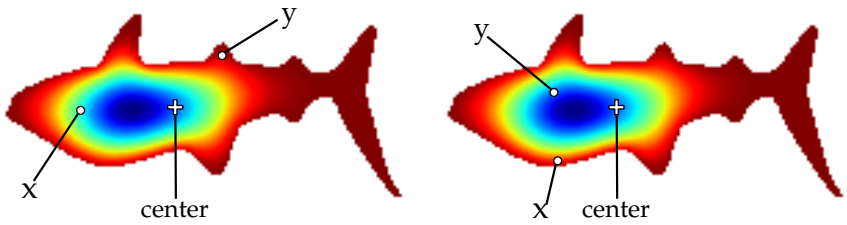
Figure 4.6: Comparison of eigenvector centrality histograms

Finally, we display the results for the *eigenvector* centrality in Figure 4.6. The eigenvector centrality had an interesting behavior in the noiseless case, it had a noticeable gradient varying from the object's center of mass towards the periphery. We also notice this pattern in the histogram of the eigenvector centrality. Although it is a long tail distribution, it spreads across all centrality values instead of having a single spike on the zero value (when compared to the betweenness centrality). When the noise was added, the distribution was flattened and more nodes were assigned the zero value. We can also notice a small spike in some random regions of the shape. What we observe is that the spike was located in the biggest closed region without noise inside the shape.

4.3.3 Random Binary Comparison of Centrality Values

One of the biggest drawbacks of representing a shape using only the histogram values is that we lose spatial information on where the high centrality values are. We only know they exist by their presence in the histogram, but we do not know where they are located.

One way to bring spatial information into our feature modeling is to use all image values as a feature vector as described in the introduction of this chapter. The problem with this kind of representation is that the algorithm will be highly biased on the training data and it will be difficult to generalize for unseen images. A powerful approach considers random binary comparisons of distinguishable locations. Querying key locations for comparison might be powerful enough to split the data correctly, such as in the work of Lepetit and Fua [70]. We do not wish to overfit the training data by using all information we have, but to learn the best way to split our centrality-based feature space. One might ask questions such: (i) Which locations should be chosen for the binary comparison? (ii) How many locations are necessary? In the approach of Lepetit and Fua [70] and Shotton et al. [114], they generate random locations and at each location they evaluate the information gain, by minimizing the entropy of leaf nodes. Therefore, they choose the locations which succeed to maximize the gain.



a) $v(\mathbf{x}) \geq v(\mathbf{y})$ b) $v(\mathbf{x}) < v(\mathbf{y})$

Figure 4.7: Binary Comparison of Centrality Values. The centrality value $v(\mathbf{x}) \geq v(\mathbf{y})$ in figure a) and it is $v(\mathbf{x}) < v(\mathbf{y})$ in b). Therefore, we could encode feature a as a quintuple $<dw_{\mathbf{x}}, dh_{\mathbf{x}}, dw_{\mathbf{y}}, dh_{\mathbf{y}}, bool>$ in which $dw_{\mathbf{x}}, dh_{\mathbf{y}}$ stand for the width and height of a point $\mathbf{x}$ with respect to the center of the shape, the same information is stored for the point $\mathbf{y}$, and we have a boolean value which indicates if the centrality of $\mathbf{x}$ is higher than the centrality of $\mathbf{y}$.

Let us consider a simple example. There are three classes: $\mathbf{a}, \mathbf{b}, \mathbf{c}$, and two images per class. We randomly choose locations for test $x_1 = (\mathbf{p}, \mathbf{q})$ such that $0 \leq p < width, 0 \leq q < height$ and $x_2 = (\mathbf{r}, \mathbf{s})$ such that $0 \leq r < width, 0 \leq s < height$ and we compare the centrality values of all images using those locations. For each location, we split the images into two sets, the ones who fulfill the condition $(v(\mathbf{p}) \geq v(\mathbf{q}))$ and the ones which did not. The same applies for location x_2. After testing x_1 in all images, we obtain $left_1 = [\mathbf{a}, \mathbf{a}, \mathbf{b}, \mathbf{c}]$ and $right_1 = [\mathbf{b}, \mathbf{c}]$. Nevertheless, when we apply the test for location x_2, we obtain $left_1 = [\mathbf{a}, \mathbf{a}]$ and $right_1 = [\mathbf{b}, \mathbf{b}, \mathbf{c}, \mathbf{c}]$. We can calculate the entropy values for all sets and choose the test location which minimizes the entropy. For instance, by using location x_2, we are able to distinguish class $\mathbf{a}$ from the other ones. Therefore, we would choose x_2 for this level, and later repeat the process until we are able to split classes $\mathbf{b}$ and $\mathbf{c}$. Therefore, the leaf nodes of the binary tree would hold a posterior

probability for each class, e.g. the probability of finding a class **b** in the right node of x_1 is 25%. There are many variants of this method and for a deeper understanding, the reader can refer to [114, 70].

Figure 4.7 shows two features using binary comparison. The idea consists of considering a certain reference point in the shape, e.g. the center of mass of the shape, the center of the window, or even the center of the graph, i.e. the node with minimum eccentricity value (Section 2.2.2). After choosing a reference location, we could compare the centrality values of two other locations with respect to this reference. We call a boolean comparison because we are only interested to see if the location $\mathbf{x}$ has a higher centrality value than location $\mathbf{y}$. Therefore, the feature representation would be $f(\mathbf{x}, \mathbf{y}) = <dw_{\mathbf{x}}, dh_{\mathbf{x}}, dw_{\mathbf{y}}, dh_{\mathbf{y}}, bool>$, where $dw_{\mathbf{x}}, dh_{\mathbf{y}}$ stands for the distance that $\mathbf{x}$ is from the reference point and the same for $\mathbf{y}$. The boolean value tells if the centrality of $\mathbf{x}$ is higher than $\mathbf{y}$ or not. We have answered question (i), choosing the locations which maximize the gain by minimizing the entropy. Question (ii) is related to the height (h) of the tree. A tree which is really deep could easily overfit the dataset. One has to keep in mind that the number of comparisons will be on the order of 2^h, since at each branching location we create more nodes. The traditional way is to avoid overfitting by cross-validating that parameter.

4.4 Training Centrality-based Features

Most of the feature vector procedures we have been talking are suitable for the so-called shallow machine learning. Although the meaning of shallow learning is not a consensus, the shallow architecture for learning became more evident in contrast to deep representations [54, 116]. In this work, we will only refer to shallow architectures. There are many alternative definitions for the concepts being described in this section.

Definition 35 (Shallow Learning)**.** *A shallow architecture is a learning procedure which models the problem in one level of representation (feature vector) which is then used for training.*

This definition means that we encode all possible information for our problem in just a single feature vector and hope that this feature vector is discriminative enough to separate our classes. The learning algorithm will be trained using those features. On a deep architecture, such as convolutional neural networks [68], there are several layers of abstractions built to represent the problem.

Definition 36 (Training Algorithm for Classification)**.** *An algorithm which can minimize a cost function based on labeled features (experience) to decide upon the class (prediction) of a certain feature (representation) is called a training algorithm.*

There is a vast literature on learning algorithms based on shallow architectures, such as the Naive Bayes Classifier, Nearest Neighbors classifier, SVM[23], and so on. We have sketched a shallow learning procedure based on centrality in Algorithm 4.1. The input for

Algorithm 4.1: Training Algorithm

Data: Images I and labels L
Result: A trained algorithm T

```
1 F ← {}
2 for i ∈ I do
3 |   G(V, E) ← build_graph(i);
4 |   C ← centrality(G(V, E));
5 |   f ←build_feature(C);
6 |   F ← F ∪ {f};
7 end
8 T ←train(F, L);
```

the algorithm consists of the images and their respective labels. For each image, we build a graph representation of it (Line 3), and later calculate the centrality (Line 4). Based on the image and the centrality values for the nodes, we build a feature representation (histogram, patches, etc.). When all features have been prepared for learning, we train the algorithm (Line 8).

The training part of a learning algorithm is the stage which takes a considerable amount of time. It depends on the algorithm, on the amount of data, and number of classes. The testing procedure is usually a fast procedure, which uses the trained algorithm to estimate the class label. After training the algorithm, we would simply classify a certain feature using that algorithm. The input is the image we would like to classify and the algorithm we trained in the previous step. From lines 1 to 3 we build the same feature representation as we did for the training algorithm, but now the algorithm only predicts the class for this feature. In online learning approaches, the classifier could be modified or updated based on the new data.

4.5 Experiments

We have performed experiments on Kimia's 99 dataset [112] using the training procedure described in Algorithm 4.1. We build a graph for each image and compute the feature vectors using the centrality measures. The feature modeling is based on the histogram described in Section 4.3.2.

Our hypothesis we would like to evaluate is: if and which centrality measures can be used (as features) in ML for shape matching tasks under the presence of topological changes (noise). We have four different versions of the dataset, the noiseless case and datasets under different noise levels: 1%, 5%, and 10%. We build five different classifiers using graph centralities. Four classifiers are computed using each individual centrality histogram and one classifier using the combination of all of them. In these experiments, we grouped the values into 40 bins.

In ML, if one classifies the set which was used for training, the results would not be meaningful since they could easily overfit the trained set. Although this measure is not

meaningful to assess how good a classifier is, it is useful to find out how discriminative the centrality-based feature is. For instance, if two different objects $\mathbf{a} \in A$ and $\mathbf{b} \in B$ generate the same feature representation A but belong to two different classes, no matter which training algorithm is used, there will be a misclassification. The algorithm will make a decision for either classes A or B. This happened due to poor representational power of the feature. This is what we aim when classify the same dataset. Nevertheless, we also evaluate how good they are with respect to images never seen before, such as the noisy ones, in which the algorithms are unaware of the topological change. In this way, we can find out which centrality measure could be a good feature representation when using the histogram.

Figure 4.8 shows the confusion matrices of the classification results for classifiers *all (A)*, *betweenness (B)*, *closeness (C)*, *degree (D)*, and *eigenvector (E)*. The first column (Figures 4.8a, 4.8e, 4.8i, 4.8m, and 4.8q) shows the results on the same trained dataset. These are the images in which we would expect a perfect confusion matrix, i.e. an identity matrix. The first row shows the results when all centrality measures were used. As expected, the trained algorithm obtained almost correct classification for all classes in the trained dataset (Figure 4.8a). We see that there are still some images misclassified. As soon as the classifier A faces the datasets with noise, the results start obtaining worst performance, for the *airplane* and *dude* classes, there was not a single correct classification. Nevertheless, classes with elongated structures had correct classification results with high accuracy (91% for *wrench* and 100% *shark*) under 10% of noise.

The second row shows the results for the betweenness centrality. We see the histogram of the betweenness centrality does not seem to be a robust representation. Most nodes have the degree equal to zero, as we expected when we plot the distribution of centrality over the shape and the histogram. In the noiseless case, Figure e), we see that some are misclassified, and as the noise increases, this centrality starts showing strong misclassification results. The classes *hand* and *rabbit*, and *dog* and *stingray* had wrong results. We believe that this centrality is robust to noise, but the histogram of it is not. A more powerful result could be obtained if the location of the high peaks of betweenness values in the shape were stored in the feature. Referring to Figure 4.2, we can see that the high peaks were stable under noise.

The centrality representation which obtained the best results was the closeness centrality, as seen in Figures 4.8i, 4.8j, 4.8k, 4.8l. Figure 4.8l shows that the confusion matrix still looks like an identity one even under the presence of heavy noise. The reason for this is that when nodes were randomly deleted, there were still alternative paths whose costs were not much higher than the original case. This centrality shows the most robust results under this scenario. The scenario in which we would expect poor results of the closeness centrality would be one in which many adjacent nodes would vanish together, since this would change abruptly the cost of the shortest paths in the graph.

The fourth row shows the degree centrality, which obtained the worst results in this experiment. The reason for that is simple, the distribution had previously a single peak in the number of neighbors used, but when the noise appeared, other peaks started appearing in the histogram, and this classifier was not able to correctly classify under

such a change. Under 10% of noise, almost all images were classified as belonging to the shark class.

The results of classification are summarized in Table 4.1. First column displays the classifier evaluated. Second column (no noise) displays the result of the classification on the training data, which shows that the classifier using all centralities obtained the best results (98% of correctness). Most of the individual centralities were able to perform well (with at least 81% of correctness). However, when 1% of noise is added, the closeness centrality starts to obtain the best performance. The other centrality measures are more sensitive to noise than closeness and their results drop remarkably: the addition of random noise had the highest impact on the results of degree and eigenvector. We observe that closeness centrality remains stable even though there are missing nodes in the graph. Even when 10% of noise is added, it still obtains better results than other centralities on the training data.

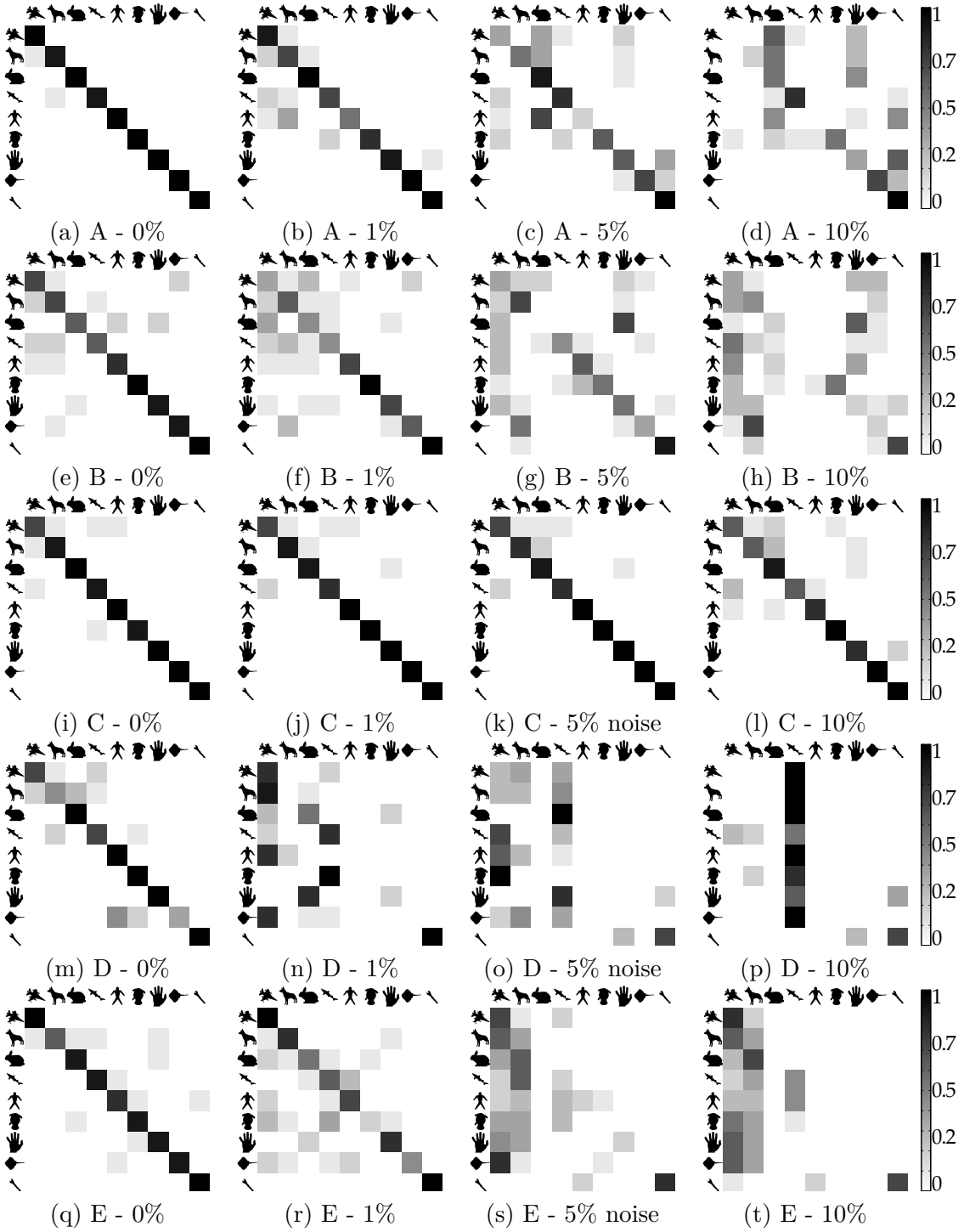
classifier	**No noise**	**1% noise**	**5% noise**	**10% noise**
All	**0.98**	0.85	0.65	0.46
betweenness	0.82	0.67	0.51	0.29
closeness	0.94	**0.93**	**0.92**	**0.83**
eigenvector	0.89	0.69	0.27	0.26
degree	0.81	0.37	0.17	0.14

Table 4.1: Classification results using different centrality measures. The closeness centrality achieved the best results.

4.6 Conclusions

It can be concluded for *Kimia's 99 dataset* and the *histogram* feature representation that the closeness centrality is the most robust centrality measure for shape matching using the histogram representation and naive Bayes classifier. The simple combination of all centralities did not obtain the best result but it was still considerably better than the other isolated centralities. Furthermore, the performance of the degree centrality changes severely as under "salt" noise. This happens due to the fact that the vertices inside the shape have degrees smaller than the number of neighbors used. This type of noise was chosen due to the topological changes it causes in the graphs. The histogram distribution changes significantly for some centrality values.

Notice, however, that these experiments only evaluate how good they were in combination to the histogram representation. When the location is incorporated to the feature representation, those results could change and for instance, we would expect that the betweenness centrality would be able to represent the data well even when the topology of the graph changed. We will apply the random binary comparison in our future work (Section 8).



(a) A - 0% (b) A - 1% (c) A - 5% (d) A - 10%

(e) B - 0% (f) B - 1% (g) B - 5% (h) B - 10%

(i) C - 0% (j) C - 1% (k) C - 5% noise (l) C - 10%

(m) D - 0% (n) D - 1% (o) D - 5% noise (p) D - 10%

(q) E - 0% (r) E - 1% (s) E - 5% noise (t) E - 10%

Figure 4.8: Confusion Matrices for the Centrality-based Learning.

CHAPTER 5

Graph-based Point Drift

One of the first problems discussed in this dissertation was the registration of point-sets. We illustrated in Chapter 3 the importance of such problem with the example of finding the correct key to open a door. It is time for us to address the registration of point-sets using graph centralities (introduced in Chapter 2). The centralities will play an important role on the estimation of correspondence, which is the step assigning the probability of how likely a node is to be matched to another node.

The results of this chapter were published in the Pattern Recognition journal by Elsevier, 2015 [27]. Elsevier has granted us permission to reuse the full content of the paper, including text and images, in this dissertation. The license is available in Appendix A.

5.1 Computing Probabilities using Centralities

Let us first build the intuition for registering points using centralities. Given a point-set $\mathbf{X}$ and its associated graph $\mathcal{X}$, we shall refer to $\mathbf{x}$ as a point belonging to both $\mathbf{X}$ and $\mathcal{X}$, hence, assuming a bijection $\mathcal{X} \leftrightarrow \mathbf{X}$. We shall first triangulate a point-set to obtain a graph (the data-graph). Chapters 6 and 7 will deal with techniques for creating data graphs out of point-sets. We color code the graph of Fig. 5.1a to display the centrality value of each node. The closeness centrality $c(\mathbf{x}_n)$ of a point $\mathbf{x}_n \in \mathcal{X}$ is chosen for this example. Blue nodes represent points with low centrality values and pink nodes represent the ones which are more central to graph according to the closeness. Let us also compute the histogram of those values (Fig. 5.2a) to find out how diverse the distribution of centrality is across the graph.

When looking at a particular bin of the histogram, we notice that many nodes have the same centrality value and there are many non empty bins. This indicates that the distribution is spread across a wide range of centrality values instead of having a single peak (such as the degree centrality described in Chapter 4). We seek to find candidates

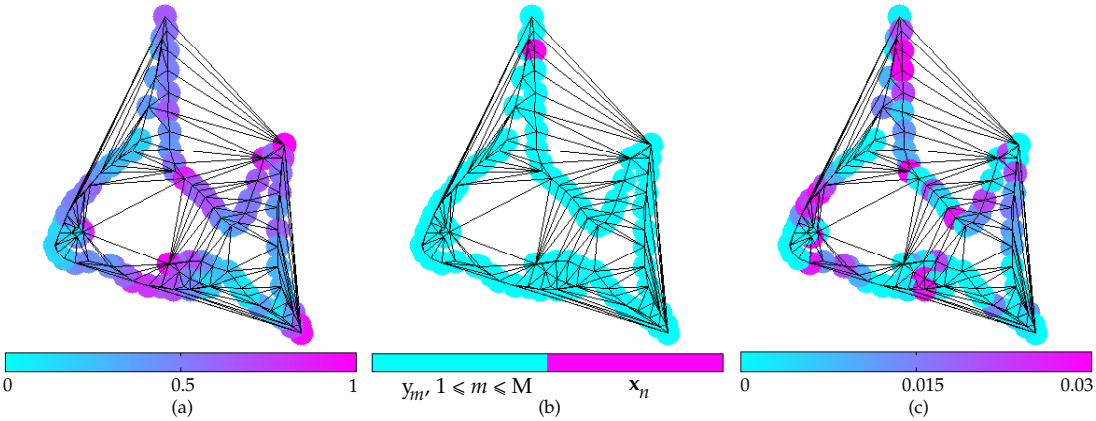


Figure 5.1: **(a)**: Graph $\mathcal{X}$ built using Delaunay triangulation color-coded using the closeness centrality. The pinkish the node is, the more central it is. **(b)**: Node $\mathbf{x}_n$ chosen for calculating the probability of being matched with respect to the centrality of the others. **(c)**: Probability $P_g(m|\mathbf{x}_n)$ of $\mathbf{x}_n$ being matched as in Eq. 5.1. Finally, $\sum_{n=1}^{N} P_g(m|\mathbf{x}_n) = 1$. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

to match our point $\mathbf{x}_n$ (the pink node displayed in Fig. 5.1b). If we are about to match nodes based solely on the centrality, we would believe that a node whose centrality value is similar to the one we seek is more likely to be the correct node and all the other nodes should be rejected. This seems to be a strong assumption, since we would ignore all the points whose centrality are not exactly the same one as $\mathbf{x}_n$. Instead of making such a hard decision, we could assume that the chances of finding the correct point will be higher if the point has the same centrality as our point $\mathbf{x}_n$ and it will be lower if this centrality is not the same. How this probability will vary is up to us. This intuitive assumption does not discard points, but it makes it highly unlikely to be matched the more different the centrality values are.

One way to model the decay of importance for matching is to center a Gaussian distribution (or any other unimodal distribution) in the location which has the highest likelihood of being the correct match (i.e. the nodes have the same centrality values as the node $\mathbf{x}_n$). Points whose centralities are higher or lower shall receive a lower probability of being matched. Thus, the probability of $\mathbf{y}_m$ being matched to $\mathbf{x}_n$ could be defined as:

$$P(m|\mathbf{x}_n) = \frac{h_m^c \; exp\left(-\frac{\|v(\mathbf{y}_m) - v(\mathbf{x}_n)\|^2}{2\phi^2}\right)}{\sum_{k=1}^{M} h_{\mathrm{k}}^c \; exp\left(-\frac{\|v(\mathbf{y}_k) - v(\mathbf{x}_n)\|^2}{2\phi^2}\right)}, \tag{5.1}$$

in which h_m^c stands for the histogram bin associated with the centrality $v(\mathbf{y}_m)$ of node $\mathbf{y}_m$ indicating how many nodes were found in that graph whose centrality is the same as $\mathbf{y}_m$. Intuitively, the nodes of the graph whose centrality values are closer to the centrality of $\mathbf{x}_m$ would have a higher probability of being matched. In fact, Eq. 5.1 is similar to the one proposed by Carcassoni and Hancock [19, Eq. 16]. The authors employ eigenvectors of a proximity matrix $\mathbf{H}$ with different weighting functions. If we apply the eigenvector centrality as $v(\mathbf{x}_n)$ for Eq. 5.1, we compute the difference between the elements of the eigenvector of the adjacency matrix, not of a proximity matrix created with a specific weighting function.

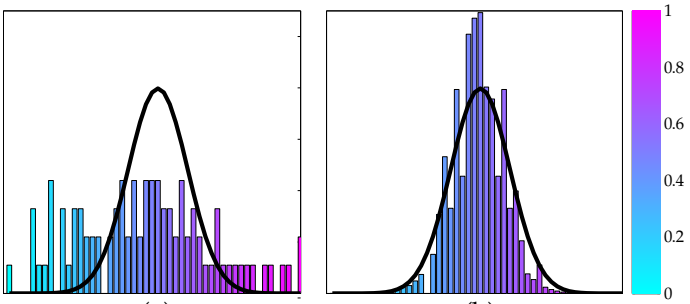


Figure 5.2: (a): The histogram of the closeness centrality of the nodes displayed in Fig. 5.1 (a) overlapped with a Gaussian distribution centered at $\mathbf{x}_n$. (b) The probability of the nodes $P(m|\mathbf{x}_n)$ as computed in Eq. 5.1. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

The probability of a node $\mathbf{y}_m$ being matched to node $\mathbf{x}_n$ is displayed in Fig. 5.1c. We can see that many nodes are assigned the blue color, meaning that they have a lower chance of being matched to $\mathbf{x}_n$ due to the fact that their centrality diverges significantly. The intuition of Eq. 5.1 is to combine a Gaussian distribution with the one estimated by the histogram to enhance the probability of nodes whose values are closer to the one we would like to match. Fig. 5.2a shows the histogram of the centrality distribution in Fig. 5.1a while the final probability is displayed in Fig. 5.2b. The color scheme displays the importance of the node according to the closeness centrality. Pink nodes have the highest closeness centrality (normalized to between zero and one). The Gaussian distribution is centered at the centrality value of point $\mathbf{x}_n$ and ϕ is the variance of the

centralities in $\mathcal{X}$ which controls how strong the influence of the Gaussian is. The area under the curve adds up to one. We notice that based only on the centrality, one could find the correct match in the first ten more likely nodes, the centrality itself provides a good prior for pruning unfeasible nodes.

5.2 Combining Centrality and Spatial Information

Although our probability estimation produced good results in the previous section (i.e. it gave a high probability to the correct point), it is clear that we can improve Eq. 5.1 if we integrate the spatial information as well. There are many nodes with similar centralities which turns the matching process into a difficult task if we only rely on the centrality and since the spatial information is available, there is no reason for not doing so.

The centrality estimation helped pruning nodes that do not represent good matches, by integrating it to CPD, we expect that the algorithm will converge faster. First of all, we define the combination of centrality and spatial information in Eq. 5.2:

$$\psi(\mathbf{x}, m) = -\left(\frac{\|\mathbf{x} - \mathbf{y}_m\|^2}{2\sigma^2} + \frac{\|v(\mathbf{x}) - v(\mathbf{y}_m)\|^2}{2\phi^2}\right), \tag{5.2}$$

where σ^2 is the variance associated with the locations of the points and ϕ^2 the variance associated with the centrality values. The probability $p(\mathbf{x}|m)$ takes the form of:

$$p(\mathbf{x}|m) = \frac{h_m^c \exp(\psi(\mathbf{x}, m))}{(2\pi\sigma^2)^{D/2}}, \tag{5.3}$$

Myronenko and Song [86, Eq. 1] define $p(\mathbf{x})$ as:

$$p(\mathbf{x}) = \sum_{m=1}^{M+1} p(m)p(\mathbf{x}|m), \tag{5.4}$$

note that the sum goes up to $M + 1$ as they introduce a term $p(M + 1)$ to account for outliers, as well as a parameter w which weighs between the two terms. Therefore, we can rewrite $p(M + 1) = \frac{w}{N}$ and $p(m) = \frac{1-w}{M} : 1 \leq m \leq M$ and decompose the sum into two parts:

$$p(\mathbf{x}) = w\frac{1}{N} + (1 - w)\sum_{m=1}^{M} \frac{1}{M} p(\mathbf{x}|m). \tag{5.5}$$

Our centrality-based probability estimation ($P_g(m|\mathbf{x})$) is called Graph-based Point Drift (GPD) and we start the derivation from Bayes' theorem:

$$P_g(m|\mathbf{x}_n) = \frac{p(m)p(\mathbf{x}_n|m)}{p(\mathbf{x}_n)}, \tag{5.6}$$

$$P_g(m|\mathbf{x}_n) = \frac{\frac{(1-w)}{M}\frac{h_m^c \exp\left(\psi(\mathbf{x}_n,m)\right)}{(2\pi\sigma^2)^{D/2}}}{w\frac{1}{N} + (1-w)\sum_{k=1}^{M}\frac{1}{M}\frac{h_k^c \exp(\psi(\mathbf{x}_n,k))}{(2\pi\sigma^2)^{D/2}}}, \tag{5.7}$$

multiplying by $NM(2\pi\sigma^2)^{D/2}$:

$$P_g(m|\mathbf{x}_n) = \frac{N(1-w)h_m^c \exp(\psi(\mathbf{x}_n,m))}{wM(2\pi\sigma^2)^{D/2} + N(1-w)\sum_{k=1}^{M} h_k^c \exp(\psi(\mathbf{x}_n,k))}, \tag{5.8}$$

dividing by $N(1-w)$:

$$P_g(m|\mathbf{x}_n) = \frac{h_m^c \exp(\psi(x,m))}{\frac{wM(2\pi\sigma^2)^{D/2}}{N(1-w)} + \sum_{k=1}^{M} h_k^c \exp(\psi(\mathbf{x}_n,k))}, \tag{5.9}$$

as CPD first did in Eq. 3.10, we also call $c = \frac{wM(2\pi\sigma^2)^{D/2}}{N(1-w)}$. Therefore, the probability function which considers both the spatial and centrality information is computed as:

$$P_g(m|\mathbf{x}_n) = \frac{h_m^c \ \exp(\psi(\mathbf{x}_n,m))}{\sum_{k=1}^{M} h_k^c \ \exp(\psi(\mathbf{x}_n,k)) + c}. \tag{5.10}$$

Each centrality measure can be applied in $v(\mathbf{x}_n)$ for computing the probabilities. We evaluate the performance of the different measures in our experiments. We use the subscript g in $P_g(m|\mathbf{x}_n)$ to differentiate between the probability $P_c(m|\mathbf{x}_n)$ originally proposed by CPD and our new probability estimation combining the centrality values. Eq. 5.10 states that the centrality will either reward or penalize the connection between $\mathbf{y}_m$ and $\mathbf{x}_n$ based on how similar (or divergent) their centrality values are.

5.3 Relation between GPD and CPD

In the previous section, we have provided a way to integrate the centralities in the probability estimation of CPD. Theorem 1 helps us better understand how GPD is related to CPD.

Theorem 1 (Relation between GPD and CPD)**.** *The centrality values of either a complete graph or a disconnected graph do not improve CPD.*

Let $\mathcal{X}_N$ and $\mathcal{Y}_N$ be complete graphs with N nodes associated with point-sets $\mathbf{X}$ and $\mathbf{Y}$ respectively. CPD treats $\mathbf{Y}$ as GMM centroids and $\mathbf{X}$ as points generated by the GMM, where $m = (1 \dots |\mathbf{Y}|)$ stands for the GMM components. Theorem 1 states that the GPD posterior for the GMM components of the complete graph $P_g(\boldsymbol{m}|\boldsymbol{x}_n)$ is equivalent to the GPD posterior of the disconnected graph $P_g(m|\mathbf{x}_n)$ and to the CPD posterior $P_c(m|\mathbf{x}_n)$ which only uses the point-sets, where $\boldsymbol{y}_m \in \mathcal{Y}_N$, $\boldsymbol{x}_n \in \mathcal{X}_N$, $\mathbf{y}_m \in \mathbf{Y}$, and $\mathbf{x}_n \in \mathbf{X}$. Thus, it establishes a relation between CPD and GPD. In order to prove Theorem 1, we need to show that for a complete graph $\mathcal{X}_N$, all nodes have the same centrality value (Lemma 1).

Lemma 1 (Centrality values of a complete graph)**.** *Centralities* $v(\mathbf{x}) = v(\mathbf{y})$ *for all* $\{\mathbf{x}, \mathbf{y}\} \in \mathcal{X}_N$.

Proof of Lemma 1. The *degree* centrality states that all $d(\mathbf{x}) = N-1$ for any node $\mathbf{x} \in \mathcal{X}_N$, thus, fulfilling Lemma 1. The *closeness* centrality is calculated as $c(\mathbf{x}) = \sum \frac{1}{\mathrm{d}(\mathbf{x},\mathbf{y})}$, for $\mathbf{y} \in \mathcal{X}_N \backslash \{\mathbf{x}\}$, where $\mathrm{d}(\mathbf{x}, \mathbf{y})$ is the shortest distance between nodes $\mathbf{x}$ and $\mathbf{y}$. As all the nodes are connected, $\mathrm{d}(\mathbf{x}, \mathbf{y}) = 1$ and the centrality $c(\mathbf{x})$ will be the same for all nodes in the graph. For the *eigenvector* centrality, let us first define $\mathbf{J}$ as a $\mathbf{1}$ matrix of dimension N, the rank of $\mathbf{J}$ is 1 with spectrum N^1, 0^{N-1}. The adjacency matrix $\mathbf{A}$ of $\mathcal{X}_N$ is $\mathbf{A} = \mathbf{J} - \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Considering that the spectrum of $\mathbf{I}$ is 1^N, the spectrum of $\mathcal{X}_N$ is $(N-1)^1$ and $(-1)^{N-1}$. In order to fulfill the condition $\mathbf{Aw} = \lambda\mathbf{w}$ for the eigenvalue $\lambda = N - 1$, the eigenvector $\mathbf{w}$ associated with it needs to be an $\mathbf{1}$-vector. Therefore, for any node $\mathbf{x} \in \mathcal{X}_N$, the centrality value $e(\mathbf{x}) = 1$. □

Considering all centrality measures which fulfill Lemma 1, we can now prove Theorem 1.

Proof of Theorem 1. Due to the fact that $\mathcal{X}_N$ and $\mathcal{Y}_N$ are complete graphs, Lemma 1 states that $v(\mathbf{x}_k) = v(\mathbf{x}_i)$, for all $\{\mathbf{x}_k, \mathbf{x}_i\} \in \mathcal{X}_N$ and the histogram value $h_c^m = N$, since all centrality values are the same, thus, they will be in the same bin. Without loss of generality, the same holds for $\mathcal{Y}_N$. The exponential of two sums is equal to the product of exponential terms. Therefore, we can split the ψ function (Eq. 5.2) into two terms and plug those previous values into Eq. 5.10:

$$P_g(m|\mathbf{x}_n) = \frac{N \exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_m,\theta)}{\sigma}\right\|^2\right) \exp\left(-\frac{1}{2}\left\|\frac{v(\mathbf{x}_n) - v(\mathbf{y}_m)}{\phi}\right\|^2\right)}{\sum_{k=1}^{M} N \exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_k,\theta)}{\sigma}\right\|^2\right) \exp\left(-\frac{1}{2}\left\|\frac{v(\mathbf{x}_n) - v(\mathbf{y}_k)}{\phi}\right\|^2\right) + c}, \tag{5.11}$$

since $v(\mathbf{x}_n) = v(\mathbf{y}_m)$:

$$P_g(m|\mathbf{x}_n) = \frac{N \exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_m,\theta)}{\sigma}\right\|^2\right) \exp(0)}{\sum_{k=1}^{M} N \exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_k,\theta)}{\sigma}\right\|^2\right) \exp(0) + c}, \tag{5.12}$$

and we arrive at:

$$P_g(m|\mathbf{x}_n) = \frac{\exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_m,\theta)}{\sigma}\right\|^2\right)}{\sum_{k=1}^{M}\exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_n - \mathcal{T}(\mathbf{y}_k,\theta)}{\sigma}\right\|^2\right) + c}, \tag{5.13}$$

showing that the probability Eq. 5.13 is equivalent to Eq. 3.10. When using disconnected graphs in GPD, the content inside the exponential function will be zero[1] and the same results would be obtained. □

Theorem 1 indicates that neither a completely disconnected graph $\mathbf{X}$ (just a point-set) nor a complete graph $\mathcal{X}_N$ could improve CPD. Any improvement could only be obtained if the graph is neither empty nor complete but when the edges play the important role of resembling the actual structure of the object being matched.

5.4 Experiments

Our experimental section aims at evaluating the behaviour of different measures of centralities in different types of data using our probabilistic estimation. We also compare our results with the original CPD algorithm. We perform two types of experiments:

1. Registration under different types of transformations (as explained in Chapter 3).
   - a Similarity transformation without noise.
   - b Affine transformation with missing points.
   - c Non-rigid transformation with noise and missing points.
2. Registration using point-sets of different classes.
   - a Clusters.
   - b Points randomly distributed.
   - c Points normally distributed.
   - d Silhouette of objects.
   - e Points sampled from a grid.

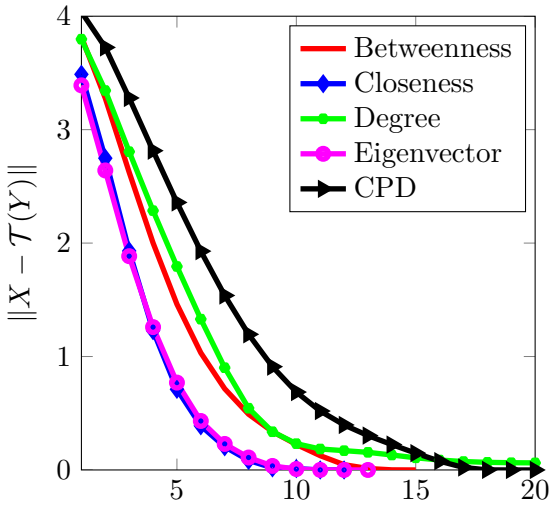
The experiments we designed have different goals: the first one aims at assessing how fast each centrality is when we vary the type of transformation. Remember when we centered the Gaussian in the histogram, the probability was pruning nodes which could not represent good matches. We also referred to noise in the first type of experiment. We

[1] Assuming that centrality of disconnected nodes are either zero or a constant but not infinity, e.g. Freeman [43] considers the distance between disconnected nodes as infinity, hence, we could consider that the closeness $c(\mathbf{x}_n) = \sum 1/\infty \to 0$. Freeman [42] also defines the betweenness of disconnected nodes as 0.

define noise as spurious points which do not belong to the original dataset and by missing points, we mean that true points of the dataset will not be present. For the second type of experiments, we evaluate 495 databases consisting of 5 classes enumerated before. Those classes of point-sets have different properties, such as the sampling functions, the distributions applied for drawing points, or points lying on the silhouette of objects. The aim of the second part is to isolate certain properties of the point-sets which could help to understand the behaviour or isolate the behaviour of the individual centrality. When we studied the centrality based learning in Chapter 4, we observed different behavior for the datasets. Therefore, to some extent, one could predict which centrality is more advantageous given some knowledge on the type of point-set being registered.

5.4.1 Registration under Similarity Transformation

Figure 5.4 shows the registration of the fish[2] dataset with 91 points under similarity transformation. The blue circles belong the point-set $\mathbf{Y}$ being aligned at each iteration towards the point-set $\mathbf{X}$ in red stars. The rows display the 3^{rd}, 6^{th}, and 9^{th} iteration of registration. Closeness and eigenvector variants obtained good results at the 6^{th} iteration, which are better when compared with CPD at the 9^{th} iteration.



(a) Rigid

Figure 5.3: Convergence of the registration under similarity transformation. All centrality values obtained a faster decay than the original CPD algorithm. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

[2] http://sites.google.com/site/myronenko/research/cpd

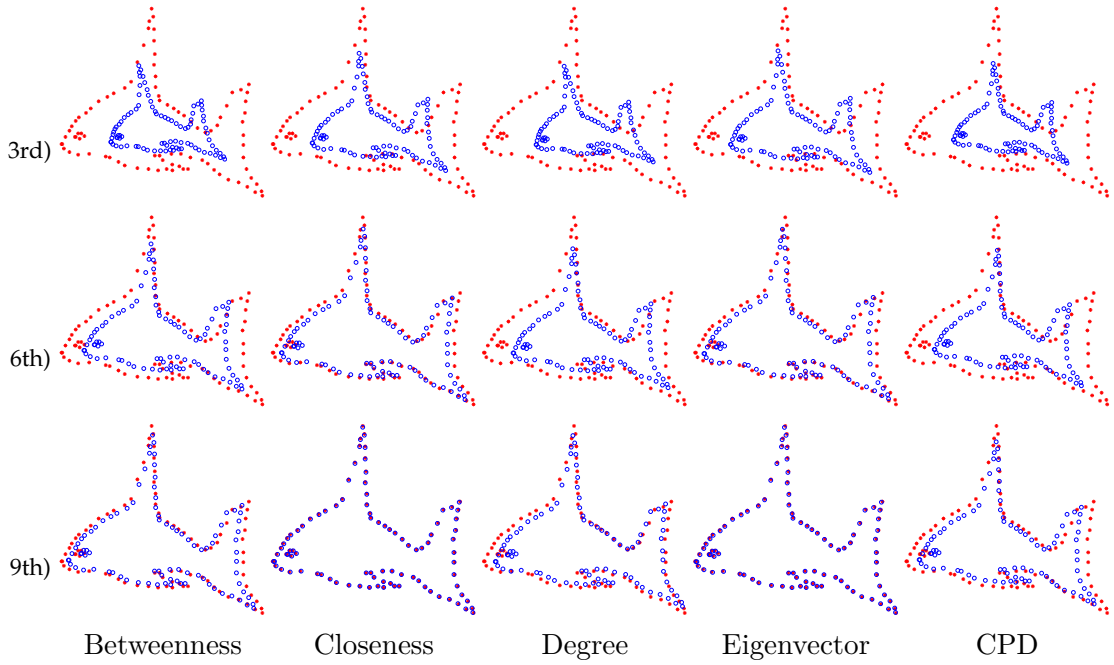


Figure 5.4: Registration of the fish point-set under similarity transformation. The first row shows results at the 3^{rd} iteration, second row at 6^{th} iteration, and third row at 9^{th} iteration. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

Our GPD variant based on the closeness centrality converges at the 9^{th} iteration while CPD needs more iterations to finish the registration. We consider the alignment error as the ℓ_2 norm between the original point set $\mathbf{X}$ and the point set $\mathbf{Y}$ after the recovered transformation: $\|\mathbf{X} - \mathcal{T}(\mathbf{Y})\|_2$. Figure 5.3 shows the alignment error with respect to the number of iterations. The lower the curve decay, the faster the registration. The curve should approach zero in the ideal case. Observing the curves, the GPD variants using closeness and eigenvector centralities obtained the quickest decay, while the degree and betweenness decayed more slowly compared with them, although all centrality variants are still faster than CPD.

The fast alignment using centrality indicates that this topological information might help the algorithm converge faster. As we expected before when we derived the probabilistic estimation, the algorithm was pruning unfeasible points (based on how wide the Gaussian is). The algorithm was able to register faster in this similarity example. Nevertheless, if such a prior is not correct, i.e. we end up pruning points which should not be pruned, the algorithm might fail to converge. For now, we notice that under similarity registration (rigid + isotropic scale) without noise, our centrality variant performed faster than CPD in the number of iterations. In the next sections, we shall vary the type of registration we perform.

5.4.2 Registration under Affine Transformation with Missing Points

The previous experiments were based on a 2D dataset. Now, we will compare the centralities on a 3D face dataset[3] with 392 points under an affine transformation and missing points. Each data set has different missing points which makes the registration more challenging.

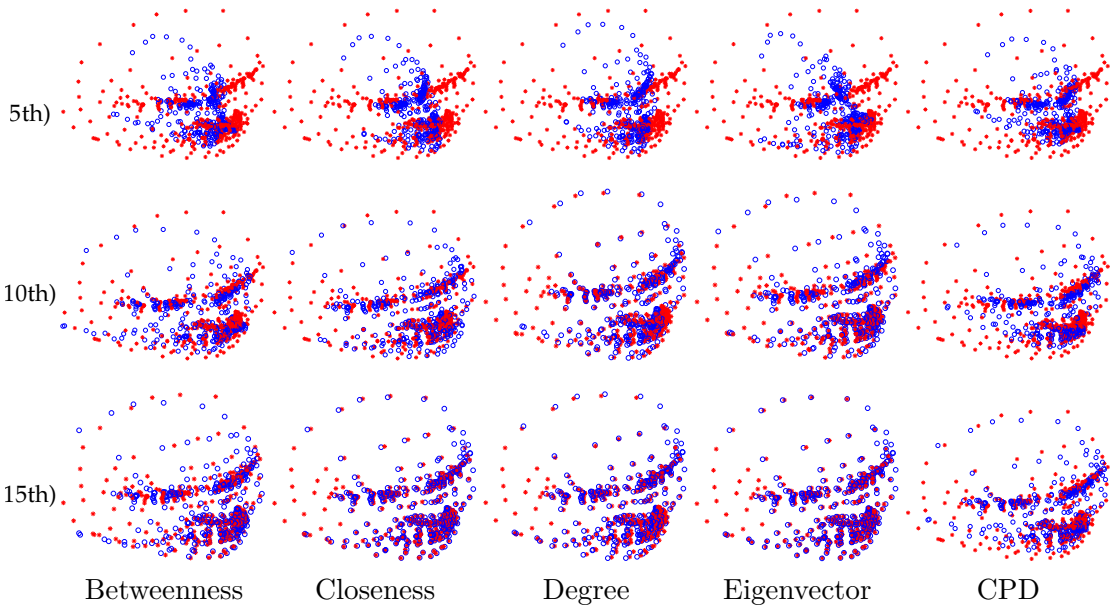
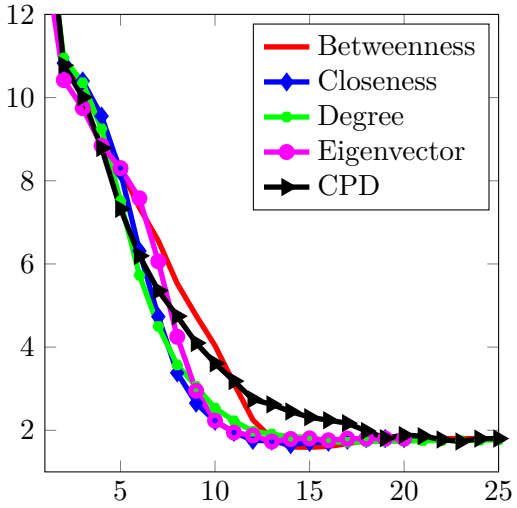


Figure 5.5: Registration of 3D point sets under affine transformation with missing points. First row shows results at the 5^{th} iteration, second row at 10^{th} iteration, and third row at 15^{th} iteration.

Fig. 5.5 shows the registration of the datasets. Notice in the last row that some blue points are missing in the left part of the face (only red points exist). The same happens on the other side of the face (missing red points and only blue points exist). Although it is harder to visualize how well registered the points are, we see that closeness centrality was able to align the point-sets well in the third row, while CPD needs some more iterations to finish the registration.

Fig. 5.6 displays the alignment error in the same way we did for the similarity example. In this example, some centralities decayed faster than CPD. Looking at the curves at iteration 10, many centralities had lower error than CPD, while CPD reaches this error around iteration 20, therefore, taking twice as many iterations to reach that error level. This is a more challenging example than the fish dataset, due to the higher number of dimensions but also due to the missing points.

[3] http://sites.google.com/site/myronenko/research/cpd



(b) Affine + Missing Points

Figure 5.6: Convergence of the Affine Registration. Many centralities approached the final error at the 10th iteration while CPD reached that value around 20th iteration. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

5.4.3 Non-Rigid Registration with Noise and Missing Points

Finally, the last scenario for the first batch of experiments is the non-rigid case with spurious nodes and missing points. Fig. 5.7 shows again the fish dataset under such type of transformation. One of the fishes does not have the dorsal fin, which is a very distinguishable part which could help the registration. This type of transformation is particularly difficult for all algorithms and the decay is presented in Fig. 5.8.

We notice in this experiment that most centrality values performed poorly compared with CPD, except the closeness centrality. The closeness centrality had initially a faster decay than CPD, until 20th iteration when CPD takes over, and later at 50th iteration, closeness takes the lead again. Betweenness and degree centrality did not converge. In fact, the degree incorrectly matched the dorsal fin of one fish to the caudal fin of the other. Such deformation would not be possible in the rigid case, but in this example we allowed non-rigid deformations. The cause for failure of the degree centrality is due to the fact that the prior provides wrong information which is highly associated with the data-graph.

Centralities are defined on a graph data structure. Therefore, the way the graph is built plays an essential role for the success of registration. When the Delaunay triangulation assigns many edges to a noisy point, this node will become more central and therefore, it will jeopardize the registration. We will come back to this issue later in

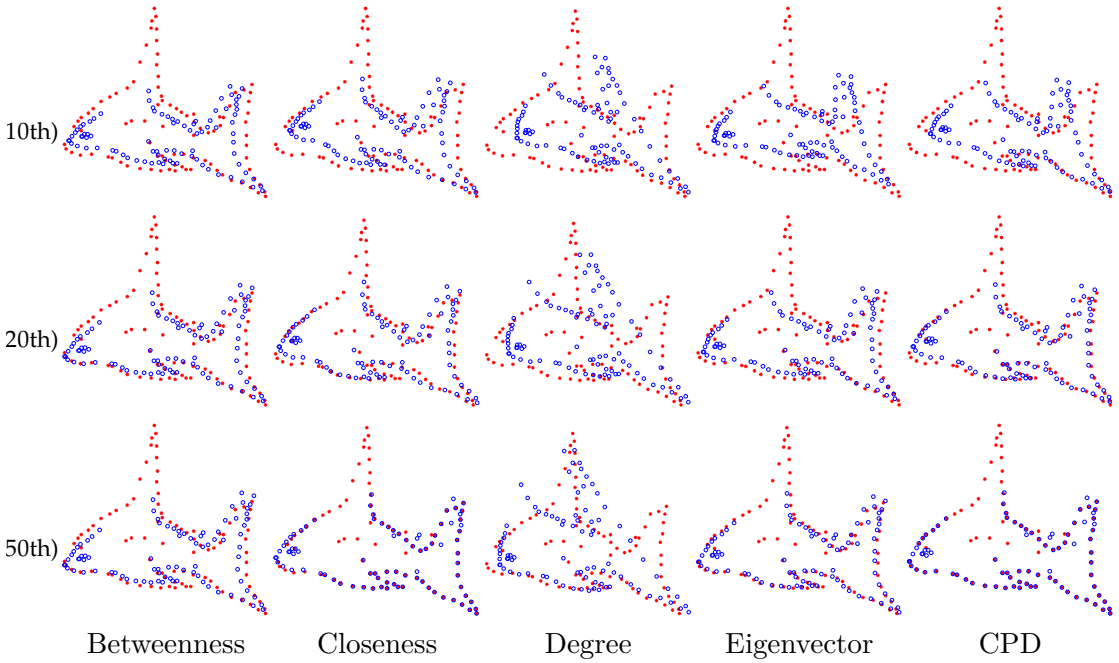


Figure 5.7: Registration of two 2D point sets under non-rigid transformation with outliers and missing points. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

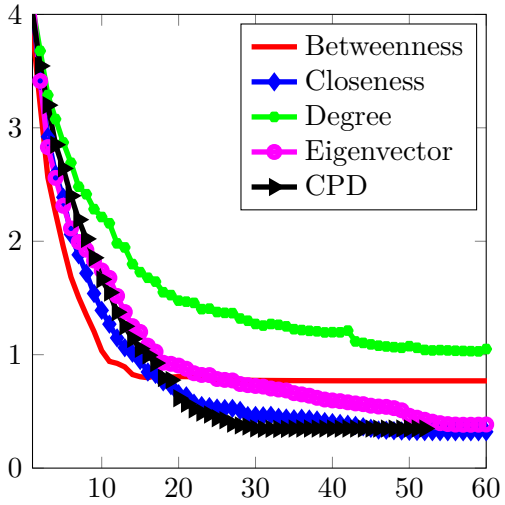
the next sections when discuss about data-graph creation.

5.4.4 Discussion

After those first three initial examples of registration, we can gather some intuition on the behaviour of centrality-based registration. The success and quality of the registration is associated with several aspects:

- The type of centrality measure,
- The technique for building the data-graph, and
- The peculiarities of each point-set, such as elongated structures, sampling, density, etc.

We already observed in Chapter 4, when we performed the centrality-based learning, that the type of centrality played an important role for the classification. This is also the case in the registration problem, since we could briefly observe that the degree obtained the worst performance and the closeness the best one. We applied the Delaunay triangulation for building the data-graph. Hence, we should expect that sets with noisy and missing points will not have significantly better performance, or even worse, as the



(c) Non-rigid + Noise + Missing Points

Figure 5.8: Convergence of the non-rigid registration case. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

prior might be wrong due to the triangulation, it can cause failure in the convergence as it happened for some centrality values. Figure 5.9 illustrates the problem associated with the Delaunay triangulation. The first two graphs show possible ambiguous configurations, but the third one shows an abrupt change in topology by the addition of a single noisy point.

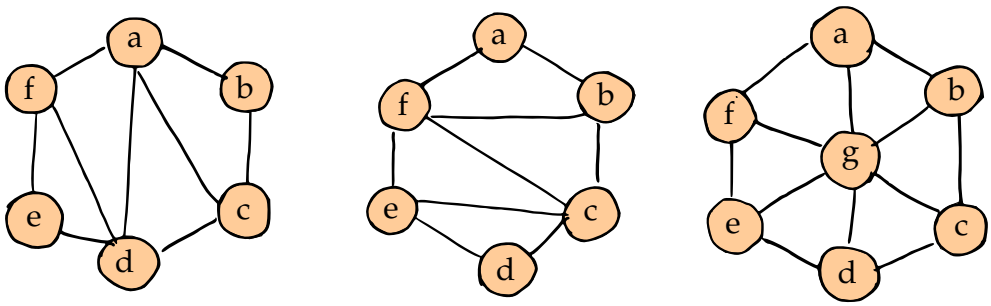


Figure 5.9: Stability of Delaunay Triangulation. A single misplaced node changed the topology of the graph abruptly. Figure a) and b) show two possible triangulations for the same point set and Figure c) shows the sensitivity of the triangulation when one node was added.

There are many options for data-graph construction. One could build the Gabriel Graph [44] to as a data-graph representation. A data-graph could also be built using a TSP or a MST formulation. An alternative which is closely related to the Delaunay triangulation is the α-shapes proposed by Edelsbrunner et al. [36], which are collections of subsets of Delaunay triangulation capturing the notions (fine, crude) of shape in point-sets. In Chapters 6 and 7 we also propose two different techniques to generate the data-graph, but it is still an open question to decide which technique is the most suitable one and we plan to investigate this in our future work (Chapter 8). The reason for this lies on the fact that the concept of noise in a point-set is context dependent. We mentioned noise in the fish example because we knew that true points were sampled from an object's surface and they should lie on the silhouette. If no information about the point-set is given and the algorithm is asked to create a graph robust to noise, it will be a challenging task to decide what noise actually is in order to be able to reduce its impact. We will be moving towards that question in our future work. Chapter 7 minimizes the impact of noise on the data-graph in comparison with Delaunay triangulation.

5.4.5 Experiments on Different Classes of Point-sets

We need to perform extensive experiments in the noise free scenario in order to be able to understand the impact that the class of object has in the registration using centralities, since we already know that a noisy dataset will not improve the results of the centralities. We fix the data-graph construction (Delaunay) and noise scenario. Therefore, we can vary properties of the point-sets and evaluate how those properties impact the results.

We provide five different classes of point-sets. For each class, there are 99 distinct point-sets and we aim to see if the registration behaves similarly per class and algorithm. Figure 5.10 shows samples of those five classes: (i) Gaussian clusters, (ii) random points, (iii) Gaussian sampling, (iv) grid sampling, and (v) contour of objects. The images of *Kimia's 99* dataset were used to generate the silhouette and the different sampling images. This is the same dataset used in Chapter 4 for the centrality-based learning.

Figure 5.10: Sample point-sets of 495 database used. Each image represent one sample of each class. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

Tables 5.1 and 5.2 show the average and standard deviation of registration on those 495 point-sets respectively. The results are displayed on the number of iterations until

convergence per class and algorithm. Results are calculated using only the registrations that succeeded to converge. The convergence of each algorithm is displayed in Table 5.3. On the first type of data, which are clusters, all centralities obtained a faster decay than CPD. In fact, the closeness centrality was in average 2.66 times faster than CPD. Among the 99 sets of this class, the best result was approximately 3 times faster than CPD. We measure the speed of registration by the number of iterations the algorithm took to converge. When analyzing the standard deviation for both degree and eigenvector centralities for this class, it is noticeable that they are really high in comparison with the other algorithms. Also, degree and eigenvector were the only algorithms not able to converge on all databases for this class.

Database	**B**(μ)	**C**(μ)	**D**(μ)	**E**(μ)	**CPD**(μ)
cluster	42.72	**22.72**	53.67	41.95	60.56
random	19.94	15.20	30.46	**14.56**	37.25
normal	36.72	**22.08**	50.45	37.20	51.54
contour	23.82	**17.68**	28.49	20.49	26.39
grid	32.60	26.31	31.23	**24.71**	34.29

Table 5.1: Average on the registration of 495 datasets. Each cell shows the average for the number of iterations until the algorithm converged. In general, Closeness centrality obtained the fastest convergence among all algorithms. The fastest algorithm in average is highlighted in bold. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

Database	**B**($\pm\sigma$)	**C**($\pm\sigma$)	**D**($\pm\sigma$)	**E**($\pm\sigma$)	**CPD**($\pm\sigma$)
cluster	4.36	**1.26**	19.89	17.15	3.91
random	1.94	1.21	15.97	**0.96**	3.90
normal	9.33	**3.37**	20.47	18.22	13.82
contour	3.74	**3.66**	5.03	8.99	4.27
grid	18.39	17.28	21.08	13.41	**13.05**

Table 5.2: Standard deviation on the registration of 495 datasets. Each cell shows the standard deviation for the number of iterations until the algorithm converged. In general, Closeness centrality obtained the fastest convergence among all algorithms. The fastest algorithm in average is highlighted in bold. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

Database	**B**	**C**	**D**	**E**	**CPD**
clusters	100%	100%	81%	86%	100%
random	100%	100%	55%	100%	100%
normal	100%	100%	75%	86%	100%
contour	100%	100%	90%	95%	100%
grid	95%	69%	35%	77%	100%

Table 5.3: Results on convergence of 495 point-sets under Rigid transformation. Degree and eigenvector centralities did not succeed to converge on all sets. However, when the point set comes from grid sampling, no centrality was able to converge 100%. Reprinted from Pattern Recognition, 48/ 2, Samuel de Sousa and Walter G. Kropatsch, Graph-based point drift: Graph centrality on the registration of point-sets, Pages 368–379, Copyright 2015, with permission from Elsevier.

For the random class, all centralities obtained faster convergence than CPD, but the degree was this time the only algorithm that did not converge on all sets. Eigenvector and closeness centralities were the fastest ones, more than 2 times faster than CPD in average. The eigenvector algorithm was quite stable for this sort of data since its standard deviation was quite low compared with the previous class. The next class evaluated is the contour of an object. We extracted the contour of images from the Kimia dataset in order to create the contour-based point-set. Closeness centrality was again the fastest algorithm to converge and we notice that the discrepancy between the results is not as high as in the other classes. The degree and eigenvector were unable to totally converge and this time the average convergence time for the degree is higher than CPD.

Finally, we sample the object using grid-based strategy. Notice that inside the object, all nodes will have the same degree, while they will differ on the boundaries. This was the only case in which no centrality was able to converge 100%. If one, for instance, create the graph out of an image using an 8-connected neighborhood, all points inside the object are likely to be matched, if one considers only the degree. Therefore, when the point set is grid sampled, it is better to rely on CPD which considers geometry of the points. It is important to mention that one could add a free parameter (e.g. $\kappa A + (1 - \kappa)B$) into Eq. 5.2 and balance the importance of each term A and B in the equation (geometry vs. topology). One could always obtain a better or equal results to CPD by crossing validating κ. In our experiments, we decided not to add such a free parameter to avoid dependence on how well κ was tuned. For instance, in the last grid experiment, one could set $\kappa = 1$ and turn off the contribution of the centralities. In fact, by letting both terms equally contribute, we can understand the behaviour better. So, by isolating certain properties of the point-sets, we now have a strong indication of when the centralities can perform better or worse than CPD.

5.5 Conclusions

In this chapter, we have presented a centrality-based approach coupled with the CPD algorithm. We have shown how the probabilistic approach was derived, the types of experiments conducted and the results obtained. In the data we analyzed, many points had the same centrality, especially when looking at the grid sampling. You can refer to the images of Chapter 4 when we displayed the distribution of centrality per shape. For instance, all points inside the object in the degree image have the same centrality value (e.g. 8 for an 8-connected neighborhood). When sampling using the degree, the distribution will have a high peak in number of neighbors (e.g. 8) and smaller peaks for the points in the boundary.

Eigenvector and closeness centralities are the most discriminative distributions under grid sampling, although there are many nodes with similar centrality values, which are concentric inside the shape. Such property would explain why the eigenvector centrality and closeness obtained the best results in the grid experiment, although all centralities suffered from the same convergence problem. Therefore, when performing registration of grid sampled points, centralities still can produce good results but it is clear that the impact on the registration should be reduced via a free parameter.

CHAPTER 6

The Minimum-Weight Maximum-Entropy Problem

In the previous chapter, we brought up the importance of the data-graph in the problem of registration. We saw that many approaches use the Delaunay triangulation as their data-graph technique [25, 27, 76]. It is not clear, though, if the Delaunay triangulation is always the best solution for all possible tasks in CV when a data-graph is required. In this chapter, we focus solely on the problem of creating a data-graph by approximating the degree centrality of the graph by a uniform distribution.

A common procedure for registration consists of (i) creating the data-graph using the Delaunay triangulation and (ii) performing the registration using an optimization procedure ([25, 76, 27]. We applied this triangulation to build the graph in the previous chapter. The Delaunay triangulation [33] is based on the condition that no other point should lie inside the circumcircle of any triangle. Alternative methods include the Reeb graph [98] which builds the graph based on a height function, the Gabriel Graph [44], which is a subgraph of the Delaunay triangulation, or even the Euclidean Minimum Spanning Tree (EMST), since it could be employed as a data-graph. Many researchers propose methods of data-graph construction focusing either on aesthetic aspects [89] or designing their own criteria, such as the fan-shaped triangulation of Lian and Zhang [72].

We would like to generate a data-graph which would ease the registration process. Instead of creating a generic graph (such as the Delaunay) and applying heavy optimization algorithms to match the two graphs, we would rather spend more energy on designing a unique data-graph representation that would make the registration process as trivial as possible. Developing a more exclusive representation for a graph is not a new concept and it has been explored before by Dickinson et al. [34] where they focus on a class of graphs with unique representation for the node labels. They develop a representation ρ of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \alpha, \beta)$, where α and β stand for functions assigning labels to the vertices and edges respectively. They can find if two graphs g' and g'' are isomorphics by comparing their representations $\rho(g')$ and $\rho(g'')$ in $O(N^2)$.

The work presented in this chapter differs in nature from the work of Dickinson et al. [34] since we are not interested in finding isomorphism between two graphs. Many isomorphisms are still possible between two point-sets when we can decide how we are going to build the graphs. So, it would still be difficult to generate graphs without ambiguous solutions. We would like to reduce the amount of ambiguous configurations until the only remaining solution corresponds to our matching. In essence, we try to maximize the entropy of the degree centrality of the graph while minimizing the total weight of the edges. We pose an optimization problem (called the MWME) and we provide a polynomial time algorithm based on DP. To the best of our knowledge, we are the first ones to tackle the matching problem in this fashion.

The results of this chapter were published in the proceedings of the Graph-based Representation in Pattern Recognition in Beijing, 2015 [28]. We have obtained permission from Springer and Business Media to reprint all content of the paper in this dissertation. The license is attached in Appendix A.

6.1 Problem Formulation

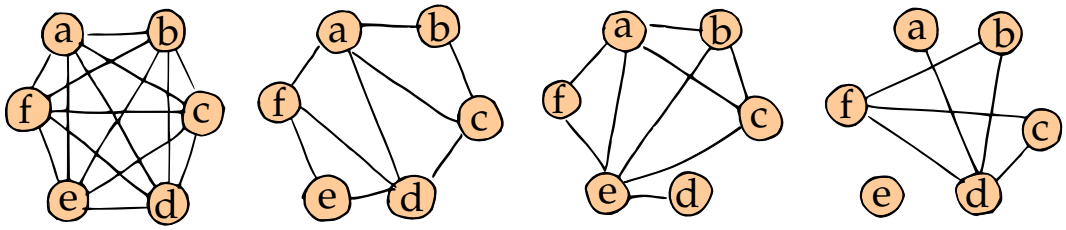
The question to be asked when we try to create a data-graph that can support the registration process is how one can define such a metric, or which property the data-graph would need to have in order to make the registration as easy as possible. Let us examine a regular graph, i.e. a graph in which all nodes have the same degree. When we try to match such graphs, we would notice that all nodes could be matched against all the other nodes, and the registration task would generate many ambiguous solutions. Therefore, if we succeed to build a graph representation which is exactly opposite of the regular graph, the registration process would be alleviated. We are seeking to obtain a graph whose degree distribution is as diverse as possible.

The Shannon entropy (H) measures the uncertainty associated with a random variable. We could use the entropy to assess how diverse the degree distribution $D\{\mathcal{X}\}$ of a graph $\mathcal{X}$ is. We can calculate the entropy of the degree distribution as follows:

$$\mathrm{H}(\mathcal{X}) = -\sum_{\mathbf{v}\in D\{\mathcal{X}\}_{\neq}} p(\mathbf{v}) \log_2(p(\mathbf{v})). \tag{6.1}$$

where $p(\mathbf{v})$ is the probability of finding a node with a degree of $\mathbf{v}$ among all distinct degree values ($D\{\mathcal{X}\}_{\neq}$ stands for the distinct degree values of graph $\mathcal{X}$). A graph $\mathcal{X}$ with high entropy according to Eq. 6.1 would indicate that the distribution has a variability of degree values. The converse is also true, a low entropy means low variability, as in a $\mathbf{k}$-regular graph in which the probability of finding a node with a degree $\mathbf{k}$ is $p(\mathbf{k}) = 1$ and, therefore, $\log(1) = 0$. This also includes the graph without edges.

A graph with a high entropy would let us match nodes more easily. In fact, if all nodes had different degree values, the matching process would be determined as soon as the graph is built since matching nodes must have the same degree. Nevertheless, even if we are able to obtain a graph with the higheset entropy, there is still the problem of ambiguity since our input is a point-set, we can always exchange the order of points



$H(\mathcal{X}_6) = 0$ (a) Complete $\mathcal{X}_6$ $H(L_6) = 0.91$ (b) Delaunay L_6 $H(\mathcal{X}_6^{\bullet}) = 2.25$ (c) NHDD $\mathcal{X}_6^{\bullet}$ $H(\mathcal{X}_6^{\circ}) = 2.25$ (d) NHDD $\mathcal{X}_6^{\circ}$

Figure 6.1: Some realizations of a six vertices graph along with their entropy. Graph (a) is a complete graph, Graph (b) is a triangulation and Graphs (c) and (d) fulfill the NHDD property with the highest entropy values.

and obtain other graph isomorphims, which is not what we want. Thus, we also need to generate a graph as unique as possible. We would like to uniquely identify the nodes unless all points in the graph are equidistant, i.e. many possible solutions still exist. If this is not the case, we should be able to match the graph. In order to achieve this goal, we minimize the total weighted edge cost of our graph when we constrain the solution to have maximum entropy in a graph.

Definition 37 (The Minimum-Weight Maximum-Entropy Problem)**.** *The Minimum-Weight Maximum-Entropy MWME is the problem of estimating an edge-induced subgraph whose entropy of the degree distribution is maximum and the total edge weight is minimum.*

Given a point set $\mathbf{X}$, we create a complete graph $\mathcal{X}_{|\mathbf{X}|}$ using a metric[1] function (Def. 12) as the edge weights. Let W denote the weighted edges of $\mathcal{X}_{|\mathbf{X}|}$. We define a binary vector $U \in \{0,1\}^{|W|}$ that induces an edge-subgraph $\mathcal{X}[U]$ composed of all nodes of $\mathcal{X}_{|\mathbf{X}|}$ and edges $\{W_i | U_i = 1\}$. We search for the vector U which minimizes the cost:

$$\begin{aligned} \underset{U}{\text{minimize}} \quad & \sum_{i=1}^{|W|} W_i U_i, \\ \text{subject to} \quad & \mathrm{H}(\mathcal{F}) \leq \mathrm{H}(\mathcal{X}[U]), \forall \mathcal{F} \\ & U \in \{0,1\}^{|W|}. \end{aligned} \tag{6.2}$$

under the constraint that the entropy $\mathrm{H}(\mathcal{X}[U])$ of the induced subgraph $\mathcal{X}[U]$ is maximum, i.e. for any graph $\mathcal{F}$, where $|\mathcal{F}| = |\mathcal{X}|$, the entropy $\mathrm{H}(\mathcal{F})$ will be lower or equal to our edge induced subgraph $\mathrm{H}(\mathcal{X}[U])$. The second constraint states that the optimization variable is discrete, we either add the edge W_i to our graph $\mathrm{H}(\mathcal{X}[U])$ when $U_i = 1$ or we do not add such an edge ($U_i = 0$). We propose a DP algorithm that minimizes Eq. 6.2 by looking deeper into some properties of the desired induced graph $\mathrm{H}(\mathcal{X}[U])$. It is

[1] e.g. Metric functions were discussed in Chapter 2.

important to mention that we cannot guarantee unique solutions. The reason for that can be visualized in Fig. 6.1a. By constructing a graph out of a regular polygon, we could rotate all nodes and the total edge cost would remain the same as well as the entropy.

6.2 The Near Homogeneous Degree Distribution

We decompose the optimization problem of Eq. 6.2 into the entropy maximization and the edge cost minimization steps. The manner we pose the problem enforces that a solution only belongs to the feasible set *iff* it has the maximum entropy. In this section, we explain how we can guarantee the maximum possible entropy for a graph and later how to incorporate the minimum edge cost in that graph.

Our pursuit for the graph with maximum entropy starts with the question of what is the maximum number of nodes in a simple graph (Def. 2) with distinct degree values. This problem is closely related to the graph realization problem [37] which consists of determining if such a sequence of degree values is feasible for a simple graph (called graphic sequence). This problem has been addressed by Erdős-Gallai [37] and Havel-Hakimi [53, 56]. Thus, our problem could be considered as generating the graphic sequence with highest variability. Theorem 2 states the maximum variability in a graph.

Theorem 2 (Maximum Number of Distinct Degree Values)**.** *For every simple graph* $\mathcal{X}(\mathcal{V}, \mathcal{E})$ *with* $|\mathcal{V}| > 2$*, there are at least two nodes with the same degree.*

The proof for Theorem 2 is based on the proof presented by Kocay and Kreher [63].

Proof. A simple graph $\mathcal{X}(\mathcal{V}, \mathcal{E})$ does neither allow parallel edges nor self-loops. Therefore, the highest degree is $|\mathcal{V}| - 1$. If we create a degree sequence in which all nodes have different degree values (the highest possible variability), this would mean that the nodes would have a degree distribution $D\{\mathcal{X}\} = (0, 1, \ldots, |\mathcal{V}| - 1)$. If one node has a degree of $|\mathcal{V}| - 1$, it means it is connected to all other nodes, but the degree zero means that there is a node not connected to any other. Thus, degree values of 0 and $|\mathcal{V}| - 1$ are mutually exclusive and cannot coexist in the same graph. As there are only $|\mathcal{V}| - 1$ values for $|\mathcal{V}|$ nodes, by the pigeon hole principle, at least two nodes must have the same degree. □

Theorem 2 establishes an important result for us, since we know that we cannot create a graph whose nodes have all distinctive degree values, the highest possible number of different node degrees is $|\mathcal{V}| - 1$ (Def. 38).

Definition 38 (Near Homogeneous Degree Distribution)**.** *A simple graph* $\mathcal{X}(\mathcal{V}, \mathcal{E})$ *fulfills the NHDD property when it contains* $|\mathcal{V}| - 1$ *nodes with different degree values.*

When we analyze the constraint on our object function, we can state that the solution is only feasible if the induced subgraph fulfills the NHDD property. We show that there are two feasible graphs that fulfill NHDD.

6.2.1 Graphs fulfilling NHDD

Definition 39 (Types of NHDD Graphs)**.** *$\mathcal{X}_N^\bullet$ is a connected graph which fulfills the NHDD property and $\mathcal{X}_N^\circ$ is a graph which fulfills the NHDD but it contains one isolated node.*

In Lemma 2, we show that the complement of a graph which fulfills the NHDD property also obeys the NHDD.

Lemma 2 (The complement graph fulfills the NHDD condition)**.** *The complement graph of $\mathcal{X}_N^\bullet$ also fulfills the NHDD property.*

Proof. The complement graph $\mathcal{X}_N^\circ = \overline{\mathcal{X}_N^\bullet}$ is obtained by taking the difference of $\mathcal{X}_N - \mathcal{X}_N^\bullet$. A $\mathcal{X}_N^\bullet$ has distribution equal to $D\{\mathcal{X}_N^\bullet\} = (1, \ldots, |\mathcal{V}| - 1)$. A complete graph $\mathcal{X}_N$ has all nodes with degree equal to $|\mathcal{V}| - 1$. Therefore, the degree distribution of $\mathcal{X}_N - \mathcal{X}_N^\bullet$ will be $D\{\mathcal{X}_N^\circ\} = (|\mathcal{V}| - 2, \ldots, 0)$ with one isolated node and maximum degree of $|\mathcal{V}| - 2$. □

Lemma 3 and Algorithm 6.1 show how we can generate the two possible graphs ($\mathcal{X}_N^\bullet$ and $\mathcal{X}_N^\circ$) described in Def. 39, and more importantly, establishes a relation between the two graphs.

Lemma 3 (Construction of graphs fulling NHDD)**.** *For any integer $N \geq 3$, it is possible to generate both $\mathcal{X}_N^\bullet$ and $\mathcal{X}_N^\circ$.*

Proof. We start with a base case of a graph $\mathcal{X}_N^\circ$ with $N = 3$, whose degree distribution (the degree values of all nodes in the graph) is $D\{\mathcal{X}_N^\circ\} = (0, 1, 1)$. The NHDD is already fulfilled. Our inductive step states that this holds up to any number $N \geq 3$. In order to build the solution for $N + 1$, there are two possibilities:

a) $\mathcal{X}_N^\circ$ has an isolated node. Then, by adding a node and connecting it to all the others, we will obtain a $\mathcal{X}_{N+1}^\bullet$ since the degree of all the other nodes will be increased by one and the graph will be connected. The graph with one disconnected node can be obtained using the complement graph: $\mathcal{X}_{N+1}^\circ = \mathcal{X}_{N+1} - \mathcal{X}_{N+1}^\bullet$.

b) $\mathcal{X}_N^\bullet$ does not have an isolated node. Therefore, we add a new isolated node and we obtain $\mathcal{X}_{N+1}^\circ$ in which the NHDD is fulfilled. The connected graph can be obtained using the complement graph: $\mathcal{X}_{N+1}^\bullet = \mathcal{X}_{N+1} - \mathcal{X}_{N+1}^\circ$. □

The `nhdd` algorithm (Algorithm 6.1) guarantees that our induced subgraph has maximum entropy. Figure 6.2 shows a visualization of the algorithm for a graph with six nodes. The graph is colored in blue if it does not fulfill the NHDD condition and in yellow otherwise. Graphs 6.2b and 6.2g are $\mathcal{X}_3^\circ$ and $\mathcal{X}_5^\circ$, and graphs 6.2c and 6.2l are

$\mathcal{X}_4^\bullet$ and $\mathcal{X}_6^\bullet$ respectively. In the proof of Lemma 3, we used the complement of a graph to show that graphs $\mathcal{X}_N^\bullet$ and $\mathcal{X}_N^\circ$ could be built.

Algorithm 6.1: The `nhdd` algorithm: Generating a graph with maximum entropy.

```
   Data: N ≥ 3;
 1 begin
 2   X ← {x_0, x_1};
     // We alternate between the two steps
 3   for i ← 2 to N do
 4     if X is connected then
         // X_N°: Add a node, but do not connect to any other.
 5       X ← X ∪ {x_i};
 6     else
         // X_N•: Add a node, and connect it to all the other nodes.
 7       X ← X ∪ {x_i | x_i ↔ x ; ∀x ∈ X};
 8     end
 9   end
10   return X.
11 end
```

Theorem 3 (Connected Graph fulfilling NHDD)**.** *For any integer number N, it is always possible to obtain a connected graph that fulfills the NHDD property.*

Proof. This proof comes out naturally as a consequence of Lemmas 3 and 2. For any integer N, we will obtain either $\mathcal{X}_N^\bullet$ or $\mathcal{X}_N^\circ$. In case we obtain $\mathcal{X}_N^\circ$, we take the complement graph and we can always obtain a connected graph which is *NHDD*. □

Fig. 6.1 shows four different induced graphs with 6 nodes and their respective entropy values (Eq. 6.1). Graph 6.1a is a complete graph $\mathcal{X}_6$, whose entropy is equal to zero. Graph 6.1b is created using a Delaunay triangulation (L_6) and its entropy is equal to 0.91. Graphs 6.1c and 6.1d are a G_6 and a $\mathcal{X}_6^\circ$ respectively. It is clear that both $\mathcal{X}_6^\circ$ and $\mathcal{X}_6^\bullet$ are more distinctive than L_6 and $\mathcal{X}_6$ and their entropy is significantly higher. The existence of an isolated node did not affect the entropy as the variability in the number of existing nodes is the same, however, as the nodes have the same distance, those solutions are not unique.

Since we have been speaking about the distinctive node degrees which would allow a one-to-one match between graphs, we can also determine which node has the repeated degree value (Lemma 4).

Lemma 4 (The repeated node)**.** *The repeated node in a graph fulfilling the NHDD has degree equal to* $\left\lfloor \frac{N}{2} \right\rfloor$.

Proof. The proof will be by induction. We start with a base case with $N = 4$. Hence, the node degree distribution is $D\{\mathcal{X}_4^\bullet\} = (1, 2, 2, 3)$ and the repeated node has degree

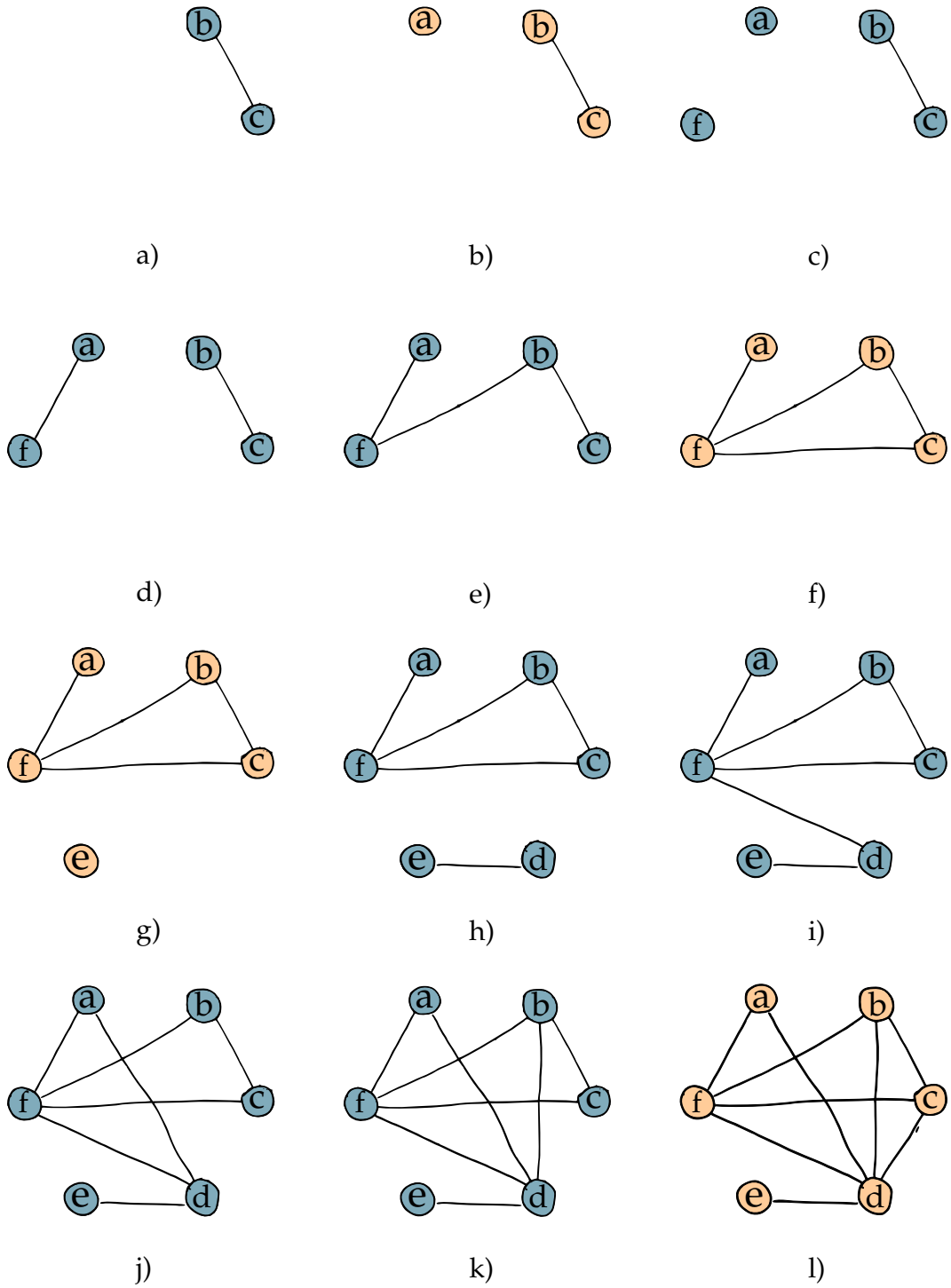


Figure 6.2: An algorithm for building a graph which fulfills the NHDD condition. The graph is colored in blue if it does not fulfill the NHDD condition and in yellow otherwise.

equal to 2. Our inductive step is:

$$D\{\mathcal{X}_N^{\bullet}\} = \left(1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor, \left\lfloor \frac{N}{2} \right\rfloor, \ldots, N-1\right). \tag{6.3}$$

By adding one isolated node as described in Lemma 3, we obtain:

$$D\{\mathcal{X}_{N+1}^{\circ}\} = \left(0, 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor, \left\lfloor \frac{N}{2} \right\rfloor, \ldots, N-1\right) \tag{6.4}$$

The connected graph can be obtained by taking the complement of $\overline{D\{\mathcal{X}_{N+1}^{\circ}\}}$:

$$D\{\mathcal{X}_{N+1}^{\bullet}\} = \left(N, \ldots, N - \left\lfloor \frac{N}{2} \right\rfloor, N - \left\lfloor \frac{N}{2} \right\rfloor, \ldots, 1\right) \tag{6.5}$$

We know that $N = \left\lfloor \frac{N}{2} \right\rfloor + \left\lfloor \frac{N+1}{2} \right\rfloor$, therefore, by plugging it back at Eq. 6.5, we obtain:

$$D\{\mathcal{X}_{N+1}^{\bullet}\} = \left(1, \ldots, \left\lfloor \frac{N+1}{2} \right\rfloor, \left\lfloor \frac{N+1}{2} \right\rfloor, \ldots, N\right) \tag{6.6}$$

□

The number of edges of $\mathcal{X}_N^{\bullet}$ and $\mathcal{X}_N^{\circ}$ can be directly calculated (Theorem 4).

Theorem 4 (The number of edges in a NHDD graph)**.** *The number of edges of graphs* $\mathcal{X}_N^{\bullet}$ *and* $\mathcal{X}_N^{\circ}$ *is equal to* $|\mathcal{E}_N^{\bullet}| = \frac{1}{2}\left(\frac{N(N-1)}{2} + \left\lfloor \frac{N}{2} \right\rfloor\right)$ *and* $|\mathcal{E}_N^{\circ}| = \frac{1}{2}\left(\frac{(N-2)(N-1)}{2} + \left\lfloor \frac{N-1}{2} \right\rfloor\right)$.

Proof. We first prove it for graph $\mathcal{X}_N^{\bullet}$. As proved in Lemma 4, the sum of degrees in a $\mathcal{X}_N^{\bullet}$ is equal to:

$$\sum_{\mathbf{x} \in \mathcal{X}_N^{\bullet}} d(\mathbf{x}) = 1 + \ldots + \left\lfloor \frac{N}{2} \right\rfloor + \left\lfloor \frac{N}{2} \right\rfloor + \ldots + N - 1 \tag{6.7}$$

which is equivalent to:

$$\sum_{\mathbf{x} \in \mathcal{X}_N^{\bullet}} d(\mathbf{x}) = \sum_{i=1}^{N-1} i + \left\lfloor \frac{N}{2} \right\rfloor = \frac{N(N-1)}{2} + \left\lfloor \frac{N}{2} \right\rfloor \tag{6.8}$$

The handshake lemma states that $|\mathcal{E}| = \sum d(\mathbf{x})/2$, we arrive that the number of edges is:

$$|\mathcal{E}_N^{\bullet}| = \frac{1}{2}\left(\frac{N(N-1)}{2} + \left\lfloor \frac{N}{2} \right\rfloor\right) \tag{6.9}$$

By rewriting Equation 6.4, we conclude that:

$$|\mathcal{E}_N^{\circ}| = \sum_{\mathbf{x} \in \mathcal{X}_N^{\circ}} d(\mathbf{x}) = \sum_{i=0}^{N-2} i + \left\lfloor \frac{N-1}{2} \right\rfloor = \frac{1}{2}\left(\frac{(N-2)(N-1)}{2} + \left\lfloor \frac{N-1}{2} \right\rfloor\right) \tag{6.10}$$

□

6.3 Optimization

Let us go back to our initial problem in this chapter: the minimization of the objective function in Eq. 6.2. If we would like to use the NHDD property during our optimization, we would need to prove that a graph fulfilling such property has the maximum possible entropy. The uniform distribution has the maximum entropy [129, 61], however, as already discussed, this cannot be achieved in a graph (Theorem 2). Theorem 5 shows that a graph which fulfills the NHDD has the maximum entropy.

Theorem 5 (Maximum Entropy in a Graph)**.** *A graph fulfilling the NHDD property has the maximum entropy among all graphs with N nodes.*

Proof. Let us suppose that there is a graph $\mathcal{X}'$ with a higher entropy than a graph fulfilling the *NHDD* condition. Our graph has $N-1$ distinctive degree values, therefore, $\mathcal{X}'$ should have N distinctive values in order to have a higher entropy than ours (the uniform distribution). Theorem 2 shows that such a graph does not exist, therefore it is a contradiction, leading that no graph can have higher entropy than the one fulfilling NHDD. □

Theorem 6 calculates the entropy of graphs $\mathcal{X}_N^\bullet$ or $\mathcal{X}_N^\circ$ analytically.

Theorem 6 (The entropy of a NHDD graph)**.** *The entropy of* $H(\mathcal{X}_N^\bullet) = H(\mathcal{X}_N^\circ) = \log_2 N - \frac{2}{N}$.

Proof. In a probabilistic interpretation of the entropy, the $p(\mathbf{v})$ is the probability of an event $\mathbf{v}$ to happen. Our event is the occurrence of a degree $\mathbf{v}$ in $\mathcal{X}_N^\bullet$ (or $\mathcal{X}_N^\circ$). There are $N-2$ events with probability $p(\mathbf{v}) = 1/N$ and one whose probability is $p(\mathbf{v}) = 2/N$. Therefore, the entropy of $H(\mathcal{X}_N^\bullet)$ and $H(\mathcal{X}_N^\circ)$ is:

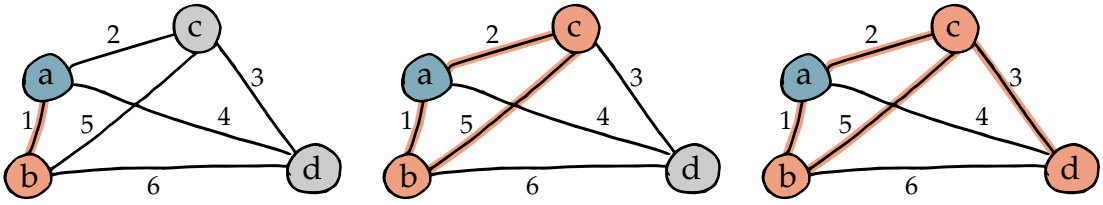
$$H(\mathcal{X}_N^\bullet) = \frac{N\log N}{N} + \frac{2\log N}{N} - \frac{2\log N}{N} - \frac{2}{N} = \log N - \frac{2}{N} \tag{6.11}$$

□

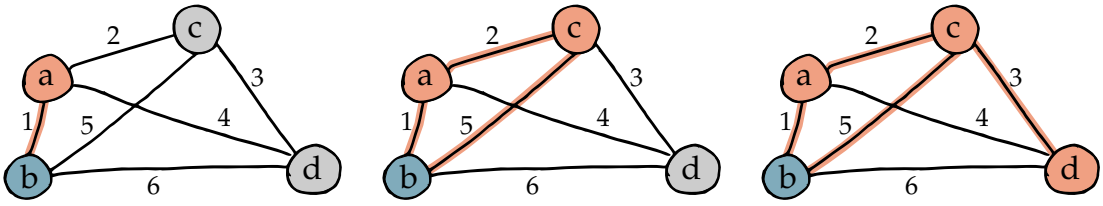
6.3.1 The `n1graph` algorithm explained

Fig. 6.3 shows the optimization process over a complete graph $\mathcal{X}_4$. The edge weights are computed using the Euclidean distance between the points. The `n1graph` algorithm (Algorithm 6.2) contains three nested loops. Each row of Fig. 6.3 displays one iteration of the outer **i** loop. We highlight in blue the current **i** node. For instance, the first row corresponds to **i** = (a), the second row corresponds to node (b), and so on. We start by allocating Matrix *dp* (line 2 of Algorithm 6.2) of dimensions N^3 whose cost is initialized to ∞. This matrix will hold the cost to be minimized.

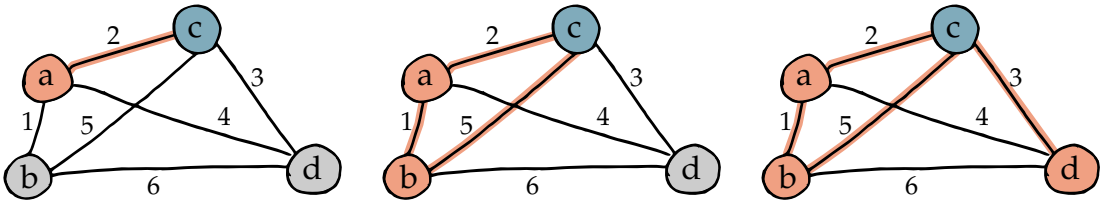
The second nested loop stands for the capacity **k** of the current edge-induced subgraph, which can be visualized by the columns in Fig. 6.3. A node belongs to the induced subgraph if it is highlighted in orange, the edges which were selected based on the cost. In the first iteration, node **i** = (a), therefore, this node is the only node in the



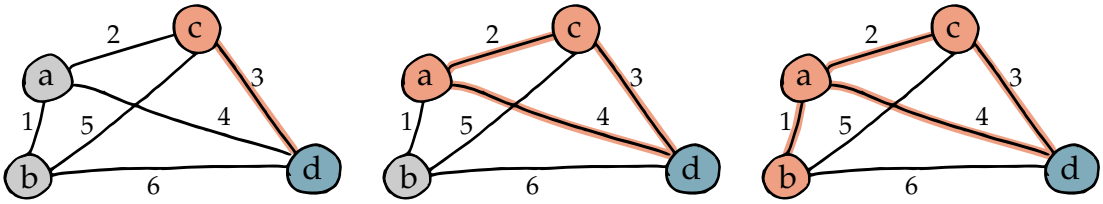
a) i=(a), k=2, c = 1 **b)** i=(a), k=3, c = 8 **c)** i=(a), k=4, c = 11



d) i=(b), k=2, c = 1 **e)** i=(b), k=3, c = 8 **f)** i=(b), k=4, c = 11



g) i=(c), k=2, c = 2 **h)** i=(c), k=3, c = 8 **i)** i=(c), k=4, c = 11



j) i=(d), k=2, c = 3 **k)** i=(d), k=3, c = 9 **l)** i=(d), k=4, c = 10

Figure 6.3: N1graph Optimization. The row represents the reference node (i) which increases from top to down (blue nodes). The column describes the capacity of the current graph (increases left to right). The optimization starts always with two nodes, i.e. the nearest neighbor of the reference node. At every iteration (column) we add one more node to the graph (the one with minimum cost) since the capacity is increased. The optimal solution is the minimum cost of the last column. In this example the cost is 10 at i=d and k=4.

induced subgraph. We increase the capacity $\mathbf{k} = 2$ and add node (b), since it is the one contributing with the lowest cost (Fig. 6.3a). Then, between the two remaining nodes, we choose the one which has the minimum weight. If (c) joins the graph, we add a cost of 7 but if we add node (d) instead, the added cost would be 9, therefore, we choose (c) (Fig. 6.3b). Finally, we need to add the last remaining node since our graph has to be connected (Fig. 6.3c). This is one solution during the optimization and it could be seen as one point in the space of solutions with a certain cost. We produce N solutions iteratively (line 3) and search for the one with the lowest cost. For instance, solution $\mathbf{i}$ will be least as low as $\mathbf{i} - 1$ (line 16).

Algorithm 6.2: The `n1graph` algorithm which generates a $\mathcal{X}_N^{\bullet}$ graph.

Data: Complete graph $\mathcal{X}_N(\mathbf{X}, W)$, $N \geq 3$;

1 **begin**
2 $dp \leftarrow Matrix(N, N, N, value = \infty)$;
// We create N solutions starting with point i (layer).
3 **for** $i \leftarrow 1$ **to** N **do**
4 $dp(i, 1, i : N) = 0; join = 0$;
// k is the capacity of the graph $\mathcal{X}_N^{\bullet}$ (row).
5 **for** $k \leftarrow 2$ **to** N **do**
6 added = false; // j is the node to be added into $\mathcal{X}_N^{\bullet}$ (column).
7 **for** $j \leftarrow 1$ **to** N **do**
8 **if** *join* **then**
9 $c_{new} \leftarrow dp(i, k-1, N) + \sum_{g \in G} W(j, g)$;
10 **else**
11 $c_{new} \leftarrow dp(i, k-1, N) + W(j, \mathbf{x}_{\mathbf{k}-1})$;
12 **end**
// If we have not added a node yet, we must accept.
13 **if not** *added* **then**
14 $dp(i, k, j) \leftarrow c_{new}$;
15 added = true;
16 **else**
17 $dp(i, k, j) \leftarrow \min(dp(i, k, j-1), c_{new})$;
18 **end**
19 **end**
20 $join = \neg join$;
21 **end**
// The new solution i is at least as good as $i-1$.
22 $dp(i, n, n) \leftarrow \min(dp(i, n, n), dp(i-1, n, n))$;
23 **end**
24 **return** `traceback`(dp).
25 **end**

Every iteration of $\mathbf{i}$ could be visualized as one layer of dp matrix and we set the

costs from $\mathbf{i}$ to N equal to zero (line 4), this is the reason why the cost is reinitialized in Fig. 6.3d. The weights lie in the edges, thus, the initial cost is zero since there are no edges in $\mathcal{X}_N^\bullet$ at capacity 1. We have one node ($\mathbf{i}$) in $\mathcal{X}_N^\bullet$ and we start to increase the capacity ($\mathbf{k}$) of our graph $\mathcal{X}_N^\bullet$ (line 6) until all nodes belong to the graph. We are allowed to add one more node ($\mathbf{j}$) whenever the capacity $\mathbf{k}$ is increased. We alternate between two steps, adding an edge to all the nodes currently in $\mathcal{X}_N^\bullet$ (line 8) and adding an edge to the last node ($\mathbf{x}_{\mathbf{k}-1}$) added to $\mathcal{X}_N^\bullet$ (line 10). The algorithm can be visualized as "pushing" the minimum weight towards the right bottom of the matrix, then the optimum for each capacity $\mathbf{k}$ is set at $(\mathbf{k}, N)$ and the optimum for each layer $\mathbf{i}$ is found at $dp(\mathbf{i}, N, N)$. Hence, the term $dp(\mathbf{i}, \mathbf{k}-1, N)$ can be understood as the best cost for the optimization when the capacity was $\mathbf{k}-1$, that's the total weight we will propagate towards the end. When the capacity k is increased, we must add a new node into the graph, therefore, the cost shall increase. We add initially the first node which is not already in the graph (line 14). Finally, on line 15, we take the minimum between the cost of adding the current node j and the previous cost at capacity $\mathbf{k}$ without node j.

Algorithm 6.2 shows only the core steps to obtain the MWME sollution. Nevertheless, more involved steps are necessary to ensure the algorithm will behave correctly. For instance, we need to prevent adding the same node twice into the graph. Every node has to be added only once. Also more involved steps which are

6.3.2 The `traceback` algorithm

Table 6.1 highlights the layer of matrix dp in which the optimum solution was found. The highlighted node (d) consists of the layer which yielded the optimum solution ($\mathbf{i} = 4$). On $\mathbf{k} = 1$, dp is initialized with node (d) $= 0$ (since at this capacity there are no edges in the graph). When capacity is increased to 2, the node (a) is added yielding a cost of 4, but within the same capacity there is a lower cost (3) if node (c) is added instead. The final cost is displayed at cell $\mathbf{j} =$ (d), $\mathbf{k} = 4$. We trace back on the same row until there is a change in cost (i.e. meaning that a node has been added). Whenever a node is added, we proceed the trace from the row with a lower capacity $(\mathbf{k}-1, N)$ and continue tracing back until all nodes are added. The node sequence obtained by tracing back is ((b),(a),(c),(d)).

The `traceback` algorithm (Algorithm 6.3) starts at the right bottom corner (row N), where the minimum is, and goes back on the same line until the cost is changed. This change in cost means that a node was added (cell highlighted in orange). The row means the capacity, when a node at the capacity $\mathbf{k}$ was added, we only need to go to row $N-1$ and search for a change of cost until we reach the first row. The algorithm produces a sequence in which the nodes were added, e.g. ((b),(a),(c),(d)) for Table 6.1. This sequence is used by the `nhdd` algorithm (Algorithm 6.1) to produce the final graph $\mathcal{X}_4^\bullet$.

$k \diagdown j$	a	b	c	d
1	∞	∞	∞	0
2	4	4	$3\ \uparrow$	$\leftarrow 3$
3	$9\ \uparrow$	$\leftarrow 9$	$\leftarrow 9$	$\leftarrow 9$
4	9	$10\ \uparrow$	$\leftarrow 10$	$\leftarrow 10$

Table 6.1: The table storing the cost during optimization. The trace back starts at (d,4) and moves on the same row until a value changes: a node is added to the graph.

Algorithm 6.3: The `traceback` algorithm

Data: Table dp and optimal layer $\mathbf{i}$

```
1  begin
2      S ← {};
3      k ← N;
       // We start at the right bottom corner.
4      while k > 0 do
           // If the cost is different, a node has been added.
5          if j = 1  or  dp(i,k,j) ≠ dp(i,k,j−1) then
6              S ← S ∪ {j};
7              k ← k − 1;
8              j ← N;
9          else
10             j ← j − 1;
11         end
12     end
13     return S;
14 end
```

6.4 Complexity Analysis

In this chapter, we provide three algorithms: `nhdd` (Algorithm 6.1) that produces a NHDD graph given an ordered set of nodes, `n1graph` (Algorithm 6.2) which finds the optimal solution for our MWME problem, and the `traceback` (Algorithm 6.3) which reconstructs the data graph by tracing back the optimum cost from the dp table. In the actual implementation, one is able to store the solution during the optimization and therefore, the tracing back step could be avoided. Nevertheless, we discuss the time complexity associated with all algorithms.

Theorem 7 (The complexity of the `nhdd` algorithm)**.** *The time complexity of the `nhdd` algorithm is* $\Theta(N + |\mathcal{E}|)$.

Proof. `nhdd` has a single loop (line 3) which accounts for the N term, and the cost added at each iteration is associated with each edge created in the graph (line 7). □

Theorem 8 (The complexity of the `n1graph` algorithm.)**.** *The time complexity of the* `n1graph` *algorithm is* $\Theta(N^3|\mathcal{E}|)$.

Proof. The three nested loops of $\mathbf{i}, \mathbf{j}, \mathbf{k}$ yield clearly a $\Theta(N^3)$ time. Inside the $\mathbf{j}$ loop, two operations will alternate: (i) adding one edge to the last node of $\mathcal{X}_N^\bullet$ and (ii) adding an edge to each node of $\mathcal{X}_N^\bullet$, the total operations will be: $(1, 2, 1, 4, 1, \ldots)$, which can be split into $\left\lfloor \frac{N-1}{2} \right\rfloor$ operations of type (ii) and $\left\lfloor \frac{N}{2} \right\rfloor$ operations of type (i). The complexity for those two operations is equivalent to Eq. 6.9, which is the number of edges in the graph, yielding in total $\Theta(N^3|\mathcal{E}|)$. □

Theorem 9 (The complexity of the `traceback` algorithm)**.** *The complexity of the* `traceback` *algorithm is* $O(N^2)$.

Proof. The `traceback` algorithm is bounded by the table size which is N^2. □

6.5 Matching

The matching is performed by bringing each point-set towards this canonical representation. Given point-sets $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_P$, we build the MWME graph for each one of them. The matching associates a node with degree $v \in \mathcal{X}_k^\bullet$ to a node with the same degree $v \in \mathcal{X}_l^\bullet$. Therefore, the matching is performed via this canonical representation. As each point-set is optimized individually, the scale of one does not affect the scale of the other. The optimization tries to minimize the distances within each point-set individually, therefore, it is able to handle the following two scenarios: rigid and similarity transformation (Chapter 3).

We took images from the Caltech-256 dataset which contained a single object in the scene. We sampled $N = 50$ points from the silhouette of the object. A random similarity transformation was applied to the point-sets. Figure 6.4 shows an example of the matching. Not all edges were displayed to avoid cluttering the scene, but the sets are correctly matched.

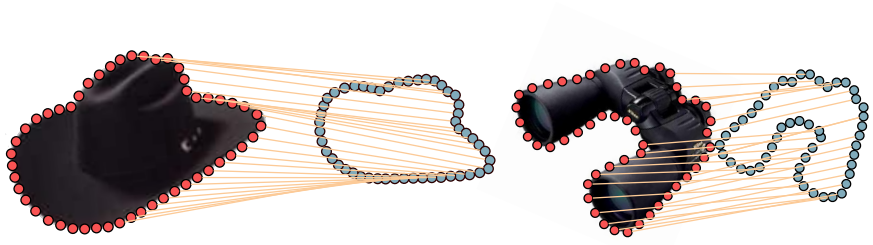


Figure 6.4: Registration of point-sets under Similarity Transformation: translation + rotation + isotropic scaling.

6.6 Conclusions

In this chapter, we proposed a novel way to match point-sets via a canonical representation in which we minimize the MWME problem. Our approach is based on a Dynamic Programming algorithm yielding a cubic complexity solution. Such a graph turns a registration procedure into a trivial task under, for instance, rigid and similarity transformation since there is a direct mapping between nodes.

When the point-sets are not related by a similarity transformation, this method cannot compute the direct matching. Also, due to global optimization, the location of every individual point has a major impact on the cost of the edge-induced subgraph, it becomes sensitive to variations in the local neighborhood of the points. We will cover some of those issues in the next chapter, but as a future work in this topic, we will extend the algorithm to a more local optimization not to be dependent on all nodes, but we will try to match regions of maximal entropy, thus, considering only a subset of (i.e. subset of nodes). This will make allow more flexibility during registration and it will make the process less dependent on this global optimization since we could perform the registration of non-linearly related point-sets by performing piece-wise matchings of a subset of nodes. It will also to be able cope with noise and partially overlapping regions since it is will no longer optimize over the whole graph.

CHAPTER 7

The Minimum Spanning Tree of Maximum Entropy

In the previous chapter, we started dealing with the problem of data-graph construction, which is the process of generating graphs out of unstructured point-sets. This type of graph does not offer topological information from the edges, since the edges we find in data-graphs do not represent actual connections between the original points. They are built according to a certain criteria such as geometrical or even aesthetical ones [89]. The choice of edges is highly dependant on the type of data-graph technique. For instance, the Delaunay triangulation is based on the condition that no other point should lie inside the circumcircle of any triangle.

One possibility for registering point-sets is to formulate the problem as Graph Matching (GM). As we have the freedom to design the data-graph, we are not restricted to use a triangulation. In fact, we could design a graph that would decrease the search space when matching points, i.e. a graph that would be tuned towards a specific application instead of acting as a general stand-alone data-graph construction technique. The Iterative Closest Point (ICP) algorithm was introduced in Chapter 3. As already mentioned, we would need to find for each point in one set the closest point in the other set, requiring us to check the whole set of points to determine which point is the closest one. Nevertheless, if we have some prior knowledge about the desired node, we could decrease this search space by only checking points which match a certain criteria (such as the degree of the node). The number of points we need to check would be greatly reduced in this fashion if this information is reliable.

In this chapter, we modify the cost function presented in Chapter 6 to build a data-graph which can cope with noise in the local neighborhood of points, a more stable graph tuned for registration. We call the problem as the Minimum Spanning Tree of Maximum Entropy (MSTME). By minimizing the total edge weight, we can allow some robustness of the data-graph with respect to small deviations of the points. By maximizing the entropy, we can generate a data-graph which could later ease the registration process.

We evaluate the current approach under different levels of noise and compare the stability of the data-graph with a Delaunay triangulation in several databases.

This chapter is available at arxiv.org, report number 1505.06319 [29]. It has won the best paper award at the 39th Annual Workshop of the Austrian Association for Pattern Recognition (ÖAGM) in Salzburg, 2015, sponsored by the Austrian Computer Society.

7.1 Objective Function

We pose the problem as an optimization procedure in which our cost function is built upon a MST formulation and entropy maximization. In this sense, our work is closely related to [84] as the authors also formulate a Voronoi problem focusing on the entropy of a MST. However, they are dealing a different problem by proposing an entropy estimator for clustering. In our work, we are interested on the data-graph construction problem and our entropy is calculated on the degree distribution of the generated MST while they estimate the data entropy based on the length of the spanning tree. The problem we formulate in this section could also be seen as a bi-criteria optimization problem. Ravi and Goemans [97] proposed a Lagrangian relaxation to solve the constrained minimum spanning tree problem, which is also a bi-criteria optimization proven to be $\mathcal{NP}$-Hard by Aggarwal et al. [3]. Neighborhood search and adjacency search heuristics for the bicriterion minimum spanning tree problem were proposed by Andersen et al. [4].

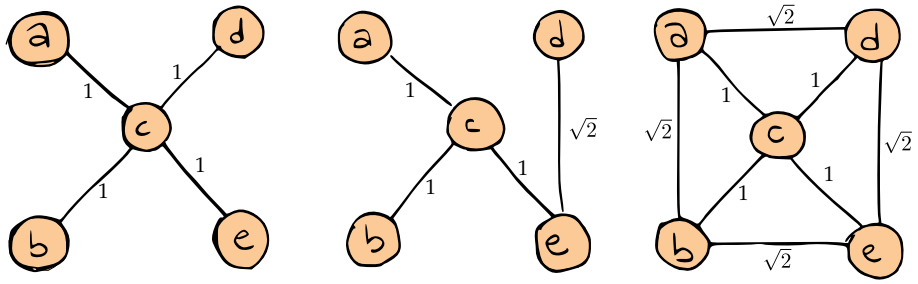
Given a point-set $\mathbf{X}$, we calculate the distance (Section 2.2.2) from each point to all the other points and store those distances in W. This generates a complete graph $\mathcal{X}_N(\mathbf{X}, W)$. We seek to find a vector $U \in \{0,1\}^{|W|}$ in such a way that when $U_i = 1$, we add W_i to our graph. We can formulate our cost function simply as:

$$\begin{aligned} \underset{U}{\text{minimize}} \quad & \sum_{i=1}^{|W|} W_i U_i - \lambda(\mathrm{H}(\mathcal{X}[U])) \\ \text{subject to} \quad & U \in \{0,1\}^{|W|}, \\ & \textstyle\sum U = |\mathcal{X}| - 1, \\ & \mathcal{X}[U] \text{ is connected.} \end{aligned} \tag{7.1}$$

where $\mathcal{X}[U]$ is an edge-induced subgraph from vector U. λ is a parameter which weighs the importance between the two terms: minimizing the total edge and maximizing the graph entropy. The first constraint states that we have a discrete problem, i.e. we either add an edge ($U_i = 1$) or we do not add it. The second constraint is a necessary condition for our graph to be a tree. We need the third constraint to force the graph to be connected.

Figure 7.1 displays three different graphs of the same point-set $\mathbf{X} = \{a, b, c, d, e\}$. The first graph (a) is simply a MST whose cost is $c(\mathrm{MST}, \lambda = 0) = 4$. It could be obtained by setting $\lambda = 0$ and therefore the entropy part vanishes from Equation 7.1. The second graph (b) considers $\lambda = 1$, a higher entropy would be able to decrease the cost of solution in comparison with the MST, the cost is $c(\mathrm{MSTME}, \lambda = 1) = 3.04$. Notice that although

the cost of the weighted edges is higher, the total cost was decreased due to the higher entropy obtained. Finally, we show in contrast that calculating a Delaunay triangulation yields an entropy as low as the MST one.



a) MST
H(MST) = 0.72

b) MSTME ($\lambda = 1$)
H(MSTME) = 1.37

c) Delaunay
H(Delaunay) = 0.72

Figure 7.1: The first graph (a) is obtained with a MST and has cost c(MST) = 4 and H(MST) = 0.72. Graph (b) is obtained using our MSTME. Since H(MSTME) = 1.37, the cost c(MSTME,$\lambda = 1$) = 3.04, hence, c(MST) > c(MSTME). The last graph (c) is a Delaunay triangulation and it has the same entropy as the MST.

7.2 Optimization

Our optimization will be based on a greedy approach evaluating the best cost which could be obtained when adding an edge. We have the constrain that our algorithm has to generate a tree, therefore, it cannot contain cycles. At each iteration, we are allowed to add one more edge until we reach a tree composed of all nodes. We add the node whose cost is minimum regarding our cost function measuring the weight of the edge and the entropy of the new generated MST.

The input is the complete graph $\mathcal{X}_N(\mathbf{X}, W)$ composed of the point-set as the vertices and the edges are weighted according to the Euclidean distance (Section 2.2.2). For each edge e, we check if the subgraph $\mathcal{X}[U]$ induced by our vector U would contain a cycle by the addition of such an edge, in which case, we discard (line 8). Otherwise, we calculate the new cost obtained by the weight of e and we estimate the entropy of the MST (line 13). In line 16, we remove this edge from our graph and continuing searching within the same capacity (i.e. the number of edges allowed at the moment) for an edge which could yield a smaller cost. At the end, we add the edge with the best cost and increase the capacity of our MST by one until it reaches $|\mathbf{X}| - 1$ (line 18).

This algorithm has a greedy strategy, which means, it will make decisions locally, based on the best solution at the moment. Advantages of greedy approaches consist of simpler algorithms with intuitive approach, but it can only converge towards local

minima and cannot guarantee the optimal solution.

Algorithm 7.1: The `mstme` algorithm.

```
   Data: 𝒳_N(X, W), λ;
   Result: 𝒳[U];
 1 begin
 2     U ← {};
       // We can only choose |X| − 1 edges
 3     for i = 0 to |X| − 1 do
 4         cost ← ∞;
 5         best ← 0;
 6         foreach e ∈ E \ U do
 7             if G[U ∪ {e}] has a cycle then
 8                 continue;
 9             else
10                 U ← U ∪ {e};
11             end
12             if W_e − λH(𝒳[U]) < cost then
13                 cost ← W_e − λH(𝒳[U]);
14                 best ← e;
15             end
16             U ← U − {e};
17         end
18         U ← U ∪ {best};
19     end
20     return 𝒳[U];
21 end
```

7.3 Experiments

Our experimental section focus on evaluating the behavior of our algorithm on a point-set sampled from the silhouette of an object. Figure 7.2 shows some point-sets connected using the MSTME. It is noticeable that in thin areas, the entropy term of the optimization function takes over, i.e. the degree values of the points which lie in sich areas are higher than in other regions. We can exemplify this by looking at the top of the fish or the feet of the dog, while other regions follow the silhouette of the object having lower degree value. It is too costly to make an edge cross the shape of the object, since we would have to add the weight of the edge to our cost. Therefore, we can up to some degree predict how the degree of the nodes behave based on their local neighborhood.

Our noise experiment is built upon the shortest pairwise distance in the point-set as ϵ. Then, we perturb each point $x \in \mathbf{X}$ by a length and an angle (l, α), where $\alpha = [0, 360)$, and the radius $l = [0, r\epsilon]$ and r varies from $[1, 10]$, e.g. 5ϵ means that each point of the dataset was perturbed within a radius of 5 times the shortest pairwise distance in the

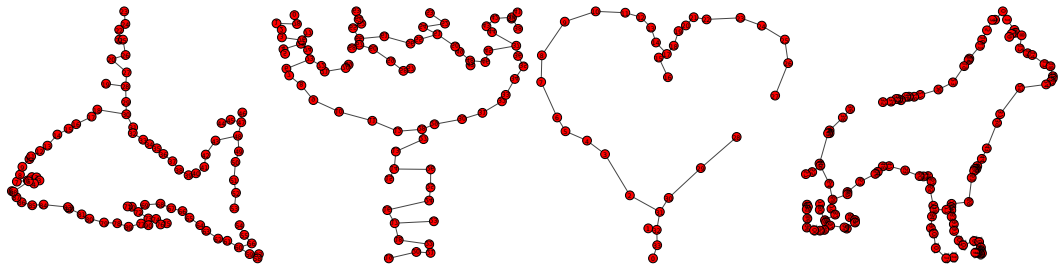
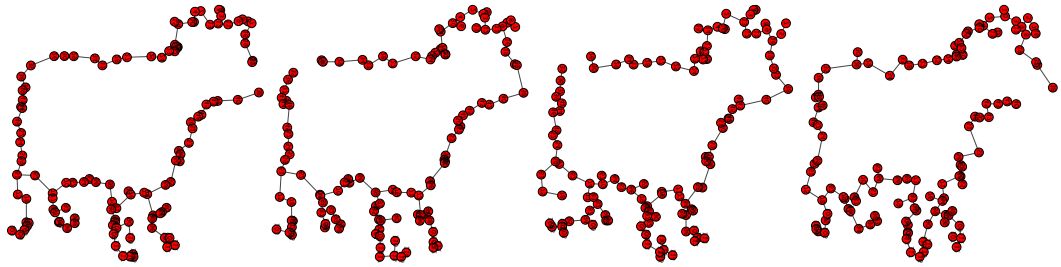


Figure 7.2: Data graphs connected using the MSTME ($\lambda = 0.5$).

graph. Certainly, the higher the noise, the lower the stability of the algorithms will be.



Noise up to 3ϵ Noise up to 5ϵ Noise up to 8ϵ Noise up to 10ϵ

Figure 7.3: Noise added to the point-set.

Figure 7.3 shows the impact of noise in the point-sets varying according to ϵ in a cow silhouette. It would not be an easy task even for humans to match such points, notice how abrupt the change in thin areas (e.g. feet). It is worth mentioning that as the algorithm searches for the optimal, it has to decide on which edges to take, therefore, a simple wrong edge will cause a much higher impact on the results when compared with Delaunay triangulation, which has many more edges.

As we are not allowed to have cycles, the silhouette of the object will "break" in locations where the weights of the edges are high. This location will vary according to the noise. Notice that the split occurred in the face of the cow when noise level was 10ϵ but in level 8ϵ, it happened at a different location. Figure 7.4 shows the quantitative analysis of stability between MSTME and Delaunay triangulation. The way we evaluate the stability is by calculating the percentage of edges which remained stable in all graphs within the same noise level. The first graph shows the results for the fish dataset[1]. The y-axis displays the percentage of stable edges, e.g. 0.8 means that 80% of edges were found across all data-graphs within the same level. For this dataset, our algorithm for the MSTME was always more stable than the Delaunay triangulation up to the highest

[1]The fish dataset was obtained from the CPD [86].

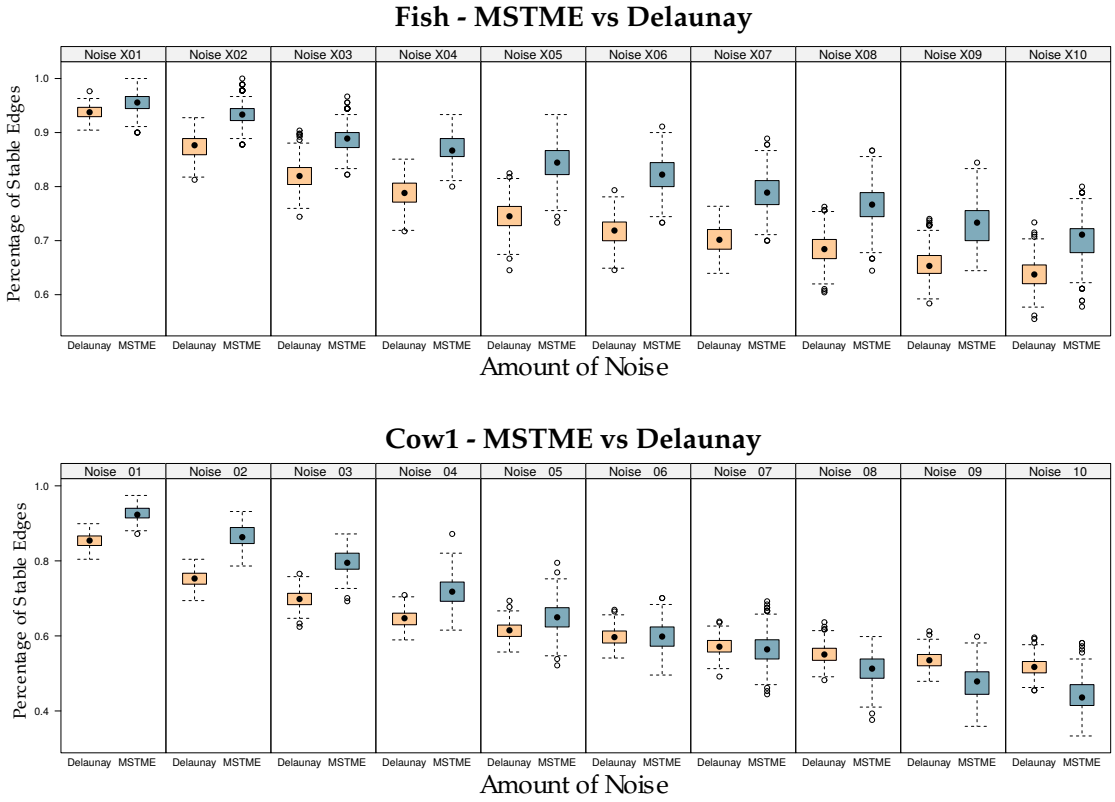


Figure 7.4: Boxplot comparison of the stability of the fish dataset. The noise consists of shifting each individual point by $\epsilon\times$level, varing from 1 to 10. For each noise level, we generated 30 perturbed datasets and compute the number of stable edges accross all results.

noise level. The boxplot displays several other types of information, such as the median, quartiles, etc. The instability observed was found on the thin parts of the fish, such as the dorsal and caudal fins of the fish. The second plot shows the results on the *cow1* dataset[2]

We observed that the MSTME was, in general, more stable than the Delaunay algorithm. Small variations in the location of the nodes affected the triangulation more than it affected the MSTME. As the triangulation generates many more edges, a single wrong edge will have a high impact on the results of the MSTME. Nevertheless, it was still able to obtain higher stability than Delaunay up to a certain level of noise. Figure 7.5 shows examples of the *dog1* point-set perturbed under noise level 3ϵ. It is easy to spot the differences between the data-graphs generated by MSTME samples, although it is hard to visualize the differents of the triangulation, we notice that the edges between the

[2]the cow1 dataset was obtained from the kimia99 dataset [112].

head and the tail of the dog vary on all data-graphs. It is also noticeable a distinction around the head and between the legs of the dog.

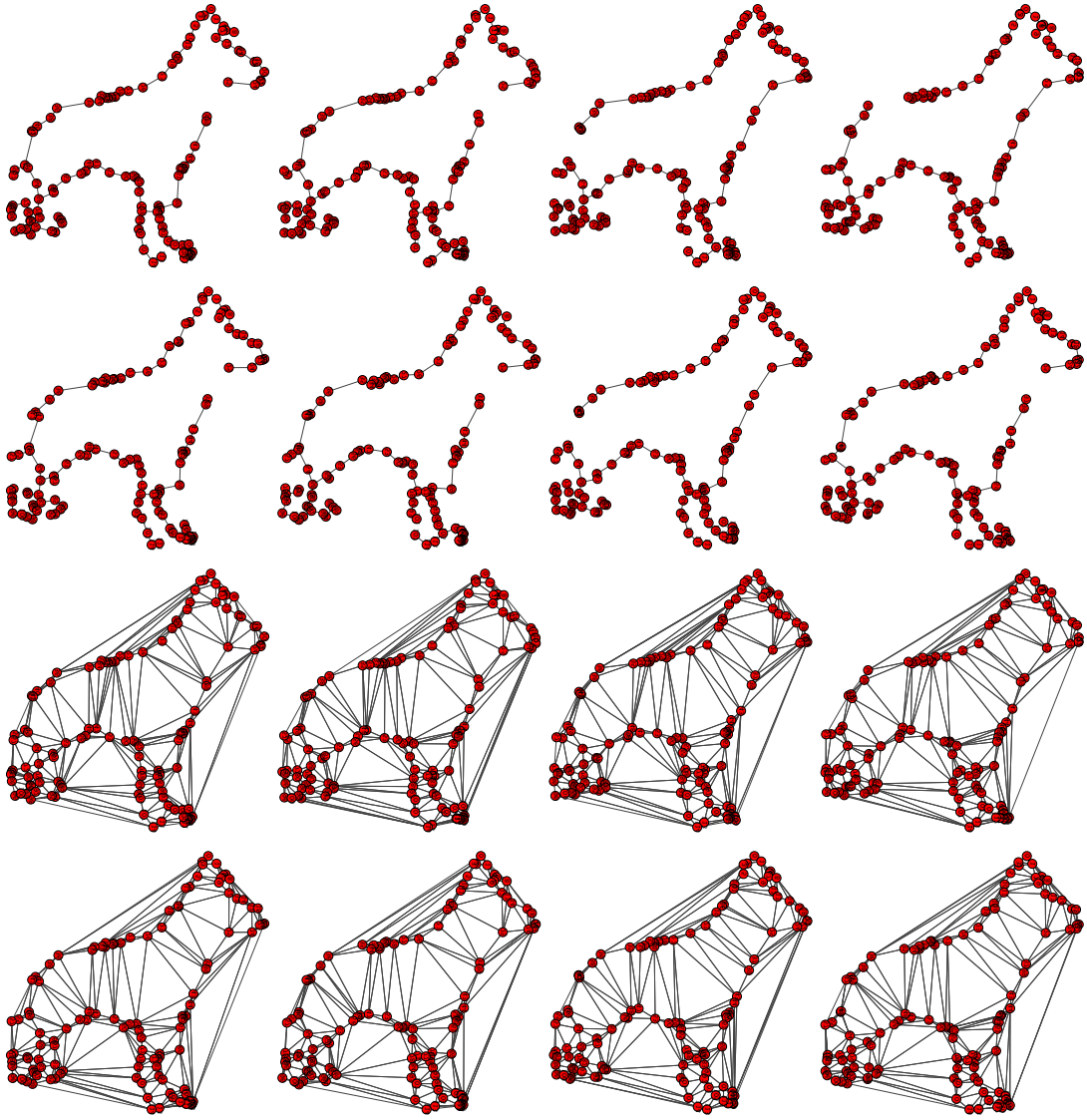


Figure 7.5: Examples of data-graphs with noise equal to 3ϵ

7.4 Conclusions

In this chapter, we tackled the problem of robustness of data-graph construction under local perturbation on the neighborhood of points. We proposed a data-graph technique by

searching for a tree whose cost is minimum regarding the total edge weight and maximum regarding the entropy of the degree distribution. We called this problem the Minimum Spanning Tree of Maximum Entropy (MSTME). The algorithm differs from the one from Chapter 6 since in the former we compute the correspondence of two point-sets by building their data-graph, which could cope only up to a similarity transformation. The proposed algorithm tries still tries to obtain a more unique data-graph (by maximizing the entropy) but it also needs to be stable against local variations (by approximating a MST). The trade-off is controlled by the parameter λ of the algorithm which if could be cross-validated in a specific scenario whose noise level is known.

In point-sets sampled from the silhouette of an object, we observed that points lying in thin areas were often unstable when there was noise. The explanation for this is based on the fact that since those points are close to each other [3], the entropy term of the objective functions takes over, i.e. we are able to decrease the cost if we maximize the entropy of that location, whereas maximizing the entropy of points whose pairwise distance is higher would bring a bigger cost for the optimization. After analyzing and comparing the robustness of the generated graph, we notice that the data-graph generated by the MSTME was more stable than the one by the Delaunay triangulation.

The proposed algorithm was based on a greedy strategy which guarantees only convergence towards a local minimum. Even though there was no guarantee of obtaining the optimum solution, the results were already more stable than the Delaunay triangulation. In the future, we would like to propose an approximation algorithm which could guarantee bounds on the MSTME problem, an algorithm which can find the optimum would only improve the results obtained in this chapter. Finally, the main motivation of this chapter started from the registration problem. Therefore, we will apply this data-graph in a framework similar to ICP and evaluate the accuracy of the results obtained as well as comparing the results using other data-graph techniques such as Delaunay [33], Gabriel Graph [44], alpha sets [36], etc.

[3] the pairwise distance is significantly smaller than the average distance in the graph, e.g. 1.5 standard deviations of the mean.

CHAPTER 8

Conclusions

> "I am here to change the future of science and eat cake and I just ran out of cake"
>
> – Unknown

Computer science has evolved since its first arithmetical usages. We have come a long way on developing smart machines and algorithms capable of learning from data, specially from images, which has been our main subject of study. Also, we have seen how graph theory developed in computer science, being part of real life problems: from social network analysis to computer vision.

We have presented a graph centrality approach to computer vision. We introduced this topological property for the first time in many vision problems. In this chapter, we present our conclusions and future directions.

8.1 Summary

We started by providing a literature review on graph theory and centrality, building up the intuition for the problems which we would tackle and later we discussed each one of them individually. Chapter 4 provided the first application of centrality as descriptors in 2D shape matching. We build a representation for the shape using the histograms of centralities and we train a Naïve Bayes classifier based on that representation. The results we obtained indicate that in the presence of random noise, the closeness centrality had the highest classification rates compared to the other centralities. One clear advantage of such a representation is the rotation invariance achieved by the using graphs. We emphasize that scale invariance can also be achieved if one normalizes the histograms. We already discussed though that the histogram representation is rather simple and looses important aspects of centrality-based representation, i.e. the location of the central nodes. If we build our system solely based on the histogram, important details of the shape might not

be captured. We also showed an alternative solution for the feature representation which could cope with that problem in Section 4.3.3 which would build random decision forests that are able to learn which locations in the shape are the most discriminative ones, by maximizing the information gain. Nevertheless, our experiments using the histograms allowed us to gather knowledge about the behaviour of each individual centrality.

Chapter 5 tackled the point-set registration problem by introducing the graph centralities into the CPD framework. We performed experimental evaluation in a different number of datasets and we observed that for some types of point-sets, the centralities acted as a good prior and were able to make the algorithm converge in fewer iterations. We evaluated the algorithm in four different scenarios: (i) point-sets which are spatially organized as clusters, (ii) points whose locations are randomly distributed, (iii) points whose locations are Gaussian distributed, (iv) points sample from the silhouette of an object, and (v) points organized as grids. Among all studied centrality measures, the closeness one obtained the fastest and most reliable results. The closeness was the fastest in three scenarios (i and iii, and iv) and the eigenvector centrality the fastest in two scenarios (ii and v). Both were significantly faster than CPD. Unfortunately not all centralities obtained good results. The reason for that is due to the fact each centrality value is defined independently, there is no consensus on which properties a centrality measure should have. Some of the consider shortest paths to define importance, whereas other measures consider the number of connections in order to call a node important or central. For instance, the degree centrality showed high instability and poor results in the registration. When we analyze the convergence of the algorithm, all centrality measures had problems with the grid based sampling, not all of them were able to converge. This is due to the fact that there is a small variability in the number of connections between points, since they are sampled in a grid, there will be always the same number of connections for points lying inside the shape, whereas only points in the border will different number of connections.

We also tackled the registration by approximating the degree distribution of a graph by a uniform distribution (Chapter 6). Our main motivation for approaching the problem in this fashion is to perform the matching by bringing the point-set towards a canonical representation, therefore, the point-sets could be matched directly, the MWME. This type of registration we proposed is able to cope up to similarity transformation, where there is a rotation, translation and scale factor. We defined a special class of graphs who can fulfill the NHDD, which is a graph with $N-1$ different node degrees, for N the number of nodes in the graph. We have proved many properties regarding such graphs, such as the duality with the complement graph (which also fulfills the NHDD), the repeated element and the entropy of such a graph.

We constrained the MWME problem to be a tree, leading to the MSTME problem in Chapter 7. In this formulation, the edge-induced subgraph is actually a tree. This would let the data-graph to be more robust to small variations in the neighborhood of each point. We provided a greedy algorithm which can approximate the solution for this problem, although we can no longer guarantee the unique solution as the unconstrained version. By solving this problem, we can obtain a data graph which is more robust to local variations.

We have evaluated the stability of the approach against the Delaunay triangulation our algorithm proved to be statistically more stable. Our noise level was based on the shortest pairwise distance in the point-set ϵ. We varied the noise up to $\pm 10 \times \epsilon$ for each point in the dataset and performed a significance test (Wilcoxon) comparing the stability of our approach against Delaunay's.

The last aspect studied in this dissertation was the usage of graph centrality in the machine learning process. We discussed about feature modeling and how we could model different machine learning features using graph centralities. We covered a histogram based feature which groups the different measures in the graph into a specific number of bins and those values can be used for training. The other type of feature we discussed was the random binary comparison within the graph, which would randomly compare two nodes in the graph and build a decision tree in case one node has a higher centrality value than the other. We trained classifiers for the different centrality values. The trained data was obtained by building graphs out of 2D shapes. The evaluation was performed using the histogram based feature of centrality values and we compared all centralities measurements under the presence of noise. Our results indicate that the closeness centrality was the most robust one under presence of noise and the degree centrality obtained the worst results in those datasets. This reassures us about the power of centrality representation in vision when we obtained similar conclusions in different problems. We noticed that closeness outperformed the other ones while the degree obtained poor results also in the registration problem discussed previously.

8.2 Open Problems and Future Direction

We see there is a potential to develop vision-oriented centrality measures since the graph we build in our applications has often spatial information associated with the points. On top of that, graphs generated from images can incorporate other types of information which could be relevant to the matching despite the spatial coordinates. For instance, we could design centralities that would incorporate gradient information from the images, or how frequent the point appears in the image (by evaluating a descriptor of the point). Aggregating more information from the scene could lead to a better centrality value, without losing the original meaning of central points in a graph.

Another approach for the design of new centralities could be the combination of the already existing ones in a fuzzy scenario. These new fuzzy-based centralities could incorporate the behaviour of standard centrality such as closeness and betweenness but having a function that would weight the importance of each one of them. Therefore, one could obtain a centrality contribute to the importance of a node using different measures and it could cope with the weakness of each individual centrality. In fact, at the end of Chapter 2, we mentioned the existence of fuzzy centralities. It would be meaningful to evaluate how fuzzy centralities could be useful in the computer vision scenario.

In Chapter 4, we observed that the eigenvector centrality had a distinguishable gradient in the shape of the object (Figure 4.2). After adding noise, that gradient vanished and a smaller gradient appeared whose center was located in the biggest closed

region without noise inside the shape. We observed this pattern, although we do not performed a deeper analysis of why this is happening. We would like in the future to prove such a statement.

When we discussed about the MSTME, we provided a greedy algorithm for it. We would like, in the future, to propose an approximation algorithm which could guarantee bounds on the MSTME problem. In Chapter 5, we discussed about registration. There are still many opened questions which can be raised and are essential to understand the remaining results of the registration:

1. What is the data-graph construction technique which can best cope with noise in point-sets?

2. Is the noise in the triangulation process the only reason for the poor performance of the centrality in the non-rigid registration?

3. If no noise is present, is the centrality always able to converge faster than CPD for any point-set?

4. What are the properties of point-sets which can be successfully addressed by the centralities?

Those questions are going to be addressed in our future work. When we discussed about the MSTME, we provided a greedy algorithm for it. We would like, in the future, to propose an approximation algorithm which could guarantee bounds on the MSTME problem. Also, we would like to integrate the MSTME in our CPD framework and in a ICP-like framework, in which we would not need to search for all points in order to find the closest one.

APPENDIX A

Licenses for Copyrighted Material

The results of this dissertation have been published in book chapters and journals. We have obtained permission from Springer and Elsevier to reprint the published material in this dissertation. We append the granted licenses in this appendix.

1. The first license refers to the contents of the paper de Sousa and Kropatsch [27] granted by Elsevier to be reused in this dissertation in both printed and online version.

2. The second license refers to the contents of the paper de Sousa and Kropatsch [28] granted by Springer Science and Business Media to be reused in this dissertation in both printed and online version.

3. The second license refers to the contents of the paper de Sousa et al. [31] granted by Springer Science and Business Media to be reused in this dissertation in both printed and online version.

4. The third license refers to the contents of the paper de Sousa et al. [32] granted by Springer Science and Business Media to be reused in this dissertation in both printed and online version.

ELSEVIER LICENSE
TERMS AND CONDITIONS

Apr 16, 2015

This is a License Agreement between Samuel de Sousa ("You") and Elsevier ("Elsevier") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Elsevier, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

Supplier	Elsevier Limited The Boulevard,Langford Lane Kidlington,Oxford,OX5 1GB,UK
Registered Company Number	1982084
Customer name	Samuel de Sousa
Customer address	Favoritenstrasse 9/4, PRIP Wien, None 1040
License number	3610750687786
License date	Apr 16, 2015
Licensed content publisher	Elsevier
Licensed content publication	Pattern Recognition
Licensed content title	Graph-based point drift: Graph centrality on the registration of point-sets
Licensed content author	None
Licensed content date	February 2015
Licensed content volume number	48
Licensed content issue number	2
Number of pages	12
Start Page	368
End Page	379
Type of Use	reuse in a thesis/dissertation
Intended publisher of new work	other
Portion	full article
Format	both print and electronic
Are you the author of this Elsevier article?	Yes
Will you be translating?	No
Title of your thesis/dissertation	A Graph Centrality Approach to Computer Vision
Expected completion date	Oct 2015
Estimated size (number of pages)	
Elsevier VAT number	GB 494 6272 12
Price	0.00 USD

VAT/Local Sales Tax	0.00 USD / 0.00 GBP
Total	**0.00 USD**

Terms and Conditions

INTRODUCTION

1. The publisher for this copyrighted material is Elsevier. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

GENERAL TERMS

2. Elsevier hereby grants you permission to reproduce the aforementioned material subject to the terms and conditions indicated.
3. Acknowledgement: If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source, permission must also be sought from that source. If such permission is not obtained then that material may not be included in your publication/copies. Suitable acknowledgement to the source must be made, either as a footnote or in a reference list at the end of your publication, as follows:
"Reprinted from Publication title, Vol /edition number, Author(s), Title of article / title of chapter, Pages No., Copyright (Year), with permission from Elsevier [OR APPLICABLE SOCIETY COPYRIGHT OWNER]." Also Lancet special credit - "Reprinted from The Lancet, Vol. number, Author(s), Title of article, Pages No., Copyright (Year), with permission from Elsevier."
4. Reproduction of this material is confined to the purpose and/or media for which permission is hereby given.
5. Altering/Modifying Material: Not Permitted. However figures and illustrations may be altered/adapted minimally to serve your work. Any other abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of Elsevier Ltd. (Please contact Elsevier at permissions@elsevier.com)
6. If the permission fee for the requested use of our material is waived in this instance, please be advised that your future requests for Elsevier materials may attract a fee.
7. Reservation of Rights: Publisher reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.
8. License Contingent Upon Payment: While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by publisher or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received on a timely basis, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.
9. Warranties: Publisher makes no representations or warranties with respect to the licensed material.
10. Indemnity: You hereby indemnify and agree to hold harmless publisher and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.
11. No Transfer of License: This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.
12. No Amendment Except in Writing: This license may not be amended except in a writing signed by both parties (or, in the case of publisher, by CCC on publisher's behalf).
13. Objection to Contrary Terms: Publisher hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and publisher (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.
14. Revocation: Elsevier or Copyright Clearance Center may deny the permissions described in this License at their sole discretion, for any reason or no reason, with a full refund payable to you. Notice of such denial will be made using the contact information provided by you. Failure to receive such notice will not alter or invalidate the denial. In no event will Elsevier or Copyright Clearance Center be responsible or liable for any costs, expenses or damage incurred by you as a result of a denial of your permission request, other than a refund of the amount(s) paid by you to Elsevier and/or Copyright Clearance Center for denied permissions.

LIMITED LICENSE

The following terms and conditions apply only to specific license types:
15. **Translation**: This permission is granted for non-exclusive world **English** rights only unless your license was granted for translation rights. If you licensed translation rights you may only translate this content into the languages you requested. A professional translator must perform all translations and reproduce the content word for word preserving the integrity of the article. If this license is to re-use 1 or 2 figures then permission is granted for non-exclusive world rights in all languages.
16. **Posting licensed content on any Website**: The following terms and conditions apply as follows: Licensing material from an Elsevier journal: All content posted to the web site must maintain the copyright information line on the bottom of each image; A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx or the Elsevier homepage for books at http://www.elsevier.com; Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.
Licensing material from an Elsevier book: A hyper-text link must be included to the Elsevier homepage at http://www.elsevier.com . All content posted to the web site must maintain the copyright information line on the bottom of each image.

Posting licensed content on Electronic reserve: In addition to the above the following clauses are applicable: The web site must be password-protected and made available only to bona fide students registered on a relevant course. This permission is granted for 1 year only. You may obtain a new license for future website posting.
17. **For journal authors:** the following clauses are applicable in addition to the above:
Preprints:
A preprint is an author's own write-up of research results and analysis, it has not been peer-reviewed, nor has it had any other value added to it by a publisher (such as formatting, copyright, technical enhancement etc.).
Authors can share their preprints anywhere at any time. Preprints should not be added to or enhanced in any way in order to appear more like, or to substitute for, the final versions of articles however authors can update their preprints on arXiv or RePEc with their Accepted Author Manuscript (see below).
If accepted for publication, we encourage authors to link from the preprint to their formal publication via its DOI. Millions of researchers have access to the formal publications on ScienceDirect, and so links will help users to find, access, cite and use the best available version. Please note that Cell Press, The Lancet and some society-owned have different preprint policies. Information on these policies is available on the journal homepage.
Accepted Author Manuscripts: An accepted author manuscript is the manuscript of an article that has been accepted for publication and which typically includes author-incorporated changes suggested during submission, peer review and editor-author communications.
Authors can share their accepted author manuscript:

- immediately
  - via their non-commercial person homepage or blog
  - by updating a preprint in arXiv or RePEc with the accepted manuscript
  - via their research institute or institutional repository for internal institutional uses or as part of an invitation-only research collaboration work-group
  - directly by providing copies to their students or to research collaborators for their personal use
  - for private scholarly sharing as part of an invitation-only work group on commercial sites with which Elsevier has an agreement
- after the embargo period
  - via non-commercial hosting platforms such as their institutional repository
  - via commercial sites with which Elsevier has an agreement

In all cases accepted manuscripts should:

- link to the formal publication via its DOI
- bear a CC-BY-NC-ND license - this is easy to do
- if aggregated with other manuscripts, for example in a repository or other site, be shared in alignment with our hosting policy not be added to or enhanced in any way to appear more like, or to substitute for, the published journal article.

Published journal article (JPA): A published journal article (PJA) is the definitive final record of published research that appears or will appear in the journal and embodies all value-adding publishing activities including peer review co-ordination, copy-editing, formatting, (if relevant) pagination and online enrichment.
Policies for sharing publishing journal articles differ for subscription and gold open access articles:
Subscription Articles: If you are an author, please share a link to your article rather than the full-text. Millions of researchers have access to the formal publications on ScienceDirect, and so links will help your users to find, access, cite, and use the best available version.

Theses and dissertations which contain embedded PJAs as part of the formal submission can be posted publicly by the awarding institution with DOI links back to the formal publications on ScienceDirect.
If you are affiliated with a library that subscribes to ScienceDirect you have additional private sharing rights for others' research accessed under that agreement. This includes use for classroom teaching and internal training at the institution (including use in course packs and courseware programs), and inclusion of the article for grant funding purposes.
Gold Open Access Articles: May be shared according to the author-selected end-user license and should contain a CrossMark logo, the end user license, and a DOI link to the formal publication on ScienceDirect.
Please refer to Elsevier's posting policy for further information.
18. **For book authors** the following clauses are applicable in addition to the above: Authors are permitted to place a brief summary of their work online only. You are not allowed to download and post the published electronic version of your chapter, nor may you scan the printed edition to create an electronic version. **Posting to a repository:** Authors are permitted to post a summary of their chapter only in their institution's repository.
19. **Thesis/Dissertation**: If your license is for use in a thesis/dissertation your thesis may be submitted to your institution in either print or electronic form. Should your thesis be published commercially, please reapply for permission. These requirements include permission for the Library and Archives of Canada to supply single copies, on demand, of the complete thesis and include permission for Proquest/UMI to supply single copies, on demand, of the complete thesis. Should your thesis be published commercially, please reapply for permission. Theses and dissertations which contain embedded PJAs as part of the formal submission can be posted publicly by the awarding institution with DOI links back to the formal publications on ScienceDirect.

Elsevier Open Access Terms and Conditions
You can publish open access with Elsevier in hundreds of open access journals or in nearly 2000 established subscription journals that support open access publishing. Permitted third party re-use of these open access articles is defined by the author's choice of Creative Commons user license. See our open access license policy for more information.
Terms & Conditions applicable to all Open Access articles published with Elsevier:
Any reuse of the article must not represent the author as endorsing the adaptation of the article nor should the article be modified in such a way as to damage the author's honour or reputation. If any changes have been made, such changes must be clearly indicated.
The author(s) must be appropriately credited and we ask that you include the end user license and a DOI link to the formal publication on ScienceDirect.
If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source it is the responsibility of the user to ensure their reuse complies with the terms and conditions determined by the rights holder.
Additional Terms & Conditions applicable to each Creative Commons user license:
CC BY: The CC-BY license allows users to copy, to create extracts, abstracts and new works from the Article, to alter and revise the Article and to make commercial use of the Article (including reuse and/or resale of the Article by commercial entities), provided the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, indicates if changes were made and the licensor is not represented as endorsing the use made of the work. The full details of the license are available at http://creativecommons.org/licenses/by/4.0.
CC BY NC SA: The CC BY-NC-SA license allows users to copy, to create extracts, abstracts and new works from the Article, to alter and revise the Article, provided this is not done for commercial purposes, and that the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, indicates if changes were made and the licensor is not represented as endorsing the use made of the work. Further, any new works must be made available on the same conditions. The full details of the license are available at http://creativecommons.org/licenses/by-nc-sa/4.0.
CC BY NC ND: The CC BY-NC-ND license allows users to copy and distribute the Article, provided this is not done for commercial purposes and further does not permit distribution of the Article if it is changed or edited in any way, and provided the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, and that the licensor is not represented as endorsing the use made of the work. The full details of the license are available at http://creativecommons.org/licenses/by-nc-nd/4.0. Any commercial reuse of Open Access articles published with a CC BY NC SA or CC BY NC ND license requires permission from Elsevier and will be subject to a fee.
Commercial reuse includes:

- Associating advertising with the full text of the Article
- Charging fees for document delivery or access
- Article aggregation
- Systematic distribution via e-mail lists or share buttons

Posting or linking by commercial companies for use by customers of those companies.

20. **Other Conditions**:

v1.7

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.

Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.

SPRINGER LICENSE
TERMS AND CONDITIONS

May 10, 2015

This is a License Agreement between Samuel de Sousa ("You") and Springer ("Springer") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Springer, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	3625271294081
License date	May 10, 2015
Licensed content publisher	Springer
Licensed content publication	Springer eBook
Licensed content title	Data Graph Formulation as the Minimum-Weight Maximum-Entropy Problem
Licensed content author	Samuel de Sousa
Licensed content date	Jan 1, 2015
Type of Use	Book/Textbook
Requestor type	Publisher
Publisher	Vienna University of Technology
Portion	Full text
Format	Print and Electronic
Will you be translating?	No
Print run	6
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	None
Title of new book	A Graph Centrality Approach to Computer Vision
Author of new book	Samuel de Sousa
Expected publication date of new book	Oct 2015
Estimated size of new book (pages)	130
Total	0.00 USD
Terms and Conditions	

Introduction

The publisher for this copyrighted material is Springer Science + Business Media. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Springer Science + Business Media makes no representations or warranties with respect to the licensed material.

Indemnity
You hereby indemnify and agree to hold harmless Springer Science + Business Media and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License
This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without Springer Science + Business Media's written permission.

No Amendment Except in Writing
This license may not be amended except in a writing signed by both parties (or, in the case of Springer Science + Business Media, by CCC on Springer Science + Business Media's behalf).

Objection to Contrary Terms
Springer Science + Business Media hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer Science + Business Media (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction
All disputes that may arise in connection with this present License, or the breach thereof, shall be settled exclusively by the country's law in which the work was originally published.

Other terms and conditions:

v1.3

SPRINGER LICENSE
TERMS AND CONDITIONS

Apr 13, 2015

This is a License Agreement between Samuel de Sousa ("You") and Springer ("Springer") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Springer, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	3607130889324
License date	Apr 13, 2015
Licensed content publisher	Springer
Licensed content publication	Springer eBook
Licensed content title	Estimation of Distribution Algorithm for the Max-Cut Problem
Licensed content author	Samuel de Sousa
Licensed content date	Jan 1, 2013
Type of Use	Book/Textbook
Requestor type	Publisher
Publisher	Vienna University of Technology
Portion	Full text
Format	Print and Electronic
Will you be translating?	No
Print run	10
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	None
Title of new book	A Graph Centrality Approach to Computer Vision
Author of new book	Samuel de Sousa
Expected publication date of new book	Oct 2015
Estimated size of new book (pages)	130
Total	0.00 USD
Terms and Conditions	

Introduction

The publisher for this copyrighted material is Springer Science + Business Media. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Springer Science + Business Media makes no representations or warranties with respect to the licensed material.

Indemnity
You hereby indemnify and agree to hold harmless Springer Science + Business Media and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License
This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without Springer Science + Business Media's written permission.

No Amendment Except in Writing
This license may not be amended except in a writing signed by both parties (or, in the case of Springer Science + Business Media, by CCC on Springer Science + Business Media's behalf).

Objection to Contrary Terms
Springer Science + Business Media hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer Science + Business Media (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction
All disputes that may arise in connection with this present License, or the breach thereof, shall be settled exclusively by the country's law in which the work was originally published.

Other terms and conditions:

v1.3

SPRINGER LICENSE
TERMS AND CONDITIONS

Apr 13, 2015

This is a License Agreement between Samuel de Sousa ("You") and Springer ("Springer") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Springer, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	3607130760709
License date	Apr 13, 2015
Licensed content publisher	Springer
Licensed content publication	Springer eBook
Licensed content title	On the Evaluation of Graph Centrality for Shape Matching
Licensed content author	Samuel de Sousa
Licensed content date	Jan 1, 2013
Type of Use	Book/Textbook
Requestor type	Publisher
Publisher	Vienna University of Technology
Portion	Full text
Format	Print and Electronic
Will you be translating?	No
Print run	10
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	None
Title of new book	A Graph Centrality Approach to Computer Vision
Author of new book	Samuel de Sousa
Expected publication date of new book	Oct 2015
Estimated size of new book (pages)	130
Total	0.00 USD
Terms and Conditions	

Introduction

The publisher for this copyrighted material is Springer Science + Business Media. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Springer Science + Business Media makes no representations or warranties with respect to the licensed material.

Indemnity
You hereby indemnify and agree to hold harmless Springer Science + Business Media and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License
This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without Springer Science + Business Media's written permission.

No Amendment Except in Writing
This license may not be amended except in a writing signed by both parties (or, in the case of Springer Science + Business Media, by CCC on Springer Science + Business Media's behalf).

Objection to Contrary Terms
Springer Science + Business Media hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer Science + Business Media (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction
All disputes that may arise in connection with this present License, or the breach thereof, shall be settled exclusively by the country's law in which the work was originally published.

Other terms and conditions:

v1.3

Bibliography

[1] A. Abbasi and J. Altmann. On the correlation between research performance and social network analysis measures applied to research collaboration networks. In *44th Hawaii International Conference onSystem Sciences (HICSS)*, pages 1–10, 2011.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34 (11):2274–2282, November 2012. ISSN 0162-8828.

[3] V. Aggarwal, Y.P. Aneja, and K.P.K. Nair. Minimal spanning tree subject to a side constraint. *Computers and Operations Research*, 9(4):287 – 296, 1982. ISSN 0305-0548.

[4] Kim Allan Andersen, Kurt Jőrnsten, and Mikael Lind. On bicriterion minimal spanning trees: An approximation. *Computers and Operations Research*, 23(12): 1171 – 1182, 1996. ISSN 0305-0548.

[5] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007. ISBN ISBN 0691129932.

[6] M. A. Audette, F. P. Ferrie, and T. M. Peters. An algorithmic overview of surface registration techniques for medical imaging. *Medical Image Analysis*, 4(3):201–217, 2000.

[7] Marco Antonio Balieiro, Samuel de Sousa, and Cleidson R. B. de Souza. Facilitating social network studies of floss using the ossnetwork environment. In *Proceedings of International Conference on Open Source Systems, OSS.*, IFIP, pages 343–350. Springer, 2008. ISBN 978-0-387-09683-4.

[8] Alex Bavelas. A mathematical model for group structures. *Human Organization*, 7, 1948.

[9] Alex Bavelas. A mathematical model for small group structures. *Human Organization*, 7:16–30, 1948.

[10] Alex Bavelas. Communication patterns in task oriented groups. *Journal of the Acoustical Society of America*, 22:271–282, 1950.

[11] MA Beauchamp. An improved index of centrality. *Behavioral Science*, 1965.

[12] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. ISSN 0162-8828.

[13] H. Blum. Biological shape and visual science (part I). *Journal of Theoretical Biology*, 38:205–287, 1973.

[14] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept 2004. ISSN 0162-8828.

[15] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. volume 1, pages 105–112, 2001.

[16] Yuri Boykov and Gareth F. Lea. Graph Cuts and Efficient N-D Image Segmentation. *International Journal of Computer Vision*, 70(2):109–131, November 2006. ISSN 0920-5691.

[17] Enrico Bozzo and Massimo Franceschet. Resistance distance, closeness, and betweenness. *Social Networks*, 35(3):460 – 469, 2013. ISSN 0378-8733.

[18] T.S. Caetano, T. Caelli, D. Schuurmans, and D. A C Barone. Graphical models and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1646–1663, 2006. ISSN 0162-8828.

[19] Marco Carcassoni and Edwin R. Hancock. Spectral correspondence for point pattern matching. *Pattern Recognition*, 36(1):193 – 204, 2003. ISSN 0031-3203.

[20] J Cheeger. A Lower Bound for the Smallest Eigenvalue of the Laplacian. *Problems in Analysis*, 1970.

[21] 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*, 2004.

[22] CarlosD. Correa and Kwan-Liu Ma. Visualizing social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 307–326. Springer US, 2011. ISBN 978-1-4419-8461-6.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 0885-6125.

[24] Xavier Cortés, Francesc Serratosa, and Carlos F. Moreno-García. On the influence of node centralities on graph edit distance for graph classification. In Cheng-Lin Liu, Bin Luo, Walter G. Kropatsch, and Jian Cheng, editors, *Graph-Based Representations in Pattern Recognition*, volume 9069 of *Lecture Notes in Computer Science*, pages 231–241. Springer International Publishing, 2015. ISBN 978-3-319-18223-0.

[25] A.D.J. Cross and E.R. Hancock. Graph matching with a dual-step em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1236–1253, Nov 1998. ISSN 0162-8828.

[26] W.J. Cukierski and D.J. Foran. Using betweenness centrality to identify manifold shortcuts. In *IEEE International Conference on Data Mining Workshops*, pages 949 –958, dec. 2008.

[27] Samuel de Sousa and Walter G. Kropatsch. Graph-based point drift: Graph centrality on the registration of point-sets. *Pattern Recognition*, 48(2):368 – 379, 2015. ISSN 0031-3203.

[28] Samuel de Sousa and Walter G. Kropatsch. Data Graph Formulation as the Minimum-Weight Maximum-Entropy Problem. In Luo B. Kropatsch W.G. Cheng J Liu, C.-L., editor, *Graph-Based Representations in Pattern Recognition*, volume 9069 of *Lecture Notes in Computer Science*, pages 244–253. Springer International Publishing, 2015.

[29] Samuel de Sousa and Walter G. Kropatsch. The Minimum Spanning Tree of Maximum Entropy. In *Proceedings of The 39th Annual Workshop of the Austrian Association for Pattern Recognition (ÖAGM)*. 2015.

[30] Samuel de Sousa, M.A. Balieiro, J.M. dos R. Costa, and C.R.B. de Souza. Multiple social networks analysis of floss projects using sargas. In *HICSS '09. 42nd Hawaii International Conference on System Sciences.*, pages 1–10, 2009.

[31] Samuel de Sousa, Nicole M. Artner, and Walter G. Kropatsch. On the evaluation of graph centrality for shape matching. In *GbRPR*, volume 7877 of *Lecture Notes in Computer Science*, pages 204–213. Springer, 2013. ISBN 978-3-642-38220-8.

[32] Samuel de Sousa, Yll Haxhimusa, and Walter G. Kropatsch. Estimation of Distribution Algorithm for the Max-Cut Problem. In WalterG. Kropatsch, NicoleM. Artner, Yll Haxhimusa, and Xiaoyi Jiang, editors, *Graph-Based Representations in Pattern Recognition*, volume 7877 of *Lecture Notes in Computer Science*, pages 244–253. Springer Berlin Heidelberg, 2013.

[33] B. N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, pages 793–800, 1934.

[34] PeterJ. Dickinson, Horst Bunke, Arek Dadej, and Miro Kraetzl. Matching graphs with unique node labels. *Pattern Analysis and Applications*, 7(3):243–254, 2004. ISSN 1433-7541.

[35] W.E. Donath and A.J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Research and Developments*, 1973.

[36] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, Jul 1983. ISSN 0018-9448.

[37] T. Erdős, P.; Gallai. Gráfok előírt fokszámú pontokkal. *Matematikai Lapok*, 11: 264–274, 1960.

[38] Leonard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.

[39] Esther Ezra, Micha Sharir, and Alon Efrat. On the performance of the ICP algorithm. *Computational Geometry*, 41(1-2):77 – 93, 2008. ISSN 0925-7721. Special Issue on the 22nd European Workshop on Computational Geometry (EuroCG) 22nd European Workshop on Computational Geometry.

[40] P.F. Felzenszwalb and J.D. Schwartz. Hierarchical matching of deformable shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 –8, june 2007.

[41] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13-14):1145 – 1153, 2003. ISSN 0262-8856.

[42] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, March 1977.

[43] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, page 215, 1978.

[44] K. Ruben Gabriel and Robert R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969.

[45] Silvia Garcia-Diez, François Fouss, Masashi Shimbo, and Marco Saerens. A sum-over-paths extension of edit distances accounting for all sequence alignments. *Pattern Recognition*, 44(6):1172 – 1182, 2011. ISSN 0031-3203.

[46] Steven Gold, Anand Rangarajan, and Eric Mjolsness. Learning with preknowledge: Clustering with point and graph matching distance measures. *Neural Computation*, 8:787–804, 1996.

[47] Janaína Gomide, Adriano Veloso, Wagner Meira, Jr., Virgílio Almeida, Fabrício Benevenuto, Fernanda Ferraz, and Mauro Teixeira. Dengue surveillance based on a computational model of spatio-temporal locality of twitter. In *Proceedings of the 3rd International Web Science Conference*, WebSci '11, pages 3:1–3:8, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0855-7.

[48] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. ISBN 013168728X.

[49] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013.

[50] L. Grady and C.V. Alvino. The piecewise smooth mumford shah functional on an arbitrary graph. *IEEE Transactions on Image Processing*, 18(11):2547–2561, Nov 2009. ISSN 1057-7149.

[51] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279, 1989. ISSN 00359246.

[52] Séverine Habert, Parmeshwar Khurd, and Christophe Chefd'hotel. Registration of multiple temporally related point sets using a novel variant of the coherent point drift algorithm: application to coronary tree matching. *Proceedings of SPIE*, 8669: 86690M–86690M–11, 2013.

[53] S. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial and Applied Mathematics*, 10(3): 496–506, 1962.

[54] Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[55] E Harary. Status and contrastatus. *Sociometry*, 22:23–43, 1959.

[56] Václav Havel. Poznámka o existenci konečných grafu. *Časopis pro pěstování matematiky*, 080(4):477–480, 1955.

[57] Ren-jie Hu, Guang-yu Zhang, and Li-ping Liao. The closeness centrality analysis of fuzzy social network based on inversely attenuation factor. In Bing-Yuan Cao and Hadi Nasseri, editors, *Fuzzy Information and Engineering and Operations Research and Management*, volume 211 of *Advances in Intelligent Systems and Computing*, pages 457–465. Springer Berlin Heidelberg, 2014.

[58] Ren-Jie Hu, Qing Li, Guang-Yu Zhang, and Wen-Cong Ma. Centrality measures in directed fuzzy social networks. *Fuzzy Information and Engineering*, 7(1):115 – 128, 2015. ISSN 1616-8658.

[59] Varun Jain and Hao Zhang. Robust 3d shape correspondence in the spectral domain. In *IEEE International Conference on Shape Modeling and Applications*, pages 19–19, 2006.

[60] H. Jiang, S. X. Yu, and D. R. Martin. Linear scale and rotation invariant matching. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 2011.

[61] George Judge, Douglas Miller, and Golan. *Maximum Entropy Econometrics: Robust Estimation With Limited Data*. John Wiley & Sons, March 1996. ISBN 0471953113.

[62] D.J. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993. ISSN 0259-9791.

[63] W. Kocay and D.L. Kreher. *Graphs, Algorithms, and Optimization*. Discrete Mathematics and Its Applications. Taylor & Francis, 2004. ISBN 9780203489055.

[64] Walter G. Kropatsch, Adrian Ion, Yll Haxhimusa, and Thomas Flanitzer. The eccentricity transform (of a digital shape). In *Discrete Geometry for Computer Imagery*, volume 4245 of *Lecture Notes in Computer Science*, pages 437–448. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-47651-1.

[65] Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

[66] Christoph H. Lampert. Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 4(3):193–285, March 2009. ISSN 1572-2740.

[67] Pat Langley. The changing science of machine learning. *Machine Learning*, 82(3): 275–279, 2011. ISSN 0885-6125.

[68] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219.

[69] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1482–1489, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2334-X-02.

[70] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, September 2006. ISSN 0162-8828.

[71] Qun Li, Zhen Qin, Lunshao Chai, Honggang Zhang, Jun Guo, and Bir Bhanu. Representative reference-set and betweenness centrality for scene image categorization. In *20th IEEE International Conference on Image Processing (ICIP)*, 2013.

[72] Wei Lian and Lei Zhang. Rotation invariant non-rigid shape matching in cluttered scenes. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *European Conference in Computer Vision (ECCV)*, volume 6315 of *Lecture Notes in Computer Science*, pages 506–518. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15554-3.

[73] Wei Lian, D. Zhang, and D. Zhang. Rotation-invariant nonrigid point set matching in cluttered scenes. *IEEE Transactions on Image Processing*, 21(5):2786–2797, 2012. ISSN 1057-7149.

[74] L. Liao, R. Hu, and G. Zhang. The centrality analysis of fuzzy social networks. *Fuzzy Systems and Mathematics*, 27(2):169–173, 2012.

[75] B. Luo and E. R. Hancock. Iterative procrustes alignment with the em algorithm. *Image and Vision Computing*, 2002.

[76] B. Luo and E.R. Hancock. Iterative procrustes alignment with the EM algorithm. *Image and Vision Computing*, 20(5-6):377 – 396, 2002. ISSN 0262-8856.

[77] Bin Luo and E. R. Hancock. A unified framework for alignment and correspondence. *Computer Vision and Image Understanding*, 92(1):26–55, October 2003. ISSN 1077-3142.

[78] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California 1965/66, 1, 281-297 (1967)., 1967.

[79] A. Mantrach, Luh Yen, J. Callut, K. Francoisse, M. Shimbo, and M. Saerens. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1112–1126, 2010. ISSN 0162-8828.

[80] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[81] M.D. McFarlane. Digital pictures fifty years ago. *Proceedings of the IEEE*, 60(7): 768–770, July 1972. ISSN 0018-9219.

[82] Bojan Mohar. Laplace eigenvalues of graphs - a survey. *Discrete Mathematics*, 109 (1-3):171 – 183, 1992. ISSN 0012-365X.

[83] S. Mukherjee, S.K. Biswas, and D.P. Mukherjee. Recognizing human action at a distance in video by key poses. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(9):1228 –1241, sept. 2011. ISSN 1051-8215.

[84] Andreas C. Műller, Sebastian Nowozin, and Christoph H. Lampert. Information theoretic clustering using minimum spanning trees. In Axel Pinz, Thomas Pock, Horst Bischof, and Franz Leberl, editors, *Pattern Recognition*, volume 7476 of *Lecture Notes in Computer Science*, pages 205–215. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32716-2.

[85] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989. ISSN 1097-0312.

[86] Andriy Myronenko and Xubo Song. Point Set Registration: Coherent Point Drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, December 2010. ISSN 0162-8828.

[87] A. B M Nasiruzzaman, H.R. Pota, and M.A. Mahmud. Application of centrality measures of complex network framework in power grid. In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, pages 4660–4665, 2011.

[88] M. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39–54, 2005.

[89] S. Ohrhallinger and S. Mudur. An efficient algorithm for determining an aesthetic shape connecting unorganized 2d points. *Computer Graphics Forum*, 32(8):72–88, 2013. ISSN 1467-8659.

[90] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. Ranking of closeness centrality for large-scale social networks. In *Proceedings of the 2nd annual international workshop on Frontiers in Algorithmics*, pages 186–195, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-69310-9.

[91] R. Pal, A. Mukherjee, P. Mitra, and J. Mukherjee. Modelling visual saliency using degree centrality. *Computer Vision, IET*, 4(3):218–229, 2010. ISSN 1751-9632.

[92] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the mumford-shah functional. In *IEEE 12th International Conference on Computer Vision*, pages 1133–1140, Sept 2009.

[93] Helmut Pottmann, Qi-Xing Huang, Yong-Liang Yang, and Shi-Min Hu. Geometry and convergence analysis of algorithms for registration of 3D shapes. *International Journal of Computer Vision*, 67(3):277–296, 2006.

[94] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[95] Huaijun Qiu and Edwin R. Hancock. Image segmentation using commute times. In *In British Machine Vision Conference*, pages 929–938, 2005.

[96] Huaijun Qiu and Edwin R. Hancock. Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11): 1873–1890, 2007. ISSN 0162-8828.

[97] R. Ravi and M.X. Goemans. The constrained minimum spanning tree problem. In Rolf Karlsson and Andrzej Lingas, editors, *Algorithm Theory âĂŤ SWAT'96*, volume 1097 of *Lecture Notes in Computer Science*, pages 66–75. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-61422-7.

[98] G. Reeb. Sur les points singuliers d'une forme de pfaff complétement intégrable ou d'une fonction numérique. *Comptes Rendus de l'Académie des Sciences Paris*, 222: 847–849, 1946.

[99] Marcos Rodrigues, Robert Fisher, and Yonghuai Liu. Special issue on registration and fusion of range images. *Computer Vision and Image Understanding*, 87(1-3): 1–7, July 2002. ISSN 1077-3142.

[100] Frank Rosenblatt. The perceptron, a perceiving and recognizing automaton. *Report 85-460-1. Cornell Aeronautical Laboratory*, 1957.

[101] J. N. Rosenquist, J. Murabito, J. H. Fowler, and N. A. Christakis. The spread of alcohol consumption behavior in a large social network. *Annals of Internal Medicine*, 152(7):426–W141, 2010.

[102] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[103] G Sabidussi. The centrality index of a graph. *Psychometrika*, 1966.

[104] Marco Saerens, François Fouss, Luh Yen, and Pierre Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *Proceedings of the 15th European Conference on Machine Learning*, pages 371–383. Springer-Verlag, 2004.

[105] Yusuf Sahillioglu and Yücel Yemez. Minimum-distortion isometric shape correspondence using em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2203–2215, 2012.

[106] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image Vision Computing*, 25(5):578–596, May 2007. ISSN 0262-8856.

[107] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, April 2002. ISSN 0920-5691.

[108] G. Scott and Longuet H. Higgins. An algorithm for associating the features of two patterns. In *Proc. Royal Society London*, volume B244, pages 21–26, 1991.

[109] T.B. Sebastian, P.N. Klein, and B.B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550 –571, may 2004. ISSN 0162-8828.

[110] Mathieu Senelle, Silvia García-Díez, Amin Mantrach, Masashi Shimbo, Marco Saerens, and François Fouss. The sum-over-forests density index: identifying dense regions in a graph. *CoRR*, abs/1301.0725, 2013.

[111] Larry S Shapiro and J Michael Brady. Feature-based correspondence: an eigenvector approach. *Image and Vision Computing*, 10(5):283 – 288, 1992. ISSN 0262-8856.

[112] Daniel Sharvit, Jacky Chan, Hüseyin Tek, and Benjamin B. Kimia. Symmetry-based indexing of image databases. *Journal of Visual Communication and Image Representation*, 9(4):366–380, December 1998.

[113] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[114] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2011.

[115] K. Siddiqi, A. Shokoufandeh, S.J. Dickenson, and S.W. Zucker. Shock graphs and shape matching. In *Sixth International Conference on Computer Vision, 1998*, pages 222–229, Jan 1998.

[116] Y. Taigman, Ming Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, June 2014.

[117] G.K.L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, F.C. Langbein, Yonghuai Liu, D. Marshall, R.R. Martin, Xian-Fang Sun, and P.L. Rosin. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1199–1217, 2013.

[118] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

[119] L. Torresani, V. Kolmogorov, and C. Rother. A dual decomposition approach to feature correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):259–271, 2013. ISSN 0162-8828.

[120] Andrea Torsello and Edwin R. Hancock. A skeletal measure of 2d shape similarity. *Computer Vision and Image Understanding*, 95(1):1 – 29, 2004. ISSN 1077-3142.

[121] A. M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950.

[122] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5): 695–703, 1988. ISSN 0162-8828.

[123] J. Utans. Mixture models and the em algorithms for object recognition within compositional hierarchies. Technical report, ICSI Berkeley, 1993.

[124] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[125] Zihou Wang, Yanni Han, Tao Lin, Hui Tang, and Song Ci. Virtual network embedding by exploiting topological information. In *IEEE Global Communications Conference (GLOBECOM)*, pages 2603–2608, 2012.

[126] Stanley Wasserman and Katherine Faust. *Social Network Analysis. Methods and Applications.* Cambridge University Press, New York, USA, 1994.

[127] Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[128] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1457–1464. IEEE, 2013.

[129] Arnold Zellner and Richard A. Highfield. Calculation of maximum entropy distributions and approximation of marginalposterior distributions. *Journal of Econometrics*, 37(2):195 – 209, 1988. ISSN 0304-4076.

[130] Feng Zhou and Fernando De la Torre. Factorized graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[131] Feng Zhou and Fernando De la Torre. Deformable graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[132] Long Zhu, Yuanhao Chen, and A. Yuille. Learning a hierarchical deformable template for rapid deformable object parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1029 –1043, june 2010.

Index

Acronyms

AI Artificial Intelligence. 1

CG Computer Graphics. 2

CPD Coherent Point Drift. 5, 9, 11, 34, 37, 38, 40, 60–67, 71, 72, 95, 100, 102

CV Computer Vision. 2, 3, 5, 20, 27–29, 31, 42, 75

DGM Deformable Graph Matching. 39

DIP Digital Image Processing. 2

DP Dynamic Programming. 11, 23, 76, 77

EM Expectation Maximization. 37–40

EMST Euclidean Minimum Spanning Tree. 75

FGM Factorized Graph Matching. 39

FSNA Fuzzy Social Network Analysis. 31

GM Graph Matching. 8, 39, 91

GMM Gaussian Mixture Models. 37, 38, 62

GPD Graph-based Point Drift. 60–63, 65

GPS Global Positioning System. 22

ICP Iterative Closest Point. 4, 5, 11, 37, 40, 91, 97, 102

ML Machine Learning. 1, 2, 52

MRF Markov Random Field. 8

MST Minimum Spanning Tree. 8, 10, 16, 17, 70, 92–94

MSTME Minimum Spanning Tree of Maximum Entropy. 11, 91, 93–97, 100, 102

MWME Minimum-Weight Maximum-Entropy. 11, 76, 77, 88, 89, 100

NHDD Near Homogeneous Degree Distribution. xvii, xxi, 77–85, 88, 100

PGM Probabilistic Graphical Models. 8

RGB Red, Green, and Blue. 5

SNA Social Network Analysis. 9

SoF Sum-over-Forest. 28

SoP Sum-over-Path. 28

SVD Singular Value Decomposition. 29

SVM Support Vector Machines. 1, 7, 51

TSP Travel Salesman Problem. 15, 70

Notation

Class of Problems

- $\mathcal{P}$ is the class of problems which can be solved by a deterministic Turing machine in polynomial time.
- $\mathcal{NP}$ is the class of problems which can be solved by a non-deterministic Turing machine in polynomial time.
- $O(.)$ refers to an upper bound on the growth of a function, i.e. the worst case computational complexity of an algorithm according to the Big-O notation.

Transformations and Geometry

- $\mathbf{R}$ is a rotation matrix.
- $\mathbf{t}$ is a translation vector.
- $\mathbf{A}$ is an affine matrix.
- $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)$ is the distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ according to a metric function.
- d_e is the Euclidean distance.
- d_m is the Manhattan distance.
- d_c is the Chevycheb distance.
- $|.|$ is the length of a vector or a graph, i.e. $|\mathcal{X}| = N$.
- $||.||_p$ is a p-norm.

Graphs and Point-sets

- $\mathbf{X}, \mathbf{Y}$ are point-sets whose dimensions are N and M.
- $\mathcal{X}(\mathcal{V}, \mathcal{E})$ is graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$. In case of vertices corresponding to the point-set $\mathbf{X}$, then $\mathcal{X}$ is a data-graph.
- $\mathbf{x}_n$ is the n^{th} point $\in \mathcal{X}$, for $1 \leq n \leq N$.

- $\mathbf{y}_m$ is the m^{th} point $\in \mathcal{Y}$, for $1 \leq m \leq M$.
- $\mathcal{X}_t$ is a complete graph with t nodes and ${}^{t(t-1)}/_{2}$ edges.
- $\overline{\mathcal{X}}$ is the complement graph of a graph $\mathcal{X}$, i.e. $\overline{\mathcal{X}} = \mathcal{X}_t - \mathcal{X}$.
- $\mathcal{X}\langle S \rangle$ is a node induced subgraph of $\mathcal{X}(\mathcal{V}, \mathcal{E})$, induced by a vector S, where $|S| = |\mathcal{V}|$.
- $\mathcal{X}[U]$ is an edge induced subgraph of $\mathcal{X}(\mathcal{V}, \mathcal{E})$, induced by a vector U, where $|U| = |\mathcal{E}|$.
- $\mathcal{X}_N^{\bullet}$ is a graph fulfilling the NHDD condition which is connected.
- $\mathcal{X}_N^{\circ}$ is a graph fulfilling the NHDD condition but it has one disconnected node.

Centrality

- $v(\mathbf{x}_n)$ is a function $v : \mathcal{X} \to \mathbb{R}$ which stands for any centrality value of a node $\mathbf{x}_n$.
- **Betwenneess**
  - $b(\mathbf{x}_n)$ stands for the betweenness centrality of a node $\mathbf{x}_n$.
  - $B\{\mathcal{X}\} = \{b(\mathbf{x}_n) | \mathbf{x}_n \in \mathcal{X} : 1 \leq n \leq N\}$.
  - $B\{\mathcal{X}\}_{\neq}$ is set of distinct values of $B\{\mathcal{X}\}$.
- **Closeness**
  - $c(\mathbf{x}_n)$ stands for the closeness centrality of a node $\mathbf{x}_n$.
  - $C\{\mathcal{X}\} = \{c(\mathbf{x}_n) | \mathbf{x}_n \in \mathcal{X} : 1 \leq n \leq N\}$.
  - $C\{\mathcal{X}\}_{\neq}$ is set of distinct values of $C\{\mathcal{X}\}$.
- **Degree**
  - $d(\mathbf{x}_n)$ stands for the degree centrality of a node $\mathbf{x}_n$.
  - $D\{\mathcal{X}\} = \{d(\mathbf{x}_n) | \mathbf{x}_n \in \mathcal{X} : 1 \leq n \leq N\}$.
  - $D\{\mathcal{X}\}_{\neq}$ is set of distinct values of $D\{\mathcal{X}\}$.
- **Eigenvector**
  - $e(\mathbf{x}_n)$ stands for the eigenvector centrality of a node $\mathbf{x}_n$.
  - $E\{\mathcal{X}\} = \{e(\mathbf{x}_n) | \mathbf{x}_n \in \mathcal{X} : 1 \leq n \leq N\}$.
  - $E\{\mathcal{X}\}_{\neq}$ is set of distinct values of $E\{\mathcal{X}\}$.