Méthodes Spectrales pour la Modélisation d'Objets Articulés à Partir de Vidéos Multiples

Diana Mateus

INSTITUT POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THESE

pour obtenir le grade de

DOCTEUR DE L'Institut Polytechnique de Grenoble

Spécialité : "IMAGERIE, VISION ET ROBOTIQUE"

préparée au laboratoire **Laboratoire Jean Kuntzmann (LJK) - INRIA Rhône Alpes**

dans le cadre de l'École Doctorale:
"Mathématiques, Sciences et Technologies de l'Information, Informatique"

préparée et soutenue publiquement

par

Diana Carolina Mateus Lamus

le

Spectral Tools for Unsupervised Modeling of Articulated Objects from Multiple-view Videos

DIRECTEUR DE THESE: Radu Horaud

JURY

Président	**Jim Crowley**
Rapporteurs	**Frédéric Jury**
	Bruno Lévy
Éxaminateurs:	**Nassir Navab**
	Edmond Boyer

Contents

Chapter 1

Introduction

Contents

1.1 Motivation and Context

The study of articulated objects has been driven, for more than two millennia, by the desire to understand human and animal motion. Early attempts to analyze the mechanical functioning of the animals and humans go back to Aristotle's (384 BC – 322 BC) book "On the Movement of Animals". Biomechanics began centuries after, with Leonardo da Vinci's (1452–1519) studies of human-body positions (Fig. 1.1-left), and by Dürer's (1471–1528) "Four books on Human Proportions". The field was later developed along with classical physics, gathering the contributions from scientists like Borelli (1608–1679), Newton (1642–1727), Bernoulli (1700–1782), Euler (1707–1783), Young (1773–1829) and Poiseuille (1799–1869).

The invention of photography marked an important milestone in the history of motion-analysis, after the studies of Muybridge (1830-1904) and Marey (1830-1904) based on photographs of humans and animals [Muybridge, 1955, Dagognet, 1992] (Fig. 1.1-right). Since then and with the technological progress of recent decades, and in particular, the rapid growth in availability of computers and cameras, the understanding of human kinematics and kinetics has continued to advance, opening the way to numerous applications where *modeling of humans and motion-analysis* play an important role; for instance, sports and medical motion-analysis, human machine interaction, modeling of humanoid robotics, computer animation, biometrics, among many others [Dariush, 2003].

Figure 1.1: Left: Leonardo da Vinci's Vitruve (Courtesy Luc Viatour). Right: Photos taken by Eadweard Muybridge (1830-1904).

A common challenge raised by applications dealing with modeling of human and animal motion is *data acquisition*, *i.e.* collecting data from the humans or animals to then measure, analyze, model, reproduce, and so on. To this end, humans and animals are generally considered as simple articulated bodies, represented by a system of rigid links connected by joints. Although this is only partially true for these biological examples, it simplifies the acquisition process, reducing it to separate measurements of each rigid body-part. Using this principle, a wide range of *motion-capture* systems has been developed, including mechanical, optical, magnetic, acoustic and inertial sensors.

Because of their accuracy, the most widely-used commercial systems for motion capture rely on optical sensing [from Vicon, AR-tracking]. The standard methodology of optical systems consists in attaching markers to the subject's body segments and detecting their projection on the images [1]. Given the loss of information in the imaging process (from 3-D motion to its 2-D projection), it is usually required to use multiple calibrated cameras to recover the position of the markers in 3-D, *i.e.* it is necessary to have knowledge about the internal parameters of the cameras and their location in the scene.

Although *marker-based* systems have proven to be effective and accurate, they are usually expensive and can only be used if it is possible to place markers on the rigid segments of the body under study. Therefore, many Computer Vision techniques have been developed in parallel, which take advantage of the large amount of information that can be recovered from conventional high-resolution cameras while observing the objects. However, the "freedom" of marker-free image-based methods comes at the cost of computational algorithms to process and interpret the videos.

There are many challenges associated with the study of articulated objects from video sequences, ranging from detection to recognition of complex motion patterns. Among these, *tracking* and

1. Different devices are used to facilitate detection for example active (*e.g.* LEDs) or passive (*e.g.* reflective) markers, along with conventional or infrared cameras

pose-recovery of articulated-objects have received significant interest from the community. The large majority of the methods proposed in the literature that aim to solve these problems are *model-based* approaches and rely on skeletons to represent the basic structure of articulated objects, and on kinematic chains to describe their motion. The description level of skeleton models is enhanced by adding geometrical primitives which better approximate the actual shape of the objects (*e.g.* rectangles, cones, ellipsoids) [Knossow et al., 2008]. Some examples of typical model-based representations are illustrated in Fig. 1.2.

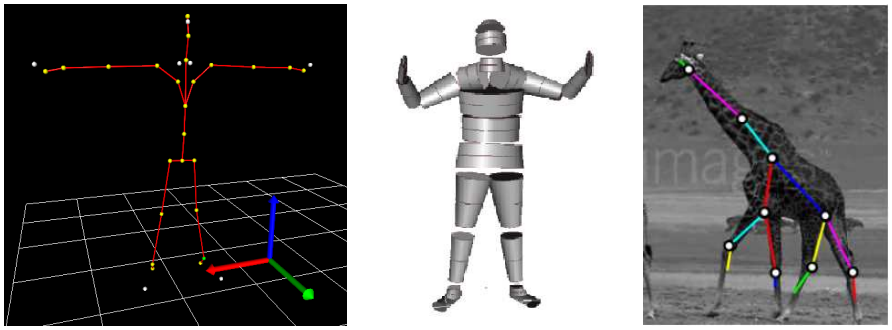


Figure 1.2: Typical models of articulated objects used in model-based approaches. a) Skeleton model b) Kinematic-chain with geometric description of the body-parts c) an automatic model built from point tracks in video sequences (courtesy [Ross et al., 2008]).

In the general framework of *model-based* approaches, problems are solved by fitting the model to the observed data within an optimization framework [Gavrila, 1999, Aggarwal and Cai, 1999]. For kinematic chains, the problem often reduces to finding the set of joint-parameters which approximates the observed pose of the articulated object. This is known to be difficult because joint-space are usually high-dimensional (*e.g.* 35 degrees of freedom for a typical human model) and non-convex [Choo and Fleet, 2001]. *Learning-based* methods overcome some of the problems linked to the optimization of the pose by training systems with examples consisting both of observations on the images and joint angles [Agarwal and Triggs, 2006] (*e.g.* captured with a classical optical system). Their limitations lie on the need for large databases and the difficulty to generalize for non trained examples. In either case, the use of model-based approaches is restricted to applications where a model built beforehand is available. This is not necessarily straightforward, since first, it implies a-priori knowledge on the exact object to be analyzed, and second, the construction of such a model is often done manually.

In recent years, there has been an increasing interest in the automation of the process to build articulated models. In general, these methods are *data-driven* solutions that abstract the model from the observed data. Examples include the discovery of skeletons from shape [Reveret et al., 2005, de Aguiar et al., 2008] or the inference of articulated shape and motion from points tracked in an image sequence [Costeira and Kanade, 1995, Yan and Pollefeys, 2005, Ross et al., 2008].

Regardless of the way the model is built or the method used to fit their configuration to observations, *model-based* methods are impractical when there is not enough a-priori knowledge about the objects in the scene, *e.g.* in the presence of multiple objects. In such cases, the acquisition of data from articulated bodies can be performed by means of feature tracking methods, optical-flow or *scene-flow* in the 3-D case. As opposed to model-based approaches, appearance-based methods

for tracking do not restrict the space of allowed transformations and deformations; this makes them general but also challenging to acquire with precision.

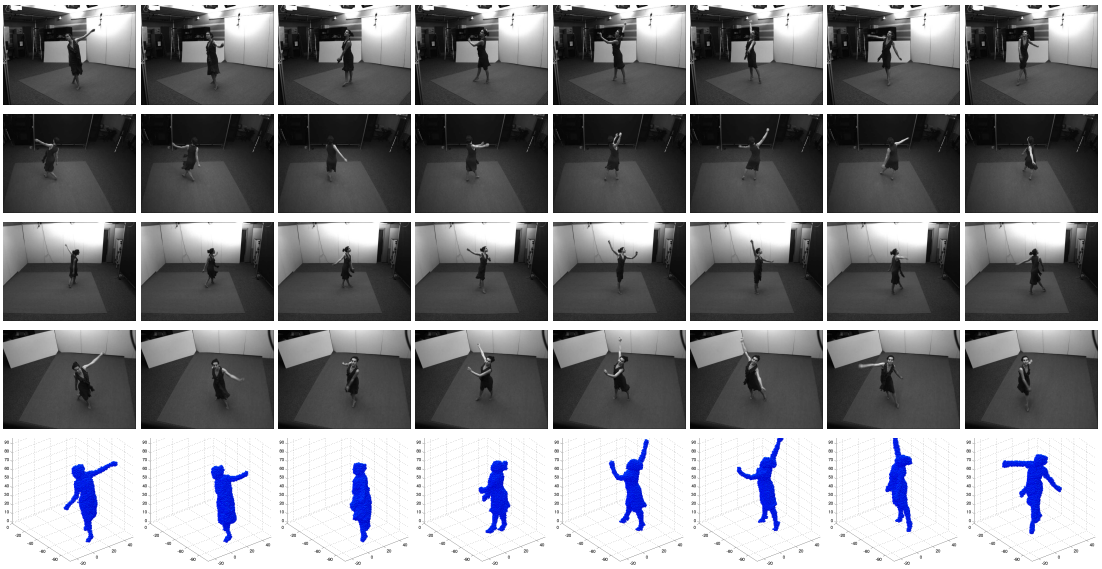


Figure 1.3: Different poses of an articulated human-body captured with a multiple-camera system. Images and silhouette-based 3-D voxel reconstruction.

With the increasing availability of multiple-camera systems (Fig. 1.3), most applications have opted for capturing the shape of the articulated object frame-by-frame, rather than the actual motion of the articulated objects. Relying on a controlled scenario (*e.g.* uniform lighting, restricted number of objects moving in the scene and static background) and a calibrated and synchronized multiple-camera system, it is possible to compute a 3-D reconstruction of the objects based, for example, on the principle of shape-from-silhouettes [sbastien Franco and Boyer, 2003]. The result of the reconstruction is generally described and stored as voxel-sets or meshed surfaces (Fig. 1.3-bottom). Efforts toward the development of multiple-camera systems over the past years have focused on the improvement of the quality and the reduction of the computational cost [Matsuyama et al., 2004]. Although such impressive reconstructions are suitable for certain applications, *e.g.* augmented-reality and free-viewpoint videos [Carranza et al., 2003], they do not provide either correspondences between the voxel-sets or meshes over time, nor information about the object's actual motion.

To recover the dynamics from independently computed 3-D reconstructions, is once again possible to fit a model to the reconstructions [Gavrila and Davis, 1996, Mikic et al., 2001, Ménier et al., 2006]. If no such a model is available, another approach is to formulate the problem as the evolution of a surface mesh over time [Matsuyama et al., 2004]. This leads to temporal smoothness but does not guarantee geometrically meaningful correspondence, as the mesh drifts over the surface, and the vertices in different frames do not necessarily correspond to the same location on the object. Appearance on the surface may be used to guide the evolution of the mesh, but this requires objects with textured surfaces [Matsuyama et al., 2004] or complex models of photometry, illuminance and reflectance [Carcernoi and Kutulakos, 2002]

Most recently proposed solutions for recovering the geometrically meaningful correspondences over time rely on finding feature correspondences to guide the evolution of the surface. Features are found either manually [Sumner and Popović, 2004, Anguelov et al., 2004, Zayer et al., 2005] or automatically, relying on geometric [Solem and Kahl, 2004] or photometric features [Starck and Hilton, 2007, K.Varanasi et al., 2008, Ahmed et al., 2008] [2]

Apart from a few exceptions [Choi et al., 2006, Starck and Hilton, 2007], the solutions proposed in the Computer Vision literature to the correspondence problem rely on the assumption of temporal smoothness and are not suitable for matching widely-separate poses of the object. Starck and Hilton [Starck and Hilton, 2007] model the search for correspondences as a constrained optimization problem with a Markov Random Field, which maximizes the photometric-correspondence of features extracted beforehand, and imposes smoothness constraints over the surface. The method requires several pre-processing and refining steps, but given that the photometric-correspondences are available and correct, it provides good results. Choi *et al.* [Choi et al., 2006] build an elastic model from the object in the first frame, and deform it to best fit the the shape of the second. This solution was probably inspired by methods used in Computer Graphics, where deforming 3-D models (*e.g.* meshes) with smoothness [Sumner and Popović, 2004], volume preserving [Zhou et al., 2005], or "as-rigid-as-possible" constraints [Sorkine and Alexa, 2007] are common ways to find correspondences between two models.

The research topics compiled in this thesis were motivated by this context of acquisition and modeling of articulated objects in the multiple-camera set-up. This work provides a detailed formulation of the problems described above, and makes important contributions to their solution.

1.2 Addressed problems

This document addresses the unsupervised [3] modeling of articulated objects, observed from multiple-view video sequences. We identify and address challenges at different stages of modeling, namely, tracking, representation, registration and segmentation. We provide advancements towards the understanding and solving these problems based on elements from Computer Vision, Computer Graphics, Machine Learning and Spectral Graph Theory.

1.2.1 Data acquisition

Different methods of acquiring 3-D information from a multiple-camera system are considered. In the first approach, we use a *sparse* representation of the objects in a scene, in terms of 3-D *appearance-based* features. We extract salient features from the images and reconstruct them in 3-D. We then propose a *scene flow* method to acquire motion information from the objects in the scene over time.

In the second approach we use shape-from-silhouette methods to compute 3-D reconstructions of articulated objects in motion, which are represented as sequences of either voxel-sets or meshes. We use these to investigate the problems of *registration* and *segmentation*. Additionally, for the registration of widely-separated pose correspondences, we also consider 3-D models of articulated objects from publicly available databases for animation.

2. Like the Scale Invariant Feature Points of SIFT [Lowe, 2004] in the photometric case; or curvature [Gal and Cohen-Or, 2006], in the geometric case

3. The term unsupervised is used throughout this document to refer to marker-free, model-free methods that do not use manual labeling or other external information.

Finally, we use the results of an model-based motion-capture system [Knossow et al., 2008] to create synthetic sequences and provide us with "ground truth", in the analysis of coherent segmentation over time.

1.2.2 Scene flow

Scene flow represents the 3-D motion of points in the scene, just as optical flow is related to their 2-D motion in the images. Contrary to classical methods which compute scene flow from optical flow, we propose to compute it by tracking 3-D points and surface elements (surfels) in a multi-camera setup (at least two cameras are needed). Two methods are proposed: in the first one, the translation of each 3-D point is found by matching the neighborhoods of its 2-D projections in each camera between two time steps; in the second one, the full pose of a surfel is recovered by matching the image of its projection with a texture template attached to the surfel, and visibility changes caused by occlusion or rotation of surfels are handled. Both methods detect lost or untrackable points and surfels. They were designed for real-time execution and can be used for fast extraction of scene flow from multi-camera sequences. Additionally, we describe an application of scene flow for motion segmentation of articulated bodies. The method is based on a similarity measure that encodes the rigidity constraint between scene-flow trajectories and spectral clustering.

1.2.3 Spectral representation of shapes

We describe a representation of articulated objects (here voxel-sets or meshes) that is invariant to pose. We derive an invariant representation from the *local-shape preservation*. The method is based on the construction of *shape-graphs* (graphs that describe the geometry of the articulated object) and their analysis with Spectral Graph Theory. The result is a set of *shape-dependent* discrete functions that are likely to be preserved under articulated motion.

We provide the connection of this type of spectral description with two related concepts in Machine Learning and Geometric Processing. We illustrate how *non-linear spectral embedding methods*, conventionally used in Machine Learning for dimensionality reduction, can be employed to recover invariant aspects of the shape (*e.g.* topology of the graph). This is also in close relation with the methods used in Geometric Processing where the Laplacian operator is used to enforce smoothness in deformations [Pinkall and Polthier, 1993, Sorkine, 2006b], and its spectral analysis for shape retrieval [Reuter et al., 2006] and to provide a "shape-aware" basis to describe functions on the shapes [Taubin, 1995, Levy, 2006].

1.2.4 Articulated shape matching

We address the problem of finding correspondences between pairs of articulated shapes in arbitrary poses, coming from the same or different sequences. We propose a general framework to solve the problem by formulating the problem in terms of graph matching. The combinatorial nature of graph matching is addressed with a solution to the inexact problem based on spectral relaxation. The derivation of the method provides a link between *spectral graph matching* and the spectral representation of shapes described above (Sec. 1.2.3). The solution is obtained in three stages: first, finding a similar set of functions from the spectral representation of the two objects; second, using the functions to embed the shape-graphs and third, performing a constrained registration in the embedding space. The later stage is formulated as probabilistic clustering and a Expectation Maximization (EM) algorithm is derived to solve it. The overall approach identifies and addresses one

of the major limitations of spectral graph matching methods when dealing with sparse and large graphs, namely their assumption on the ordering of the eigenvalues.

1.2.5 Coherent shape segmentation

Finally, we study the segmentation of articulated shapes coherently over time, *i.e.* leading to segments that are related to the same physical parts of the object in the first and last frame. We target sequences that include deformations, noise, and topological changes of the graph (for example due to self-contacts). These difficulties mean that a temporally consistent segmentation cannot be obtained from any algorithm that relies only on the isometric invariance of the shape. The proposed approach relies on the spectral representation (Chap. 3) and spectral clustering. The time coherence is ensured by the propagation of the cluster centers over time and a merge/split algorithm that recovers the appropriate number of clusters at each frame, automatically handling topological changes in the graph.

1.3 Structure of the document

This document follows two streams addressing two aspects of the motion-capture of articulated objects. The two streams correspond two the main types of data-acquistion methods considered. The first, is dedicated to the approaches that deal with dense 3-D reconstructions of articulated shapes, and it comprises the related methods for representation, registration and segmentation. A background chapter opens the first part (Chap. 2), explaining definitions and concepts of Spectral Graph Theory. Chap. 3 describes the creation of the spectral representation of the objects. The representation is used then for finding a solution to the correspondence problem (Chap. 4 and Chap. 5), and for segmentation (Chap. 6). Relevant state of the art to each problem is presented at the beginning of each chapter. The second stream, deals with the acquisition of scene-flow (Chap. 7) and its application to motion segmentation of articulated objects (Chap. 6). The dependencies between the chapters are illustrated in Fig. 1.4. The reader may also use Table. 1.1 for a guided comparison of the algorithms.

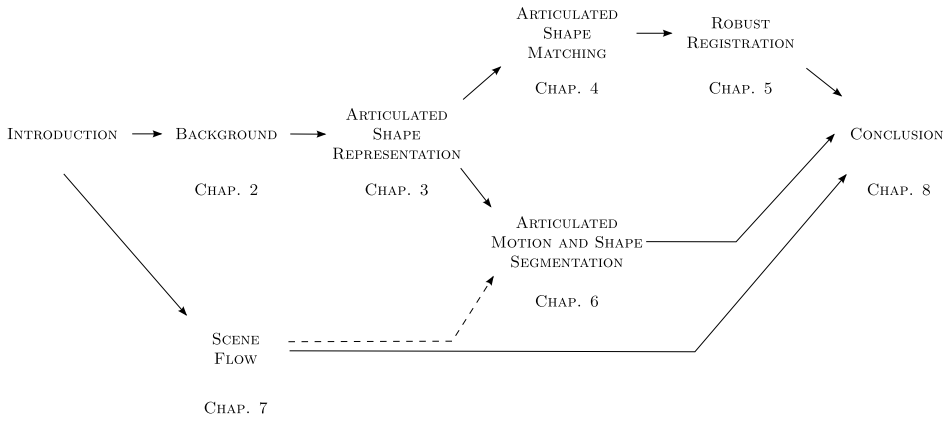


Figure 1.4: Structure of the document.

	Scene-Flow	Shape Registration	Motion segmentation	Shape segmentation
Input to the algorithm	multi-view videos, calibration, 3-D set of feature points	pair of (topologically similar) articulated objects	sparse set of scene-flow trajectories	sequence of articulated objects (voxel-sets or meshes)
Relies on temporal smoothness	yes	yes/no [a]	no	yes
Output of the algorithm	scene-flow trajectories	dense correspondence map	clusters of rigid motion	segmentation in shape-based clusters and correspondences between clusters over time
Representation of the articulated object	sparse set of features (template and positions)	graph	set of clusters	graph, set of clusters
Type of correspondences	sparse	dense	sparse	sparse
Type of information used	photometry and geometry	geometry and topology	photometry [b] and geometry	geometry and topology
Formulation of the problem	non-linear least squares problem	inexact graph matching.	clustering	clustering
Solution	Gauss-Newton algorithm	1. Common subspace selection/alignment [c] 2. probabilistic point registration (EM)	spectral clustering	spectral clustering with seed propagation
computational performance	real-time	offline	batch	offline

Table 1.1: Comparison of the methods described in this document.

a. depends on algorithm for subpspace alignement/selection: yes if out-of-sample, no if eigenfunction histogram matching.
b. captured by scene-flow
c. out-of-sample or eigenfunction histogram matching

Chapter 2

Background

Contents

2.1 Introduction

This chapter introduces basic definitions, concepts and theorems from *spectral graph theory*. Together, these elements build a unified background suitable for solving different types of problems related to the analysis of the connectivity of graphs. In brief, the spectral graph theory studies the relationship between the eigenvalues and eigenvectors of matrices describing the graph connectivity (*e.g.* the *Laplacian matrix*) with the global properties of the graph such at its connectedness, volume, bottlenecks, minimal cut, *etc*. These theoretical results have promoted the development of a large diversity of methods in computer science and other applicative domains that use graphs to model problems. Algorithms based on the spectral graph theory are attractive both because they are simple and general. The generality comes from the fact that the only input required is a graph and no condition is made on the way in which it is built.

Besides the theoretical background, this chapter illustrates in an unified fashion the formulation of several problems that can be solved through the usage of the spectral graph theory, like clustering, the analysis of random-walks and the reduction of data dimensionality. These examples give an insight into the solutions proposed later and help understanding the versatile character of the spectral graph theory. In this document, the spectral graph theory supports the methods presented later for articulated shape representation Chap. 3, shape matching Chap. 4 and shape segmentation Chap. 6.

Finally, we briefly introduce the generalization of the *Laplacian matrix* as a *discrete Laplace operator* as well as its relationship to the continuous *Laplace* and *Laplace-Beltrami operators*. This connection gives a geometric interpretation to the spectral graph theory. Geometry is convenient for understanding the outcomes of the spectral methods, in particular when the nodes of the graph are assumed to represent samples of some (possibly unknown) continuous space. The link to the continuous set-up has also been used in the literature to describe the discrete methods as approximations of physical models based on the differential Laplacian operator, *e.g.* the wave equation, the electrical conductance or the diffusion processes. The analogy serves as a means to predict the output in the ideal continuous case. As we will see in the next chapter, the relationship justifies our use of spectral graph theory in the generation of articulated-invariant shape representations which are suitable for shape matching.

The chapter is organized as follows. In Sec. 2.2, we briefly review the relevant definitions of spectral graph theory (graphs, Laplacian matrices, etc) and introduce the notation that will be used in the rest of the document. Only the required introductory results and concepts are presented here; proofs and further theoretical results may be found in books such as [Chung, 1997]. In Sec. 2.4 a series of problems are modeled on the basis of the theory, namely the study of random-walks, the graph embedding for dimension reduction and clustering. These problems are directly related to the ones we describe in detail in the following chapters related with shape analysis.

2.2 Definitions and Concepts

2.2.1 Graphs and adjacency matrices

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical object defined by a set of *vertices* or *nodes* $\mathcal{V}(\mathcal{G})$ and a collection of *edges* $\mathcal{E}(\mathcal{G})$ connecting these nodes. Vertices form a discrete set $\mathcal{V}(\mathcal{G}) = \{v_i | i \in 1, \ldots, N\}$, where N is the total number of nodes in the graph. N is usually referred as the graph's size or *cardinality* and noted $|\mathcal{G}| = N$. Edges describe connections between pairs of vertices $\mathcal{E}(\mathcal{G}) = \{e_{ij} | (i,j) \in \mathcal{V} \times \mathcal{V}\}$. Two vertices v_i and v_j connected by an edge $e_{i,j}$ are said to be *adjacent*. In order to alleviate notation and whenever there is no place for confusion, we identify the vertices and edges with their indices, *i.e.* $\mathcal{V}(\mathcal{G}) = \{1, 2, \ldots, N\}$ and $\mathcal{E}(\mathcal{G}) = \{(i,j) | (i,j) \in \mathcal{V} \times \mathcal{V}\}$. Graphs can be pictorially displayed as in Fig. 2.1, where circles represent vertices and lines or arrows represent the edges.

Different types of graphs exist according to the type of edge connections. A graph is called *directed* when its edges have a particular orientation (Fig. 2.1.b), otherwise the graph is said to be *undirected* (Fig. 2.1.a) and its edge-set is formed with *unordered* pairs of adjacent vertices. The connectivity pattern defined by the edges also determines whether the graph is complete, connected or disconnected. A graph is *complete* when there exists an edge between every pair of nodes; it is *connected* if there exists a series of edges forming a path connecting every two nodes, or *disconnected* if such path does not exist for some nodes (Fig. 2.1.c). Finally, both nodes and edges can be enriched

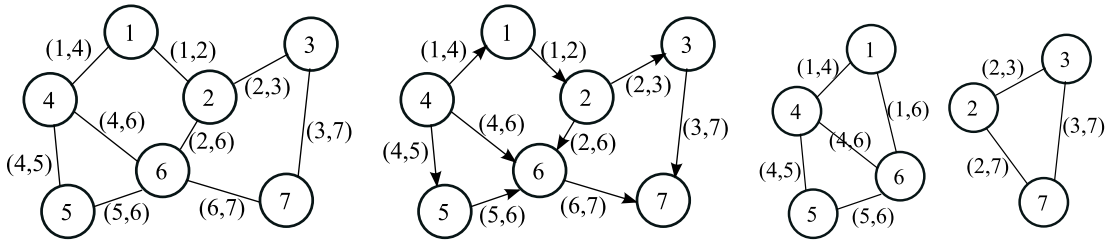


Figure 2.1: Examples of simple graphs constructed with the node-set $\mathcal{V} = \{1, 2, 3, 5, 6, 7\}$. Edge-sets are adapted to show the differences between: a) an undirected graph, b) a directed graph and c) a disconnected graph.

with further information by associating *attributes* to each node and/or adding *weights* w_{ij} to each edge. A weighted graph has an additional entry $\mathcal{W}(\mathcal{G})$, *i.e.* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, which specifies a weighting value w_{ij} for each edge (i, j).

The graph's connectivity $\mathcal{E}(\mathcal{G})$ and weights $\mathcal{W}(\mathcal{G})$ may be expressed as an $N \times N$ *Adjacency Matrix* $\mathbf{W}$, whose rows and columns are indexed by the vertices of the graph. The elements of $\mathbf{W} = [w_{ij}]_{N \times N}$ correspond either to the weights for two adjacent nodes, or are filled with zeros for non-adjacent nodes.

$$\mathbf{W} = \begin{cases} w_{ij} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

The way the connectivity pattern is built, as well as the weighting values assigned to the edges are application and/or data dependent. In general, the weights are determined by a distance, cost or similarity function defined over pairs of nodes. In the simple unweighted case, binary values are assigned to the edge (i, j), indicating whether the edge exists ($w_{ij} = 1$) or not ($w_{ij} = 0$). More information about the nature of the connection can be added by assigning continuous values to the weights. In general, weights are usually restricted to be positive, and often normalized to be in the range of the unit interval ($w_{ij} \in [0, 1]$).

Unless otherwise specified, in this document we deal with undirected, connected and weighted graphs. This implies that the weight matrix $\mathbf{W}$ is symmetric ($w_{ij} = w_{ji}$).

2.2.2 The Laplacian and Normalized Laplacian

Spectral Graph Theory studies the connectivity properties of a graph through the spectral analysis of a particular type of matrices, namely the *Laplacian matrix* $\mathbf{L}$ and the *Normalized Laplacian Matrix* $\mathcal{L}$. These matrices are built from the graph's original adjacency matrix $\mathbf{W}$ but include the information of each nodes' *degree*, which measures the distribution of the connections in the graph. The *degree* of a vertex i is found by summing up the weights of all the edges incident on it, *i.e.* $d_{ii} = \sum_j w_{ij}$. The degrees of all vertices are then stacked in the diagonal of a matrix called the *Degree matrix* $\mathbf{D}$. Finally, the *combinatorial or unnormalized Laplacian* matrix $\mathbf{L}$ is defined as the difference between the *Degree matrix* $\mathbf{D}$ and the *Adjacency matrix* $\mathbf{A}$:

$$(\textit{combinatorial Laplacian}) \qquad \mathbf{L} = \mathbf{D} - \mathbf{W} \tag{2.2}$$

The elements of $\mathbf{L} = [L_{ij}]_{N \times N}$ correspond then to:

$$L_{ij} = \begin{cases} -w_{ij} & \text{if } (i,j) \in \mathcal{E}(\mathcal{G}) \\ d_{ii} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

In a similar fashion, the *normalized Laplacian* is defined as:

$$\textit{(normalized Laplacian)} \qquad \mathcal{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2} \tag{2.4}$$

$$\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \tag{2.5}$$

Element by element, the definition of $\mathcal{L}$ amounts to normalize the weight of each edge (i,j) according the square root degree of the vertices it connects:

$$\mathcal{L} = \begin{cases} \frac{-w_{ij}}{\sqrt{d_{ii}d_{jj}}} & \text{if } (i,j) \in \mathcal{E}(\mathcal{G}) \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The normalization is used to remove the influence of the connection density, avoiding to give more importance to vertices with higher degree. The spectral analysis of $\mathcal{L}$ reveals the large-scale structure of a graph [Chung, 1997]. This is the reason why it serves as a basis for clustering applications interested in the analysis of the graph substructures. In such practical tasks, the normalized Laplacian has been shown to perform better than the combinatorial Laplacian [Meila and Shi, 2001]. The superiority of normalized Laplacian for clustering applications has been proved following a statistical analysis [von Luxburg et al., 2005]. Finally, normalization also helps to smooth out irregular samplings when nodes represent samples of a presumed uniform distribution.

Once the combinatorial or normalized Laplacian is defined, the spectral analysis of the matrix should be carried out. In the following sections we detail the multiple variants existent in the literature for this procedure.

2.2.3 Spectral analysis of the Laplacian matrices

The *spectral analysis*, for a general square matrix $\mathbf{M}$, consists of finding a set of *eigenvalues* λ that solve the *characteristic equation*:

$$0 = \det(\mathbf{M} - \lambda\mathbf{I}), \tag{2.7}$$

where det stands for the matrix determinant and $\mathbf{I}$ is the identity matrix. The polynomial that results from evaluating this determinant is called the *characteristic polynomial* of the matrix. Since the determinant is equated to zero, the solutions to Eq. 2.7 correspond to the roots of the polynomial which make the matrix $(\mathbf{M} - \lambda\mathbf{I})$ singular. Then, for every λ for which Eq. 2.7 is true there is a vector $\boldsymbol{v} \neq 0$ which verifies:

$$(\mathbf{M} - \lambda\mathbf{I})\boldsymbol{v} = 0. \tag{2.8}$$

The collection of vectors associated to each eigenvalue are the *eigenvectors* of $\mathbf{M}$. Thus, reorganizing the terms in Eq. 2.8, the eigen-decomposition of $\mathbf{M}$ in eigenvalues and eigenvectors can be found by solving the following linear system:

$$\mathbf{M}\boldsymbol{v} = \lambda\boldsymbol{v}. \tag{2.9}$$

The expression to be solved for the eigendecomposition of Laplacian matrices is obtained by replacing the generic matrix $\mathbf{M}$ in Eq. 2.9 with the combinatorial Laplacian $\mathbf{L}$:

$$\mathbf{L}\boldsymbol{v} = \lambda\boldsymbol{v}, \tag{2.10}$$

$$(\mathbf{D}-\mathbf{W})\boldsymbol{v} = \lambda\boldsymbol{v}, \tag{2.11}$$

(combinatorial Laplacian)

and the normalized Laplacian $\mathcal{L}$:

$$\mathcal{L}\boldsymbol{v} = \lambda\boldsymbol{v}, \tag{2.12}$$

$$\mathbf{D}^{-1/2}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-1/2}\boldsymbol{v} = \lambda\boldsymbol{v}. \tag{2.13}$$

(normalized Laplacian)

The expression in Eq. 2.13 is only one of the possible definitions for the eigen-decomposition of the normalized Laplacian are found in the literature. Several systems equivalent to Eq. 2.13 are shown in the following equations:

$$(\mathbf{I}-\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\boldsymbol{v} = \lambda\boldsymbol{v} \tag{2.14}$$

$$\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\boldsymbol{v} = (1-\lambda)\boldsymbol{v} \tag{2.15}$$

(spectral-clustering Laplacian)

Using the substitution $\boldsymbol{u} := \mathbf{D}^{-1/2}\boldsymbol{v}$,

$$\mathbf{D}^{-1}\mathbf{W}\boldsymbol{u} = (1-\lambda)\boldsymbol{u} \tag{2.16}$$

(random-walk Laplacian)

Transforming the system to a *generalized eigenvalue problem*[1]:

$$\mathbf{W}\boldsymbol{u} = (1-\lambda)\mathbf{D}\boldsymbol{u} \tag{2.17}$$

$$(\mathbf{D}-\mathbf{W})\boldsymbol{u} = \lambda\mathbf{D}\boldsymbol{u} \tag{2.18}$$

(generalized normalized Laplacian)

The systems in equations 2.13 and 2.18 have been used separately for different applications. For example, the classical *normalized Laplacian* formulation in Eq. 2.13 is used for the derivation of theoretical results in Chung's book [Chung, 1997], but was also employed in the description of the well known *normalized-cuts* algorithm for image-segmentation [Shi and Malik, 2000]. We recognize in Eq. 2.15 the formulation used by Ng's *et al.* [Ng et al., 2002] for their spectral clustering algorithm. Expressions in Eq. 2.16 and Eq. 2.17 are used by some authors [Meila and Shi, 2001, Qiu and Hancock, 2007] to relate the Laplacian to random-walks. Finally, the equation in Eq. 2.18 is used by Belkin and Niyogi [Belkin and Niyogi, 2003] for describing the *Laplacian-eigenmap* algorithm for dimensionality reduction.

There are two important facts that should be noticed about these relationships. The first, is that they show how spectral graph theory serves as a basis for developing spectral methods which solve different problems: segmentation, clustering, dimensionality reduction. Therefore, these solutions are intrinsically related. For example, from Eq. 2.13 and Eq. 2.18, one deduces that the embedding computed with the Laplacian Eigenmaps method (built from the eigenvectors of Eq. 2.18) is actually the same embedding that is computed with the *normalized-cuts* algorithm [Shi and Malik, 2000] (using Eq. 2.13), up to a component-wise scaling of the eigenvectors.

The second important fact about these equivalences is that they allow us to exploit desirable properties of a given system for which eigen-decomposition algorithms are more efficient or accurate.

1. The generalized eigenvalue problem is characterized by the presence of a matrix $\mathbf{B}$ in the right hand side of the linear system: $\mathbf{M}\boldsymbol{v} = \lambda\mathbf{B}\boldsymbol{v}$.

For instance, we use the eigenvalues and eigenvectors obtained with *generalized normalized Laplacian* to derive those of the *normalized Laplacian*, because it is numerically more stable. Similarly, it might be better to find the eigen-decomposition of the *random-walk* by first finding the eigenvalues and eigenvectors of the *normalized Laplacian*, which is symmetric, and then use the equivalences described above.

Nevertheless, some caution is advised when using the equations 2.13 - 2.18. First, most of the time the equivalences do not imply that the same exact eigenvalues and eigenvectors are obtained; meaning that some transformation might be required to go from one system to another [2].

Secondly, in most of the applications only a few eigenvalues and their associated eigenvectors are retained. While expressions for the normalized Laplacian (Eq. 2.13 and Eq. 2.18) keep the *smallest* eigenvalues, the remaining formulations (Eq. 2.15, Eq. 2.16 and Eq. 2.17), where the term $(1-\lambda)$ appears, retain the *largest* ones.

Thirdly, notice that some of the preceding linear systems involve the inversion of the degree matrix. In those cases, the graph needs to be checked for isolated nodes which have degree $d_{ii}=0$ and make $\mathbf{D}$ singular (non-invertible). Similar considerations have to be taken into account for small degrees which make the ratios $1/d_{ii}$ numerically unstable. This explains why the generalized version of the normalized Laplacian (Eq. 2.18) may be preferred to the decomposition in Eq. 2.13.

Finally, as we will see in Sec. 2.2.4, the resultant matrices do not necessarily have the same properties and may need different algorithms to solve the eigen-decomposition. The actual choice of an algorithm will also depend on other factors such as the graph's connectivity pattern (*e.g.* sparse or full), the size of the matrix and the number of eigenvectors and eigenvalues to be computed. Refer to [Golub and Loan, 1996], for the implementation of various eigen-decomposition algorithms.

2.2.4 Spectral properties of the Laplacian matrices

The eigen-decomposition procedure from the previous section is only informative because Laplacian matrices have interesting spectral properties. For a graph $\mathcal{G}$ with Laplacian matrices $\mathbf{L}$ and $\mathcal{L}$ and eigenvalues arranged in ascending order of magnitude $\lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_N$, the following statements are true:

1. Both $\mathbf{L}$ and $\mathcal{L}$ are symmetric, semi-definite and positive matrices [3]. This implies that their eigenvalues are real and non-negative and that their algebraic multiplicity coincides with their geometric multiplicity [4].

2. Only, Eq. 2.13 and Eq. 2.14 are strictly equivalent, meaning their eigenvalues and eigenvectors are identical. The *random-walk* Laplacian in Eq. 2.16 is easily proven to be *similar* to $\mathbf{I}-\mathcal{L}$ following the next equations:

$$\begin{aligned} \mathcal{L} &= \mathbf{I}-\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \\ \mathbf{I}-\mathcal{L} &= \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \\ \mathbf{D}^{-1/2}(\mathbf{I}-\mathcal{L})\mathbf{D}^{1/2} &= \mathbf{D}^{-1}\mathbf{W} \end{aligned}$$

The similarity means that both, the *random-walk Laplacian* $\mathbf{D}^{-1}\mathbf{W}$ and $\mathbf{I}-\mathcal{L}$, have the same eigenvalues and the same eigenvectors, factor a point-wise multiplication by $\mathbf{D}^{-1/2}$. This is also the case between the *normalized Laplacian* Eq. 2.13 and its *generalized* formulation Eq. 2.18, which have the same eigenvalues but whose eigenvectors relate through $\boldsymbol{u}=\mathbf{D}^{-1/2}\boldsymbol{v}$.

Finally, the *spectral-clustering* Laplacian Eq. 2.15, given by $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, has the same eigenvectors as the *normalized Laplacian* $\mathcal{L}$, and its eigenvalues can be obtained by subtracting one to those of $\mathcal{L}$.

3. As opposed to the adjacency matrix which is real and symmetric but not semi-definite positive.

4. The *algebraic multiplicity* of an eigenvalue λ is the number of times a certain eigenvalue appears as a solution of the characteristic polynomial Eq. 2.7. The algebraic multiplicity is in general different from the eigenvalue's *geometric multiplicity*, which is defined as the the dimension of the nullspace of $\lambda\mathbf{I}-\mathbf{A}$ for a given matrix $\mathbf{A}$.

2. The eigenvectors of both $\mathbf{L}$ and $\mathcal{L}$ form an orthogonal basis.
3. λ_1 is always 0.
4. In a connected graph, the eigenvector of the Laplacian matrix $\mathbf{L}$ associated to λ_1 is equal to the constant (all-ones) $\mathbf{1}$ vector [5]. Additionally, the sum of the elements of any other eigenvector is 0, given its orthogonality with $\mathbf{1}$. Finally, for the normalized Laplacian $\mathcal{L}$ the eigenvector associated to λ_1 is equivalent to $\mathbf{D}^{-1/2}\mathbf{1}$.
5. When graphs are disconnected, each connected component works as a separate graph. As a consequence, the first eigenvalue λ_1 of a disconnected graph is *multiple* (has an *algebraic multiplicity* greater than 1). The number of times 0 appears as an eigenvalue of $\mathbf{L}$ or $\mathcal{L}$ is equal to the number of connected components in the graph.
6. The smallest non-trivial (non-zero) eigenvalue of $\mathbf{L}$ is known under the names of *Fiedler value* or *algebraic connectivity* and it is an indicator of the connectedness (how well connected) of the graph. Together with the *Fiedler vector* they give important information that can be employed for clustering (See Sec. 2.4.4).

Table Table. 2.1 summarizes certain spectral properties for the most common forms of Laplacian. These properties explain some of the reasons why the spectral study of the Laplacian matrices is important for the analysis of graph connectivity. Further insight justifying spectral methods may be obtained from the Spectral Theorem, which we recall in the next section.

Matrix	Definition	sym.	p.s.d	spectrum
Adjacency	$\mathbf{W}$	yes	no	$\lambda \in I\!R$, $\lvert\lambda\rvert \leq \max(d_{ii})$
Combinatorial Lap $\mathbf{L}$	$\mathbf{D}-\mathbf{W}$	yes	yes	$\lambda \in [0, 2\max(d_{ii})]$
Normalized Lap $\mathcal{L}$	$\mathbf{D}^{-1/2}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-1/2}$	yes	yes	$\lambda \in [0, 2]$
Random-walk Lap	$\mathbf{D}^{-1}\mathbf{W}$	no	no	$\lambda \in [-1, 1]$

Table 2.1: Spectral properties of different Laplacians of undirected graphs. Here, p.s.d. stands for positive semi-definite and sym. for symmetric

2.2.5 The spectral theorem

The spectral theorem provides conditions under which an operator or a matrix can be diagonalized (represented as a diagonal matrix in some basis). The theorem states that a linear transformation $\mathbf{T}$ of a vector $\boldsymbol{f}$ (or a linear operator applied on $\boldsymbol{f}$) can be expressed as a linear combination of its eigenvectors, in which the coefficient of each eigenvector is equal to the corresponding eigenvalue times the scalar product of the eigenvector with the vector $\boldsymbol{f}$. Formally, the linear combination can be written as:

$$T(\boldsymbol{f}) = \sum_{i=1}^{N} \lambda_i \langle \boldsymbol{v}_i, \boldsymbol{f} \rangle \boldsymbol{v}_i, \tag{2.19}$$

where $\{\boldsymbol{v}_1\}_{i=1}^N$ and $\{\lambda_i\}_{i=1}^N$ stand for the eigenvectors and eigenvalues of $\mathbf{T}$. The theorem is valid for all self-adjoint linear transformations [6], and for the more general class of (complex) normal matrices.

5. This is easy to prove. Each element of the degree matrix d_{ii} corresponds to the ith-row-sum of the Adjacency matrix ($d_{ii} = \sum_j w_{ij}$). Then, the product $\mathbf{W1}$ is equal to $\mathbf{D1}$, leading to $(\mathbf{W}-\mathbf{D})\mathbf{1} = 0$. The only way $(\mathbf{W}-\mathbf{D})\mathbf{1} = \lambda\mathbf{1}$ holds, is for λ to be zero $\lambda_1 = 0$.

6. linear transformations given by real symmetric matrices and Hermitian matrices

According to the Table Table. 2.1, the matrices $\mathbf{W}$, $\mathbf{L}$ and $\mathcal{L}$ satisfy the required conditions for the spectral theorem. In the following, we focus on the normalized Laplacian matrix $\mathcal{L}$, but similar results apply for $\mathbf{W}$ and $\mathbf{L}$. The applicability of the spectral theorem to the normalized Laplacian implies that $\mathcal{L}$ can be diagonalized to yield a complete set of orthonormal basis functions:

$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0 \quad \forall i \neq j, \tag{2.20}$$
$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 1 \quad \forall i = j. \tag{2.21}$$

Given the orthogonal basis in Eq. 2.20 and Eq. 2.21, $\mathcal{L}$ can be reconstructed as follows:

$$\mathcal{L} = \sum_{i=1}^{N} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top. \tag{2.22}$$

In matrix form, Eq. 2.22 can be also written as $\mathcal{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where the eigenvalues have been placed in the diagonal of the matrix $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \ldots, \lambda_N)$ and the eigenvectors have been piled in the columns of the matrix $\mathbf{V} = [\boldsymbol{v}_1 | \boldsymbol{v}_2 | \ldots | \boldsymbol{v}_N]$.

Using the relationships in Sec. 2.2.3 together with the spectral theorem, it is even possible to construct orthogonal basis for the non-symmetric *random-walk Laplacian*. To do so, one should consider that the scalar product (and thus the orthogonality) between two eigenvectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^\top \mathbf{D} \boldsymbol{b}$, where $\mathbf{D}$ is the degree matrix [Rustamov, 2007, Lafon and Lee, 2006].

2.3 Discrete and continuous Laplace operators

The spectral theorem can be extended to other functions or transformations, such as the analytic functions and operators. Consider an operator $\mathbf{T}$ defined on functions $f(x)$. The spectral analysis of such operator takes the form:

$$[\mathbf{T}f](x) = \lambda f(x), \tag{2.23}$$

Similarly, for operators T defined on infinite-dimensional spaces :

$$\int Tf(x)dx = \lambda f(x). \tag{2.24}$$

There is a direct correspondence between In Eq. 2.23, and the eigen-decomposition described in Eq. 2.9. The difference being that the discrete eigenvectors $\boldsymbol{v}$ have being replaced by functions f. For this reason, the solutions associated to the eigenvalues λ in Eq. 2.23 and Eq. 2.24 are called *eigenfunctions* instead of eigenvectors.

These extensions have lead to the interpretation of the Laplacian matrices as the discrete counterparts of the Laplace operator in continuous spaces. A first approach consists in considering the generalization of the Laplacian matrix to graphs with an infinite number of vertices and edges. This generalization is known as the discrete Laplace operator, that we also note for convenience $\mathbf{L}$. As showin in [Luxburg, 2007], applying the discrete Laplace operator $\mathbf{L}$ to a discrete function $\boldsymbol{f} \in \mathbb{R}^N$ defined on the generalized (infinite) graph $\mathcal{G}$ leads to the following expression:

$$[\mathbf{L}\boldsymbol{f}](i) = \sum_{(i,j) \in \mathcal{E}(\mathcal{G})} w_{ij} \Big(\boldsymbol{f}(i) - \boldsymbol{f}(j) \Big), \tag{2.25}$$

where $\boldsymbol{f}(i)$ are the elements of $\boldsymbol{f}$, and w_{ij} correspond to the weights in the graph's adjacency matrix. In a similar fashion, for every function $\boldsymbol{f} \in I\!R^N$, the quadratic form $\boldsymbol{f}^\top \mathbf{L}\boldsymbol{f}$ leads to:

$$\boldsymbol{f}^\top \mathbf{L}\boldsymbol{f} = \langle \boldsymbol{f}, \mathbf{L}\boldsymbol{f} \rangle = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} w_{ij} \Big(\boldsymbol{f}(i) - \boldsymbol{f}(j) \Big)^2 \tag{2.26}$$

Transforming each w_{ij} into a distance ϵ_{ij} by using $w_{ij} = 1/\epsilon_{ij}^2$:

$$w_{ij} \Big(\boldsymbol{f}(i) - \boldsymbol{f}(j) \Big)^2 \approx \left(\frac{\boldsymbol{f}(i) - \boldsymbol{f}(j)}{\epsilon_{ij}} \right)^2, \tag{2.27}$$

makes the expression inside the sum of Eq. 2.26 look like a difference quotient. Thus, Eq. 2.26 may be interpreted as the discrete version of the quadratic form associated to the standard Laplace operator on $I\!R^n$.

The *Laplace operator* or *Laplacian*, denoted by ∇^2 or $\triangle$, is a second order differential operator in the n-dimensional Euclidean space, defined as the divergence of the gradient. If f is a twice-differentiable real-valued function, then the Laplacian of f is defined as:

$$\triangle f = \nabla^2 f = \nabla \cdot \nabla f \tag{2.28}$$

The spectral analysis of the Laplacian operator is determined by the solution of the following equation:

$$\triangle f = \lambda f \tag{2.29}$$

$$\langle f, \triangle f \rangle = \int |\nabla \phi|^2 dx \tag{2.30}$$

where ∇ is the *nabla* or *grad* operator.

The Laplace operator is only defined for Euclidean spaces. We will be also interested in establishing an analogy between graphs living in a non-Euclidean space, *e.g.* defined on a manifold $\mathcal{M}$, and a continuous operator. In these cases, it is convenient to refer instead to the Laplace-Beltrami operator $\triangle_\mathcal{M}$, which is a generalization of the Laplace operator to non-Euclidean spaces. $\triangle_\mathcal{M}$ can be defined in terms of its Riemannian coordinates x_i, as follows:

$$\triangle_\mathcal{M} f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{2.31}$$

The Laplace operator has the following properties:

- Constant eigenfunction: $\triangle f = 0$ for any $f = const$.
- Symmetry: $\langle \triangle, f \rangle = \langle f, \triangle \rangle$.
- Locality: $\triangle f(x)$ is independent of $f(x')$ for any $x' \neq x$.
- Positive semi-definite $\langle \triangle f, f \rangle \geq 0$.

We can relate these properties to those discussed in the precedent sections for the Laplacian matrices. These analogies between discrete and continuous Laplacians are an active subject of study [Hein et al., 2005, Belkin and Niyogi, 2005, Hein, 2006, Gine and Koltchinskii, 2005] and have been explicitly used as a theoretical background for applications such as graph embedding methods and clustering [Belkin, 2003, Lafon, 2003]. Recent works have proved that under certain conditions graph Laplacians are discrete versions of continuous Laplace operators. In particular, if the graph

Laplacian is constructed on a similarity graph of randomly sampled data points, then it converges to some continuous Laplace operator (or Laplace-Beltrami operator) on the underlying space. Several researches in the manifold learning and vision communities have produced formal demonstrations of the convergence of the different types of graph Laplacians to the Laplace-Beltrami operator as the sample size N of the graph grows to infinity[Belkin, 2003, Lafon, 2003, Belkin and Niyogi, 2005, Hein et al., 2005, von Luxburg et al., 2005, Gine and Koltchinskii, 2005, Hein, 2006] for both uniform and more general distributions of points.

The interest of the analogy between symmetric matrices and linear operators, and that of eigenvectors defined on a vector space and eigenfunctions on a function space is, first, to a geometrical interpretation to the spectral analysis of graphs, and second, to predict the response of the discrete eigenvalue problems according to some well studied physical process governed by linear differential operators. Indeed, Eq. 2.29 is at the core of very well studied problems in physics, mathematics and quantum mechanics. These problems have served as inspiration for the interpretation of the spectral analysis of Laplacian matrices in the context of diffusion processes [Kondor and Lafferty, 2002] or the wave equation [Reuter et al., 2006]. In Chap. 3, we relate the the graph Laplacian with the continuous Laplacian operating on functions and the Laplace-Beltrami for shape description.

2.4 Solving problems with spectral graph theory

The use of spectral theory is very broad and numerous applications from a large diversity of scientific communities take advantage of its results. In this section we will give examples of how the spectral graph theory may be used for modeling and solving problems such as the long-term study of random walks, dimensionality reduction and clustering. These problems are directly related to the ones we describe in detail in the following chapters.

2.4.1 Random walks analysis

The spectral graph theory is of particular interest when studying the steady-state properties of random walk defined on graphs. Indeed the *random walk Laplacian* $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ in equation Eq. 2.16 is by construction a *stochastic matrix* (whose row sums are equal to one) and thus defines a random walk on the graph. Thus, random walks on the graph can be represented as Markov chains in which the transition probabilities are computed from the edge weights.

Consider a random variable x describing the state of the random walk process at time $t : x(t) = i$; here, i is one among the nodes in the graph ($i \in \mathcal{V}(\mathcal{G})$). From this starting node, the transition to the next state is governed by the matrix $\mathbf{P} = [p_{ij}]_{N\times N}$. Each value p_{ij} indicates the transition probability between the nodes i and j in one time step. This is illustrated in Fig. 2.2 and formalized as:

$$Pr\{x(t+1) = j | x(t) = i\} = \frac{1}{d_{ii}} w_{ij}. \tag{2.32}$$

For longer periods of time, the probability of the random walk to arrive at node j after t time steps, given $x(0) = i$ as starting location, is defined by $p(t, j|i)$. According to the Markov-chain rules, this probability can be obtained by raising the transition matrix to the power of t and retaining the i^{th} row. In equations, $p(t, j|i) = \boldsymbol{e}_i \mathbf{P}^t$, where $\boldsymbol{e}_i$ is the indicator row vector for the i^{th} row [7].

7. The indicator vector $\boldsymbol{e}_i$ is a row vector with zeros everywhere except in the i^{th} coordinate

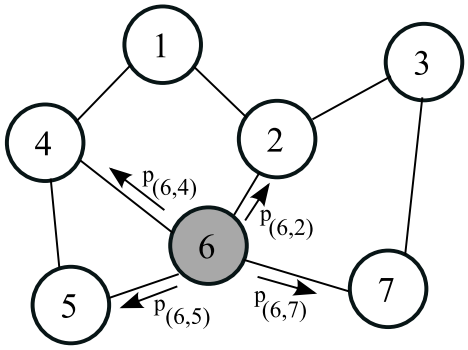


Figure 2.2: The elements of the random-walk Laplacian interpreted as the probability to go from a starting node (here node 6) to its neighbors (nodes $2, 4, 5, 7$) in one time step.

The spectral theory is very useful for evaluating these long term probabilities. The powers of the matrix $\mathbf{P}$ can be computed very efficient by noting that this operation only affects the eigenvalues and not the eigenvectors, as shown by induction in the following equations:

$$\mathbf{P}^1 = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top \tag{2.33}$$

$$\mathbf{P}^2 = \mathbf{P}\mathbf{P} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top = \mathbf{V}\mathbf{\Lambda}\mathbf{\Lambda}\mathbf{V}^\top = \mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^\top \tag{2.34}$$

$$\mathbf{P}^3 = \mathbf{P}\mathbf{P}^2 = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^\top = \mathbf{V}\mathbf{\Lambda}^3\mathbf{V}^\top \tag{2.35}$$

$$\vdots \quad \vdots \tag{2.36}$$

$$\mathbf{P}^n = \mathbf{V}\mathbf{\Lambda}^n\mathbf{V}^\top \tag{2.37}$$

The importance of Eq. 2.37 is that $p(t, j|i)$ can be easily computed from the powers of the eigenvalues of $\mathbf{P}$, which only need to be computed once, reducing the computational burden of matrix multiplication. Spectral graph theory also gives a closed-form solution to the long-term (steady-state) of the random walk, which is characterized by the largest eigenvector of Eq. 2.16. Finally, the path length distribution can be computed from the spectrum of the eigenvalues of the adjacency matrix [Biggs, 1993]. These and other results have been used in applications such as diffusion embeddings [Lafon and Lee, 2006], image segmentation [Shi and Malik, 2000] and spectral clustering [Ng et al., 2002]. The principle exploited here is the fact that a random walk starting in a given node has a higher probability to remain within highly connected regions than jumping across weakly connected ones.

As in Sec. 2.3, similar discrete-continuous considerations exist between the analysis of random-walks and continuous-time Markov chains [D.Aldous and Fill:, 2002]. From the continuous point of view, an evolving field $\phi_{(t)}$ can be interpreted as a probability distribtuion describing the likelihood of occupying state i at time t given initial probabilities $\phi(0)$. A random walk in a continuous space defines a linear operator T such that:

$$[T\phi](x) = \int_\Sigma M(y|x)\phi(y)p(y)dy \tag{2.38}$$

where $M(x|y)$ is the transition probability from y to x in infinitesimal time and $p(x)$ explains the distribution of the nodes in the underlying space. As shown in [Coifman and Lafon, 2006], in the limit where $N \mapsto \infty$, the eigenvalues and the extensions of the discrete eigenvectors of the

finite matrix $\mathbf{P}$ converge to the eigenvalues and eigenfunctions of the integral operator T. If there are enough data-points for accurate statistical sampling, the structure and characteristics of the eigenvalues and eigenfunctions are also similar.

Independent of the starting point, in the long term $t \mapsto \infty$ all the points get connected; as a consequence, $\mathbf{P}^t$ becomes an irreducible aperiodic Markov Chain with a stationary probability distribution: $\lim_{t\mapsto\infty}(t, y|x) = \gamma_0(y)$. As it is shown in [Coifman and Lafon, 2006], γ_0 corresponds to the left eigenvector of $\mathbf{P}$ associated to the greatest eigenvalue, and is serves as an empirical density estimate at the point x, since:

$$\gamma_0(x_i) = \frac{D_{ii}}{\sum_j D_{jj}} \tag{2.39}$$

In fact, for a shift invariant kernel $\mathcal{K}(x-y)$, and specially for the Gaussian Kernel, γ_0 is equivalent to the *Parzen window* density estimator[Nadler et al., 2005, Duda and Hart, 1973].

2.4.2 Least-squares optimization problems and regularization

The eigenvalues of Hermitian matrices have been largely used to find the solution of optimization problems that can be related to the *Rayleigh-Ritz ratio*[Horn and Johnson, 1985]. Consider an $N \times N$ Hermitian matrix $\mathbf{M}$, with eigenvalues of $\mathbf{M}$ ordered in increasing value of magnitude $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$. The *Rayleigh-Ritz theorem* states that for a given $N \times 1$ column vector $\boldsymbol{f}$ the following equation holds:

$$\lambda_1 \boldsymbol{f}^\top \boldsymbol{f} \leq \boldsymbol{f}^\top \mathbf{M} \boldsymbol{f} \leq \lambda_N \boldsymbol{f}^\top \boldsymbol{f}. \tag{2.40}$$

Eq. 2.40 allows to characterize the extrema eigenvalues of $\mathbf{M}$ such that:

$$\lambda_{max} = \lambda_N = \max_{\boldsymbol{f}\neq 0} \frac{\boldsymbol{f}^\top \mathbf{M} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}} = \max_{\boldsymbol{f}^\top \boldsymbol{f}=1} \boldsymbol{f}^\top \mathbf{M} \boldsymbol{f}, \tag{2.41}$$

$$\lambda_{min} = \lambda_1 = \min_{\boldsymbol{f}\neq 0} \frac{\boldsymbol{f}^\top \mathbf{M} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}} = \min_{\boldsymbol{f}^\top \boldsymbol{f}=1} \boldsymbol{f}^\top \mathbf{M} \boldsymbol{f}, \tag{2.42}$$

See [Horn and Johnson, 1985] for the proof. In this way, the Rayleigh-Ritz theorem gives a variational characterization of the largest and smallest eigenvalues of the matrix $\mathbf{M}$ and gives a solution to the quadratic optimization problems of the form:

$$\max \frac{\boldsymbol{f}^\top \mathbf{M} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}}, \qquad \text{and} \qquad \min \frac{\boldsymbol{f}^\top \mathbf{M} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}}, \tag{2.43}$$

in terms of these eigenvalues.

To characterize intermediate eigenvalues, the orthogonality constraint is used. If we call the first eigenvector $\boldsymbol{v}_1$, using the constraints $\boldsymbol{f} \neq 0$ and $\boldsymbol{f} \perp \boldsymbol{v}_1$, allows to characterize $\boldsymbol{v}_2$:

$$\min_{\substack{\boldsymbol{f}\neq 0 \\ \boldsymbol{f}\perp \boldsymbol{v}_1}} \frac{\boldsymbol{f}^\top \mathbf{M} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}} = \min_{\substack{\boldsymbol{f}^\top \boldsymbol{f}=1 \\ \boldsymbol{f}\perp \boldsymbol{v}_1}} \boldsymbol{f}^\top \mathbf{M} \boldsymbol{f} = \lambda_2 \tag{2.44}$$

This result can be further extended for the remaining eigenvectors, by always finding an orthogonal solution to the current subspace. Similar considerations can be done for the max case. Notice that

equations from 2.40 to 2.44 apply in the complex case as well, by replacing transpose with the complex conjugate.

We may use the Rayleigh-Ritz theorem to characterize the eigenvalues of the Laplacian matrix $\mathbf{L}$. Expanding the quadratic form $\boldsymbol{f}^\top \mathbf{L} \boldsymbol{f}$, one notices that the eigenvalue problems above actually correspond to a *least-squares energy function* that we note by $\mathcal{F}_\mathbf{L}$:

$$\mathcal{F}_\mathbf{L} = \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} = \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} w_{ij} \big(\boldsymbol{f}(i) - \boldsymbol{f}(j)\big)^2 \tag{2.45}$$

Thanks to the Rayleigh-Ritz theorem, we know that minimum of this energy can be found by the spectral analysis of $\mathbf{L}$ and corresponds to its smallest eigenvalue. When using the normalized Laplacian $\mathcal{L}$, the energy becomes:

$$\mathcal{F}_\mathcal{L} = \boldsymbol{f}^\top \mathcal{L} \boldsymbol{f} = \frac{1}{\sqrt{d_{ii}}} \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} w_{ij} \left(\frac{\boldsymbol{f}(i)}{\sqrt{d_{ii}}} - \frac{\boldsymbol{f}(j)}{\sqrt{d_{jj}}} \right)^2 \tag{2.46}$$

Finally, for the random-walk Laplacian the equivalent function is :

$$\mathcal{F}_\mathbf{P} = \boldsymbol{f}^\top \mathbf{P} \boldsymbol{f} = \frac{1}{\sqrt{d_{ii}}} \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} \frac{w_{ij}}{\sqrt{d_{ii}}} \left(\boldsymbol{f}(i) - \boldsymbol{f}(j)\right)^2 \tag{2.47}$$

If the weights w_{ij} measure similarity between nodes, the criteria in Eq. 2.45, Eq. 2.46 and Eq. 2.47 can be used to model an *smoothness constraints* penalizing vectors that have large differences for neighboring points. Once again the analogy the continuous Laplacian operator appears, which is a generally used as a *regularization* term in optimization problems.

2.4.3 Spectral embedding

An embedding is a method to find a mapping of the available data onto a space (the feature space) where it is easier or more efficient to perform certain tasks. Several approaches exist to build such maps. Some of the embedding techniques are modeled as optimization problems where the embedding tries to find the *optimal* representation of the data. The optimality measures how well the mapping preserves the relevant information for the given task. When the embedding space is restricted to be Euclidean, the optimizations can be expressed as minimum eigenvalue problems, such as the ones presented in 2.4.2.

When dealing with graph representations constructed from data and pairwise similarity measures, the information to be preserved is usually coded in the edge weights and in consequence, contained in the Laplacian matrix of the graph. The preservation criteria can be formulated as a least-squares optimization problem by using the energies defined in Eq. 2.45, Eq. 2.46 and Eq. 2.47. In the one-dimensionall case, we want to find the mapping that optimizes one of the following constrained least-squares problems:

$$\min_{\boldsymbol{f}_1} \mathcal{F}_\mathbf{L} \qquad \text{s.t} \qquad \boldsymbol{f}_1^\top \boldsymbol{f}_1 = 1 \tag{2.48}$$

$$\min_{\boldsymbol{f}_1} \mathcal{F}_\mathcal{L} \qquad \text{s.t} \qquad \boldsymbol{f}_1^\top \boldsymbol{f}_1 = 1 \tag{2.49}$$

$$\min_{\boldsymbol{f}_1} \mathcal{F}_\mathbf{P} \qquad \text{s.t} \qquad \boldsymbol{f}_1^\top \boldsymbol{f}_1 = 1 \tag{2.50}$$

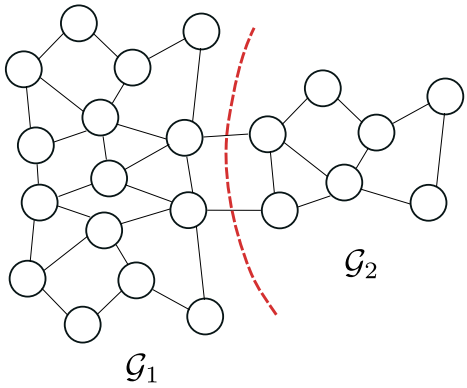


Figure 2.3: Clustering as a graph-cut problem. A graph $\mathcal{G}$ is decomposed in two components $\{\mathcal{G}_1, \mathcal{G}_2\}$ by finding an optimal cut.

The constraint $\boldsymbol{f}_1^\top \boldsymbol{f}_1$ avoids mapping the graph to the trivial solution $\boldsymbol{f} = 0$. According to the Rayleigh-Ritz theorem the solution is given by the eigenvector associated to the smallest eigenvalue. For higher-dimensional mappings, one may impose an orthogonality constraint between the different dimensions of the embedding. As in Sec. 2.4.2 the orthogonality can be incrementally enforced for the remaining dimensions. For example for the second dimension:

$$\boldsymbol{f}_2 = \arg\min_{\boldsymbol{f}_2} \mathcal{F}. \qquad \text{s.t} \qquad \begin{matrix} \boldsymbol{f}_2^\top \boldsymbol{f}_1 = 0 \\ \boldsymbol{f}_2^\top \boldsymbol{f}_2 = 1 \end{matrix} \tag{2.51}$$

The procedure leads to an optimal mapping defined by the set of eigenvectors that correspond to the smallest eigenvalues of the selected Laplacian. The obtained eigenvectors can be used as a basis in which the graph can be represented. Furthermore by choosing only the k first minimal solutions, the eigen-basis may serve as a mapping to a reduced-dimension space that only retains the most important information. For these reasons, graph embedding methods are commonly used to build vector representations of relational data and for dimensionality reduction or de-noising applications.

2.4.4 Spectral clustering

Because spectral graph theory analyzes the global connectivity of the graph, it naturally serves as a support for clustering applications. Clustering is the problem of finding natural non-overlapping groups within a given data-set in unsupervised manner. The challenges of such problems lie first, in its combinatorial nature; second, in the difficulty to define a general criteria to measure the quality of a given partition; and third, on the lack of *a-priori* knowledge on the number of clusters n. A common way to model the problem is to use a graph representation of the data-set (with a vertex per each data-point and edges between every pair of nodes) and to define a pairwise similarity function to weight the edges. The graph is partitioned in two clusters, $\mathcal{G}_1$ and $\mathcal{G}_2$, by "cutting" the edges that link any pair of clusters, as shown for example in Fig. 2.3. The set of removed edges is called a *cut-set* and its cost is defined by the sum over the cut-edge weights:

$$\text{cut}(\mathcal{G}_1, \mathcal{G}_2) = \sum_{i \in \mathcal{G}_1, j \in \mathcal{G}_2} w(i, j). \tag{2.52}$$

An optimal bipartite clustering can be found by minimizing the *Eq.* 2.52 over all possible edge-cut sets, which is known as the "min-cut" criterion. Unfortunately, this naive objective function favors the creation of small clusters towards the boundaries of the graph [Verma and Meila, 2003]. In order to obtain representative clusters, the size of each partition must be taken into account. A common way to address the problem is to balance the cost of the cut against the number of nodes in the partitions (rcut(Δ)) [Hagen and Kahng, 1992], or against the volume of the clusters (ncut(Δ)) [Shi and Malik, 2000] [8], as follows:

$$\text{rcut}(\Delta) = \frac{\text{cut}(\mathcal{G}_k, \mathcal{G}_l)}{\min(|\mathcal{G}_k|, |\mathcal{G}_l|)}, \tag{2.53}$$

$$\text{ncut}(\mathcal{G}_k, \mathcal{G}_l) = \frac{\text{cut}(\mathcal{G}_k, \mathcal{G}_l)}{\text{vol}(\mathcal{G}_k)} + \frac{\text{cut}(\mathcal{G}_k, \mathcal{G}_l)}{\text{vol}(\mathcal{G}_l)}, \tag{2.54}$$

where the volume of a cluster k is defined as the sum of the degrees of the nodes inside the cluster $\text{vol}(\mathcal{G}_k) = \sum_{i \in \mathcal{G}_k} d_i$ and $d_i = \sum_{(i,j) \in \mathcal{E}} w(i,j)$.

The extension of ncut(Δ) to a partition with n clusters ($\Delta = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_n\}$) is known as the *multi-way normalized cut* "mncut" [Shi and Malik, 2000]:

$$\text{mncut}(\Delta) = \sum_{k=1}^{n} \sum_{l=1}^{n} \text{ncut}(\mathcal{G}_k, \mathcal{G}_l). \tag{2.55}$$

Intuitively, mncut measures the quality of a partition, assuming that a good clustering should maximize both the intra-similarity within each group, and the dissimilarity between groups. The criteria to be optimized is:

$$\Delta^* = \arg\min_{\Delta} \text{mncut}(\mathcal{G}). \tag{2.56}$$

Unfortunately, introducing balancing conditions (normalization) makes the previously simple to solve "min-cut" problem become NP hard [Luxburg, 2007]. To solve it, the optimization in Eq. 2.56 can be reformulated in terms of a labeling problem: one looks for a discrete function assigning the label of a cluster to each node. In the case of a bi-partition, the range of the sought function is constrained to a pair of values (*e.g.* γ_1 and γ_2), and Eq. 2.56 can be equivalently rewritten as:

$$\min_{\boldsymbol{f}} \boldsymbol{f}^\top \mathcal{L} \boldsymbol{f} \text{ subject to } \boldsymbol{f}^\top \mathbf{1},\ ||\boldsymbol{f}|| = \sqrt{N},\ \boldsymbol{f} = \begin{cases} \gamma_1 \\ \gamma_2 \end{cases}. \tag{2.57}$$

In order to find an approximate solution to he discrete optimization in Eq. 2.57 (which is still NP-hard), the problem is decomposed in two stages: first, finding a tractable relaxation and second recovering a feasible solution from the relaxed one [Zhang and Jordan, 2008]. The *spectral relaxation*, leading to the *spectral clustering* algorithms, consists of neglecting the discreteness condition, *i.e.* allowing f to take arbitrary values in $\mathbb{R}$. This leads to the relaxed optimization problem:

$$\min_{f \in \mathbb{R}} f^\top \mathcal{L} f \text{ subject to } f^\top \mathbf{1},\ ||f|| = \sqrt{N}. \tag{2.58}$$

As discussed in Sec. 2.4.2, the optimization of Eq. 2.58 yields a Rayleigh quotient. Therefore, in the bipartite case, the relaxed optimal cut can be found by solving an eigenvector problem and analyzing the Fiedler eigenvector [9] (c.f. Sec. 2.2.4).

8. These normalizations correspond to the ones describing the unnormalized and normalized Laplacians.

9. The associated Fiedler eigenvalue value is an indicator of how well connected is a graph $\mathcal{G}$, and how much does a random subgraph of $\mathcal{G}$ look like original graph $\mathcal{G}$ [Meila and Shi, 2001].

The extension of the bipartite problem to multiple clusters, can be achieved either by iteratively applying the bipartite algorithm to each segment, or by using several eigenfunctions at a time. Algorithms in the former category focus on defining an automatic way to go from the real values in the eigenvector to a binary partition [Feng and Perona, 1998, Hagen and Kahng, 1992, Weiss, 1999, Shi and Malik, 2000]. In contrast, the second approach exploits the equivalence between Eq. 2.58 and the optimization of embedding [10] introduced in Sec. 2.4.3. One can therefore use multiple eigenfunctions to map the points to an embedding space where k-means or other simple clustering method is applied to recover the discrete labeling solution [Ng et al., 2002, Meila and Shi, 2001].

In Chap. 6 we explore the applications of spectral clustering for shape and motion analysis of articulated objects.

2.5 Conclusions

This background chapter presents the pillars on which the spectral graph theory is founded. It compiles the required essential material for chapters Chap. 3, Chap. 4 and Chap. 6 of this document, where we present methods having roots in the spectral graph theory and dedicated to the analysis of articulated-shapes.

As shown in Sec. 2.4, the flexibility of spectral graph theory enables us to address a large variety of problems related to the analysis of graph connectivity. The connectivity pattern can be designed in accordance with the desired application. A large class of methods rely on the automatic generation of the connectivity pattern from similarity measures between pairs of data elements. The theory does not impose any conditions on the design of the similarity measure, or in general, on the connectivity pattern of the graph. This is one of the most advantageous features of spectral graph theory: the "important information" for each application can be encoded in a flexible manner.

10. In general, embedding functions are found by optimizing over the preservation of weights. In the graph-cut model of clustering, weights are defined by high values of similarities. Thus, in the context of clustering, the weight preservation results in embedding functions results that map similar points together (ideally to the same point).

Chapter 3

Articulated Shape Representation using Spectral Methods

Contents

3.1 Introduction

As briefly introduced in Chap. 1, we are interested in finding solutions to the problems of *registration and segmentation* in the context of articulated objects. An important step towards this goal consists on building an appropriate *model or representation* of the objects which would facilitate the registration and segmentation tasks. In this chapter we are primarily concerned with finding an *automatic* method to build such a representation.

The proposed approach relies on the *geometric-invariant properties* which characterize the *shape* of any articulated object. In order to capture these properties, we describe the shapes of the objects as graphs and study the invariance of their local geometry, making use of spectral graph theory (c.f. Chap. 2) and in particular, of non-linear spectral embedding methods. This chapter describes in detail the algorithms used for their construction.

In practice, we build such invariant representations from observations of an articulated object. In our case, the observations consist of 3-dimensional data acquired with a multiple-camera system, which is in the form of point-clouds, voxel-sets or mesh-sets. The proposed approach could however be used for other types of data such as silhouettes and range data and could also be extended to handle photometric information when available.

The goal of this chapter is to describe an *automatic* procedure for the elaboration of a representation which facilitates solving the problems of *articulated-shape registration* and *time-coherent body-part segmentation*. The complexity of the articulated-object registration problem lies in the difficulty of comparing or measuring the distance between two different poses of the object. Therefore, a convenient way to create a representation in this case is to characterize the invariance of an articulated object with respect to its natural motion. This representation will also be helpful to ensure the coherence of segmented objects in time.

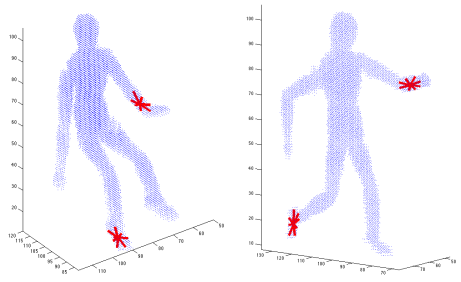
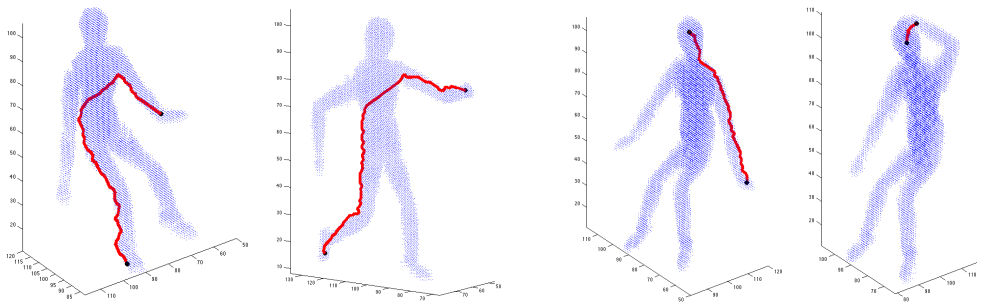


Figure 3.1: The 3D point clouds representing an articulated wooden mannequin in two different poses. The distances between a point and its local neighbors (in red) are not affected by the pose. Articulated motion preserves shape locally.

Ideally, the motion of an articulated object can be approximated by a constrained piece-wise rigid motion which *preserves the limb (local) structure of the object*. This principle is illustrated in Fig. 3.1; locally the shape of does not change from one pose to another. We exploit this local invariance to automatically build a generic invariant representation of the object. In brief, the solution proposed in this chapter uses as input a sample-set of the object's shape. The samples and their distribution in space are used to construct a neighborhood graph with the relevant (local) geometric information encoded in the weights of the graph's edge-set. These adjacency relationships define an *intrinsic distance measure* that is invariant to the pose of an articulated object. Finally, spectral embedding methods (*e.g.* Laplacian eigenmaps) are used to map the sample-set onto a Hilbert space which preserves the local measures as much as possible while removing other unimportant aspects of the shape, *e.g.* pose.

The idea of embedding shapes to create pose-invariant representations has been used before for marker-free motion capture [Chu et al., 2003] and for shape matching [Elad and Kimmel, 2003]. As opposed to our method which is based on the preservation of the local-structure, [Chu et al., 2003, Elad and Kimmel, 2003] rely on the invariance of the geodesic distances to articulated pose; geodesics are also preserved by articulated motion as shown in figure Fig. 3.1-a). However, geodesic distances are difficult to estimate with precision and are very sensitive to topological changes in the graph due to self-contacts. An example of a self-contact and its effect on the geodesic path is shown in Fig. 3.2-b). Furthtermore, [Chu et al., 2003, Elad and Kimmel, 2003] define shape graphs that are completely connected, whose processing is computational expensive when compared to that of locally-connected graphs.

Similar to our approach, Chellappa *et al.* [Sundaresan and Chellappa, 2008] use a local embedding method to uncover the 1-D structure of the 3-D shape as an intermediate step to register data to an a-priori model. As opposed to [Sundaresan and Chellappa, 2008], our method uses directly the



a)Preservation of geodesic distances b)Geodesics are affected by self-contacts

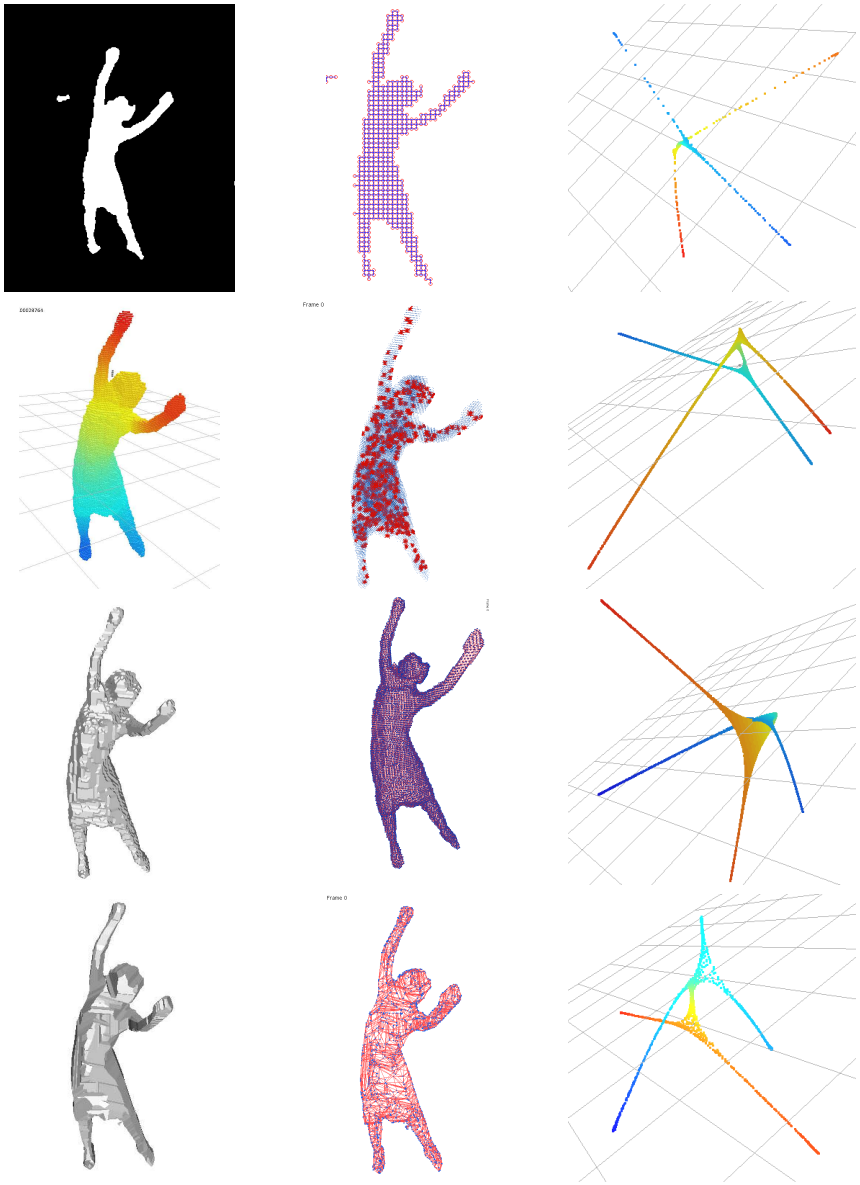
Figure 3.2: A self-contact produces a topological change in the graph. The 3D point clouds representing an articulated wooden mannequin in two different poses. a) The geodesic path between two points of the shape (in red) is not affected by the pose. b) Due to the self-contact of the arm with the head, the geodesic distance between the points changes.

results from the embedding, avoiding the requirement of an a-priori model of the object. A second contemporary related solution was proposed by Rustamov [Rustamov, 2007], who uses a Laplacian-based embedding of a meshed surface to create shape descriptors for shape-classification. Like ours, the works of Chellapa [Sundaresan and Chellappa, 2008] and Rustamov [Rustamov, 2007] use embedding methods based on manifold learning. However, different type of embedding algorithms for representing articulated objects are also possible. For instance, Hilton *et al.* [Starck and Hilton, 2005] proposed to embed both the shape and appearance of 3-D articulated objects to a spherical domain.

In this chapter, we explain how to create an unsupervised representation of articulated shapes through the use of spectral graph theory. In Sec. 3.2.1, we describe the construction of the shape graph. Sec. 3.2.2 explains how the use of spectral theory to analyze the shape graphs is related to non-linear embedding methods. The two relevant embedding algorithms are detailed in Sec. 3.2.2.3 and Sec. 3.2.2.4. Finally a discussion and analysis of the representation are presented in Sec. 3.3.

3.2 Spectral graph theory for shape representation

This section describes first, the procedure to build shape graphs from observations of an articulated shape and second, the spectral graph methods to embed the graphs and obtain the pose-invariant representations. The method consists in modeling the shape of the articulated object as a graph. Then, the graph is then embedded (mapped) to a space which preserves the local structure while removing other unimportant aspects of the shape. Such maps can be generated from the data-samples without supervision, using spectral-graph-embedding methods such as Laplacian Eigenmaps and Locally Linear Embedding (LLE) (c.f. Chap. 2). The resultant representation can be readily used for registration and segmentation. In the case of registration, the result of the embedding can be interpreted as a pose-invariant representation which facilitates the search of correspondences by removing the differences introduced by the pose. Furthermore, the invariant representation allows us to segment the shape consistently either across different poses or in time. Fig. 3.3 illustrates the main steps of the method for different types of input data.



1) Sample the shape. 2) Build shape graphs. 3) Embed shape graphs.

Figure 3.3: The process of building articulated shape representations from four different types of input data: from top to bottom silhouettes, voxel-sets, marching-cubes meshes and polygonal meshes . 1) Sampling observations of an articulated shape from the image, surface or volume of the shape. Samples are respectively, pixels, mesh vertices or 3D points. 2) Using samples as nodes of the shape graph and define local relationships among the nodes to define the edges. 3) Using a non-linear spectral embedding method to create the new representation of the shape .

3.2.1 Shape graphs

In the absence of a built-in model, articulated objects can be naturally represented by graphs with nodes modeling different types of features and edges explaining relationships between the nodes. Examples of graphs for articulated shape representations are shock-graphs [Demirci et al., 2006], reeb-graphs [Reeb, 1946, biasotti et al., 2000] and articulated chain models [Sundaresan and Chellappa, 2006]. In this chapter, we model the shape of the articulated objects by considering the shape samples as nodes. The edges linking the neighboring nodes encode local structure of the shape.

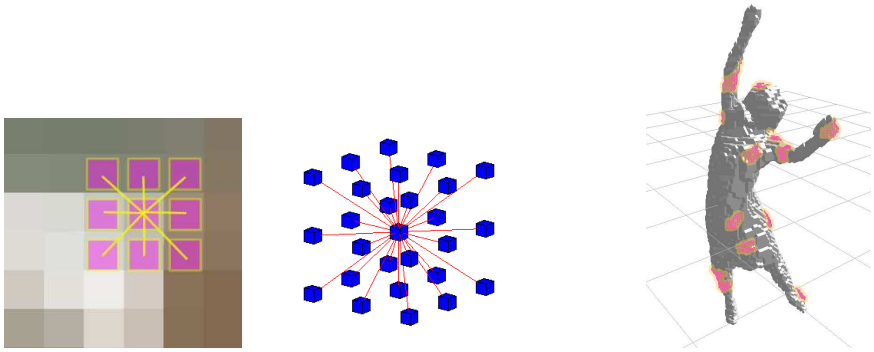
3.2.1.1 Sampling the shape

The features used to build graph-representations of articulated objects can be of different types, *e.g.* color, geometry, *etc.* In this chapter, we focus on the *shape* of the object, *i.e.* on its *geometry* and *topology*. In order to automatically extract features which are representative of objects' shape, we sample the space the occupied by the object. Sampling can be done by a number of methods. The most straightforward method is to use range or laser scans, which sample the surface of the object as a 3-D point-cloud. Alternatively, the samples can be obtained using computer vision techniques, *i.e.* using one or multiple cameras to observe the articulated object. From these images it is then possible to extract silhouettes and/or build a 3-D reconstruction of the object if the cameras are calibrated and synchronized. For the single-view case, the pixels inside the silhouette can be directly used as samples. Using multiple-views one can build 3-D reconstructions that are either voxel-based or a polygonal (mesh-based). In the first case, each voxel is considered as a sample. In the latter, the mesh inherently defines a graph. Once the samples are obtained a graph is built. The nodes of the graph describe each of the samples and some type of distance measure between neighboring samples is used to define and weight the edges.

3.2.1.2 Building shape graphs from samples

Formally, the sampling procedure leads to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where each vertex $i \in \mathcal{V}(\mathcal{G})$ represents one of the N samples ($|\mathcal{V}(\mathcal{G})| = N$), each edge $(i, j) \in \mathcal{E}(\mathcal{G})$ stands for a connection between a pair of nodes ($i, j \in \mathcal{V}(\mathcal{G})$), and each weight w_{ij} holds the geometric information related to the edge (i, j). There are different ways in which the *connectivity pattern* $\mathcal{E}(\mathcal{G})$ of the graph can be constructed. As mentioned in the introduction of this chapter, the local structure of the shape is preserved under articulated motion. In order to capture this local invariance we establish a *local* connectivity pattern by linking each node only to its neighbors. In the examples of Fig. 3.4 the pattern is already determined by the acquisition process. For instance, a 4 or 8 neighborhood can be used to connect the pixels Fig. 3.4-a), and a 6 or 27 neighborhood for a voxel-set,Fig. 3.4-b). Finally, the graph connectivity can be directly inherited from the mesh in Fig. 3.4-c).

When only point-clouds are available, the connectivity pattern is built by assuming that initially the graph is fully connected. To enforce locality, edges with high values, reflecting distant nodes, are pruned to form an ϵ or a KNN neighborhood graph. *ϵ-neighborhood graphs* retain only the edges bellow a certain ϵ value. *K-Nearest-Neighbors (KNN) graphs* fix an identical neighborhood (k) size for every node.The ϵ and k parameters have an important impact on the final representation. In [Cuzzolin et al., 2008], we proposed a heuristic to tune their values relying on the detection of "anomalous" neighborhoods, i.e. neighborhoods which do not correspond to local relationships in terms of the intrinsic geometry of the shape Fig. 3.5-b). The heuristic verifies that the distance



a) 8-neighborhood b) 26 neighborhood c) mesh exemplary neighborhoods.

Figure 3.4: a) an 8-neighborhood in an image, b) a 26 neighborhood in a voxel-set. c) examples of neighborhoods in a mesh.

between the nodes which belong to a neighborhood are all within a comparable range. Anomalies are detected by measuring and comparing the sum of the distances from each point in a given neighborhood ($j \in \mathcal{N}(i)$) to all the others points in the neighborhood:

$$p(j) = \sum_{j,l \in \mathcal{N}(i)} ||\boldsymbol{X}_j - \boldsymbol{X}_l||^2, \tag{3.1}$$

where $\boldsymbol{X}_j$ and $\boldsymbol{X}_l$ are the coordinate values of the sample points represented respectively by nodes i and l. An admissible value of k or ϵ produces similar values of $p(j)$ for every neighborhood. Inappropriate values instead lead to abrupt changes in the $p(j)$ for some neighborhoods. These abrupt changes can be thus detected, as explained in Fig. 3.5, and their corresponding values of k or ϵ rejected.

The graph edge weights $w(i,j) \in \mathcal{W}(\mathcal{G})$ are pairwise *similarity* values computed based on the *distance* $\tilde{w}(i,j)$ between connected nodes:

$$\tilde{w}(i,j) = \begin{cases} \text{dist}_\gamma(i,j) & (i,j) \in \mathcal{E}(\mathcal{G}) \\ \infty & \text{otherwise} \end{cases}, \tag{3.2}$$

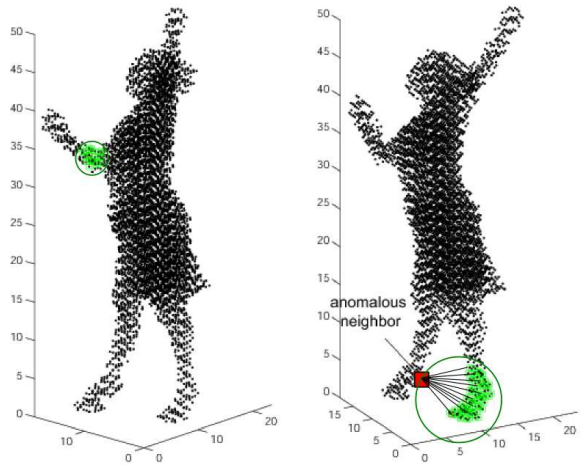
where γ is an index variable to denote the different distance types. Any distance measure can be used as long as it reflects the intrinsic geometry of the samples. Common choices for dist$\gamma(i,j)$ are:

$$\text{dist}_1(i,j) = 1, \tag{3.3}$$

$$\text{dist}_2(i,j) = ||\boldsymbol{X}_i - \boldsymbol{X}_j||^2, \tag{3.4}$$

$$\text{dist}_3(i,j)^{(q)} = \sum_{(k,l) \in path(i,j)^{(q)}} \text{dist}_{\{1,2\}}(i,j) \tag{3.5}$$

where $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ are sample coordinates of nodes i and j; and $path(i,j)^{(q)}$ is collection of edges representing the shortest path between nodes i and j in the initial neighborhood graph. dist_1 is a combinatorial measure assigning ones to all existent edges. dist_2 is the Euclidean distance between the sample coordinates. Finally, the *local geodesic* distance dist_3 approximates the geodesic locally by finding the shortest paths in the neighborhood graph ($path(i,j)^{(q)}$). The neighborhood extension



a) Admissible neighborhood b) Anomalous neighborhood

Figure 3.5: How to select from the data the correct neighborhood size when connectivity is not provided. To build a KNN or an ϵ-neighborhood graph correct values of k and ϵ should be provided. Non-admissible values of k and ϵ are characterized by "anomalous" neighborhoods which do not reflect local relationships. a) An admissible neighborhood showed in green. b) An anomalous neighborhood is detected because the sum of the distances from the red point to all other green points is very large compared to the other values of the neighborhood.

is implemented as marching front, which iteratively updates the edge-subset $(i,j) \in \mathcal{E}(\mathcal{G})$ associated to a node i, by adding at each iteration (q) the connection to the immediate surrounding neighbors [1] following Eq. 3.6. dist_3 is designed to capture information at different scales according to the values of q.

$$\mathcal{E}^{(q+1)} = \left\{ \mathcal{E}^{(q)} \cup (i,l) | (i,j) \in \mathcal{E}^{(q)}, (j,l) \in \mathcal{E}^{(0)} \right\}. \tag{3.6}$$

Distances are transformed to similarity measures using a Gaussian kernel:

$$w(i,j) = \exp^{-\frac{\tilde{w}(i,j)}{2\sigma^2}}, \tag{3.7}$$

where σ is a parameter that controls the ratio of influence of the distances. Since it is characteristic of the sampling of the shape it can be automatically obtained from the statistics of data, for example using the median:

$$\sigma = \text{median}\ (\text{dist}_\gamma(i,j)) \quad (i,j) \in \mathcal{E}(\mathcal{G}) \tag{3.8}$$

One great advantage of defining our shape graphs with a local connectivity pattern is the fact that its corresponding adjacency matrix is sparse (Fig. 3.7). This is convenient for the forthcoming spectral analysis.

3.2.2 Spectral analysis of shape graphs

As explained in Chap. 2, the *spectral graph theory* is a general tool for analyzing connectivity of graphs. After the graph assembly Sec. 3.2.1, a second step towards the automatic construction of an

1. This corresponds to a 1-ring expansion in the case of manifold meshes

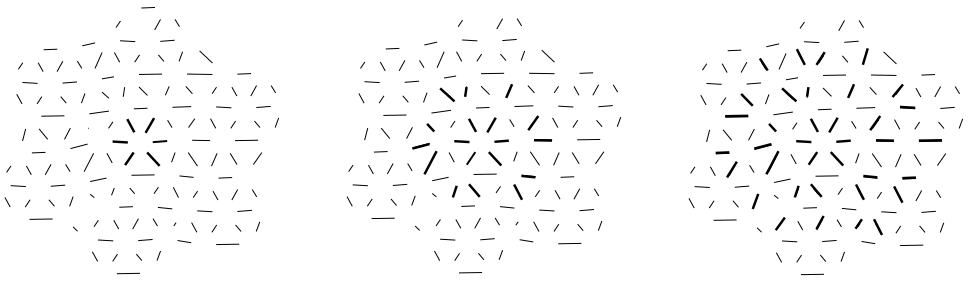


Figure 3.6: Front propagation to efficiently approximate the local geodesic distances. Three iterations of the algorithm starting from the center (blue) point. a) Iteration 1, only those nodes included in the initial neighborhood graph (light gray lines) are considered. b) c)At iteration two and three, the neighbors of the previous iteration are included in the new graph connectivity. The pairwise distances are estimated by summing the distances along the shortest paths (dark lines).

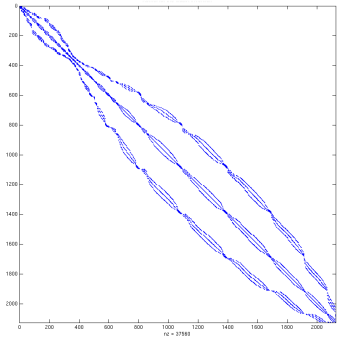


Figure 3.7: Adjacency matrices of neighborhood graphs are sparse.

invariant representation of articulated shapes, is to perform a spectral analysis of the shape graph's connectivity. Intuitively, since only the local connections have been retained, the connectivity patterns of two shape graphs obtained from different poses of the same object should be similar. Fig. 3.8 illustrates the principle.

Because a graph can be exactly specified by its spectrum and the associated eigenvectors, we can use the projection in the eigenspace to fully represent the graph. Since the geometrical information is contained in the connectivity pattern of the shape graphs (by construction), and the connectivity of a graph is reflected in its eigenvalues and eigenvectors, the projection of a shape-graph into the eigenspace reflects the geometric properties of the shape.

The projection of graphs into their eigenspace has been studied in the machine learning community, where non-linear graph embedding methods based on manifold learning have been extensively used for dimensionality reduction applications. There is a strong link between these methods and the procedure described in this chapter to obtain our representations of articulated shapes.

In the next section we study in more detail the graph-embedding application of the spectral graph theory, introduced in section Chap. 2. Starting from a formal definition, passing through a fast

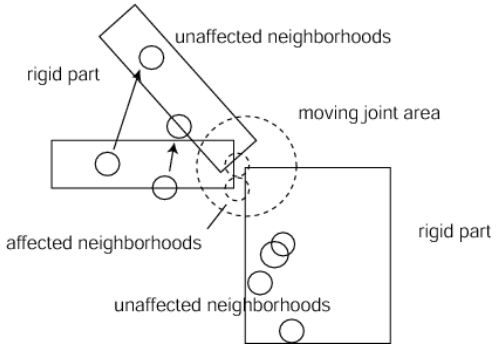


Figure 3.8: The structure of neighborhoods in rigid parts is preserved. Neighborhoods in joints or self-contacts may however be affected by non rigid deformations.

review of the most popular methods, and finally explaining why spectral graph embedding methods are useful for shape representation. Our invariant representation for articulated shape registration and segmentation, is based on the *Laplacian Eigenmaps* and the *Locally Linear Embedding* methods, detailed in Sec. 3.2.2.3 and Sec. 3.2.2.4 respectively.

3.2.2.1 Spectral graph embedding methods

A graph embedding is an injective map Φ of the vertices of the graph $\mathcal{V}(\mathcal{G})$ to some *feature space* $\mathcal{F}$, *i.e.* $\Phi : \mathcal{V}(\mathcal{G}) \mapsto \mathcal{F}$. The map (Φ) is found by the optimization of a criterion related to the preservation of some properties of the graph, usually the similarity relationships between the nodes. Embedding methods generally describe both the design of the graph construction as well as the algorithm to find the map function. The approaches can be characterized according the following criteria:

Type of input data: In most of the cases input data lives in a vector space. However, embedding methods based on pairwise similarity measures are more general and can also be applied to relational data.

Type of mapping: The mapping determines which type of relationships between the data samples are supported by the method. One of the most important distinctions is between methods that support only linear relationships and those which can handle non-linearities. In the first case, the map corresponds to a linear transformation. In the latter case, the support of non-linearities comes at the price of more complex mappings which are usually non-invertible.

Type of feature space: *metric* methods constrain the embedding space to be metric, while *non-metric* approaches allow for arbitrary target spaces.

Map optimization method: The most common approaches include heuristic methods, iterative optimization methods (*e.g.* gradient descent) and spectral methods.

Preservation criteria: Both the graph design as well as the conditions imposed on the embedding map lead to the preservation of different properties of the graph, *e.g.* variance of data points, the neighborhood relationships, geodesic distances, *etc.*

We will now focus on the spectral solution to the embedding problem. Spectral methods cast the optimization of the map into an eigenvalue problem (c.f. Sec. 2.4.3). There is a closed-form

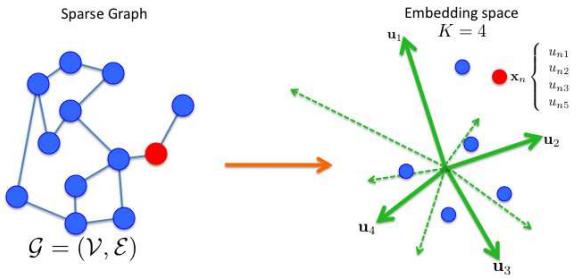


Figure 3.9: From a shape graph to a representation in an embedding or feature space

solution which is not affected by local-minima (like the iterative methods) and thus guarantees that a global solution is always found. The target space $\mathcal{F}$ is obtained from the eigen-basis formed by the eigenvectors of the matrix describing the graph's connectivity (*e.g.* the adjacency or the Laplacian matrix). Therefore, the embedding space is metric and its Euclidean distance measure reflects the information encoded in the weight edges of the graph. Fig. 3.9 illustrates the input and output of a typical spectral embedding method. There exists several s̈pectralälgorithms which provide a solution based on the spectral analysis of graph matrices. We now discuss briefly the most relevant methods.

The *Principal Component Analysis (PCA)* algorithm is a linear embedding method which finds the optimal embedding space by minimizing the squared reconstruction error or, equivalently, by enforcing the preservation of the data variance. As with most of the following embedding approaches, PCA is mainly used to reduce the dimensionality of data living in a high-dimensional vector space by mapping the input data points to a lower-dimensional subspace. The mapping function is a linear transformation followed by a subspace selection. A generalization to relational data can be obtained by preprocessing similarity relationships within the Multidimensional Scaling (MDS) framework [Cox and Cox, 2001]. MDS finds a metric representation which preserves pairwise similarities. Non-linear extensions for PCA include the *Kernel PCA* method proposed by Schöelkopf [Schölkopf et al., 1998] and several manifold-learning methods for embedding.

Manifold-learning methods are founded on the *manifold assumption* (c.f. Sec. 3.2.2.2), which states that structured data takes the form low-dimensional manifold living in a high-dimensional space. The structure of the manifold is exploited to recover a lower-dimensional representations of the data. Methods such as Isomap [J. B. Tenenbaum and Langford, 2000], Local Linear Embedding (LLE) [Roweis and Saul, 2000], Laplacian and Hessian Eigenmaps [Belkin and Niyogi, 2003, D.Donoho and Grimes, 2003] belong to this category and have an spectral solution. Saul *et al.* [Saul et al., 2006] study in depth the spectral embedding methods in the context of dimensionality reduction.

Isomap [J. B. Tenenbaum and Langford, 2000] finds an embedding which preserves *geodesic* distances along the manifold. Then, a classical metric MDS is applied to the matrix of pairwise geodesic distances. MDS minimizes a stress function that measures the deviation between the geodesic distances in the original space and the Euclidean distances in the embedding space. As a result, the Euclidean distance in the embedding space reflects the geodesic in the manifold.

Instead of trying to preserve the global aspect of the data, several of the manifold learning methods opt for preserving the local structure, which naturally leads to non-linear embeddings. Examples of local-preserving algorithms are Locally Linear Embedding, Laplacian and Hessian

Eigenmaps, and Diffusion maps. These local methods rely on the structure of the manifold to extract the global information. *Locally Linear Embedding (LLE)* [Roweis and Saul, 2000, Saul et al., 2003] assumes that small neighborhoods can be approximated by linear manifolds, in which the position of each point can be reconstructed from the weighted linear combination of its nearest neighbors. Then the algorithm looks for the lower-dimensional space that best preserves the reconstructing weights (the barycentric coordinates relative to its neighbors).

The *Eigenmaps* methods compute a low dimensional representation of the data-set that optimally preserves the local neighborhoods. The optimization is done by minimizing a quadratic form: either the squared gradient (Laplacian) [Belkin and Niyogi, 2003] or the squared Hessian [D.Donoho and Grimes, 2003], over all functions mapping the manifold into the embedding space. Once again, the optimization problem becomes a sparse matrix eigenvalue problem and is readily solved.

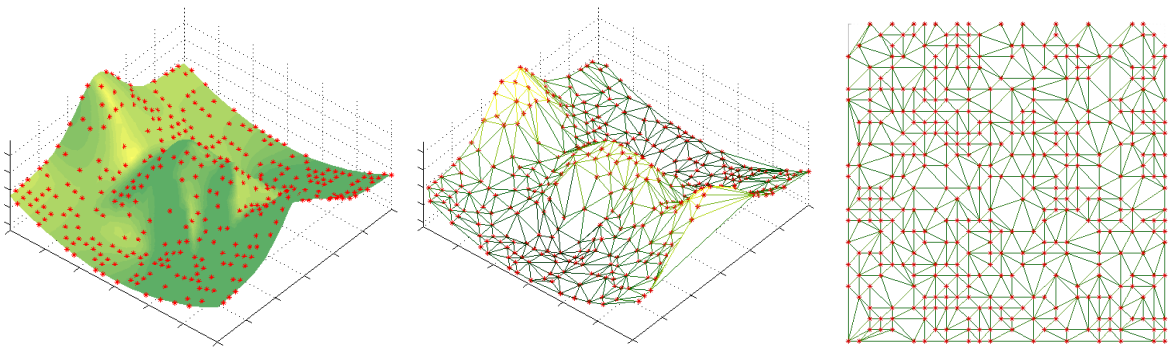
Diffusion maps [Kondor and Lafferty, 2002, Lafon and Lee, 2006, Nadler et al., 2007] follows the same construction from the previous methods, but is inspired on a probabilistic set-up. The goal is to find an embedding that preserves the *diffusion distance*, which is intimately related elated to the random-walk interpretation of the Laplacian operator (c.f. Sec. 2.4.1). As a result, the *diffusion distance* between nodes in the original space is converted to dot products that correspond to the Euclidean distance in the embedding space.

Manifold-learning spectral-embedding methods are closely related [Weiss, 1999, Williams, 2001, Bengio et al., 2004b]. Donoho and Grimes [D.Donoho and Grimes, 2003] identify LLE with an empirical implementation of the Laplacian principle, while Belkin[Belkin, 2003], finds some aproximation relating the two. Scholkopf [Ham et al., 2004] has shown that although Isomap, LLE and Laplacian Eigenmaps have different motivations and derivations they can all be studied within the kernel framework. In the three cases, the local structure of the data is described as a graph and used as constraints to define a global mapping of the manifold into a lower dimensional space. Finally, Bengio *et al.* [Bengio et al., 2004a] relate LLE, Isomap, MDS and the eigenmaps to spectral clustering, and formalize a generic algorithm which essentially follows three steps: first, sample the manifold and find neighbors for each sample; second, compute a Gram matrix and third, solve eigenvalue problem for the Gram matrix. The main difference lies in the way the gram matrix is computed.

3.2.2.2 The geometric interpretation of the manifold assumption

As mentioned before, the manifold learning approaches build on the assumption that structured data has the form of a low-dimensional manifold $\mathcal{M}$ living in a high-dimensional space (Fig. 3.10-a). In general, the goal of embedding algorithms in this context is to recover a low-dimensional representation (Fig. 3.10-c) for the data exploiting the manifold structure. Input data is considered to be a point-set $\mathcal{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\}$ sampled from manifold (Fig. 3.10-a). Since the manifold structure is not known in advance, its geometry and topology have to be approximated from the data and neighborhood relationships (Fig. 3.10-b), *e.g.* distances along the manifold (where the data lives) can be approximated by shortest paths in the neighborhood graph. Relying on the structure of the manifold, *local* embedding methods capture the global topology as an aggregation of the local geometry. As a consequence, their performance is highly dependent on the quality of the sampling, *i.e.* better results are obtained if the samples are evenly distributed over a compact manifold with little noise.

Since the geometric information is taken from local neighborhoods, most of the local embedding methods define optimization criteria for the mapping functions, related to the preservation of the



a) Low-dimensional manifold $\mathcal{M}$ in a high-dimensional space and samples $\mathcal{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\}$ of the manifold (in red).

b) Approximating the manifold with a neighborhood graph.

c) Low-dimensional representation of the approximated manifold.

Figure 3.10: The manifold learning approach for embedding and dimensionality reduction.

local structure of the manifold. From a differential point of view, the manifold assumption is equivalent to restricting the mapping functions to be smooth with respect to the underlying geometry. In other words, a mapping function f should verify:

$$f(\boldsymbol{X}_i + \delta) = f(\boldsymbol{X}_i) + \epsilon \qquad \forall \boldsymbol{X}_i \in \mathcal{X} \tag{3.9}$$

with δ and ϵ small bounded numbers.

If we consider the continuous manifold and continuous mapping functions f defined on it, we can relate the manifold assumption to the Laplace Beltrami operator $\triangle_{\mathcal{M}}$ on the manifold. In the continuous setting the local geometry preservation constraint is satisfied by functions $f : \mathcal{M} \rightarrow \mathbb{R}$ mapping nearby points on $\mathcal{M}$ onto nearby values in $\mathbb{R}$. Intuitively, this achieved if f has a small gradient, *i.e.* if moving point $\boldsymbol{X}_i$ on $\mathcal{M}$ results on small changes of $f(\boldsymbol{X}_i)$. Finding functions with minimal gradient over $\mathcal{M}$ can be done by solving for the eigenfunctions of the Laplace-Beltrami operator $\triangle$ defined on $\mathcal{M}$. Indeed, the Laplace-Beltrami operator $\triangle_{\mathcal{M}}$ is defined as the divergence $(\nabla\cdot)$ of the gradient (∇) of a function f on $\mathcal{M}$, *i.e.*:

$$\triangle_{\mathcal{M}} f = \nabla \cdot (\nabla f) \leq \delta. \tag{3.10}$$

Thus, $\triangle_{\mathcal{M}} f$ measures the rate of change of the gradient ∇f. Constraining this quantity to be small implies enforcing f to be smooth.

In the discrete case, the graph Laplacian can be seen as an approximation to the Laplace-Beltrami operator (if the manifold is uniformly sampled). Finding discrete mapping functions that preserve the local geometry is equivalent to searching for smooth discrete functions $\boldsymbol{f}$ defined on the graph, *i.e.* functions whose values do not vary significantly (below a small δ value) between neighboring (adjacent) vertices:

$$\boldsymbol{f}(i) - \boldsymbol{f}(j) \leq \delta \quad \forall (i,j) \in \mathcal{E}(\mathcal{G}) \tag{3.11}$$

The constraint in Eq. 3.11 can be expressed in terms of the the graph Laplacian operator $\mathbf{L}$ described in Sec. 2.3. The Laplacian embedding algorithm introduced in the following section (Sec. 3.2.2.3)

uses shows how to use $\mathbf{L}$ to constraint the preservation of neighborhoods. In fact, finding mapping functions that satisfy Eq. 3.11 leads to embeddings that preserve the local structure and are able to capture and model non-linear relations between samples efficiently. This is one of the reasons why these methods are of interest for shape analysis. In the following two sections we describe in more detail the two non-linear and local spectral embedding methods used in this document to create representation of articulated shapes that can be used for registration and segmentation.

3.2.2.3 The Laplacian embedding method

The Laplacian embedding method is the straightforward result of the spectral analysis of the Laplacian matrices describing the shape graphs $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$ constructed in Sec. 3.2.1. The method looks for the embedding functions which optimally preserve the local structure of the neighborhood graph, such that connected nodes stay as close together as possible after the embedding. Let the embedding Φ be formed of a collection of discrete functions $\Phi = [\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_N]$, one per dimension in the feature space.

For a one dimensional embedding, the local preservation can be expressed as the minimization of the following criteria:

$$\boldsymbol{f} = \arg\min_{\boldsymbol{f}} \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} w_{ij}\Big(\boldsymbol{f}(i) - \boldsymbol{f}(j)\Big)^2 \qquad i,j \in \mathcal{V}(\mathcal{G}) \qquad w_{ij} \in \mathcal{W}(\mathcal{G}) \tag{3.12}$$

The problem can be cast into a least-squares optimization:

$$\boldsymbol{f} = \frac{1}{2}\arg\min_{\boldsymbol{f}} \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} w_{ij}\Big(\boldsymbol{f}(i) - \boldsymbol{f}(j)\Big)^2 = \arg\min_{\boldsymbol{f}} \boldsymbol{f}^\top \mathbf{L}\boldsymbol{f} \tag{3.13}$$

under two constraints. The first, enforces the orthogonality and removes the arbitrary scaling factor $\boldsymbol{f}^\top \mathbf{D}\boldsymbol{f} = \mathbf{I}$. The second, ensures translation invariance $\boldsymbol{f}^\top \mathbf{D}\mathbf{1} = 0$ (with $\mathbf{D}$ the degree matrix introduced in Sec. 2.2.2). Extending the problem to higher dimensions we obtain:

$$\Phi = \frac{1}{2}\arg\min_{\Phi} \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} w_{ij} \parallel \Phi(i) - \Phi(j) \parallel^2 \tag{3.14}$$

$$\Phi = \arg\min_{\Phi} \operatorname{trace}(\Phi^\top \mathbf{L}\Phi). \tag{3.15}$$

As explained in Sec. 2.4.2, the square optimization problems in Eq. 3.15 can be minimized by solving for the smallest eigenvectors of the *graph Laplacian* $\mathbf{L}$ and using the eigenvectors as embedding functions; this is known as the *Laplacian embedding algorithm* [Belkin and Niyogi, 2003] which is summarized in Algorithm 1.

3.2.2.4 The Locally-Linear Embedding method (LLE)

Similarly to the Laplacian embedding, the *Locally Linear Embedding* (LLE) [Roweis and Saul, 2000] computes an embedding Φ which preserves the local structure of the point-set. For each data point $\boldsymbol{X}_i$ in the sample-set $\mathcal{X}$, the algorithm computes the weights C_{ij} that best linearly reconstruct $\boldsymbol{X}_i$ from its neighbors, by solving the following constrained least-squares problem:

$$C_{ij} = \arg\min_{C_{ij}} \sum_i ||\boldsymbol{X}_i - \sum_{(i,j)\mathcal{E}(\mathcal{G})} C_{ij}\boldsymbol{X}_j||^2 \tag{3.16}$$

Algorithm 1 The Laplacian embedding algorithm.

1. Build the neighborhood shape graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$:
   - 1.1. Use the elements of the sample-set $\mathcal{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\}$ as the nodes $\mathcal{V}(\mathcal{G})$.
   - 1.2. Establish the connectivity $\mathcal{E}(\mathcal{G})$ using the samples natural connectivity or KNN or ϵ-neighborhoods.
   - 1.3. Compute the edge weights $\mathcal{W}(\mathcal{G})$ as similarities between data points based on an intrinsic distance measure (*e.g.* local geodesic distances).
2. Build the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ with $\mathbf{D}$ being a diagonal Degree matrix with elements: $d_{ii} = \sum_j w_{ij}$.
3. Solve the eigensystem $\mathbf{L}\boldsymbol{v} = \lambda \mathbf{D}\boldsymbol{v}$.
4. Use the eigenvectors $\boldsymbol{v}_j$, with $j \in \{1, \cdots, N\}$ to form the embedding $\Phi = \{\boldsymbol{f}_1, \cdots \boldsymbol{f}_N\}$. Each $\boldsymbol{f}_j$ is a function $\boldsymbol{f}_j : \mathcal{V}(\mathcal{G}) \mapsto \mathbb{R}^N$. The embedding for a node $i \in \mathcal{V}(\mathcal{G})$ is defined as $\Phi(i) = [\boldsymbol{v}_1(i), \boldsymbol{v}_2(i), \ldots, \boldsymbol{v}_N(i)]^\top$, where the notation $\boldsymbol{v}_j(i)$ indicates the i-th entry of the $\boldsymbol{v}_j$ vector.
5. As a result, each sample is mapped to a point in $\mathbb{R}^N$. The embedded graph leads to the point set $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$.

The embedding functions $\boldsymbol{f}_i$ that form the map $\Phi = [\boldsymbol{f}_1, \cdots \boldsymbol{f}_N]$ are obtained by solving:

$$\arg\min_{\Phi} \sum_i ||\Phi(\boldsymbol{X}_i) - \sum_j C_{ij}\Phi(\boldsymbol{X}_j)||^2, \tag{3.17}$$

where $\boldsymbol{x}_i = \Phi(i) = [\boldsymbol{f}_1(i), \boldsymbol{f}_2(i), \ldots, \boldsymbol{f}_N(i)]^\top$. Similarly to the Laplacian embedding algorithm Sec. 3.2.2.3, a closed form solution for the embedding map Φ can only be found by constraining Eq. 3.17. Thus, $\Phi(\mathcal{X})$ is constrained to be centered at the origin $\sum_i \Phi(\boldsymbol{X}_i) = \mathbf{0}$ and to have unit covariance $\Phi^\top\Phi = \mathbf{I}_{N\times N}$. In this way the translational and rotational degrees of freedom are removed and the orthogonality between the different dimensions enforced. The objective function can be expressed as a quadratic form $\sum_{ij} M_{ij}\boldsymbol{f}_i \cdot \boldsymbol{f}_j$ with M_{ij} the elements of the matrix $\mathbf{M}$ defined as follows:

$$\mathbf{M} \doteq (\mathbf{I} - \mathbf{C})^\top(\mathbf{I} - \mathbf{C}), \tag{3.18}$$

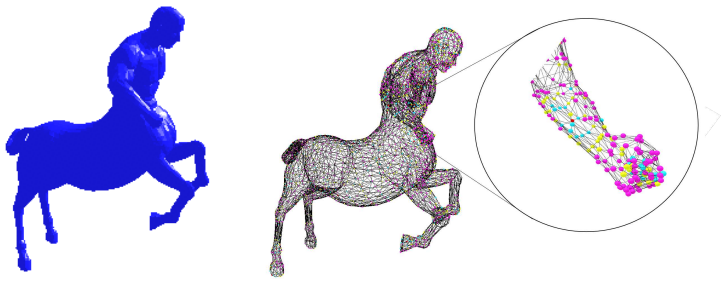
where the entries of the matrix $\mathbf{C}$ are the weights C_{ij} found in Eq. 3.16 the The optimal embedding $\Phi = [\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N]$ (up to a global rotation) is found by computing the eigenvectors of $\mathbf{M}$, and discarding the bottom (unitary) one. The algorithm is summarized in Algorithm 2.

The LLE algorithm is related to Laplacian embedding and it thus shares some of its properties. Belkin [Belkin, 2003] has suggested that the affinity matrix $\mathbf{M}$ can be approximated by the square Laplacian $\mathbf{M}\boldsymbol{f} = (\mathbf{I} - \mathbf{C})^\top(\mathbf{I} - \mathbf{C})\boldsymbol{f} \approx \frac{1}{2}\mathcal{L}^2\boldsymbol{f}$. The main practical difference between the LLE algorithm (Algorithm 2) and the Laplacian embedding (Algorithm 1) consists in the way the weights are built. Another relevant difference is the fact that LLE requires the initial sample-set to be points in a vector space in order to calculate the reconstruction weights; the Laplacian embedding can be performed from more general pairwise similarity relationships. Finally, as the Laplacian embedding is founded on spectral graph theory and its relation to continuous functional analysis, it has a stronger mathematical support than LLE.

Algorithm 2 The locally-linear embedding algorithm

1. Build the neighborhood shape graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$ as in Algorithm 1.
2. For each sample point in $\mathcal{X}$, find the best reconstructing weights C_{ij} using Eq. 3.16
3. Build the matrix $\mathbf{M} \doteq (\mathbf{I} - \mathbf{C})^\top (\mathbf{I} - \mathbf{C})$ (c.f. Eq. 3.18).
4. Find the the eigenvectors $\boldsymbol{v}$ and eigenvalues λ of the matrix $\mathbf{M}$ solving for $\mathbf{M}\boldsymbol{v} = \lambda \boldsymbol{v}$
5. Use the eigenvectors $\boldsymbol{v}_j$, with $j \in \{1, \cdots, N\}$ to form the embedding $\Phi = \{\boldsymbol{f}_1, \cdots \boldsymbol{f}_N\}$. Each $\boldsymbol{f}_j$ is a function $\boldsymbol{f}_j : \mathcal{V}(\mathcal{G}) \mapsto \mathbb{R}^N$. The embedding for a node $i \in \mathcal{V}(\mathcal{G})$ is defined as $\Phi(i) = [\boldsymbol{v}_1(i), \boldsymbol{v}_2(i), \ldots, \boldsymbol{v}_N(i)]^\top$, where the notation $\boldsymbol{v}_j(i)$ indicates the i-th entry of the $\boldsymbol{v}_j$ vector.
6. As a result, each sample is mapped to a point in $\mathbb{R}^N$. The embedded graph leads to the point set $\varkappa = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$.

3.2.2.5 Embedding visualization

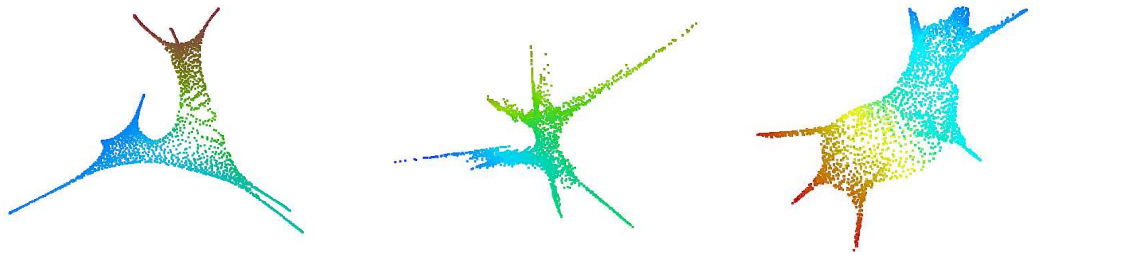


a) Original mesh b) Shape graph

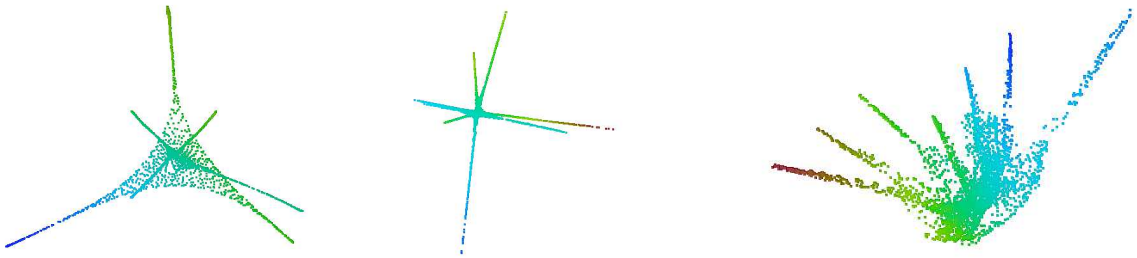
Figure 3.11: Example mesh of representing a centaur and its corresponding shape graph. Shape from the Tosca database (http://tosca.cs.technion.ac.il/data.html)

The two embedding algorithms described in the previous section (Algorithm 1 and Algorithm 2) take as input the shape graphs of an object as shown in Fig. 3.11 (c.f. Sec. 3.2.1) and give as a result a set of embedding functions $\Phi = \{\boldsymbol{f}_1, \cdots, \boldsymbol{f}_N\}$. Mapping the initial nodes to the feature space using the embedding functions leads to a point-set in the feature space (c.f. Fig. 3.9). The point-sets can be therefore visualized as 3-D subspace projections. In Fig. 3.12, we illustrate the outcome of the Laplacian and LLE embedding in comparison with those of the Isomap embedding, applied to the shape graph of the centaur in Fig. 3.11.

As opposed to the Laplacian and the LLE algorithms sharing the same neighborhood (local) graph, Isomap uses a fully-conected graph with the geodesic distances between pairs of nodes as edge-weights. The locality of the neighborhood graph leads to sparse adjacency matrices whose spectral analysis can be performed very efficiently. Furthermore, as highlighted by Scholkopf *et al.* [Ham et al., 2004] Isomap is based upon shortest paths on the graph induced by the data points, whereas Laplacian Eigenmaps uses commute times of a Markov chain on the graph. In other words, Laplacian-based methods do not consider only the shortest path but an integrate over all paths



a) Laplacian (dims 1,2,3) b) LLE (dims 1,2,3) c) Isomap (dims 0,1,2)



a) Laplacian (dims 4,5,6) b) LLE (dims 4,5,6) c) Isomap (dims 3,4,5)

Figure 3.12: Result of embedding the shape graph of the centaur in image Fig. 3.11 using three different non-linear spectral embedding methods: a) Laplacian embedding, b) Locally Linear Embedding, c) Isomap. Only the 6 eigenvectors corresponding to the most significant eigenvalues in each case are shown. The embedding results are shown as 3-D projections on groups of 3 selected dimensions.

connecting points on the graph. As a consequence, they are more robust to noise and topological changes in the graph.

The results from the embedding methods lead to a new representation of the shape which have good properties for shape analysis. In particular, the representation of the Laplacian embedding algorithm has the property of being isometry invariant. In the next section we will discuss some of these properties based on the interpretation of the Laplacian Matrix as an operator.

3.2.2.6 Embeddings and eigenfunctions

An alternative way to interpret the result of an embedding is by thinking of the mapping function as individual descriptions of the shape. As discussed in Sec. 2.3, the spectral analysis of a graph can be interpreted in the context of functional analysis and linear operators. For a given graph $\mathcal{G}$, its Laplacian $\mathbf{L}$ and normalized Laplacian matrices $\mathcal{L}$ can be seen as a discrete operators on the space of functions $\boldsymbol{f} : \mathcal{V}(\mathcal{G}) \mapsto I\!R$ [Chung, 1997] which satisfy:

$$[\mathbf{L}\boldsymbol{f}](i) = \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} w_{ij}\Big(\boldsymbol{f}(i) - \boldsymbol{f}(j)\Big) \tag{3.19}$$

$$[\mathcal{L}\boldsymbol{f}](i) = \frac{1}{\sqrt{d_{ii}}} \sum_{(i,j)\in\mathcal{E}(\mathcal{G})} \left(\frac{\boldsymbol{f}(i)}{\sqrt{d_{ii}}} - \frac{\boldsymbol{f}(j)}{\sqrt{d_{jj}}} \right) \tag{3.20}$$

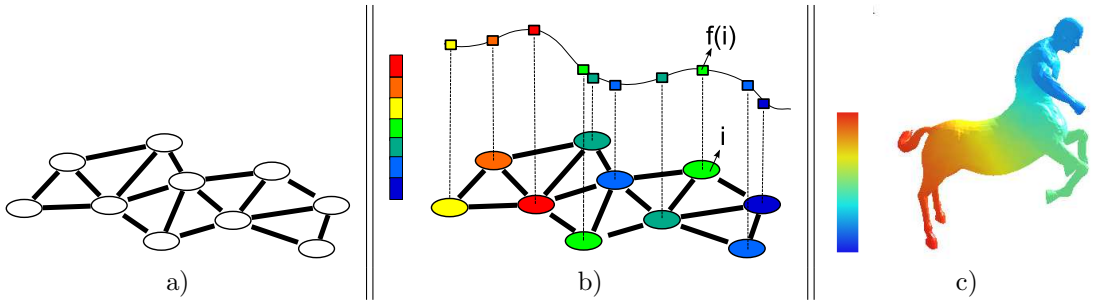


Figure 3.13: Notion and visualization of an eigenfunction. a) Initial shape graph b) Each eigenfunction $\boldsymbol{f}_j$ assigns a value to every node i in the graph. b-c) Given the arbitrary order of the nodes, it is common to visualize the result using the values of the eigenfunction as a colorscale: each sample point i in the original shape is painted with the color corresponding to the value of $f(i)$. In (c), the visualization of the first non-constant eigenfunction of the Laplacian embedding of the shape in Fig. 3.11 is shown.

The eigenvectors of $\mathbf{L}$ and $\mathcal{L}$ can be interpreted as discrete functions assigning a real value to each vertex of the graph $\boldsymbol{f} : \mathcal{V}(\mathcal{G}) \mapsto I\!R$, as illustrated in Fig. 3.13. In this sense, each $\boldsymbol{f}$ can be called an *eigenfunction*. Eigenfunctions are the function-space equivalent of eigenvectors.

As mentioned in Sec. 3.2.2.2, in the context of linear operators the embedding algorithms described in Sec. 3.2.2.3 and Sec. 3.2.2.4 can be seen as procedures to find a collection of smooth mapping functions $f : \mathcal{X} \mapsto I\!R$ defined on the shape samples $\mathcal{X}$. Eq. 3.21 shows a comparison between the optimization problem leading to the Laplacian embedding can be compared to the spectral analysis of the continuous Laplace $\triangle$ and Laplace-Beltrami operators $\triangle_{\mathcal{M}}$:

Continuous	*Discrete*
$\min_{f\in L_2(\mathcal{M})} \int_{\mathcal{M}} f(X)\triangle_{\mathcal{M}} f(X)\mathrm{d}a$	$\min_{f\in I\!R^N} \boldsymbol{f}^\top L_{\mathcal{X}} \boldsymbol{f}$
$s.t. \quad \|\|f\|\|^2_{L_2(\mathcal{X})} = 1$	$s.t. \quad \boldsymbol{f}^\top \boldsymbol{f} = 1$

(3.21)

with da a differential element of the manifold $\mathcal{M}$. Eq. 3.21 also applies for the Laplace operator $\triangle$, by defining the integral over the Euclidean space [2].

Establishing an analogy between symmetric matrices and certain linear operators (and respectively between eigenvectors defined on a vector space and eigenfunctions on a function space) is of great interest to shape analysis. Theoretical studies on shape analysis on planar plates and compact manifolds, have shown that the spectral analysis of the Laplace-Beltrami operator reveals important information about the shape. For instance, Kac [Kac, 1966] demonstrated that it is possible

2. The Laplace-Beltrami operator $\triangle_{\mathcal{M}}$ generalizes the Laplace $\triangle$ to non-Euclidean metrics and thus is appropriate for the description of a manifold $\mathcal{M}$.

to obtain the total Gaussian curvature, the Euler characteristic and the area of a vibrating surface, from the spectrum of the Laplace-Beltrami operator applied to the surface of the plates. For more complex shapes the Laplace-Beltrami operator can be assimilated to the Fourier operator defined on a manifold[Levy, 2006, Vallet and Lévy, 2008].

The geometric properties of the eigenfunctions of the Laplacian operator are suitable for other shape-related applications. In particular, in the Computer Graphics community, where the work of Floater on shape preserving parameterization of meshes [Floater, 1997] has inspired their use for mesh editing and processing [Floater and Hormann, 2005, Sorkine, 2006a]. Recent works have also explored their applicability for shape analysis, including spectral clustering and matching of shapes [Jain and Zhang, 2007, Rustamov, 2007, Mateus et al., 2008, Cuzzolin et al., 2008].

In practice, the analogy with the Laplace operator gives a theoretical support to the fact that eigenfunctions relate to the symmetries and protrusions of their domains. In other words, there is a relationship between the eigenfunctions, and the underlying geometry and topology of the shape. An example of these relationships is illustrated in Fig. 3.14, where the most significant eigenfunctions are used as a color scale (c.f. Fig. 3.13) to paint the vertices and facets of the centaur shape in Fig. 3.11. The Laplacian eigenfunctions are shown in comparison to those of the LLE and Isomap embedding algorithms. It is interesting to see a relation between the eigenfunctions of Isomap and the Laplacian Embedding, confirming the power of the Laplacian method to describe global properties of the shape from local neighborhood graphs.

Levy *et al.* [Levy, 2006, Vallet and Lévy, 2008] have also discussed the use of the Laplacian eigenfunctions for creating a geometric-aware basis useful for data compression or to project functions on the shape; the principle is illustrated in Fig. 3.15. The eigenfunctions are used as basis where the coordinates of the sample points are projected. Only the eigenfunctions corresponding to the most significant eigenvalues are retained and the coordinate functions are reconstructed from their projection on the selected eigenfunctions. Fig. 3.16 shows the reconstruction of shape Fig. 3.11 in comparison with the usage of the LLE or the Isomap eigenfunctions. The first eigenfunctions of the Laplacian embedding contain shape-dependent geometric and topological information which is not present in the two other cases.

3.2.2.7 Convergence

Some conditions need to be fullfilled, in order for the results attributed to the continuous operators to be consistent with their discrete versions, specially regarding the sampling of the shape. It is desirable that the solution to the discrete PDE converges to the solution of the continuous operator when the number of samples grows to infinity. In recent years, several studies have focused their attention on the convergence of different discrete operators and discretizations of the continuous operator. Belkin and Niyogi have showed in [Belkin and Niyogi, 2003] that when data is uniformly sampled from a low dimensional manifold in $\mathbb{R}^p$, the first largest eigenvectors of the graph Laplacians $\mathbf{L}$ and $\mathcal{L}$ are discrete approximations of the Laplace-Beltrami operator on the manifold. Coifman *et al.* address [Coifman and Lafon, 2006] the convergence for non-uniform samplings. Rustamov [Rustamov, 2007] and Levy *et al.* [Levy, 2006, Vallet and Lévy, 2008] discuss the use of the cotangent-weights approximation for meshed surfaces, based on the initial work of Meyer and Desbrun [Meyer et al., 2002]. Overall, there exist many approximations of the Laplace-Beltrami operator satisfying different properties, but there is no discretization satisfying all of the desired properties [Wardetzky et al., 2007]. Nevertheless, good approximations can be expected for well behaved shapes.

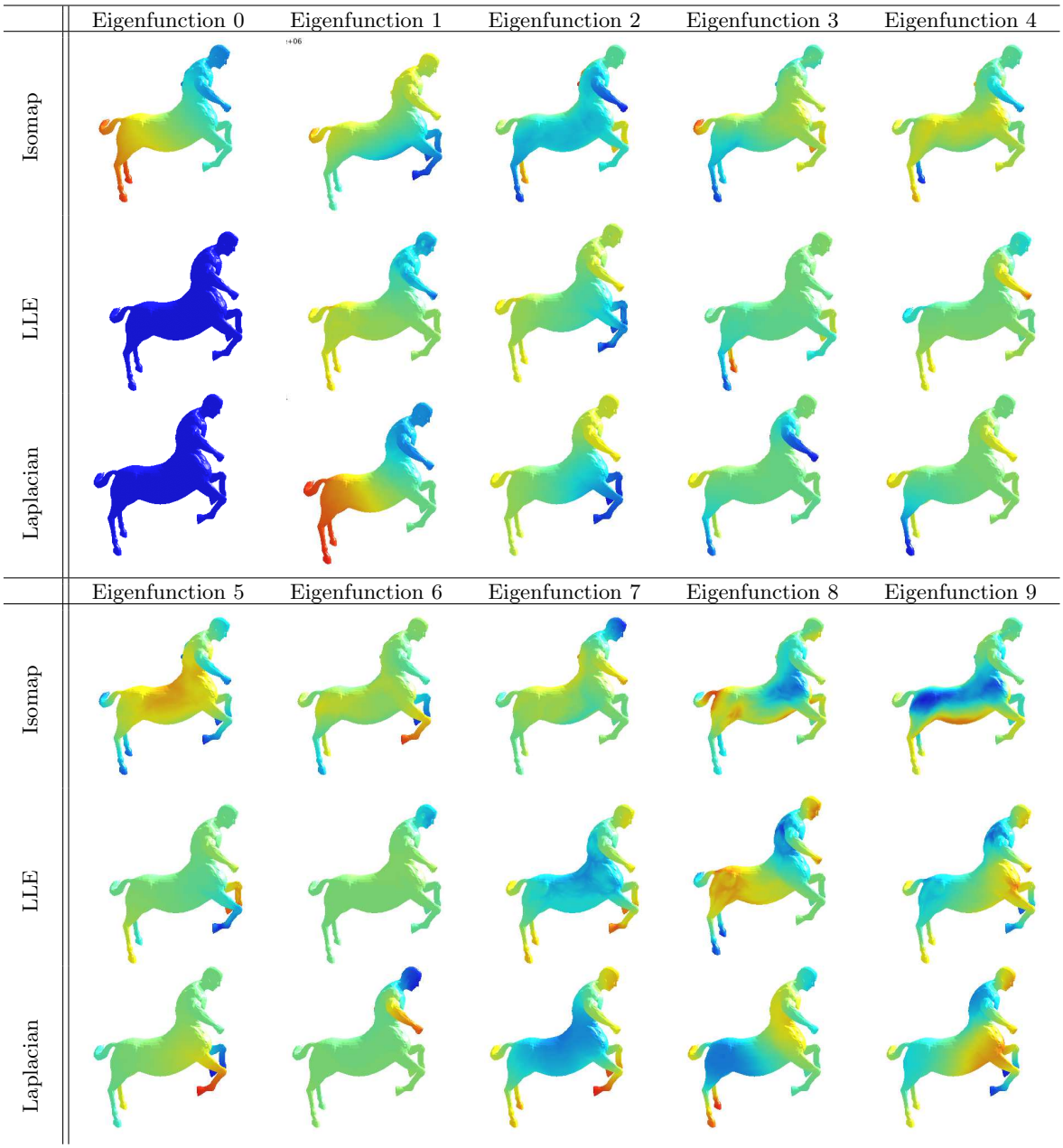


Figure 3.14: Illustration of the embedding functions obtained as the eigenvectors of the graph Laplacian operator describing the mesh lying on the surface of a centaur. Each figure shows an eigenfunction corresponding to one of the 6 most significant eigenvalues for different types of embedding: (left) Isomap, (center) LLE, (right) Laplacian. The values of the function are used to color the vertices of the mesh, using a colorscale that varies from blue (min), through green (zero), to red (max). Notice how the different protrusions of the centaur are distinctively highlighted by the Laplacian embedding. Note that the first eigenfunctions of Laplacian and LLE embedding methods are constant.

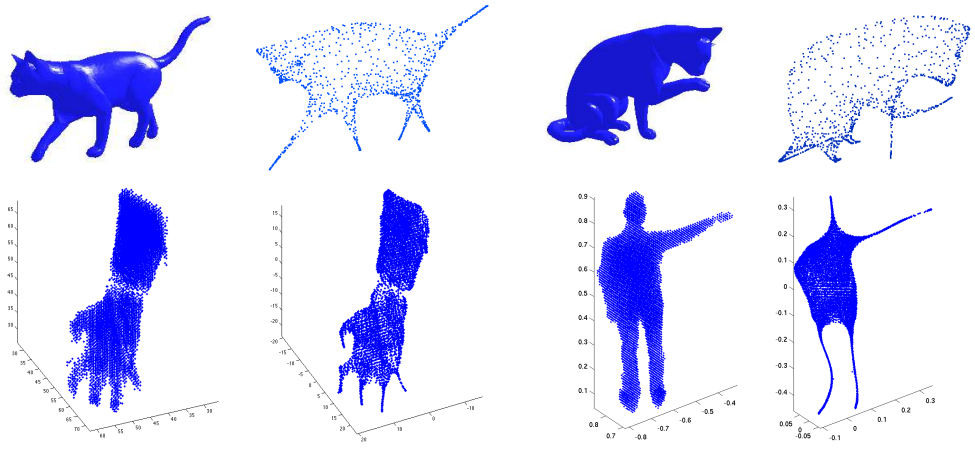


Figure 3.15: Reconstructing the coordinates of the samples from their projection on to the Laplacian eigenfunctions corresponding to the 10 most significant (non-null and smallest) eigenvalues for meshed surfaces (top) and voxel-sets (bottom). In each of the 4 examples, the original shape is shown on the left and the reconstructed set of points on the right.

In practice, the convergence of the Laplacian embedding applied to voxel-sets is ensured for the most-significant eigenfunctions given the regular sampling. In the case of meshes, the good behavior of the Laplacian embedding method is dependent on a uniform distribution of the vertices on the surface. In order to ensure the convergence in the case of irregular surface samplings on manifold meshes, the weights in Eq. 3.2 can be replaced by the symmetric cotangent weights in Eq. 3.22 as suggested by Vallet and Levy [Vallet and Lévy, 2008]. This substitution guarantees the symmetry of the operator and removes the dependency on the sampling and the triangulation.

$$w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{\sqrt{A_i A_j}} \tag{3.22}$$

In Eq. 3.22, α_{ij} and β_{ij} are the angles opposite to the edge (i, j) and A_i and A_j are the Voronoi

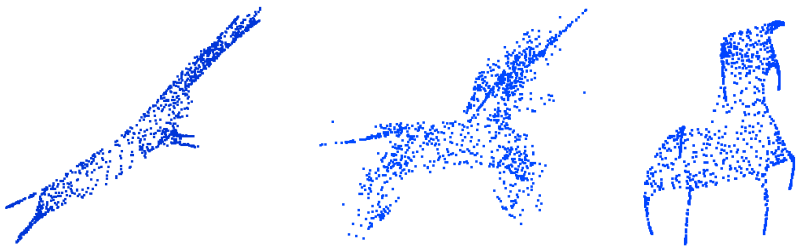


Figure 3.16: Reconstruction of the coordinate points of the centaur shape in Fig. 3.11 and Fig. 3.14 using the 10 most significant eigenfunctions of different types of embedding (left) Isomap, (center) LLE, (right) Laplacian. The projection basis constructed from the Laplacian eigenfunctions best captures the geometry and topology of the shape.

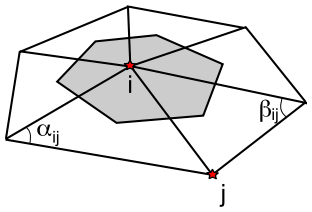


Figure 3.17: Angles α_{ij} and β_{ij} for the computation of the cotangent weights w_ij for two vertex nodes i and j. The Voronoi area A_i associated to the node i is shaded in gray.

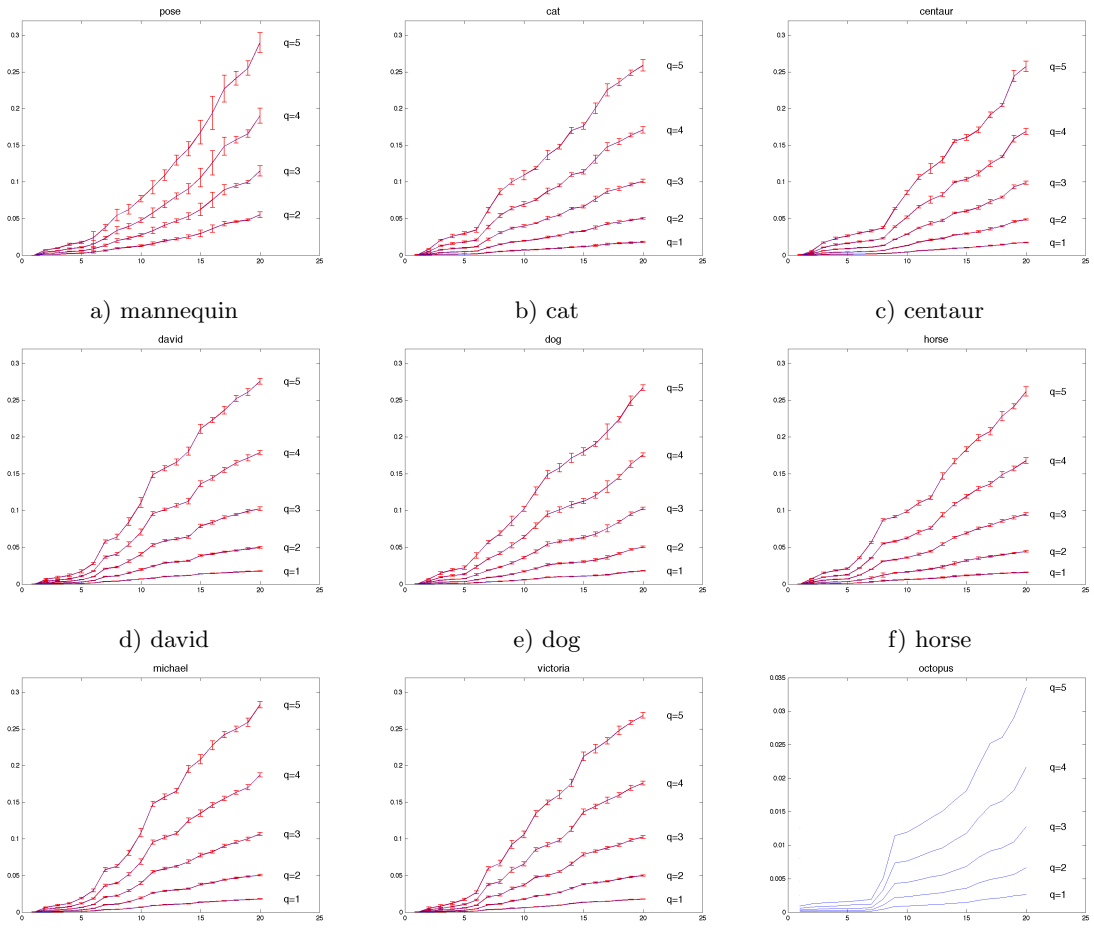
areas of the nodes i and j, as illustrated in gray in Fig. 3.17 for the area A_i. Voronoi areas are computed from the incident triangle's circumcenters (or edge midpoints for obtuse angles). This is in accordance to the generalized eigenvalue problem for Algorithm 1 and the normalized Laplacian $\mathcal{L}$ definition. The only difference being that the area of the surrounding facets are used to fill the degree matrix $\mathbf{D}$ instead of just summing weights to adjacent nodes.

3.2.2.8 Isometry invariant shape representations

Similarly to the Laplacian embedding, the spectral analysis of the Laplace-Beltrami operator applied to a compact shape gives a discrete set of eigenvalues and eigenfunctions (c.f Eq. 3.21). Since the Laplace-Beltrami operator $\triangle\mathcal{M}$ is symmetric, its eigenfunctions form an *orthogonal basis* for the space of functions defined on the metrics of the manifold. The eigenvalues and eigenfunctions of $\triangle\mathcal{M}$ are not affected by isometric transformations which means they are *isometry invariants* of a given shape. In other words, if one shape is mapped to another by a transformation which preserves distances, the resultant shape will have exactly the same eigenvalues and eigenfunctions as the original one. Furthermore, the eigenvalues and eigenfunctions characterize shapes uniquely.

Exploiting the invariance of the eigenvalues to isometries, Reuter *et al*. [Reuter et al., 2006] have proposed the use of the spectrum of a finite element discretization of the Laplace-Beltrami operator to build an invariant shape descriptor (signature): the shape "DNA" . With this descriptor it is possible to measure distances between shapes as required in applications such as shape indexing and retrieval. An example of this behavior is illustrated in Fig. 3.18, In these plots the variance (shown with bars) of the eigenvalues after the Laplacian Embedding in Algorithm 1 across different poses of a articulated object is small.

In this work, we exploit instead the invariance of the eigenfunctions. In order to do so, we consider that articulated motion transforms the shape of an objet in a *quasi-isometric* fashion (distances along the shape are preserved except for local deformations occurring in joints c.f. Fig. 3.8). Under this assumption, the eigenvalues and eigenfunctions of a shape of an articulated object in different poses should be similar. In this sense, the Laplacian eigenfunctions are suitable for creating pose-invariant representations, and in consequence, appropriate both for finding correspondences between two similar articulated shapes and for segmenting them in a meaningful and consistent manner. An example of eigenfunction quasi-invariance is shown in Fig. 3.19 and Fig. 3.20. It is interesting to see for example, how shapes with self-contacts still look similar away from the affected regions.



a) mannequin b) cat c) centaur

d) david e) dog f) horse

g) michael h)victoria i) octopus

Figure 3.18: First 20 eigenvalues for different shapes varying the propagation of the local-geodesic distance q. Blue curves show the mean of the eigenvalues across the different articulated poses. Bars show a small one standard deviation meaning that the eigenvalues are relatively well preserved by the transformations induced by articulated motion. a) voxel-based reconstruction of a wooden mannequin Fig. 3.19, b-h) Shapes from the Tosca database (http://tosca.cs.technion.ac.il/data.html) i) Single pose for an octopus from the aim@shape shape repository (http://shapes.aimatshape.net/).

3.3 Analysis and discussion

A crucial issue of spectral embedding methods is the choice of the dimensionality of the feature space, i.e. the number of eigenvectors retained after the spectral analysis. In the case of linear embedding algorithms (*e.g.* PCA), it is common to constrain the percentage of the variance *e.g.* 95% to be preserved, to choose the dimensionality. It is also common to look at the curve of eigenvalues, searching for a drastic change. Unfortunately, these solutions are not suitable for

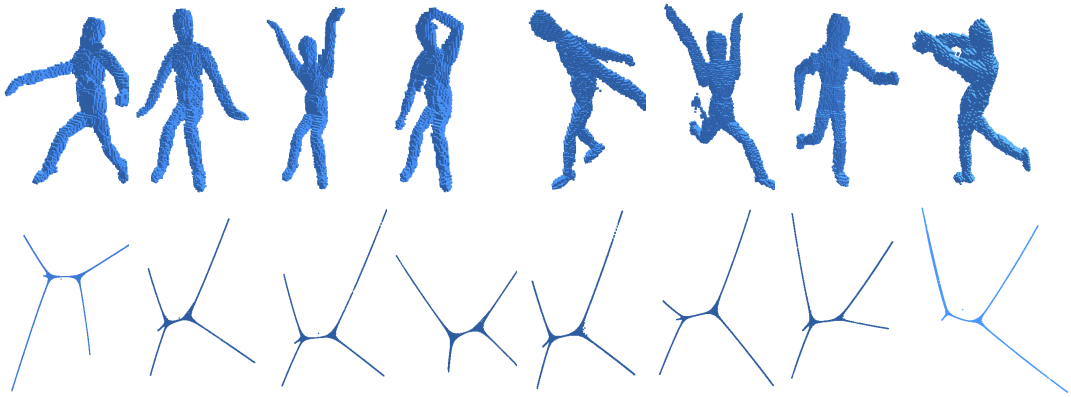


Figure 3.19: The eigenfunctions of the Laplacian operator lead to similar to similar representations of different poses of an articulated mannequin. The voxel-sets come from a multiple-view 3-D reconstruction and have over 10.000 sample points.

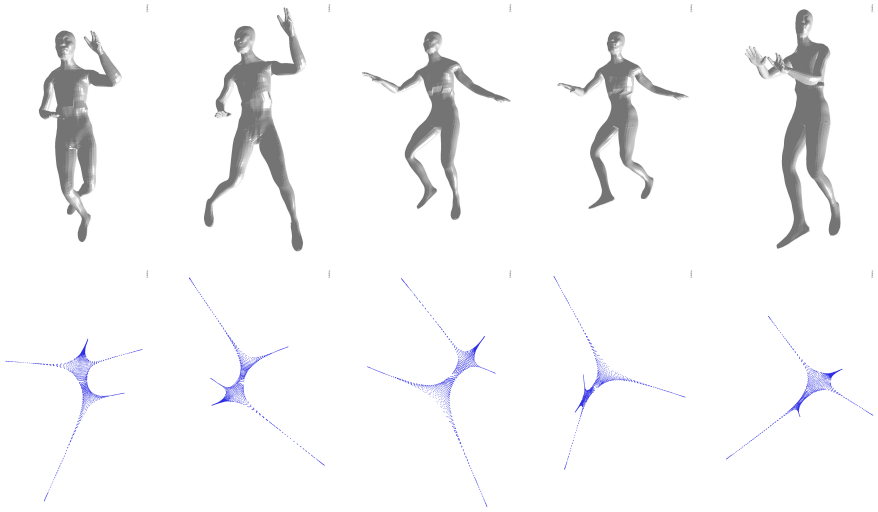


Figure 3.20: The eigenfunctions of the Laplacian operator lead to similar to similar representations of different poses of a dancer. The meshes where synthetically generated and have over 7000 sample vertices.

locally-preserving embedding algorithms, as can be seen in Figure 3.21 where a typical Laplacian spectrum is shown for a voxel-set $\mathcal{X}$ formed by 1,300 3-D points representing a human. There is no clear change in the curve and in consequence of the number of eigenvalues/eigenfunctions to retain. In Chap. 4, we will address a particular solution for the dimensionality selection in the context of matching. For segmentation and clustering, on the other hand, it has been pointed out that the number of dimensions to select should be around the desired number of segments/clusters [Polito and Perona, 2002, Ng et al., 2002]. For the tasks we aim to solve, registration and segmentation, it is enough to retain a few of these eigenfunctions capable of representing the main geometric and

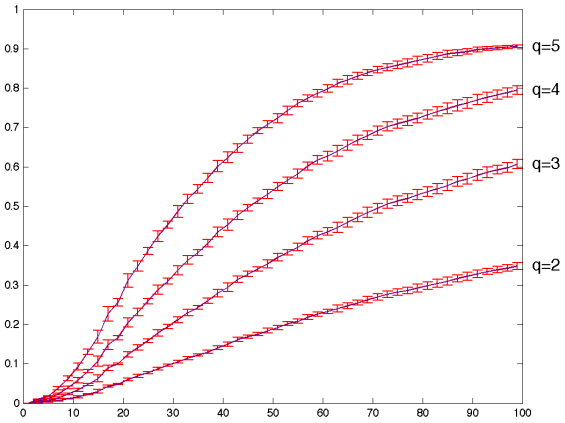


Figure 3.21: Typical behavior of the eigenvalues of a Laplacian embedding. Here the first 100 eigenvalues shown for different poses of the mannequin Fig. 3.19 and different values of local-geodesic propagation (q).

topological characteristics of the shape. In practice, more complex shapes require a larger number of eigenfunctions (see Fig. 3.22).

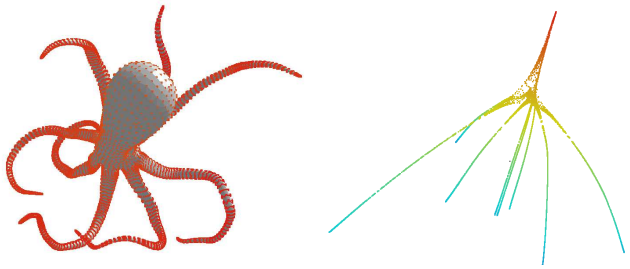


Figure 3.22: A complex shape requires more eigenvalues/eigenfunctions to get a good description (c.f. Fig. 3.18). a) octopus b) embedding. Shape from the shape repository (http://shapes.aimatshape.net/).

The theory presented in this chapter is based on two assumptions: the first, that the shapes of the objects can be captured with shape-graphs and reduced to a point-set via a locality-preserving embedding algorithm. The second, that the articulated motion is a quasi-isometric transformation and therefore the Laplacian eigenfunctions of the shape-graphs representing an articulated object in motion are similar. These assumptions are verified for synthetic articulated shapes. In the case of real articulated objects, the shape graphs are constructed from noisy and incomplete data (due for example, to the limitations of computer-vision techniques). Therefore, errors, missing parts and non-isometric deformations have to be taken into account. In the following chapters we describe some solutions to handle these difficulties.

It is also interesting to point out that when one can choose between meshes or voxels to construct the shape-graphs, the latter are to be preferred since they guarantee a dense and uniform sampling. Furthermore, voxel-sets are easy to construct, manipulate and update, opposite to meshes. However, voxel-representations are obviously less compact; so, for attaining high-resolutions, meshes are probably unavoidable.

Chapter 4

Articulated Shape Matching using Spectral Embedding Methods

Contents

4.1 Introduction

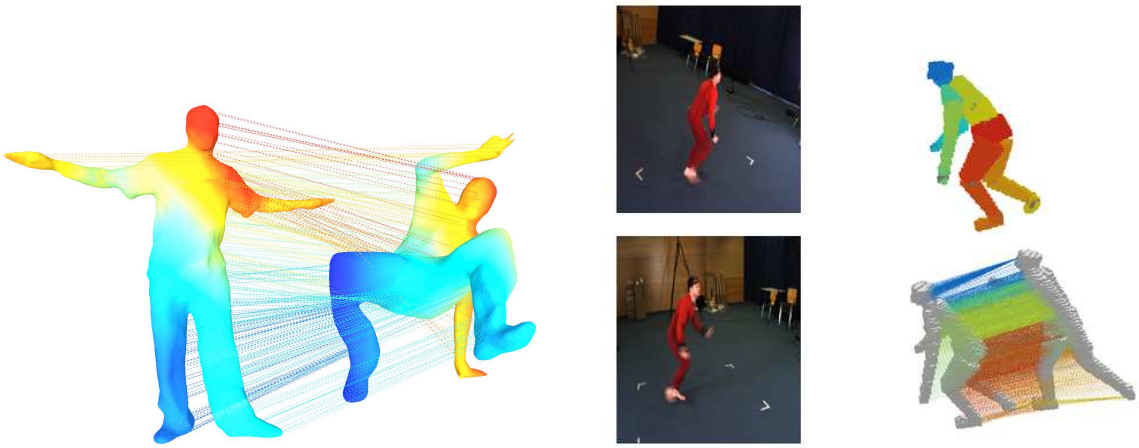
In many applications of Computer Vision and Computer Graphics, modeling shapes plays an important role. The "shape" of an object, characterizes its external form through distinctive geometric and topological attributes. These attributes allow us to describe the object or to create a model of it. The descriptor or model can then be used to compare the object to others. In the previous chapters we have built a representation of articulated shapes that is robust to pose changes. In this chapter, we will discuss how to use this representation for matching articulated objects. In the recent past, there has been an increasing interest in both 2-D and 3-D articulated shape matching. However, *the problem of matching 3-D articulated shapes* remains difficult, mainly because it is not yet clear how to choose and characterize the group of transformations under which such shapes should be studied.

There are principally two ways in which the *shape matching* problem can be presented. In the first approach, the matching problem is interpreted as *finding correspondences* between similar parts or elements of the shapes being compared. In the second, the problem is equivalent to *estimating a similarity measure* between the shapes. This shape-similarity measure is used for indexing, retrieval and classification applications. The two approaches are closely related but, in general, the solutions are dedicated to one or the other formulation of the problem. For example, transforming the correspondences to obtain a similarity measure is possible but not necessarily efficient.

In this document we interpret the the shape-matching problem as a correspondence problem. In particular, *our goal is to find a dense correspondence map between pairs of articulated objects in an unsupervised manner*. An example of the expected result can be seen in Fig. 4.1, where two poses of the articulated body are matched by finding the correspondences (shown with lines) between the nodes of the two meshed surfaces. As mentioned before, we are particularly interested in the analysis of articulated objects observed in multiple-view video sequences (Fig. 3.3).

In Computer Vision, the solutions to the articulated shape-matching problem may be used for applications such as tracking, building consistent representations of the shape along time, motion capture and motion analysis. In Computer Graphics, the correspondence map can be employed for shape morphing, pose transfer and texture transfer. Finally, the evaluation of the correspondence-map quality allows to define a similarity measure useful for shape classification and retrieval. In general, finding correspondences is a fundamental problem and often a bottleneck for higher-level applications.

The solution to the articulated-shape-matching problem presented in this document is an original method combining the Laplacian embedding algorithm (c.f. Sec. 3.2.2.3) with probabilistic point matching. The overall approach is summarized in Fig. 4.2. First, we represent articulated shapes by locally connected graphs, called shape graphs, as explained in Sec. 3.2.1. Due to the graph



a) b)

Figure 4.1: The articulated shape matching problem interpreted as finding a dense correspondence map between sample-sets of the objects' shape. a) The mesh model of two distinct poses of an articulated shape are matched with the algorithm proposed in this document. The correspondences are shown with lines joining the matched points. b) A similar example for two poses of a voxel-set sequence.

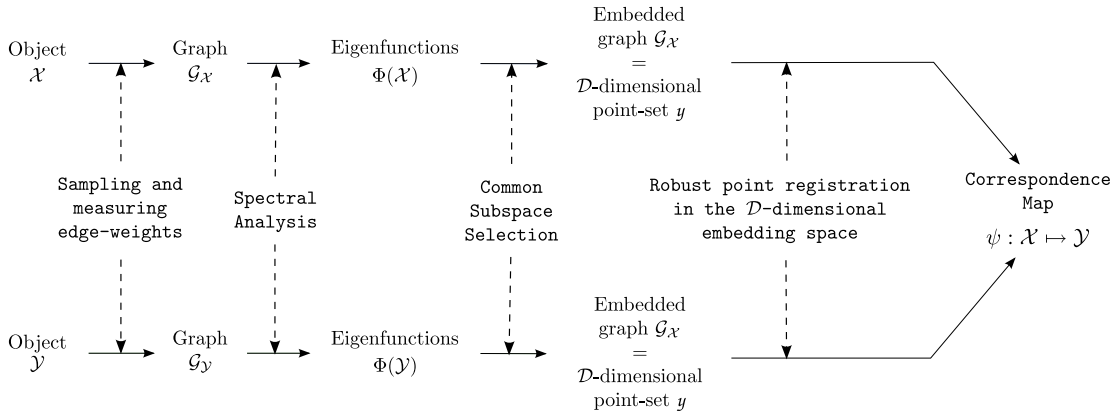


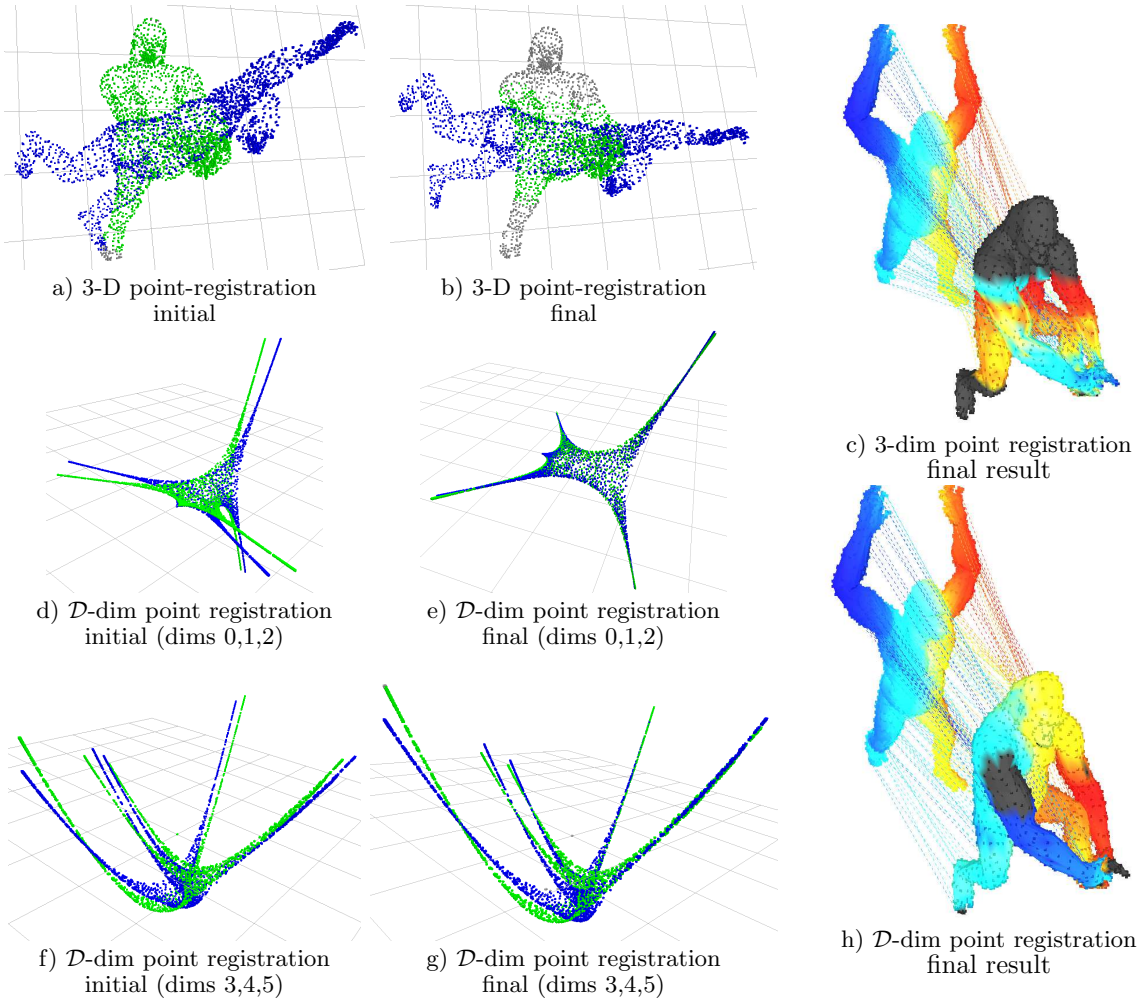
Figure 4.2: Overview of our method to match articulated shapes

representation, we can treat the *articulated-shape-matching problem* in the framework of *graph-matching*. The shape-graphs are embedded in to a normed vector space while preserving the local intrinsic geometry of the shape. The map is constructed from the eigenfunctions of the graph-Laplacian operator applied to the shape-graphs. In the context of matching, the motivations to use such map are multiple. In particular, the Laplacian-based map allow us to attain invariance to bending, rigid transformations and uniform scaling. The embedding also permits to deal with graphs of different sizes. Due to these desirable properties we can cast the shape-graph-matching problem in to the registration of point-sets in the embedding space. In brief, if two shapes are similar, their embedded shape-graphs are (close to)*congruent* and thus, easy to register.

Notice that there is a significant difference between applying a point-registration algorithm to shape-samples in the original 3-D space, and using the same algorithm to register embedded shape-graphs, as proposed here. The difference is illustrated through an example in Fig. 4.3. In 3-D, the point-registration algorithm easily falls into a local minimum (Fig. 4.3-c). After the map, the invariance ensures a good final solution (Fig. 4.3-h). Finally, to deal with the structural errors, the point-registration algorithm is modeled within a probabilistic framework, as an unsupervised clustering problem. The details of the point-registration algorithm in the $\mathcal{D}$-dimensional embedding space are the subject of Chap. 5.

To make the problem more computationally tractable the dimension of the embedding space is reduced by means of manifold-learning methods. We will see that in order to effectively compare and match two of these embedded representations it is necessary to select a *common* eigen-subspace. The selection is not straightforward for large graphs and/or in the presence of noise and structural errors (spurious nodes and edges). Only after a proper eigen subspace selection and alignment the matching can be performed with a point-registration algorithm in the embedding subspace. We conclude that *articulated shape matching is equivalent to point-to-point registration under an orthogonal transformation in* $\mathbb{R}^\mathcal{D}$, where $\mathcal{D}$ is the dimension of the selected common eigenspace.

This chapter explains the relation between matching the coordinated-points of the embedded shape-graphs and performing spectral graph matching in a reduced eigenspace. This relation serves to identify the weaknesses of classical spectral methods for graph matching and led us to propose



a) 3-D point-registration initial

b) 3-D point-registration final

c) 3-dim point registration final result

d) $\mathcal{D}$-dim point registration initial (dims 0,1,2)

e) $\mathcal{D}$-dim point registration final (dims 0,1,2)

f) $\mathcal{D}$-dim point registration initial (dims 3,4,5)

g) $\mathcal{D}$-dim point registration final (dims 3,4,5)

h) $\mathcal{D}$-dim point registration final result

Figure 4.3: Using an iterative rigid point-set registration algorithm to match two poses of an articulated object represented by a mesh. In (a,b,d,e,f,g), the vertex of meshes 1 and 2 are displayed respectively as a blue or a green point-set. Figures (a,b,c) illustrate the point-registration algorithm in the original 3-D space, using the mesh vertices as point coordinates. Starting from the initial alignment in (a), the registration algorithm falls in a local minimum (b). The final set of correspondences, shown as colored lines in (c), is therefore erroneous. On the contrary, when the vertex coordinates are embedded into a common eigenspace with a local-isometry-preserving algorithm, the resultant point-sets, (d) and (f), can be effectively aligned and registered as shown in (e) and (g). The correspondences in the embedding space can be directly used as a match in the initial 3-D space. As seen in (h), registering in the embedding space, leads to a correct set of correspondences, despite the initially large change in pose.

alternative solutions to cope with these difficulties. In fact, classical spectral methods for graph-matching assume that graphs are close to isomorphic and rely on the *eigenvalue ordering* to select and align the eigenspaces of the two graphs. These assumptions do not hold for large graphs with

structural errors, so as a consequences, the eigenvalue ordering is not reliable. If these problems have been identified and mentioned in several publications [Caelli and Kosinov, 2004][Carcassoni and Hancock, 2003a][Jain and Zhang, 2006], they are still often neglected or treated with heuristics. We have derived an extension to the classical work of Umeyama[Umeyama, 1988] which allows us to apply classical spectral methods to large graphs, such as the ones coming from articulated shapes captured with Computer Vision techniques (c.f. Fig. 3.3) or meshed models for animation as the one shown in Fig. 4.1. Specifically, we offer two original alternatives to both the structural-error and eigenvalue-ordering problems. In [Mateus et al., 2007], we have proposed the use of *out-of-sample* extensions of spectral embedding methods to deal with the alignment of embedded representations coming from a temporal sequence. In [Mateus et al., 2007], we treated the matching of widely different poses, by selecting the common subspace based on the comparison of Laplacian eigenfunction histograms.

4.2 Shape matching as graph matching

As described in Chap. 3, spectral graph theory serves as basis for non-linear spectral embedding methods. In this chapter, we explore the spectral solution to the graph matching problem, which is also founded on spectral graph theory. As a result of this common support, it is easy to relate spectral graph matching methods to the registration of embedded shape-graphs. In other words, since the construction of the embedding functions is based on the spectral properties of the graph describing the shape, *there is a direct connection between the methods that perform registration in the embedding space and the algorithms for spectral graph matching*. The connection serves as a bridge between the theoretical and practical aspects of spectral embedding and spectral matching. For example, the use of spectral embedding methods for dimensionality reduction suggests that choosing a small number of eigenfunctions may also be effective for graph matching. Therefore, we formally introduce the use of a reduced dimensionality in the classical set-up of spectral graph matching, taking into account that the comparison between two shapes requires the selection of a common eigenspace.

4.2.1 Graph matching

When shapes are represented by locally connected sets of points, *i.e.* graphs, the problem of shape matching becomes that of graph matching [M. Hilaga et al., 2001, Sundar et al., 2003, Leymarie and Kimia, 2003, Cornea et al., 2005, Bespalov and Shokouf, 2004, Zhang et al., 2005, Bespalov et al., 2006, Biasotti et al., 2006]. Graph matching is the process of finding a correspondence between the nodes and edges of two graphs under certain constraints ensuring that sub-structures of one graph are mapped to similar substructures in the other graph. There are mainly two categories of graph-matching problems according to the type of solution sought. The reduced taxonomy is shown in Fig. 4.4. The first category, known as the *exact matching* problem, imposes a strict correspondence between the two graphs. The second, *inexact graph matching* allows to compare and match graphs that are, to some extent, structurally different.

Given two graphs $\mathcal{G}_\mathcal{X} = (\mathcal{V}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathcal{W}_\mathcal{X})$ and $\mathcal{G}_\mathcal{Y} = (\mathcal{V}_\mathcal{Y}, \mathcal{E}_\mathcal{Y}, \mathcal{W}_\mathcal{Y})$, the *exact graph matching* problem consists of finding a one-to-one mapping $\psi : \mathcal{V}_\mathcal{X} \mapsto \mathcal{V}_\mathcal{Y}$ such that there exits a strict correspondence between the edges of the two sets, *i.e.* every edge connecting two nodes in the first graph, $(i, j) \in \mathcal{E}_\mathcal{X}$,

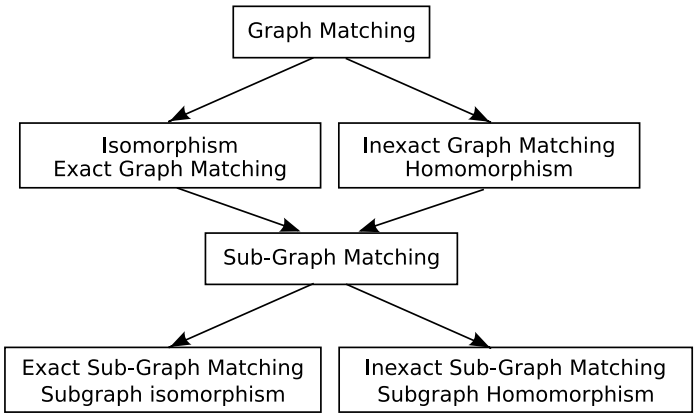


Figure 4.4: Types of graph matching problems

maps to an edge in the second graph $(\psi(i), \psi(j)) \in \mathcal{E}_\mathcal{Y}$, or:

$$\forall (i,j) \in \mathcal{E}_\mathcal{X} \implies \exists \quad (\psi(i), \psi(j)) \in \mathcal{E}_\mathcal{Y} \tag{4.1}$$

If such a map exists, it is called an *isomorphism*[1]. Equivalently, if $\mathcal{G}_\mathcal{X}$ and $\mathcal{G}_\mathcal{Y}$ are isomorphic then, then there is a permutation $\pi : |\mathcal{V}_\mathcal{X}| \times |\mathcal{V}_\mathcal{X}|$ which allows us to re-label the vertices of the second graph such that the two graphs become identical. The permutation π, can be expressed in terms of a matrix $\mathbf{P}$ applied to the adjacency matrix $\mathbf{W}_\mathcal{Y}$ of one of the graphs $\mathcal{G}_\mathcal{Y}$. With the matrix notation, the edge-preservation condition in Eq. 4.1 required for graph matching, can be written as:

$$\mathbf{W}_\mathcal{X} = \mathbf{P}\mathbf{W}_\mathcal{Y}\mathbf{P}^\top \tag{4.5}$$

under the constraints that force $\mathbf{P}$ with elements $\{P_{ij}\}_{i,j=1,\dots,N}$ to be a permutation matrix, *i.e.* binary entries ($P_{ij} \in \{0,1\}$), and row and column unitary sums ($\sum_i P_{ij} = 1$ and $\sum_j P_{ij} = 1$).

A weaker form of exact graph-matching is the *sub-graph isomorphism* problem, in which it is only required that an isomorphism exists between the nodes of one of the graphs ($\mathcal{V}_\mathcal{X}$) and a node-induced sub-graph of the other $\overline{\mathcal{V}_\mathcal{Y}} \subseteq \mathcal{V}_\mathcal{Y}$. Finally, the isomorphism may be only enforced between sub-graphs of both $\mathcal{G}_\mathcal{X}$ and $\mathcal{G}_\mathcal{Y}$, *i.e.* $\overline{\mathcal{V}_\mathcal{X}} \subseteq \mathcal{V}_\mathcal{X}$ and $\overline{\mathcal{V}_\mathcal{Y}} \subseteq \mathcal{V}_\mathcal{Y}$. This problem is known as the *maximum common sub-graph* and consists in looking for the largest sub-graph in each of the graphs, for which an isomorphism exists.

Because of its combinatorial nature and complexity[2], graph matching is either solved exactly in a restricted setting or approximately[Cour et al., 2007]. Examples of algorithms to solve the exact

1. An isomorphism is a bijective (one-to-one and onto) mapping ψ between two objects, which preserves the structure of the objects. The preservation is understood mathematically as one or several of the following preserving operations:

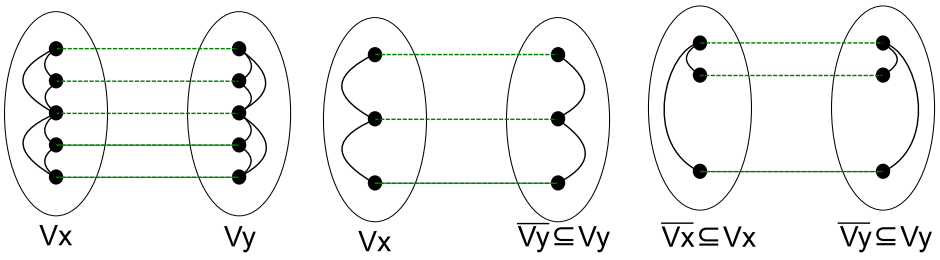
$$\text{sum} \qquad \psi(a+b) = \psi(a) + \psi(b) \tag{4.2}$$

$$\text{product} \qquad \psi(ab) = \psi(a)\psi(b) \tag{4.3}$$

$$\text{distances} \qquad \psi(||a-b||) = ||\psi(a) - \psi(b)|| \tag{4.4}$$

In the specific case of graphs, a the relevant operation is the preservation of edges in Eq. 4.1

2. All the above-mentioned graph-matching problems are NP-complete [M.R.Garey and Johnson, 1979] except for the exact graph isomorphism, for which the complexity has not yet been demonstrated.



a) Isomorphism b) Sub-graph c) Max. common sub-graph

Figure 4.5: Types of exact graph matching. Each graph is represented as an ellipse, with dots and lines illustrating nodes and edges respectively. Matches are shown as green dashed lines between the nodes and unmatched nodes and edges are shown in gray. a) In a isomorphism, all nodes and edges correspond. b) In a sub-graph exact matching, the vertex of the first graph, $\mathcal{V}_\mathcal{X}$, are mapped only to the subset of $\mathcal{V}_\mathcal{Y}$ for which the isomorphism is verified. c) Only sub-graphs of $\mathcal{G}_\mathcal{X}$ and $\mathcal{G}_\mathcal{Y}$ are isomorphic and thus matched.

graph matching problem are tree search and incomplete enumeration [Ullmann, 1976, Schmidt and Druffel, 1976, Cordella et al., 1999]. These methods give an optimal solution but are only practical for small or sparse graphs.

In the majority of applications, an *exact* match is too ambitious; specially when the compared graphs are the product of some process subject to noise, *e.g.* inducing missing or extra nodes. Therefore, it is common to address the *inexact* graph matching problem instead. The inexactness is a result of a formulation of the graph-matching problem that breaks one (or more) of the constraints imposed by isomorphisms. For example, allowing for many-to-one instead of one-to-one matchings, or using a soft-matching relaxing the binary values for a match. Relaxing theses constraints leads to bigger search spaces, so solutions are are usually approximate and suboptimal. Approximate algorithms are however more scalable (usually polynomial) with respect to the exact case.

Both the nature of the different possible relaxations and the variety of cost function definitions determine the various existent approaches to solve the inexact matching problem. Here, we discuss three of the most common approaches. The first consists of modeling the structural differences between the two graphs as graph-editing operations (*e.g.* add node, remove edge) and assigning a cost to each of these operations. The best match can be found looking for the set of operations that transforms one graph into another with the minimum cost. Examples of this approach are the graph-edit distance [Bunke, 1999] and the error-correcting graph matching [Lladós et al., 2001][Bunke, 1997] [Neuhaus and Bunke, 2006].

A second approach for inexact matching consists of relaxing the isomorphism constraints by defining a cost function that no longer forbids certain configurations but instead penalizes them. The matching solution is then found by minimization of the cost function. This formulation of the problem links graph matching to the fundamental combinatorial optimization problem known as the *assignment problem* (AP). According to the nature of the cost function and the constraints, the assignment problem is categorized as linear (LAP) or quadratic (QAP). LAP and QAP are respectively subcategories of Linear and Quadratic Programs. In this context, the inexact matching problem is solved by minimizing non-convex cost functions over discrete constraints. The inexact algorithms differ in the way they minimize the non-convex cost over the relaxed constraints [Zass and Shashua, 2008]. The *linear* program formulation, for example [Almohamad and Duffuaa, 1993], has the disadvantage of neglecting the connectivity structure of the two graphs and therefore it

often leads to erroneous matches. To account for the consistency between the edge-set of the two graphs, the cost function may no longer be defined in terms of a distance between nodes of the two graphs, as it is the case in the linear assignment, but instead, in terms of the compatibility between pairs of correspondences. Considering the compatibility between all possible matches explicitly, leads to a quadratic cost function with integer constraints (IQP). A successful approach to resolve this NP matching problem is to cast the discrete optimization into a continuous one, as done by the graduated assignment [Gold and Rangarajan, 1996], the semi-definite positive approximation [Schellewald and Schnorr, 2005, Torr, 2003] and the spectral solution in [Leordeanu and Hebert, 2005, Cour et al., 2007], which finds matches by finding consistent clusters of edges over the set of candidate edge correspondences.

Finally, the third approach to solve inexact graph matching is a different class of spectral methods, which account for the graph's structure by first embedding the graph in to a vector space whose metric reflects the graph's connectivity, *i.e.* the edge weights. Then, linear or non-linear matching algorithms can be deployed in the embedding space to find the correspondences[Umeyama, 1988, Luo and Hancock, 2001, Mateus et al., 2008]. Our work presented in this chapter lies in this category. Spectral graph matching was pioneered by [Umeyama, 1988], followed by the works of Longuet-Higgins *et al.* [Scott and Longuet-Higgins, 1991a], Shapiro and Brady, [Shapiro and J.M.Brady, 1992], and a series of publications from Hancock*et al.* [Luo and Hancock, 1999, Carcassoni and Hancock, 2003b, Wang and Hancock, 2006].The main advantages of spectral methods are their speed and simplicity. Performing the embedding step greatly reduces the size of the problem with respect to the methods in [Leordeanu and Hebert, 2005, Cour et al., 2007] which explicitly consider all the potential matches and their compatibilities. Spectral graph matching seems thus appropriate to deal with the kind of large graphs that we use to represent articulated shapes.e

The categories above are not exhaustive, and there are many other approaches for graph-matching including the Expectation Maximization framework [Cross and Hancock, 1998], matching of structures in terms of generalized maximum clique search [Pavan and Pelillo, 2003], interpolation-based matching [van Wyk and van Wyk, 2003], metric embedding [Demirci et al., 2004], matching by graph seriation and sequence alignment [Robles-Kelly and Hancock, 2005], and exact probabilistic inference using sparse graphical models with computationally feasible junction trees [Caetano and Caelli, 2006]. Further methods can be found in recent surveys [by Sven Dickinson et al., 2001, by H. Bunke and Caelli, 2004, Conte et al., 2004].

The method presented in this chapter addresses the *inexact graph matching* problem using a *spectral method*. In the following we will review in detail the fundaments of the spectral approach.

4.2.2 Spectral graph matching

Spectral graph matching methods rely on the eigendecomposition of an adjacency matrix describing a graph (or a matrix derived from the adjacency matrix such as the Laplacian Matrices, $\mathbf{L}$ and $\mathcal{L}$). The underlying principle supporting spectral graph matching methods is the invariance of the eigenvalues and eigenvectors of symmetric matrices [3] to similarity transformations, and in particular to permutations. As mentioned in Sec. 3.2.2.8, the eigenvalues of two isomorphic graphs are identical [4]. The similarity of the eigenvalues has been then used as an isomorphism test, in shape retrieval applications to determine whether two shapes are similar [Niethammer et al., 2007],

3. This is true for more general matrices, such as the Hermitian matrices or the normal matrices.
4. If it is true that there exist non-isomorphic graphs with the same spectra, these are rare and examples have only been found in 2-D

or for protein comparison in chemical analysis [Koehl, 2001]. However, although the similarity of the eigenvalues may be an efficient test for isometry, it does not directly solve for the underlying correspondence map.

To find the correspondences between two graphs, spectral methods use the eigenvectors of the graph's adjacency matrix to build a new representation that characterizes its structure. In particular, spectral methods offer an attractive solution to the shape matching problem since they provide a compact and easy-to-compute representation that can be used to globally characterize the graph structure. If used effectively, the spectral representations can be used for matching by comparing patterns of eigenvalues and eigenvectors.

Research in spectral graph matching was pioneered by Umeyama [Umeyama, 1988], who proposed a solution to the *weighted inexact graph-matching problem* based on eigen-decomposition of the graphs' adjacency matrices. Scott and Longuet-Higgins [Scott and Longuet-Higgins, 1991a], and Shapiro and Brady [Shapiro and J.M.Brady, 1992] introduced the use of a Gaussian proximity matrix in the spectral approach in order to match sets of rigid points. Sclaroff *et al.* [Sclaroff and Pentland, 1993, 1994, Sclaroff and Pentl, 1995] *et al.* propose a modal framework for correspondence and shape description based on the interpretation of the shape description and matching problems under a physical model (a deforming elastic body).

Classical spectral matching techniques are computationally impractical for matching large graphs from point-sets representing articulated shapes. First, they use completely connected graphs which implies calculating distances between every pair of points. Second, they rely on the Euclidean distances between points, which does not reflect the intrinsic geometry of the shape, and thus is not invariant to articulated-pose. Finally, they compute the entire set of eigenvalues and eigenvectors for each shape, which can be computationally costly, especially for full matrices. In addition, classical spectral methods for graph matching are known to be sensitive to structural errors. Indeed, small perturbations of the adjacency matrix, as well as the presence of eigenvalues with algebraic multiplicity, engender radical changes in the spectral representation of graphs. Different ways to cope with these problem have been suggested in the literature. For example, [Carcassoni and Hancock, 2003a,b] replace the Gaussian kernel by robust weight functions, whereas Caelli *et al.* propose the use of clustering [Caelli and Kosinov, 2004] and Carcassoni *et al.* [Carcassoni and Hancock, 2003a] use a hierarchical approach. Probabilistic methods have also been explored. Luo and Hancock [Luo and Hancock, 1999] [Luo and Hancock, 2001] give a likelihood interpretation of the sets of matches, where the rigid match of point-sets is formulated in an iterative framework combining Procrustes alignment with an Expectation Maximization (EM) algorithm. Finally, in [Carcassoni and Hancock, 2003a,b] Hancock *et al.* proposed a probabilistic framework for point-to-point matching relying on the concept of matching in the embedding space.

The algorithm proposed in this chapter makes use of the Laplacian matrices instead of any proximity, Gaussian proximity or similarity matrix. This choice has an important impact in the representations of articulated shapes, which comes as a consequence of the properties of the Laplacian operator discussed in Chap. 2 and Chap. 3 including the isometry-invariance and sparsity. The eigendecomposition of the Laplacian matrix is used in graph matching to embed the structure of the graph in to a vector space where linear and non-linear registration algorithms can be deployed. The procedure is very similar to the non-linear manifold methods described in Sec. 3.2.2.1, which embed graphs in to vector spaces. In fact, the spectral representations of graphs used in graph matching are equivalent to the embedding of the shape-graphs created following the Laplacian embedding (Algorithm 1). Similar results associating non-linear embedding methods for manifold learning and spectral graph matching have been presented in [Wang and Hancock, 2006].

A second key idea of the work presented here is the use of a reduced eigenspace, as commonly used for spectral clustering or dimension reduction applications. The reduced eigenspace allows to compare graphs with different cardinality and removes the influence of the less significant modes that may be attributed to noise, sampling and details. Thus, as also suggested in [Caelli and Kosinov, 2004, Carcassoni and Hancock, 2003b], our spectral matching algorithm works with a reduced eigenspace. As we will see in Sec. 4.2.5, reducing the eigenspace of the graph transforms the articulated-shape matching problem into the problem of point-set alignment in the selected subspace [Bai et al., 2004].

There are three main *contributions* presented in this chapter. First, modeling the articulated shape matching problem as a spectral graph matching problem and associating the latter with methods for non-linear embedding. Second, a theoretical result signaling the limits of the assumptions under which the spectral embedding of graphs lead to congruent shapes in the embedding spaces. Finally, we propose two original strategies to deal with these limitations, the first using *out-of-sample* extrapolations and the second based on *eigenfunction histograms*. These two methods allow us to select the relevant eigenspaces for matching and to give a first solution to the sub-sequent point-registration algorithm.

In the following, we recall Umeyama's setting for graph matching, we discuss the different weaknesses of the methods and propose two alternatives to make Umeyama's theorem effective for matching sparse large graphs, such as the ones created in Chap. 3 to represent articulated shapes.

4.2.3 Umeyama's theorems for graph matching

In [Umeyama, 1988], Umeyama proposed a method to perform graph matching by recovering the permutation matrix that maximizes the correlation of the adjacency matrix for graphs of the same size. Consider two graphs $\mathcal{G}_\mathcal{X} = (\mathcal{V}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathcal{W}_\mathcal{X})$ and $\mathcal{G}_\mathcal{Y} = (\mathcal{V}_\mathcal{Y}, \mathcal{E}_\mathcal{Y}, \mathcal{W}_\mathcal{Y})$, with the same cardinality ($N = |\mathcal{V}_\mathcal{X}| = |\mathcal{V}_\mathcal{Y}|$). The criterium to evaluate a graph correspondence ψ, with ψ a node-to-node map $\psi : \{1, \cdots, |\mathcal{V}_\mathcal{X}|\} \mapsto \{1, \cdots, |\mathcal{V}_\mathcal{Y}|\}$, is defined as the cost function $J(\psi)$:

$$J(\psi) = \sum_{i=1}^{N}\sum_{j=1}^{N}\Big(w_\mathcal{X}(i,j) - w_\mathcal{Y}\big(\psi(i),\psi(j)\big)\Big)^2, \tag{4.6}$$

where $w_\mathcal{X}(i,j)$ (respectively $w_\mathcal{Y}(i,j)$) stands for the (i^{th},j^{th}) element of the weighted adjacency matrix, called here $\mathbf{L}_\mathcal{X}$ (respectively $\mathbf{L}_\mathcal{Y}$) for convenience. As mentioned in Sec. 4.2.1, a node correspondence ψ based on an isomorphism can be written in terms of a $N \times N$ permutation matrix $\mathbf{P}$. Under these conditions, minimizing $J(\psi)$ is equivalent to recovering the permutation $\mathbf{P}$ that reorders the set of nodes of one of the graphs to match those of the second, such that the weighted edges are preserved. Using this relation, $J(\mathbf{P})$ can be rewritten in terms the weighted adjacency matrices:

$$J(\mathbf{P}) = ||\mathbf{L}_\mathcal{X} - \mathbf{P}\mathbf{L}_\mathcal{Y}\mathbf{P}^\top||_F^2, \tag{4.7}$$

where $||\cdot||_F$ is the Frobenious norm. For two isometric graphs Eq. 4.7 is null and thus,

$$\mathbf{L}_\mathcal{X} = \mathbf{P}\mathbf{L}_\mathcal{Y}\mathbf{P}^\top, \tag{4.8}$$

which means there is an exact solution for $\mathbf{P}$. Minimizing J is a purely combinatorial problem. For non isometric graphs, it is possible to extend the domain of J to the set of orthogonal matrices in a natural way, since the permutation matrices are a subset of orthogonal matrices. This is convenient

since a closed-form solution exists for $J(\mathbf{Q})$ in the continuous space [5]. The solution is found using the following two theorems.

Theorem 1 *For any two Hermitian matrices* $\mathbf{L}_\mathcal{X}$ *and* $\mathbf{L}_\mathcal{Y}$*, with eigenvalues* $\lambda_{\mathcal{X}1} \geq \lambda_{\mathcal{X}2} \geq \cdots \geq \lambda_{\mathcal{X}N}$ *and* $\lambda_{\mathcal{Y}1} \geq \lambda_{\mathcal{Y}2} \geq \cdots \geq \lambda_{\mathcal{Y}N}$ *respectively, it is true that:*

$$\| \mathbf{L}_\mathcal{X} - \mathbf{L}_\mathcal{Y} \|^2 \geq \sum_{i=1}^{N} (\lambda_{\mathcal{X}i} - \lambda_{\mathcal{Y}i})^2 \tag{4.9}$$

Theorem 2 *Consider two Hermitian matrices* $\mathbf{L}_\mathcal{X}$ *and* $\mathbf{L}_\mathcal{Y}$ *with* distinct eigenvalues $\lambda_{\mathcal{X}1} > \lambda_{\mathcal{X}2} > \cdots > \lambda_{\mathcal{X}N}$ *and* $\lambda_{\mathcal{Y}1} > \lambda_{\mathcal{Y}2} > \cdots > \lambda_{\mathcal{Y}N}$ *respectively and their eigendecomposition:*

$$\mathbf{L}_\mathcal{X} = \mathbf{V}_\mathcal{X} \mathbf{\Lambda}_\mathcal{X} \mathbf{V}_\mathcal{X}^\top \tag{4.10}$$
$$\mathbf{L}_\mathcal{Y} = \mathbf{V}_\mathcal{Y} \mathbf{\Lambda}_\mathcal{Y} \mathbf{V}_\mathcal{Y}^\top \tag{4.11}$$

where $\mathbf{V}_\mathcal{X}$ *and* $\mathbf{V}_\mathcal{X}$ *are orthogonal matrices with column eigenvectors, and* $\mathbf{\Lambda}_\mathcal{X}$ *and* $\mathbf{\Lambda}_\mathcal{Y}$ *diagonal matrices of ordered eigenvalues. Then, the minimum of* $\| \mathbf{Q}\mathbf{L}_\mathcal{X}\mathbf{Q}^\top - \mathbf{L}_\mathcal{Y} \|$*, with* $\mathbf{Q}$ *defined over the group of orthogonal matrices, is attained at:*

$$\mathbf{Q} = \mathbf{V}_\mathcal{Y} \mathbf{S} \mathbf{V}_\mathcal{X}^\top \tag{4.12}$$

where $\mathbf{S}$ *is a diagonal matrix with elements equal to* $s_{ii} = 1$ *or* $s_{ii} = -1$ *and the minimum value attained is* $\sum_{i=1}^{N} (\lambda_{\mathcal{X}i} - \lambda_{\mathcal{Y}i})^2$.

The following proof of Theorem 2 follows the guidelines of the one provided by Umeyama in [Umeyama, 1988]. First, Theorem 1 holds for any orthogonal matrix. Since the eigenvalues of an Hermitian matrix $\mathbf{L}_\mathcal{X}$ remain unchanged after applying a similarity transformation $\mathbf{T}$, it is also true that:

$$\| \mathbf{T}\mathbf{L}_\mathcal{X}\mathbf{T}^\top \|^2 \geq \sum_{i=1}^{N} (\lambda_{\mathcal{X}i} - \lambda_{\mathcal{Y}i}). \tag{4.13}$$

Replacing $\mathbf{T}$ by the optimal $\mathbf{Q}$ (Eq. 4.12 Theorem 2) and using the decompositions in Eq. 4.10 and Eq. 4.11, we can develop the expression in Eq. 4.13 as follows:

$$\| \mathbf{Q}\mathbf{L}_\mathcal{X}\mathbf{Q}^\top - \mathbf{L}_\mathcal{Y} \|^2 = \| \mathbf{V}_\mathcal{Y}\mathbf{S}\mathbf{V}_\mathcal{X}^\top \mathbf{L}_\mathcal{X} \mathbf{V}_\mathcal{X}\mathbf{S}\mathbf{V}_\mathcal{Y}^\top - \mathbf{L}_\mathcal{Y} \|^2 \tag{4.14}$$
$$= \| \mathbf{V}_\mathcal{Y}\mathbf{S}\mathbf{V}_\mathcal{X}^\top \mathbf{V}_\mathcal{X} \Lambda_\mathcal{X} \mathbf{V}_\mathcal{X}^\top \mathbf{V}_\mathcal{X}\mathbf{S}\mathbf{V}_\mathcal{Y}^\top - \mathbf{V}_\mathcal{Y}\Lambda_\mathcal{Y}\mathbf{V}_\mathcal{Y}^\top \|^2 \tag{4.15}$$
$$= \| \mathbf{V}_\mathcal{Y}\mathbf{S}\Lambda_\mathcal{X}\mathbf{S}\mathbf{V}_\mathcal{Y}^\top - \mathbf{V}_\mathcal{Y}\Lambda_\mathcal{Y}\mathbf{V}_\mathcal{Y}^\top \|^2 \tag{4.16}$$
$$= \| \mathbf{V}_\mathcal{Y}(\mathbf{S}\Lambda_\mathcal{X}\mathbf{S} - \Lambda_\mathcal{Y})\mathbf{V}_\mathcal{Y}^\top \|^2 \tag{4.17}$$

The norm of the product of a matrix $\mathbf{A}$ with an orthogonal matrix $\mathbf{V}$ equals the norm of $\mathbf{A}$, *i.e.* $\| \mathbf{V}\mathbf{A} \| = \| \mathbf{A}\mathbf{V}^\top \| = ||\mathbf{A}||$. Furthermore, since $\mathbf{S}$ and Λ are diagonal matrices $\mathbf{S}\Lambda\mathbf{S} = \mathbf{S}^2\Lambda$ and $\mathbf{S}^2 = \mathbf{I}$. These facts allow us to further simplify the expression and conclude the proof of Theorem 2:

$$\| \mathbf{Q}\mathbf{L}_\mathcal{X}\mathbf{Q}^\top - \mathbf{L}_\mathcal{Y} \|^2 = \| \mathbf{S}\Lambda_\mathcal{X}\mathbf{S} - \Lambda_\mathcal{Y} \|^2 \tag{4.18}$$
$$= \sum_{i=1}^{N} (\lambda_{\mathcal{X}i} - \lambda_{\mathcal{Y}i})^2 \quad \square \tag{4.19}$$

5. This corresponds to the relaxation of the integer constraint of the isomorphism.

In practice, Theorem 2 implies that in the presence of an isomorphism between two graphs, the optimal permutation leading to the map ψ can be derived from the optimal $\mathbf{Q}$ using the expression: $\mathbf{Q} = \mathbf{V}_{\mathcal{Y}}\mathbf{S}\mathbf{V}_{\mathcal{X}}^{\top}$. This would solve the correspondence problem if the sign matrix $\mathbf{S}$ could be determined algebraically, which is not the case. Nevertheless, it is still possible to claim that under the isomorphism assumption (Eq. 4.8), there exists an optimal diagonal sign matrix $\hat{\mathbf{S}}$ that makes $\mathbf{Q}$ equivalent to the optimal permutation matrix $\hat{\mathbf{P}}$. Indeed, using the eigendecomposition in Eq. 4.10 and Eq. 4.11 and assuming isomorphism, Eq. 4.8 should be verified for $\mathbf{Q}$:

$$\begin{aligned} \mathbf{Q}\mathbf{L}_{\mathcal{X}}\mathbf{Q}^{\top} &= \mathbf{L}_{\mathcal{Y}} && (4.20)\\ \mathbf{Q}\mathbf{V}_{\mathcal{X}}\Lambda_{\mathcal{X}}\mathbf{V}_{\mathcal{X}}^{\top}\mathbf{Q}^{\top} &= \mathbf{V}_{\mathcal{Y}}\Lambda_{\mathcal{Y}}\mathbf{V}_{\mathcal{Y}}^{\top} && (4.21)\\ \mathbf{Q}\mathbf{V}_{\mathcal{X}} &= \mathbf{V}_{\mathcal{Y}}\mathbf{S} && (4.22)\\ \mathbf{Q} &= \mathbf{V}_{\mathcal{Y}}\mathbf{S}\mathbf{V}_{\mathcal{X}}^{\top}. && (4.23) \end{aligned}$$

Thus, for two isomorphic graphs: $\hat{\mathbf{Q}} = \hat{\mathbf{P}} = \mathbf{V}_{\mathcal{Y}}\hat{\mathbf{S}}\mathbf{V}_{\mathcal{X}}^{\top}$. In other words to use Theorem 2 to find the optimal correspondence map matrix, one needs to solve for $\hat{\mathbf{S}}$.

The solution proposed in Umeyama [Umeyama, 1988], is an approximation that avoids the explicit calculation of $\mathbf{S}$. The approximation relies on the assumption that for close-to-isomorphic graphs it is enough to compare the absolute values of the elements of $\mathbf{V}_{\mathcal{X}}$ and $\mathbf{V}_{\mathcal{Y}}$. To justify the method, Umeyama uses the absolute value of the elements in the eigenvector matrices, here noted $\overline{\mathbf{V}}_{\mathcal{X}}$ and $\overline{\mathbf{V}}_{\mathcal{Y}}$, and shows that for any permutation matrix:

$$\text{trace}(\mathbf{P}^{\top}\overline{\mathbf{V}}_{\mathcal{X}}\overline{\mathbf{V}}_{\mathcal{Y}}) \leq N. \tag{4.24}$$

For the optimal $\hat{\mathbf{Q}}$, the following expression also holds:

$$\text{trace}(\hat{\mathbf{P}}^{\top}\mathbf{V}_{\mathcal{X}}\hat{\mathbf{S}}\mathbf{V}_{\mathcal{Y}}) = \text{trace}(\hat{\mathbf{P}}^{\top}\hat{\mathbf{P}}) = N. \tag{4.25}$$

Notice that N is the maximum of Eq. 4.24. This relation demonstrates that *in case of an isomorphism* between the two graphs, it is sufficient to maximize $trace(\mathbf{P}^{\top}\overline{\mathbf{V}}_{\mathcal{X}}\overline{\mathbf{V}}_{\mathcal{Y}})$. Finally, Umeyama argues that when $\mathcal{G}_{\mathcal{X}}$ and $\mathcal{G}_{\mathcal{Y}}$ are only nearly isomorphic, the permutation matrix estimated in this manner can be at least a good first solution to the matching problem. The maximization of $trace(\mathbf{P}^{\top}\overline{\mathbf{V}}_{\mathcal{X}}\overline{\mathbf{V}}_{\mathcal{Y}})$ is an instance of the *linear assignment problem*, also known as the *bipartite maximum weighted matching*[6], which can be solved by the Hungarian method [Frank, 2005].

In the following sections we describe the limitations of this approach showing that several considerations have to be taken into account in order to apply the spectral method to general close-to-isomorphic graphs.

4.2.4 Intrinsic ambiguities in classical spectral methods

In the core calculation of spectral methods for inexact graph matching important assumptions are made. First, the starting graphs are considered to be identical (only differing in ordering of the vertex indices) and to have the same cardinality. Second, the eigenvalues are assumed to be

6. The assignment between the vertices of two different graphs can be represented as bipartite graph. A bipartite graph is an undirected graph where vertices can be partitioned into two sets such that no edge connects vertices in the same set. A perfect match between vertices of a bipartite graph, is a one-to-one mapping between the vertices of each sub-graph. The vertices which belong to one edge are a considered to be matched. In a bipartite graph, one vertex has at most one edge which ends at a vertex in the other sub-graph.

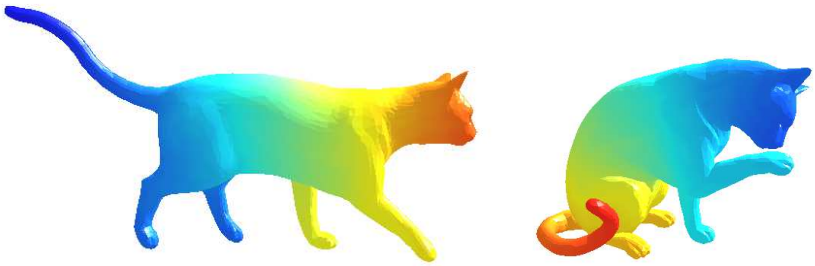


Figure 4.6: Visualization of the sign-reversal ambiguity. We display the eigenfunction corresponding to the first non-zero eigenvalue for two instances of the cat data-set using the values assigned by the eigenfunction to each vertex on the meshed surface, using a color scale as explained in Fig. 3.13. The positive (red) and negative (values) have been inverted in the right figure.

distinct such that they can be reliably ordered, which implies that none of the selected eigenvalues can be multiple. In real scenarios, for example when matching articulated shapes, the graphs are rarely identical (due to errors in reconstruction, local deformations, occlusions, missing parts and sampling) and they usually fail to verify the conditions above. In general, these assumptions do not hold for large sparse graphs. In consequence, two types of *ambiguities* have to be considered in order to effectively use an spectral method for the problem of articulated shape matching:

Sign-reversal ambiguity: Eigenvectors are only defined up to sign which introduces 2^N possible alignments between the eigen-bases of the two compared graphs. A sign flip corresponds to a reflection in the spectral domain but does not violate orthonormality of the basis. An example of such ambiguity is shown in Fig. 4.6. This ambiguity corresponds to the sign matrix $\mathbf{S}$ in Umeyama's set-up.

Eigenvalue-ordering ambiguity: The eigenvalue ordering is an important requirement for the validity of Theorem 2. There are however, several factors having a direct impact on the distinctness of the eigenvalues and therefore on the reliability on the eigenvalue ordering as a means to align the eigenbasis of two compared graphs. First, a particular shape-graph can yield several similar intrinsic elongations [7] and hence, several similar eigenvalues. Second, there may be eigenvalues with algebraic multiplicity greater than 0. Finally, numerical instabilities can directly affect the eigenvalue ordering, especially when comparing pairs of large graphs. In fact, since the eigenvalues of the Laplacian matrices are bounded (c.f. Table. 2.1), the difference between consecutive eigenvalues is constrained to be in the order of $1/N$. For graphs with a large number of nodes N, this difference may reach machine precision and produce an *eigenvalue crossing*. If the eigenvalue ordering is not reliable, one has to consider the $N!$ possible assignments between the eigenvalues (and their corresponding eigenvectors) of the two compared graphs.

In the following section we briefly review some results of Matrix Perturbation theory concerning the effect of algebraic multiplicities and the stability of eigenspaces, which contribute to the ambiguities discussed above.

7. For example, symmetries in the geodesic sense.

4.2.4.1 Insights from Matrix Perturbation theory

Computing the spectral decomposition of a symmetric matrix is among the most frequent tasks to numerical mathematics. When the eigenvalues are pairwise different and/or the size of the matrix spectrum (number of different eigenvalues) can be determined in advance, the diagonalizing procedure can be performed effectively with available software libraries [Ziegler and Brattka, 2001]. However, in general the case, methods for spectral analysis suffer from instabilities and convergence problems when applied to degenerate matrices, *i.e.* those having multiple eigenvalues. The problems come from the discontinuous behavior of eigenvectors under infinitely small *perturbations* of the input matrix (*e.g.* due to floating point approximations). This behavior has been studied in Matrix Perturbation theory, leading to two relevant results. First, the accuracy up to which the eigenvectors can be computed depends on the separation of their corresponding eigenvalues. Second, the perturbation affects the eigenvalues in a bounded manner.

Formally, let $\mathbf{A}$ be a $N \times N$ square matrix with eigenvalues $\Lambda = \{\lambda_1, \cdots, \lambda_N\}$ and eigenvectors $\mathbf{V} = \{\boldsymbol{v}_1, \cdots \boldsymbol{v}_N\}$. A non-degenerate eigenvector defines a one-dimensional subspace that is invariant with respect to pre-multiplication by $\mathbf{A}$, *i.e.* $\lambda\boldsymbol{v} = \mathbf{A}\boldsymbol{v}$. This invariance concept can also be generalized to subspaces of larger dimension. Let $\mathbf{Z}$ be such a subspace of $\mathbb{R}^N$. $\mathbf{Z}$ is said to be an *invariant subspace* of $\mathbf{A}$, if for any vector $\boldsymbol{x}$ in the subspace, *i.e.* $\forall \boldsymbol{x} \in \mathbf{Z}$, pre-multiplication by the matrix $\mathbf{A}$ also leads a vector in $\mathbf{Z}$, *i.e.* $\mathbf{A}\boldsymbol{x} \in \mathbf{Z}$ [Golub and Loan, 1996]. The sensitivity of an invariant subspace to perturbations depends upon the separation of the associated eigenvalues from the rest of the spectrum. In fact, a pair of eigenvectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, associated with nearby eigenvalues λ_i and λ_j, cannot be computed accurately. However, if their eigenvalues are well separated from a third eigenvalue λ_k, they define a two-dimensional subspace that is not sensitive with respect to λ_k.

Furthermore, the eigenvalues themselves are affected by the perturbation in a bounded manner. According to *Wielandt-Hoffman Theorem*, if a $N \times N$ symmetric $\mathbf{A}$ is perturbed by a $N \times N$ symmetric matrix $\mathbf{E}$, its eigenvalues do not change by more than $\| E \|_F$ [Golub and Loan, 1996].

$$\sum_{i=1}^{n} (\lambda_i(\mathbf{A} + \mathbf{E}) - \lambda_i(\mathbf{A}))^2 \leq \| \mathbf{E} \|_F \tag{4.26}$$

In conclusion, it is not reliable to use the eigenvalue-ordering to align the eigenspaces of the two graphs. Specially when dealing with graphs from vision-based acquisitions which are noisy and/or when matching large graphs that, as we have shown, reduce the separation between eigenvalues.

4.2.4.2 Effects of ambiguities in matching

If we consider all the possible sign-reversals and eigenvalue crossings there are $2^N N!$ alignments between the two eigenbasis. Table 4.1 summarizes the number of possible alignments as a function of the dimensionality of the considered eigenspace, N. This is a truly combinatorial problem because the number of solutions (or configurations) increases dramatically with the dimension of the embedded space.

N	2	3	4	5	6
alignments	8	48	384	3840	46080

Table 4.1: The number of possible shape alignments as a function of the dimensionality of the eigenspace.

It is interesting to notice that none of the methods mentioned in Sec. 4.2.2 formally addresses these ambiguities. Although the *sign-reversal* is a classical issue in every problem considering a eigendecomposition, most of the current solutions for spectral graph matching either ignore it or address it with heuristics. For example, [Caelli and Kosinov, 2004] forces every eigenvector to have a larger number of positive entries, whereas and [Jain and Zhang, 2006] proposes to match the first two eigenvectors in a similar manner and then add one dimension at a time.

With few exceptions [Carcassoni and Hancock, 2003b, Feng and Liu, 2006, Sun et al., 2008], most of the literature in spectral graph matching neglects the *eigenvalue-ordering ambiguity* as an improbable case. However, in our experience, eigenvalue crossings occur rather often for articulated shape-graphs, specially for large cardinalities. Carcassoni and Hancok [Carcassoni and Hancock, 2003b], explicitly treat the uncertain eigenvalue ordering with a non-heuristic method by using Gaussian prior-distribution to model the deviation between eigenvalues within an Expectation Maximization (EM) framework. Feng and Liu [Feng and Liu, 2006] address in particular the problems associated with eigenvalue multiplicities. In order to match the embedded-representations under multiplicities, [Feng and Liu, 2006] constrain the transformation between the two embedding spaces to a rotation over the sensitive eigenvalues. Finally, Sun *et al.* [Sun et al., 2008] acknowledge that the eigenfunctions associated with the repeated eigenvalues introduce rotation symmetries in the embedding. However, they claim that rotational symmetries are hard to detect in high-dimensional spaces. Instead, the problematic eigenvalues are removed and the attention is restricted to the eigenfunctions associated with non-repeated eigenvalues. Both [Feng and Liu, 2006] and [Sun et al., 2008] rely on the ability to detect the multiplicities. As shown in Fig. 3.21 and discussed in Sec. 3.3, such detection is not easy in the case of smoothly increasing eigenvalue curves.

4.2.5 Graph matching in a reduced eigenspace

The ambiguities explained in the preceding sections are important limitations of spectral graph matching methods. In fact, Umeyama's theorems hold only *weakly* for *large* and *sparse* graphs. Especially, one cannot guarantee that the eigenvalues of the Laplacian matrix are all distinct. In the absence of an absolute eigenvalue ordering, Umeyama's theorem is impractical, since one would need to find the best out of all possible $N!$ configurations. One elegant way to overcome this problem is to reduce the dimension of the eigenspace along the lines of spectral nonlinear reduction techniques. This can be easily done by retaining the $\mathcal{D}$ most significant eigenvalues [8] with their associated eigenspace ($\mathcal{D} \ll N$).

Recall that the Laplacian embedding algorithm described in Sec. 3.2.2.3, performs the eigendecomposition of the Laplacian matrix ($\mathbf{L}_\mathcal{X} = \mathbf{V}_\mathcal{X}\Lambda_\mathcal{X}\mathbf{V}_\mathcal{X}^\top$) [9] describing a shape-graph $\mathcal{G}_\mathcal{X}$. We denote by $\mathbf{V}_\mathcal{X}^\mathcal{D}$ a $N \times \mathcal{D}$ block matrix of $\mathbf{V}_\mathcal{X}$, with columns corresponding to the $\mathcal{D}$ eigenvectors associated with the selected eigenvalues. These eigenvectors (or eigenfunctions) are interpreted as mapping functions that form a truncated embedding map $\Phi^\mathcal{D}$. Equivalently, the rows of $\mathbf{V}_\mathcal{X}^\mathcal{D}$ represent the coordinates of the embedded shape graph $\mathcal{X}$ in the $\mathcal{D}$-dimensional embedding space, *i.e.* $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} = \Phi_\mathcal{X}^\mathcal{D}(\mathcal{X})$. The same procedure is applied to the second shape $\mathcal{Y}$. The matrix $\mathbf{V}_\mathcal{Y}^\mathcal{D}$ is used to derive the embedded shape-graph $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\} = \Phi_\mathcal{Y}^\mathcal{D}(\mathcal{Y})$. Each row of the Laplacian matrix gives a description of a graph-vertex in a vector space $\mathbb{R}^\mathcal{D}$ in terms of its connectivity. Therefore, selecting a $\mathcal{D}$-dimensional space of $\mathbf{L}_\mathcal{X}$ (respectively $\mathbf{L}_\mathcal{Y}$) is thus a mapping

8. leaving out most significant eigenvalue and its constant eigenvector.

9. In the case of a generalized eigendecomposition and the normalized Laplacian, an equivalent matrix reconstruction may be obtained by using $\mathcal{L}_\mathcal{X} = \mathbf{U}_\mathcal{X}\mathbf{D}^{-1/2}\Lambda_\mathcal{X}\mathbf{D}^{-1/2}\mathbf{U}_\mathcal{X}^\top = \mathbf{V}_\mathcal{X}\Lambda_\mathcal{X}\mathbf{V}_\mathcal{X}^\top$.

$\mathbb{R}^N \mapsto \mathbb{R}^{\mathcal{D}}$.

Because it preserves local geometry, the Laplacian embedding projects a pair of isomorphic shapes onto two congruent point-sets in $\mathbb{R}^{\mathcal{D}}$ as required by Umeyama's framework. If the $\mathcal{D}$ eigenvalues of the reduced embeddings were distinct and reliably ordered, one could directly use Umeyama's theorem and the minimizer of $J(\mathbf{Q})$ (Eq. 4.7) in the reduced eigenspace:

$$\mathbf{Q}^* = \mathbf{V}_{\mathcal{Y}}^{\mathcal{D}} \mathbf{S}^{\mathcal{D}} \mathbf{V}_{\mathcal{X}}^{\mathcal{D}\top}. \tag{4.27}$$

Notice that $\mathbf{S}^{\mathcal{D}}$ is now $\mathcal{D} \times \mathcal{D}$.

If we cannot reliably order the eigenvalues: $\{\lambda_{\mathcal{X}1}, \ldots, \lambda_{\mathcal{X}\mathcal{D}}\}$, $\{\lambda_{\mathcal{Y}1}, \ldots, \lambda_{\mathcal{Y}\mathcal{D}}\}$, we need to apply a permutation $\Pi^{\mathcal{D}}$ among the corresponding selected eigenvectors:

$$\mathbf{Q}^* = \mathbf{V}_{\mathcal{Y}}^{\mathcal{D}} \mathbf{S}^{\mathcal{D}} \Pi^{\mathcal{D}} \mathbf{V}_{\mathcal{X}}^{\mathcal{D}\top}. \tag{4.28}$$

Again, $\Pi^{\mathcal{D}}$ is only $\mathcal{D} \times \mathcal{D}$. Let $\mathbf{R}^{\mathcal{D}} = \mathbf{S}^{\mathcal{D}} \Pi^{\mathcal{D}}$ and extend the domain of $\mathbf{R}^{\mathcal{D}}$ to all possible orthogonal matrices of size $\mathcal{D} \times \mathcal{D}$. This is done both, to find a closed-form solution and to deal with algebraic multiplicities. As a result:

$$\mathbf{Q}^* = \mathbf{V}_{\mathcal{Y}}^{\mathcal{D}} \mathbf{R}^{\mathcal{D}} \mathbf{V}_{\mathcal{X}}^{\mathcal{D}\top}. \tag{4.29}$$

Therefore, $\mathbf{R}^{\mathcal{D}}$ works as an orthogonal transformation aligning the $\mathcal{D}$-dimensional coordinates of the two point-sets :

$$\widetilde{\mathbf{V}}_{\mathcal{X}}^{\mathcal{D}\top} = \mathbf{R}^{\mathcal{D}} \mathbf{V}_{\mathcal{Y}}^{\mathcal{D}\top}. \tag{4.30}$$

Under the transformation $\mathbf{R}^{\mathcal{D}}$, the points $\boldsymbol{x}_i$ in the embedding space are transformed to points $\widetilde{\boldsymbol{x}}_i$, as follows:

$$[\widetilde{\boldsymbol{x}}_1 \quad \widetilde{\boldsymbol{x}}_2 \quad \ldots \quad \widetilde{\boldsymbol{x}}_N] = \mathbf{R}^{\mathcal{D}} [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \ldots \quad \boldsymbol{x}_N], \tag{4.31}$$

with $\boldsymbol{x}_i$ and $\widetilde{\boldsymbol{x}}_i$ $\mathcal{D}$-dimensional column vectors ($\mathcal{D} \times 1$). Under an exact local-isomorphism, the transformed set, $\widetilde{\boldsymbol{x}}$, perfectly matches the point-set $\boldsymbol{y}$:

$$[\boldsymbol{y}_1 \quad \boldsymbol{y}_2 \quad \ldots \quad \boldsymbol{y}_N] = \mathbf{R}^{\mathcal{D}} [\boldsymbol{x}_{\psi(1)} \quad \boldsymbol{x}_{\psi(2)} \quad \ldots \quad \boldsymbol{x}_{\psi(N)}], \tag{4.32}$$

In other words, the alignment matrix $\mathbf{R}^{\mathcal{D}}$ makes the estimation of the node permutation matrix $\mathbf{P}$ (or equivalently, of the graph correspondence map ψ c.f. Eq. 4.6) trivial, *i.e.* $\mathbf{P}$ can be found using a simple nearest-neighbor algorithm. To conclude, we can state the following proposition:

Proposition: Let two articulated shapes be described by two shape graphs as defined in section Sec. 3.2.1. Consider the Laplacian embeddings of these two graphs onto a $\mathcal{D}$-dimensional space. Solving for the graph isomorphism is equivalent to finding a solution for the point registration problem under the group of orthogonal transformations. Namely, estimating a $\mathcal{D} \times \mathcal{D}$ orthogonal matrix $\mathbf{R}^{\mathcal{D}}$ that aligns the reduced eigenspaces, and finding the trivial assignment ψ.

However, in real scenarios the number of samples defining each graph can be different. To be able to recover the assignment ψ, the problem must be reformulated as a *maximum common sub-graph isomorphism*, i.e. finding the largest match between sub-graphs in the two shapes. In practice, the spectral method gives a solution to the inexact graph-matching problem obtained by relaxing the constraints of the assignment ψ being strictly one-to-one and the two graphs having the same number of nodes. Furthermore, the the two shapes are never locally identical, due to the presence of spurious points, missing, bad, and/or noisy data, and so on. Thus, the two sets of points in the embedding-space will rarely match exactly. As a consequence, the embedding spaces may not

be perfectly aligned. So, to solve for ψ outliers need to be detected and a *residual* transformation may be required. One possible solution is to devise an algorithm that alternates between estimating the $\mathcal{D} \times \mathcal{D}$ orthogonal transformation $\mathbf{R}^{\mathcal{D}}$, which aligns the $\mathcal{D}$-dimensional coordinates of the two points sets, and finding an assignment ψ. One may observe that $\mathbf{R}^{\mathcal{D}}$ belongs to the orthogonal group $O(\mathcal{D})$. It is known that optimizing directly over the orthogonal group is difficult because it is composed of two connected components [10].

Since an alternation approach may lead to local-minima, we propose a solution to the *inexact graph matching* problem in two stages. First, we ensure the selection of a common eigen-subspace as explained in Sec. 4.2.6. Then, we rely on a robust point registration method, based on maximum likelihood optimization with latent variables, to find the correspondences, detect outliers and refine the alignment.

We present two methods for the common eigenspace selection which avoid relying on eigenvalue ordering or performing an expensive exhaustive search. The first method, based on *out-of-sample* extensions of non-linear embedding methods Sec. 4.2.7, gives an estimate of the sign matrix $\mathbf{R}^{\mathcal{D}}$ for temporal sequences. The second method is a more general approach founded on a direct *comparison of the eigenfunction histograms*, which guarantees a common eigen-subspace by estimating both matrices $\Pi^{\mathcal{D}}$ and $\mathbf{S}^{\mathcal{D}}$ and retaining only the best matching eigenfunctions (see Sec. 4.2.8). The algorithm that performs the registration follows the clustering formulation of the Expectation Maximization (EM) framework (c.f. Chap. 5, and is initialized with the $\Pi^{\mathcal{D}}$ and $\mathbf{S}^{\mathcal{D}}$ matrices obtained in the previous stage.

4.2.6 Common eigen-subspace selection

In addition to the ambiguities in Sec. 4.2.4, spectral methods applied to shape-graphs should guarantee that both graphs are mapped to a common eigen-subspace when performing the dimension-reduction. Spectral dimension reduction methods determine the reduced subspace by selecting its most significant eigenvalues and their corresponding eigenvectors. As we explain in the following, this method of selecting the subspace is not enough to compare articulated shape-graphs, because significant structural variations of the shape (*e.g.* self-contact changing the topology of the graph) may drastically affect some of the eigenvectors or even add/remove some of them. In consequence, the selection of a given number of eigenvectors according to their eigenvalues does not imply the selection of a common eigen-subspace.

For our application, the selection of a common eigen-subspace between pairs of embedded representations is crucial, in order to get a proper alignment and good node-to-node correspondences. A robust selection of such subspace is not straightforward because, as perturbation theory explains, small changes in the matrix can produce abrupt changes on eigenvectors associated to sensitive eigenvalues. Unfortunately, Laplacian matrices tend to give continuously decreasing curves for the eigenvalues from which it is difficult to tell apart the invariant subspaces. In addition to the problems related with multiplicities and noise, the selection of a common eigen-subspace has to deal with the effects of articulated motion and the structural changes in the shape-graph produced by sampling and self-contacts.

If we modeled both the articulated motion and the capture procedure in a parametric fashion, the Laplacian matrix could be expressed as a function of those parameters. Kato [Kato, 1995] has demonstrated that for the case of a single parameter θ, the eigenvalues and eigenvectors of a

10. The first formed by the orthogonal matrices with determinant -1, and the second, the group of rotation matrices (forming the special orthogonal group SO(3)) with determinant equal to 1.

symmetric matrix are analytic functions of θ on the real line. As a consequence, if the eigenvalue is plotted against θ, the result is a continuous function. The most relevant result in Kato [Kato, 1995] is that, the curves of different eigenvalues can cross at various exceptional points as θ varies. At each one of these values of θ, the plot of the crossing eigenvalues as a function of θ is smooth. However, the corresponding eigenvectors, change abruptly at these crossing points, as they change "from one eigenvector to a different one".

In our application, articulated motion is determined by the number of degrees of freedom, which "parametrize" the Laplacian matrices under study. As degrees of freedom increase, motion becomes more complex and more of the exceptional points at which eigenvalues cross are to be expected. Although the eigenvalue ordering is not reliable, the good geometric properties of the Laplacian still hold, and as we will show later, it will be possible to find the common eigenspace of two graphs by comparing eigenvectors.

An example of "crossing eigenvalues" is shown in Fig. 4.8 for a 10-frame sequence of voxel-sets of a simple articulated motion (Fig. 4.7). The figure was generated by matching similar eigenvectors across the sequence with a method that will be introduced in Sec. 4.2.8. As expected, similar eigenvectors do not necessarily correspond to the eigenvalue ordering (vertical axis). In particular, when self-contacts produce structural changes in the graph, as in frames 2-4.

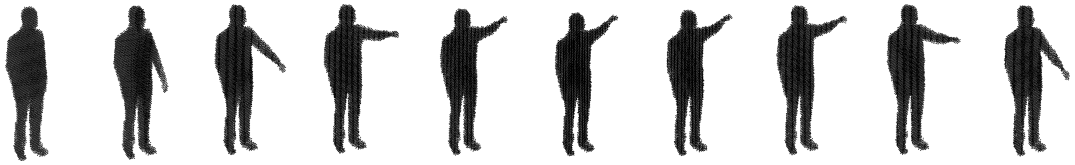


Figure 4.7: 10-frame sequence of voxel-sets captured with a multiple-camera system. The sequence describes a simple articulated motion of a man raising his arm.

The understanding of the ambiguities in Sec. 4.2.4 and the necessity of guaranteeing a common eigenspace have lead us to propose two extensions to spectral-graph matching methods, in order to effectively apply them for solving the articulated shape matching problem: a *temporal out-of-sample propagation* method and the *eigenfunction-histogram matching* described in Sec. 4.2.7 and Sec. 4.2.8.

Both methods start with a large enough reduced subset of eigenvectors, from which the common $\mathcal{D}$-dimensional subspace will be selected. One may argue that it is sufficient to consider a low-dimensional embedded space and to perform an exhaustive search. Indeed, is practical for $\mathcal{D} \leq 3$ provided that there is a reliable and efficient way to test each solution. However, in typical cases the dimension of the embedded space is higher than that. Although the literature is abundant with experiments aimed at guessing the dimensionality of the embedded space and its exploration for clustering, there is little information about the choice of $\mathcal{D}$ for shape matching. In [Jain and Zhang, 2006] it is claimed that $\mathcal{D} = 6$ yields satisfactory mesh correspondence and good results are obtained with a greedy method. We conclude that for reliable spectral shape matching, the dimension of the embedded space should not be too small, most certainly in the order of 10.

4.2.7 Out-of-sample extrapolation

This section describes an original method to select a common-eigenspace for a pair of graphs. The approach takes advantage of the *temporal coherence* of articulated shapes in a sequence and aims to relate the embedding of two shape-graphs coming from subsequent time frames. As the

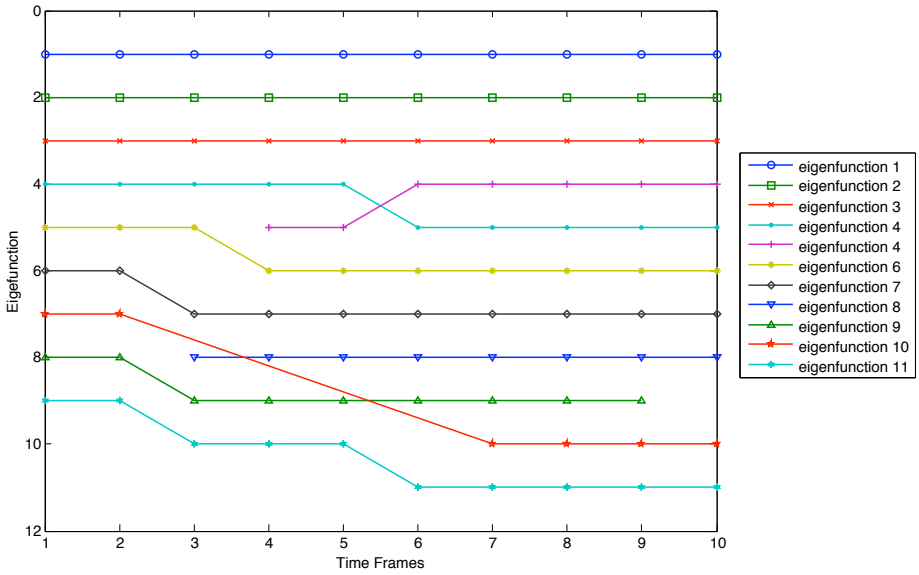
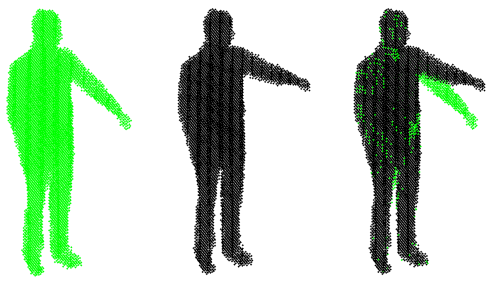


Figure 4.8: Behavior of the eigenvectors produced by articulated motion with self-contacts. We use the first frame as reference and keep track of each eigenvector along the sequence (colored lines). The vertical axis corresponds to the index initially assigned to each eigenvector using eigenvalue ordering. The figure uses the technique described in Sec. 4.2.8 to find the matches between the eigenvectors.

structure of the articulated object is not expected to drastically change from one frame to the next, it is reasonable to assume that the two point-sets of samples are similar and close to each other, as shown in the example of Fig. 4.9. If this assumption holds, the shape-graph at frame (t) can find a suitable representation in the embedding space of the shape-graph at $(t-1)$. Finding such representation can be related to the problem of extrapolating an embedding map to account for *out-of-sample* data-points [11] (c.f. Appendix A.1, Sec. A.1.4). The algorithms which solve this problem are known as *out-of-sample extensions* or *extrapolations* and aim to update the current embedded graph by incorporating the information of new samples, while avoiding computation of the entire embedding from scratch.

We follow the lines of the out-of-sample extension proposed by Bengio *et al.* [Bengio et al., 2004b], This solution relies first, on the interpretation of the nonlinear spectral embedding algorithms as kernel methods[Schölkopf and Smola, 2002](also c.f. Appendix A.1), and second, on the use of the Nyström formula [Baker, 1977] to approximate continuous kernel functions starting from discrete sets. The result is a general framework that allows us to compute the embedding of new samples for those non-linear embedding methods that can be related to kernels [Schölkopf et al., 1999, Bengio et al., 2004a], *e.g.* LLE, Laplacian-based embedding or Isomap. The goal of kernel-based out-of-sample extensions is to improve the convergence of eigenvalues and eigenvectors of a kernel matrix towards the eigenvalues and eigenfunctions of their ideal continuous kernel function (c.f. Sec. 2.3), by updating the embedding map as more samples are added [Williams and Seeger, 2000, Shawe-taylor et al., 2003, Bengio et al., 2004b].

11. Out-of-sample means they were not included in the initial set of samples from which a manifold was approximated.



frame (t-1) frame (t) (t-1) and (t)

Figure 4.9: Contiguous frames of a sequence of a simple articulated motion. The figure on the right shows the center of the voxels at time (t-1) is displayed in blue, and that of the voxels at time (t) in green.

The use of the out-of-sample extrapolation in this document is somewhat different from its original setting. As mentioned in the previous section, our objective is to estimate matrices $\Pi^{\mathcal{D}}$ and $\mathbf{S}^{\mathcal{D}}$, in order to initialize a point-registration algorithm in the reduced embedding-space. Thus, we are not interested in updating the initial graph and its embedding. In fact, for the final registration step, it is preferable to use the independently calculated embedding maps, since they truly reflect the intrinsic geometry of the original shape and are also more accurate than their approximate out-of-sample counterparts.

To use the out-of-sample extension in [Bengio et al., 2004b], the non-linear spectral embedding methods presented in Sec. 3.2.2.1 are interpreted in terms of kernel functions and kernel matrices (c.f. Appendix A.1). Given the sample-set $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ and its corresponding graph $\mathcal{G}_{\mathcal{X}}$ at time (t), a symmetric and positive semi-definite $N \times N$ matrix $\mathbf{K}$ is defined, encoding pairwise relationships between the nodes. In the case of the Laplacian embedding, the kernel matrix corresponds to the Laplacian matrices $\mathbf{L}$ or $\mathcal{L}$, whereas for LLE it is equivalent to the matrix $\mathbf{M}$ in Eq. 3.18. The values of $\mathbf{K}$ are seen as the result of evaluating a continuous kernel function $\mathcal{K}(\boldsymbol{a}, \boldsymbol{b})$ defined over pairs of nodes $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{X}$. Finally, the embedding map $\Phi^{\mathcal{D}}_{\mathcal{X}}(\mathcal{X})$ is constructed from the eigenvectors $\{\boldsymbol{v}^1_{\mathcal{X}}, \ldots, \boldsymbol{v}^{\mathcal{D}}_{\mathcal{X}}\}$ and eigenvalues $\{\lambda_{\mathcal{X}1}, \ldots, \lambda_{\mathcal{X}\mathcal{D}}\}$ of $\mathbf{K}$. Similarly, it is possible to obtain $\Phi^{\mathcal{D}}_{\mathcal{Y}}(\mathcal{Y})$ following the same procedure for the set $\mathcal{Y}$ captured at time $(t+1)$.

However, to relate the embedding of the two sets, $\Phi^{\mathcal{D}}_{\mathcal{X}}(\mathcal{X})$ can be extrapolated to account for points $\boldsymbol{Y}_j \in \mathcal{Y}$, which is equivalent to finding a map $\widetilde{y}_j = \Phi^{\mathcal{D}}_{\mathcal{X}}(Y_j)$. Based on the solution proposed in [Bengio et al., 2004b], such a map can be estimated by extending the kernel function $\mathcal{K}(\boldsymbol{a}, \boldsymbol{b})$, with $a, b \in \mathcal{X}$, to a more general kernel function that is also defined over samples in $\mathcal{Y}$. This new kernel function is denoted here $\widetilde{\mathcal{K}}(\boldsymbol{a}, \boldsymbol{b})$, with $a, b \in \{\mathcal{X}, \mathcal{Y}\}$. The method finds the map $\widetilde{y}_j = \Phi^{\mathcal{D}}_{\mathcal{X}}(Y_j)$ by approximating the eigenfunctions and eigenvalues of the extended kernel $\widetilde{\mathcal{K}}(\boldsymbol{a}, \boldsymbol{b})$.

To define the extended kernel $\widetilde{\mathcal{K}}(\boldsymbol{a}, \boldsymbol{b})$, the initial graph $\mathcal{G}_{\mathcal{X}}$ is augmented with a node $\mathbf{Y}_j$. Since there is no "natural" or "intrinsic" connectivity between the samples of two frames, an ϵ or a KNN-neighborhood strategy needs to be used to connect the new vertex. The edge weights can be estimated as for the original shape, following Eq. 3.2. The calculated edge weights are then transformed according to a particular embedding method (see below for the Laplacian and LLE methods). After these steps, the kernel $\widetilde{\mathcal{K}}(\boldsymbol{a}, \boldsymbol{b})$ can be estimated for every pair of points in $\{\mathcal{X}, \mathcal{Y}\}$. In [Bengio et al., 2004b], the approximated eigenvalues $\widetilde{\mu}_{\mathcal{X}k}$ and eigenfunctions $\widetilde{\boldsymbol{f}}^k_{\mathcal{X}}(\boldsymbol{Y}_j)$ of $\widetilde{\mathcal{K}}(\boldsymbol{a}, \boldsymbol{b})$

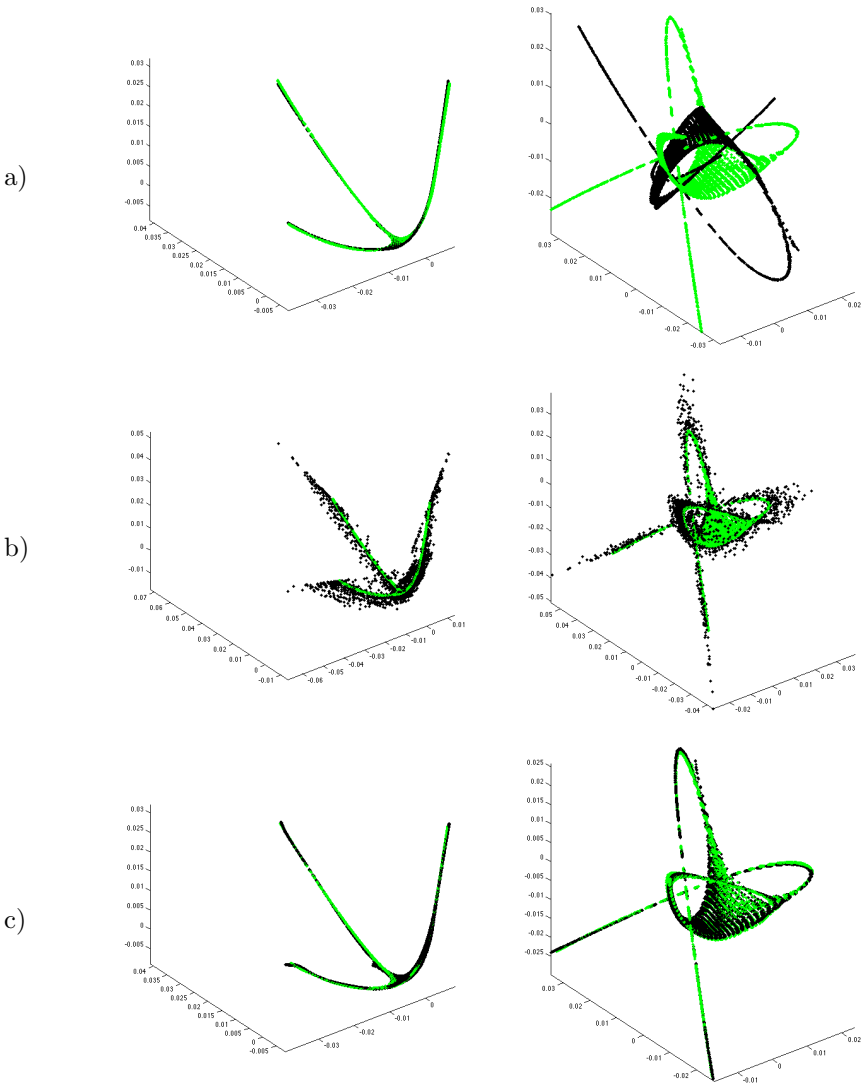


Figure 4.10: Aligning embedded shape-graphs with the out-of-sample extension technique in Algorithm 3. **(a)** Embedded graphs from shapes Fig. 4.9 obtained independently. The green point-set represents the shape at time (t-1) and the black one, the shape at frame (t). The point-sets corresponds to the subspace projection of the embedded graphs in to dimensions 1-3 (left) and 4-6 (right). Dimensions 4-6 are not aligned. **(b)** Out-of-sample approximation of the embedding of the shape-graph at time (t) from the embedding at time (t-1). Although the approximation leads to a scattered point-cloud, the point-sets are correctly aligned. **(c)** Alignment produced by estimating the transformation between the embedded shape-graphs and its out-of-sample approximation.

are derived as follows:

$$\widetilde{\mu}_{\mathcal{X}k} = \frac{\mu_{\mathcal{X}k}}{N} \tag{4.33}$$

$$\widetilde{\boldsymbol{f}}^k_{\mathcal{X}}(\boldsymbol{X}_l) = \frac{\sqrt{N}}{\mu_{\mathcal{X}k}} \sum_{i=1}^{N} \boldsymbol{v}^k_{\mathcal{X}}(i)\widetilde{\mathcal{K}}(\boldsymbol{X}_i, \boldsymbol{X}_l) = \sqrt{N}\boldsymbol{v}^k_{\mathcal{X}}(i) \tag{4.34}$$

$$\widetilde{\boldsymbol{f}}^k_{\mathcal{X}}(\boldsymbol{Y}_j) = \frac{\sqrt{N}}{\mu_{\mathcal{X}k}} \sum_{i=1}^{N} \boldsymbol{v}^k_{\mathcal{X}}(i)\widetilde{\mathcal{K}}(\boldsymbol{X}_i, \boldsymbol{Y}_j) \tag{4.35}$$

where $\boldsymbol{v}^k_{\mathcal{X}}(i)$ denotes the i^{th} entry of the vector $\boldsymbol{v}^k_{\mathcal{X}}$. Based on these values, the resultant embedding for a new point $\boldsymbol{Y}_j$, namely $\widetilde{\boldsymbol{y}}_j = \Phi^{\mathcal{D}}_{\mathcal{X}}(\boldsymbol{Y}_j)$, is:

$$\widetilde{\boldsymbol{y}}_j(k) = \frac{\tilde{\boldsymbol{f}}^k_{\mathcal{X}}(\boldsymbol{Y}_j)}{\sqrt{N}} = \frac{1}{\mu_{\mathcal{X}k}} \sum_{i=1}^{N} \boldsymbol{v}^k_{\mathcal{X}}(i) \widetilde{\mathcal{K}}(\boldsymbol{X}_i, \boldsymbol{Y}_j) \tag{4.36}$$

where $\widetilde{\boldsymbol{y}}_j(k)$ stands for the k^{th} coordinate of $\widetilde{\boldsymbol{y}}_j$. Notice that only computations of the form $\widetilde{\mathcal{K}}(\boldsymbol{X}_i, \boldsymbol{Y}_j)$ are required to estimate the embedding of the new point.

In the case of Laplacian-based embedding methods, the approximate kernel function is identified with the kernel used for weight estimation, *e.g.* the Gaussian kernel in Eq. 3.2, for the standard formulation of the Laplacian matrix $\mathbf{L}$ Eq. 2.11. Here, we note this kernel $\mathcal{K}(\boldsymbol{a}, \boldsymbol{b})$. For the spectral-clustering formulation of the Laplacian Eq. 2.15 an additional normalization step is required:

$$\widetilde{\mathcal{K}}(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{N} \frac{\mathcal{K}(\boldsymbol{a}, \boldsymbol{b})}{\sqrt{E_i[\widetilde{\mathcal{K}}(a, \boldsymbol{X}_i)]} \sqrt{E_j[\widetilde{\mathcal{K}}(a, \boldsymbol{Y}_j)]}} \tag{4.37}$$

where $E_i[\cdot]$ is the expectation computed over the set $\mathcal{X}$. The expected values relate to the node degree values introduced in Sec. 2.2.2. Finally, this extension uses the spectral-clustering formulation of the Laplacian Eq. 2.15. In order to convert eigenvalues and eigenvectors to the generalized formulation Eq. 2.18 used to calculate y (c.f. Sec. 3.2.2.3, the conversion $\lambda_{\mathcal{X}k} = 1 - \mu_{\mathcal{X}k}$ and a pre-multiplication by $\mathbf{D}^{1/2}$ of the eigenvectors are required.

For LLE, one possible approximate kernel is:

$$\widetilde{\mathcal{K}}(\boldsymbol{a}, \boldsymbol{b}) = \begin{cases} (\mu - 1) w(\boldsymbol{a}, \boldsymbol{b}) & \text{if } \boldsymbol{a} \in \mathcal{X} \text{ and } \boldsymbol{b} \in \mathcal{Y} \\ C_{ab} + C_{ba} + \sum_{l \in \mathcal{X}} C_{la} C_{lb} & \text{if } \boldsymbol{a} \in \mathcal{X} \text{ and } \boldsymbol{b} \in \mathcal{X} \\ 0 & \text{if } \boldsymbol{a} \in \mathcal{Y} \text{ and } \boldsymbol{b} \in \mathcal{Y} \end{cases} \tag{4.38}$$

where the elements C_{ij} are the linear reconstruction values used in the LLE embedding of $\mathcal{X}$ (c.f. Eq. 3.16) and $w(\boldsymbol{a}, \boldsymbol{b})$ are the linear reconstruction weights of a sample $\boldsymbol{b} \in \mathcal{Y}$ from its neighborhood points in $\mathcal{X}$.

Let the sample-sets $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ and $\mathcal{Y} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_M\}$ be the sample sets acquired respectively at time (t) and $(t+1)$. Apply independently Algorithm 1 or Algorithm 2 to obtain the embedded graphs $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$. The algorithm to find a common-egienspace between $\mathcal{X}$ and y reads as follows:

Algorithm 3 Aligning with Laplacian out-of-sample extended embedding

1. For a selection of points point $\boldsymbol{Y}_j$ from the $\mathcal{Y}$ shape at time $(t+1)$, find the neighborhood of $\boldsymbol{Y}_j$ in the shape $\mathcal{X}$, and compute the related adjacency weights.
2. Find the coordinates of $\boldsymbol{y}_i$ in the embedding space $y = \Phi_{\mathcal{X}}(\mathcal{Y}) = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$ using Eq. 4.36.
3. Estimate the rigid orthogonal transformation $\widetilde{\mathbf{R}}$ between the out-of-sample approximation $\widetilde{\boldsymbol{y}}$ and the independently calculated embedding y.
4. Apply transformation $\widetilde{\mathbf{R}}_{\mathcal{D}}$ to the point-set y.

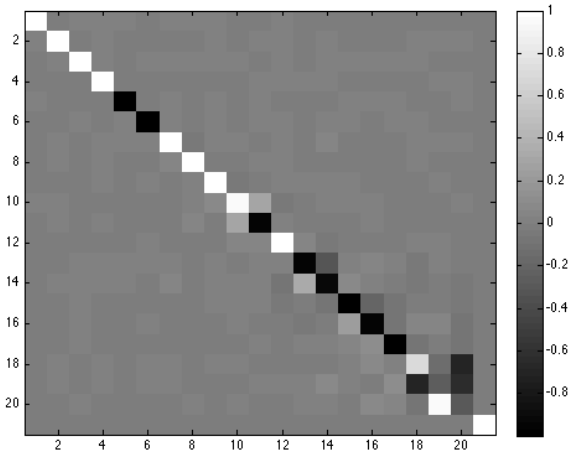


Figure 4.11: Estimated rigid transformation between the 20-dimensional embedding of the shape-graphs for two contiguous frames in the sequence. We can clearly see that up to the 9-th dimension, the transformation is essentially a diagonal matrix, with values close to (1) (in white) or (-1) (in black), which solves for the sign matrix $\mathbf{S}^{\mathcal{D}}$. Off-diagonal elements appear progressively as the dimension increases. To identify similar eigenfunctions we can impose a minimum absolute value on the diagonal of the transformation matrix. In this case, if the threshold $|diag(T)| > 0.99$ is imposed, the algorithm retains the eigenfunctions 1 to 9, but also 12.

An example of the method is shown in Fig. 4.9, Fig. 4.10 and Fig. 4.11. Although the change in poses in Fig. 4.9 is small, the eigenfunctions obtained by applying the Laplacian Embedding to the sample-sets in Fig. 4.9 are not "aligned", as shown for the subspace projections in Fig. 4.10-a). By using the out-of-sample extrapolation of the first set, it is possible to find a suitable representation for a selection of points in the second shape Fig. 4.10-b). The trivial correspondence between between the out-of-sample approximation and the independently computed embedding are used to obtain the transformation between the two sets with a least-squares estimation. In the ideal case, the transformation Fig. 4.10 is a diagonal matrix with 1 or -1 values, solving for the sign matrix $\mathbf{S}^{\mathcal{D}}$ in Eq. 4.28. A high threshold can be imposed on the diagonal matrix in order to guarantee that the registration takes place in a common eigenspace.

4.2.8 Alignment using Laplacian eigenfunctions histogram matching

A more sophisticated way to solve for the initial match is needed when the sample-sets do not come from subsequent frames in a sequence. In this section, we describe a method to find a common eigenspace relying on *matching the histograms of the Laplacian eigenfunctions*.

Histograms are a used extensively as a tool for matching [Lowe, 2004, S. Belongie and Puzicha, 2002]. Methods addressing a correspondence problem use histograms to create descriptors that are both invariant to the order in which data is presented, and simple to construct. For instance, it is common to use histogram-based descriptors to represent features and find image correspondences. Examples of histogram-based feature descriptors go from the basic gray-level and color histograms over a region of pixels, to the well-known SIFT[Lowe, 2004] and SURF [Bay et al., 2008] descriptors. Histograms have also been used for geometric shape matching. In [S. Belongie and Puzicha,

2002], Belongie *et al*. represent 2-D feature points by their *shape context*: a histogram of the relative positions from the point to other feature points. Other 2-D applications include image retrieval [Rubner et al., 2000][Indyk and Thaper, 2003][Grauman and Darrell, 2005],contour matching [Grauman and Darrell, 2004] and articulated and articulated shape matching [Ling and Jacobs, 2005]. In 3-D, histogram-based descriptors such as the *spin-image* [Johnson and Hebert, 1999] and the local-descriptor in [Gelfand et al., 2005], have been used for shape matching. Finally, in a closely related work, Rustamov [Rustamov, 2007] uses histograms combined with a Laplacian embedding of meshed surfaces. In brief, once the shape is mapped, points in the embedding space are selected. A collection of histograms describing the pairwise distances between the selected points is used to represent the shape. This compact representation leads global descriptor of the shape suitable for retrieval applications, where the distance between shapes is defined as a comparison of histograms.

In this chapter we discuss an original approach to create global descriptors of the shape from the histograms of the Laplacian eigenfunctions. As explained in the previous chapter (Sec. 3.2.2.6), the Laplacian embedding of shape-graphs is constructed from the eigenvectors of the Laplacian matrix, that is, the columns of the matrix $\mathbf{V}_{\mathcal{X}}^{\mathcal{D}} = [\boldsymbol{v}_{\mathcal{X}}^1 \quad \boldsymbol{v}_{\mathcal{X}}^2 \dots \boldsymbol{v}_{\mathcal{X}}^{\mathcal{D}}]$ (c.f. Algorithm 1). Each $\boldsymbol{v}_{\mathcal{X}}^k$, with $k = \{1, \dots, \mathcal{D}\}$, may be viewed as function mapping the graph $\mathcal{G}_{\mathcal{X}}$ onto $\mathbb{R}$. Similarly, a column of $\mathbf{V}_{\mathcal{Y}}^{\mathcal{D}}$, namely $\boldsymbol{v}_{\mathcal{Y}}^k$, maps $\mathcal{G}_{\mathcal{Y}}$ onto $\mathbb{R}$. As explained before, the eigenfunctions of two isomorphic graphs are identical up to a node-to-node assignment, under the condition that the order of the eigenvalues enough permits to identify corresponding eigenfunctions of the two graphs. However, empirical evidence points out that eigenvalue ordering is not reliable (hence the presence of matrix $\Pi^{\mathcal{D}}$) and we do not have a node-to-node assignment.

Let $\boldsymbol{v}_{\mathcal{X}}^k$ (respectively $\boldsymbol{v}_{\mathcal{Y}}^k$) be the k^{th} eigenfunction of the Laplacian associated with the graph $\mathcal{G}_{\mathcal{X}}$ (respectively $\mathcal{G}_{\mathcal{Y}}$). The vector $\boldsymbol{v}_{\mathcal{X}}^k$ can be seen as a set, whose histogram $\text{hist}(\boldsymbol{v}_{\mathcal{X}}^k)$ is invariant with respect to the order of the vector's components; in this case, to the order in which the graph nodes are considered. The histogram can be regarded then as an *eigenfunction signature*.

The problem of finding an estimate for the matrix $\Pi^{\mathcal{D}}$ can therefore be addressed as the problem of finding a set of assignments $\{\boldsymbol{v}_{\mathcal{X}}^k \mapsto \pm\boldsymbol{v}_{\mathcal{Y}}^l, 1 \leq k, l \leq \mathcal{D}\}$ based on the comparison of the eigenfunction signatures, namely their histograms. This is an instance of the standard bipartite maximum matching problem whose complexity is $O(\mathcal{D}^3)$. Notice however that the eigenfunctions are defined up to a sign (recall the matrix $\mathbf{S}^{\mathcal{D}}$ in Eq. 4.28). So, two different histograms $\text{hist}(\boldsymbol{v})$ and $\text{hist}(-\boldsymbol{v})$ can be associated with each eigenfunction and need to be compared. Let $\text{diss}(\text{hist}(\boldsymbol{u}), \text{hist}(\boldsymbol{v}))$ be a measure of dissimilarity between the histograms of two eigenfunctions $\boldsymbol{u}$ and $\boldsymbol{v}$. To ease the notation we use the following definitions:

$$H_{\mathcal{X}}^k = \text{hist}(\boldsymbol{v}_{\mathcal{X}}^k) \tag{4.39}$$

$$H_{\mathcal{Y}}^l = \text{hist}(\boldsymbol{v}_{\mathcal{Y}}^l) \tag{4.40}$$

$$H_{\mathcal{Y}}^{-l} = \text{hist}(-\boldsymbol{v}_{\mathcal{Y}}^l) \tag{4.41}$$

By computing the dissimilarity of all pairs of eigenfunctions $(\boldsymbol{v}_{\mathcal{X}}^k, \pm\boldsymbol{v}_{\mathcal{Y}}^l)$ we can build a $\mathcal{D} \times \mathcal{D}$ matrix $\mathbf{E}$ whose entries E_{kl} are defined by:

$$E_{kl} = \min(\, \text{diss}(H_{\mathcal{X}}^k, H_{\mathcal{Y}}^l),\ \text{diss}(H_{\mathcal{X}}^k, H_{\mathcal{Y}}^{-l})\,), \tag{4.42}$$

The matrix $\mathbf{E}$ can be interpreted as the cost of associating each pair of eigenfunctions. Since we are interested in finding the most compatible set of similar eigenfunctions to build our common eigenspace, the dissimilarity function has to be chosen carefully. This is discussed in the next section, along with some practical algorithms to calculate $\text{diss}(\text{hist}(\boldsymbol{u}), \text{hist}(\boldsymbol{v}))$.

4.2.8.1 Dissimilarity between eigenfunction histograms

There exist mainly two approaches to measure the difference between histograms: the *bin-to-bin* and the *cross-bin* methods. The former assumes that histograms are previously aligned and thus can be compared by defining a distance measure between corresponding pairs of bins. Examples of bin-to-bin measures are the L_p distances (Euclidean distance and Manhattan Distance), the χ^2 statistics [Snedecor and Cochran, 1989], the KL-divergence[Kullback, 1997] and the Jensen-Shannon JS-divergence[Lin, 1991, Puzicha et al., 1999]. However, for many applications histograms are not necessarily aligned. In such cases, *cross-bin* differences are more adequate. The class of cross-bin measures includes the Earth-Movers Distance (EMD)[Rubner et al., 2000], its optimized L1 version [Ling and Okada, 2006, 2007] and the diffusion distance [Ling and Okada, 2007].

In order to effectively compare eigenfunctions through their histograms several considerations have to be taken into account. First, unless the shapes are synthetically generated, there is a risk that the shape acquisition procedure leads to structural errors in the graph, such as missing or extra nodes. Up to a certain extend these errors can be handled by considering *cross-bin* differences. Furthermore, the difference in the number of nodes can be large, still for very similar objects (*e.g.* change in scale). In these cases, a normalization step that enforces the sum of bins to be one is necessary to ensure that the two histograms are comparable. Finally, intrinsic symmetries in the shape generate horizontal symmetries in the eigenfunction's histograms. As a consequence, the positive and negative histograms resemble each other, and it may not be possible to solve for the corresponding sign entry. The ambiguities generated by symmetries are a well known problem in matching. Perfect symmetries can only be solved by including a-priori knowledge in to the solution.

For the reasons above we chose to use the fast implementation of the Earth-Movers Distance (EMD)-L_1 introduced by Ling and Okada [Ling and Okada, 2006], and described in more detail in Appendix B.1. The advantage of EMD is that it naturally extends the notion of distance between single elements to distance between sets of elements, or distributions. This is a an interesting property in the context of shape matching problems, since it allows for partial matching. When used to compare distributions that have the same overall mass, the EMD is a true metric. Because of the cross-bin structure the EMD is less sensitive to fixed binning, and so it can be applied to signatures with different sizes. In practice, we use the code provided by the authors [Ling and Okada, 2007] for our experiments.

4.2.8.2 Linear assignment of histograms

Once the matrix $\mathbf{E}$ has been filled, the assignment problem can be solved using the Hungarian algorithm [Frank, 2005]. In our case, the Hungarian algorithm provides an optimal solution to the problem of finding an assignment between eigenfunctions of the two graphs (shapes), taking $\mathbf{E}$ as input and producing as output a permutation matrix. Because the sign ambiguity has been explicitly taken into account, this solves as well for $\mathbf{S}^{\mathcal{D}}$. This method provides a good initialization for the point registration method described in the following chapter (Chap. 5).

Let the sample-sets $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ and $\mathcal{Y} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_M\}$ be the sample sets acquired respectively at time (t) and $(t+1)$. The algorithm to find a common-egienspace between $\mathcal{X}$ and $\mathcal{Y}$ based on the comparison of the eigenfunction histograms is summarized in Algorithm 4 and illustrated in Fig. 4.12.

Finally, the result for the histogram matching used to illustrate the eigenvalue crossings (Sec. 4.2.6, Fig. 4.8) for a sequence of articulated motion, and particularly in the presence of a simple self-contact, it is shown in Fig. 4.13.

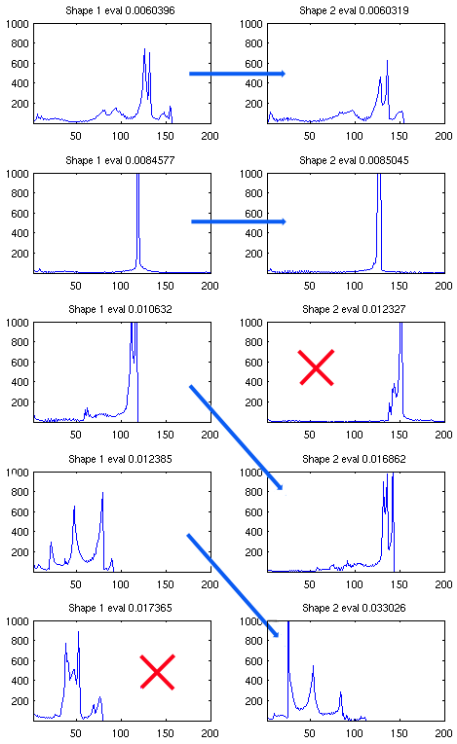
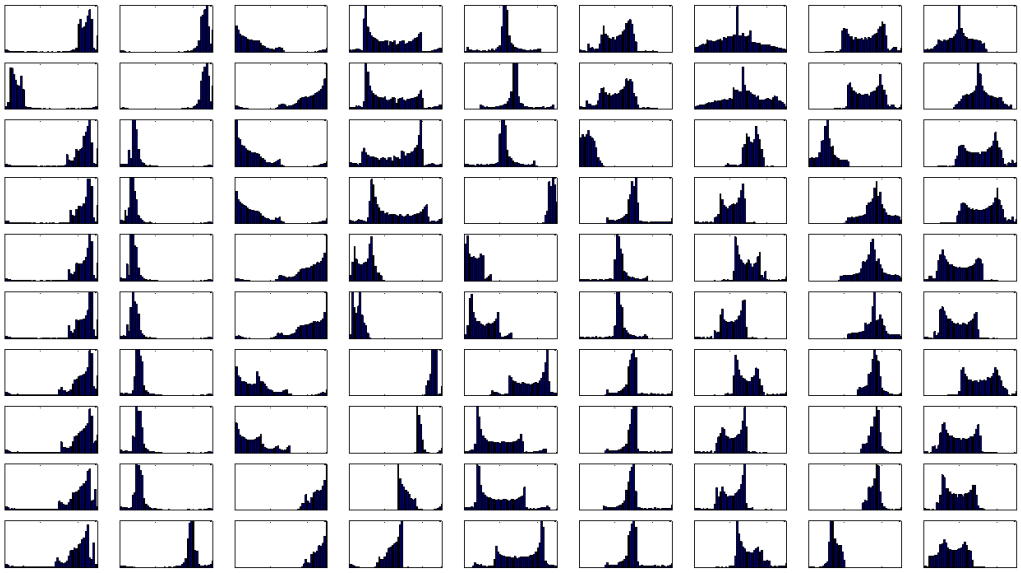


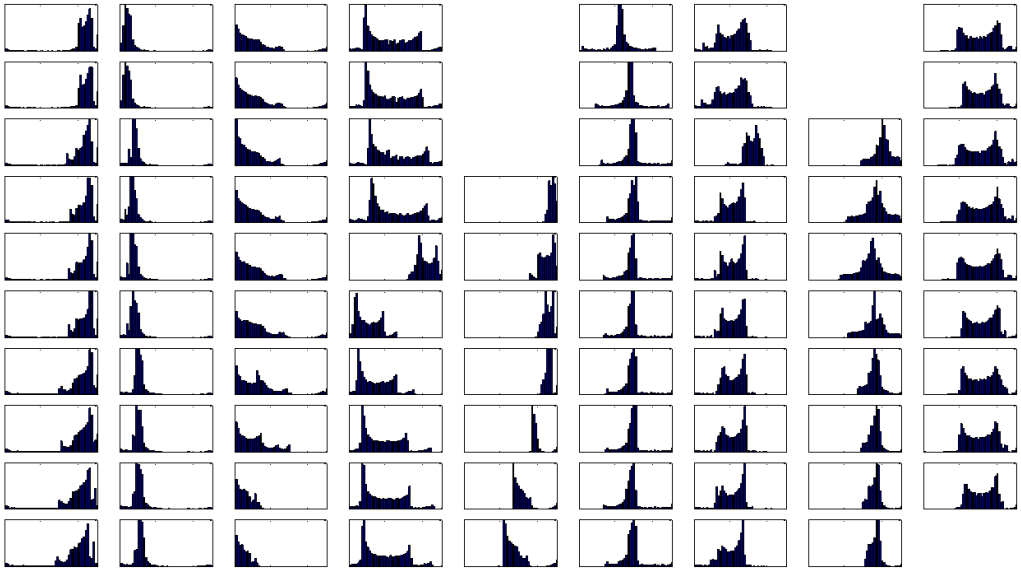
Figure 4.12: Histograms for two poses of the wooden-mannequin in Fig. 3.19. Each column represents the histogram of one of the poses. First 5 histograms out of the 20 compared. Assignments are shown with blue lines. Notice the corrected sign flip when the fourth histogram on the left is aligned with the fifth histogram on the right (horizontal reflection). Histograms marked with a cross are automatically discarded. The discarded histogram on the left, is eliminated since its comparison to histograms on the right does not lead to any distinctive low cost value. The histogram on the right is discarded by the hungarian algorithm.

Algorithm 4 Common eigenspace by matching eigenfunctions

1. Apply independently Algorithm 1 or Algorithm 2 to obtain the embedded graphs $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$. Retain a certain number of eigenfunctions ($\sim$20-30).
2. Build a histogram for each eigenfunction and for each shape. Normalize the histogram to have unit mass.
3. Compare the histograms based on a cross-bin measure and taking into account the sign ambiguity (compare both to the positive and the negative eigenfunction). For example, the EMD (Earth Mover's Distance) (c.f. Appendix B.1). Fill the matrix $\mathbf{E}$ with the values.
4. Prune the matrix $\mathbf{E}$ by removing lines for which the cost is not discriminant.
5. Apply the Hungarian algorithm to the matrix $\mathbf{E}$ to solve for the assignment between the eigenfunctions and thus for an initial matrix $\mathbf{R}^{\mathcal{D}} = \Pi^{\mathcal{D}}\mathbf{S}^{\mathcal{D}}$.
6. Apply $\mathbf{R}^{\mathcal{D}}$ to the point-set $\mathcal{Y}$.



a)Unordered eigenfunction histograms of the sequence in Fig. 4.7.



b)Eigenfunction histograms after selection and alignment.

Figure 4.13: Common eigenspace selection for a sequence of voxel-sets performing a simple articulated motion Fig. 4.7. The selection and alignment are obtained with the eigenfunction histogram matching algorithm described in Algorithm 4.

4.3 Conclusion

In this chapter we have presented the context and motivation for the articulated shape matching problem. We propose a solution to the specific case of articulated shape matching based on spectral methods. We establish a formal link between the classical theory for the spectral-graph matching methods with the non-linear embedding methods for creating a pose-invariant representations. This chapter further presents two theoretical extensions to Umeyama's theorem that allow us to use a classical spectral-graph matching algorithm on large graphs describing captured shapes (meshes or voxelsets). First, we use a reduced eigenspace where the correspondence problem can be solved as a point-registration under a rigid transformation. Second, we assure that the reduced eigenspace is common to both representations, by selecting and aligning eigenfunctions that are similar.

In the next chapter we will present the details of the algorithm for rigidly matching the embedded representations in the $\mathcal{D}$-dimensional embedding space.

Chapter 5

Robust Registration

Contents

5.1 Introduction

As a result of the common eigenspace alignment/selection methods presented in Chap. 4, we have a collection of eigenfunctions assigning a value to each vertex in each shape-graph. These values can be interpreted as the $\mathcal{D}$-dimensional coordinates of a point-set in the common eigenspace. Therefore, the final step towards finding a dense match between two articulated objects consists of registering the two $\mathcal{D}$-dimensional point-sets (c.f. Fig. 4.2). This step is straightforward for strictly local-isometric shapes. However, the local-isometry is only approximate in most of the data-sets which we deal with. As a consequence, the registration method should be able to handle different cardinalities, noise, missing data and other eventual discrepancies between the two point-sets.

The registration of point-sets is a classical problem in Pattern Recognition. Approaches mainly differ in the type of transformations applied to the point-sets, and in the way point-to-point correspondences are assigned. Given that both correspondences and transformations are unknown, the problem is usually addressed with an iterative solution. A classical method in the case of point-set registration under 3-D rigid transformations is the Iterative Closest Point (ICP) algorithm proposed by Besl and McKay [Besl and McKay, 1992]. Most of the recent approaches focus on the treatment of noise and missing data, and on handling deformations [Tsin and Kanade, 2004, Myronenko et al.,

2007, Zheng and Doermann, 2006]. Despite the advancement in the modeling of the problem, these non-rigid point-registration methods are still highly dependent on good initializations and are not very scalable.

Previous work in registering point-sets resulting from embedding algorithms include [Luo and Hancock, 2001, Jain and Zhang, 2007]. Zhang *et al.* [Jain and Zhang, 2007] propose to solve the problem using ICP for registration of point-sets resulting from embedding meshes of ~ 100 nodes. As discussed before, the performance of such a method can only be expected to be good for "clean" data. Hancock *et al.* [Luo and Hancock, 2001] proposed a probabilistic point-registration after embedding of generic graphs. Similar to their work, we model the problem in a probabilistic manner. However, as opposed to [Luo and Hancock, 2001] we provide a formal derivation of the Expectation Maximization (EM) algorithm, inspired by the probabilistic framework for clustering; this allows us to consider probabilistic assignments instead of the binary weights. Furthermore, we explicitly consider a uniform outlier class that, as shown in the experiments, will contribute to handling challenging registrations produced by discrepancies, noise and missing data.

As mentioned above, our model of the registration problem is inspired on the probabilistic approach to clustering. Each point in the first point-set is modeled as a probability density function, while points in the second set are treated as observations. The goal is to find the probability of each observation to belong to a cluster. The correspondences are modeled as latent variables and we solve the problem using Expectation Maximization (EM). We additionally constrain the set-up to consider the quasi-congruence of the point-sets. Under this constraint, the maximization step (M-step) results in the optimization over an orthogonal transformation that aligns the two point-sets. Finally, the correspondences are obtained from the posterior probabilities of the latent variables, which are updated during the Expectation step (E-step).

Results of the method are shown over different data-sets, including voxel-sets and meshes. We illustrate the performance of the EM algorithm both for point-sets sequences aligned under the temporal smoothness assumption (Sec. 4.2.7), and for widely-separated poses of articulated objects using the histogram eigenspace alignment/selection method (Sec. 4.2.8). We show registration results under challenging cases including topological-changes in the graph, 3-D reconstruction artifacts and significant discrepancies in the shape-graphs. Finally, we demonstrate the capability of the algorithm to register different articulated objects with similar topology.

5.2 Registration in the $\mathcal{D}$-dimensional embedding space

After the local non-linear spectral embedding of the shape-graphs (using the Laplacian embedding or LLE c.f. Chap. 3) and the eigenfunction selection/alignment procedure described in Chap. 4, two poses of an articulated object are mapped to a pair of quasi-congruent $\mathcal{D}$ point-sets, denoted by $\mathcal{X}$ and $\mathcal{Y}$. As a consequence, the articulated shape matching is cast into a *point-registration* problem. In this chapter we propose a solution based on the modeling of point-registration in the framework of probabilistic *clustering*.

The goal of probabilistic clustering is to find natural groups within a data-set. Each cluster is modeled as a probability density function (p.d.f) and data-points are treated as observations. The goal is to find the best assignments of observations to clusters by maximizing a likelihood function over the parameters of the p.d.f. that describe the clusters. To fit our registration problem into this framework, we treat points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ as observations, while each point in the second point-set, $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$, is considered to be the mean of a cluster modeled with a

multivariate normal distribution. Thus, the likelihood of an observation $\boldsymbol{x}_n$ to belong to a cluster m, with $1 \leq m \leq M$, has a $\mathcal{D}$-dimensional Gaussian distribution with mean $\boldsymbol{\mu}_m$ and covariance $\boldsymbol{\Sigma}_m$. Additionally, we consider an *outlier component*, which corresponds to the $(M+1)^{th}$ cluster, with uniform distribution that allows us to capture eventual differences in the shape-graphs produced by noise, missing or additional data-points. The likelihood of an observation to be an outlier is uniformly distributed over the volume V which contains the point-sets in the embedding space Formally, this yields:

$$p(\boldsymbol{x}_n|z_n = m) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \quad 1 \leq m \leq M, \tag{5.1}$$

$$p(\boldsymbol{x}_n|z_n = M+1) = \mathcal{U}(\boldsymbol{x}_n|V, 0). \tag{5.2}$$

where z_n is a discrete random variable taking values in the set $\{1, \ldots, M+1\}$. In other words, z_n indicates the assignment of a point $\boldsymbol{x}_n$ to one of the $M+1$ clusters. Our goal is to estimate the set of cluster parameters $\{\boldsymbol{\mu}_m\}$, $\{\boldsymbol{\Sigma}_m\}$, as well as the assignments z_n. This can be achieved by maximizing the the joint likelihood of observations and assignments $p(\mathcal{X}, \mathcal{Z})$. Given the definitions in Eq. 5.1 and Eq. 5.2, this likelihood is given by:

$$p(\mathcal{X}, \mathcal{Z}|\{\boldsymbol{\mu}_m\}, \{\boldsymbol{\Sigma}_m\}) = \prod_{n=1}^{N} \prod_{m=1}^{M+1} \big(p(\boldsymbol{x}_n|z_n = m)p(z_n = m)\big)^{\delta_m(z_n)}, \tag{5.3}$$

where the function $\delta_m(z_n)$ is equal to 1 if $z_n = m$ and to 0 otherwise. The corresponding log-likelihood function yields:

$$\ln p(\mathcal{X}, \mathcal{Z}|\{\boldsymbol{\mu}_m\}, \{\boldsymbol{\Sigma}_m\}) = \sum_{n=1}^{N} \sum_{m=1}^{M+1} \delta_m(z_n)\big(\ln p(\boldsymbol{x}_n|z_n = m) + \ln p(z_n = m)\big), \tag{5.4}$$

However, in our registration problem it is not possible to observe the assignments, therefore $\mathcal{Z} = \{z_1, \ldots, z_n\}$ are interpreted as latent variables with marginal distribution $p(z_n = m) = \pi_m$, where $0 \leq \pi_m \leq 1$ and $\sum_1^{M+1} \pi_m = 1$. Due to the presence of the latent variables $\mathcal{Z}$, the maximization of the log-likelihood in Eq. 5.4 does not have a closed-form solution. Nevertheless, the optimization can be carried out using the Expectation Maximization (EM) algorithm by considering the expectation of Eq. 5.4 with respect to the posterior of $\mathcal{Z}$ [Bishop, 2006]. As a result, the point registration problem can be solved by maximization of the following criteria:

$$E_{\mathcal{Z}}[\ln p(\mathcal{X}, \mathcal{Z}|\{\boldsymbol{\mu}_n\}, \{\boldsymbol{\Sigma}_n\}, \{\pi_n\})] = \sum_{n=1}^{N} \sum_{m=1}^{M+1} \alpha_{nm}\Big(\ln p(\boldsymbol{x}_n|z_n = m) + \ln \pi_m\Big), \tag{5.5}$$

where $\alpha_{nm} \doteq E[\delta_m(z_n)|\mathcal{X}]$ (see below). The EM algorithm maximizes Eq. 5.5 by iteratively updating the values α_{nm} in the *E-step*, and by estimating the parameters $\{\boldsymbol{\mu}_n\}$, $\{\boldsymbol{\Sigma}_n\}$ and $\{\pi_n\}$ that maximize Eq. 5.5 in the *M-step*.

The values of α_{nm} are computed in the *E-step* and correspond to:

$$\alpha_{nm} \doteq E[\delta_m(z_n)|\mathcal{X}] = \sum_{i=1}^{M+1} \delta_m(z_n = i)p(z_n = i|\boldsymbol{x}_n). \tag{5.6}$$

Here, $p(z_n = m|\boldsymbol{x}_n)$ is the posterior probabilities of z_n given the data, and can thus be interpreted as the posterior probability of a point $\boldsymbol{x}_n$ to be registered with a point $\boldsymbol{y}_m$. Aditionally, by means

of Bayes' theorem: $p(z_n = m|\boldsymbol{x}_n) = p(z_n = m, \boldsymbol{x}_n)/p(\boldsymbol{x}_n)$. Assuming that all points m have equal prior probabilities, *i.e.* $\pi_1 = \ldots = \pi_M = \pi_{in}$, and $\pi_{M+1} = 1 - M\pi_{in} = \pi_{out}$, and denoting by d_Σ the Mahalanobis distance [1], we obtain:

$$\alpha_{nm} = \frac{\exp\left(-\frac{1}{2}d_{\Sigma_m}(\boldsymbol{x}_n, \boldsymbol{\mu}_m)\right)}{\sum_{i=1}^{M}\exp\left(-\frac{1}{2}d_{\Sigma_i}(\boldsymbol{x}_n, \boldsymbol{\mu}_i) + \kappa\right)}, \quad 0 \leq m \leq M \tag{5.7}$$

where the parameter κ represents the contribution of the outlier cluster. The goal of the *E-step* is to update the values of the the posteriors α_{nm} over each iteration.

Given α_{nm}, the *M-step* estimates the remaining parameters $\{\boldsymbol{\mu}_n\}, \{\boldsymbol{\Sigma}_n\}$ by maximizing Eq. 5.5. Although it is possible to differentiate Eq. 5.5 with respect to the parameters and obtain a closed-form solution [Bishop, 2006], these estimates will be highly dependent on initialization, since Eq. 5.5 has many local-maxima. Therefore, we further constrain the problem to account for the a-priori knowledge about the quasi-congruence of the point-sets. Instead of M independent $\mathcal{D}$-dimensional means, we estimate a *global* rotation [2] $\mathbf{R}$, such that $\boldsymbol{\mu}_m = \mathbf{R}\boldsymbol{y}_m$. Therefore, we replace the likelihood of the observations (Eq. 5.1) by:

$$p(\boldsymbol{x}_n|z_n = m) = \mathcal{N}(\boldsymbol{x}_n|\mathbf{R}\boldsymbol{y}_m, \boldsymbol{\Sigma}_m), 1 \leq m \leq M. \tag{5.8}$$

We compute a *global diagonal* $\mathcal{D} \times \mathcal{D}$ covariance matrix $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_{\mathcal{D}}^2)$, constrained to be common to all the clusters, in the interest of avoiding convergence problems when a point $\boldsymbol{x}_n$ is infinitely close to a cluster center $\mathbf{R}\boldsymbol{y}_m$.

With these considerations the maximization step of the EM algorithm results in optimizing the following expression:

$$\mathbf{R} = \arg\max_{\mathbf{M}} E_{\mathcal{Z}}[\ln p(\mathcal{X}, \mathcal{Z}|\mathbf{M}, \boldsymbol{\Sigma}) \,|\, \mathcal{X}], \quad \mathbf{M} \in SO(\mathcal{D}) \tag{5.9}$$

which maximizes the *conditional expectation* taken over $\mathcal{Z}$ of the joint log-likelihood (of observations and assignments) *given the observations*.

Finally, denoting the current parameter estimate with superscript (q) and dropping out the constant terms, the negative expectation in Eq. 5.9 can be written as:

$$E\left(\mathbf{R}|\mathbf{R}^{(q)}\right) = \sum_{m=1}^{M}\xi_m\left(d_\Sigma\left(\overline{\boldsymbol{x}}_m, \mathbf{R}\boldsymbol{y}_m\right) + \ln\det\boldsymbol{\Sigma}\right), \tag{5.10}$$

where,

$$\overline{\boldsymbol{x}}_m = \sum_{i=1}^{N}\frac{\alpha_{im}}{\xi_m}\boldsymbol{x}_i \quad \text{and} \quad \xi_m = \sum_{i=1}^{N}\alpha_{im}. \tag{5.11}$$

The formal derivation outlined above guarantees the equivalence between the maximization of Eq. 5.4 and the minimization of Eq. 5.10 [McLachlan and Krishnan, 1997].

In the case of an isotropic covariance $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$, there exists a closed-form solution for the $\mathcal{D} \times \mathcal{D}$ rotation matrix $\mathbf{R}$, based on an extension of [Arun et al., 1987] to deal with weighted sum of squared differences and rotation matrices of arbitrary dimension (rather than 3-D). The optimal

1. $d_\Sigma(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a} - \boldsymbol{b})^\top\Sigma^{-1}(\boldsymbol{a} - \boldsymbol{b})$
2. The reader may remark the connection to the transformation $\mathbf{R}^{\mathcal{D}}$ in Chap. 4

rotation is recovered as the result of a point-set *Procrustes Alignment* in the $\mathcal{D}$-dimensional space, by minimizing the weighted sum of square differences:

$$\mathbf{R}^* = \arg\min_{\mathbf{R}} \sum_{i}^{M} \xi_m (\overline{\boldsymbol{x}}_m - \mathbf{R}\boldsymbol{y}_m)^2, \quad \mathbf{R} \in SO(\mathcal{D}). \tag{5.12}$$

As described in [Arun et al., 1987], the rotation that aligns the (already centered) point-sets $\mathcal{x}$ and y can be found in closed-form solution by computing:

$$\mathbf{R}^* = \mathbf{V}\mathbf{U}^\top, \tag{5.13}$$

where $\mathbf{U}$ and $\mathbf{V}$ result from the singular value decomposition of:

$$\mathbf{H} = \sum_{i=1}^{M+1} \xi_m \overline{\boldsymbol{x}}_m \boldsymbol{y}_m^\top, \tag{5.14}$$

Additionally, as in [Arun et al., 1987], we constrain the solution to belong to the $SO(\mathcal{D})$ group.

Additionally in the *M-step*, the covariance parameter is estimated from Eq. 5.10 by using the standard method [Bishop, 2006]:

$$\sigma^{2*} = \frac{1}{\sum_{m=1}^{M}\sum_{n=1}^{N} \alpha_{nm}} \sum_{m=1}^{M}\sum_{n=1}^{N} \alpha_{nm} ||\boldsymbol{x}_n - \mathbf{R}\boldsymbol{y}_m||^2 \tag{5.15}$$

Notice that the $\mathbf{R}^*$ and σ^{2*} are only estimated over the M inlier components. Indeed, the posteriors (α_{nm}) and weights (ξ_m) allow the *classification* of the observations into inliers and outliers.

At convergence, EM provides a locally optimal value for the maximum likelihood, as well as final estimates for the posterior probabilities. The posteriors assign each "mean" observation $\overline{\boldsymbol{x}}_m$ (Eq. 5.11) to a "cluster center" $\boldsymbol{y}_m$. Each $\overline{x}_m$ is the mean over all $\boldsymbol{x}_i \in \mathcal{x}$, built by weighting each $\boldsymbol{x}_i$ with its posterior probability of being assigned to $\boldsymbol{y}_m \in y$, and normalizing each such assignment by ξ_m. Therefore, our algorithm finds *one-to-one* assignments $\overline{\boldsymbol{x}}_m \Leftrightarrow \boldsymbol{y}_m$.

Given the posteriors and weights an inlier/outlier classification is performed. An observation is an outlier if $\alpha_{nM+1} > \alpha_{ni}$, with $1 \leq i \leq M$. On the other hand, if ξ_m is close to zero, the cluster does not have enough supporting observations and the corresponding data-point $\boldsymbol{y}_m$ is left unmatched.

For the remaining inlier features it is possible to obtain a *one-to-one* correspondence map ψ by assigning to each class $\boldsymbol{y}_m$ its most representative observation $\boldsymbol{x}_n$, according to the maximum ratio $\alpha_{nm}/\sum_{i=1}^{N}\alpha_{im}$. However, if the one-to-one condition is not required, a *many-to-one* correspondence map is derived by considering that an observation can support several clusters[3]. In this case ψ is obtained by assigning to each observation $\boldsymbol{x}_n$, a fixed number (β) of clusters. These clusters correspond to the β highest values of α_{nm}. The procedure is summarized in Algorithm 5.

In order to initialize the algorithm, we use the common subspace selection/alignment algorithms, presented in Sec. 4.2.7 and Sec. 4.2.8. After these methods are applied we set $\mathbf{R}^{(0)} = \mathbf{I}$. The variance σ^2 is estimated from the data, by taking into account the variance of the distances of a rough correspondence map obtained with using kNN neighbors, with a large number of neighbors, *e.g.* choose k to be 20% of the cardinality of the graph.

3. This is not unrealistic given the way samples, *e.g.* voxels and vertices, are obtained.

Algorithm 5 Robust point-registration in $\mathcal{D}$-dimensional space.

1. Initialize covariance $\mathbf{\Sigma}^{(0)}$ and the initial transformation $\mathbf{R}^{(0)}$. Determine the outlier component κ.
2. Iterate until convergence:

   E-step: compute the posterior probabilities α_{nm} associated with each observation using Eq. 5.7 from current estimates of $\mathbf{R}^{(q)}$ and $\mathbf{\Sigma}^{(q)}$

   M-step: estimate the transformation $\mathbf{R}$ and the covariance $\mathbf{\Sigma}$ using Eq. 5.12 and Eq. 5.15.

3. Classify observations in outliers and inliers, and detect the unmatched clusters.
4. Find correspondence map ψ from the inliers.

For the outlier component, we initially considered determining κ as a function of the volume occupied by the clusters, such that $\kappa = (2\pi)^{\mathcal{D}/2}(\det \mathbf{\Sigma})^{1/2}\pi_{out}/V\pi_{in}$, where V is the entire volume containing the two point-sets. However, this leads to numerical instabilities in high-dimensional spaces. We therefore fix κ to a small value (~ 0.1) from the beginning of the algorithm. Finally, the convergence is measured in terms of change of the likelihood.

5.3 Experimental results

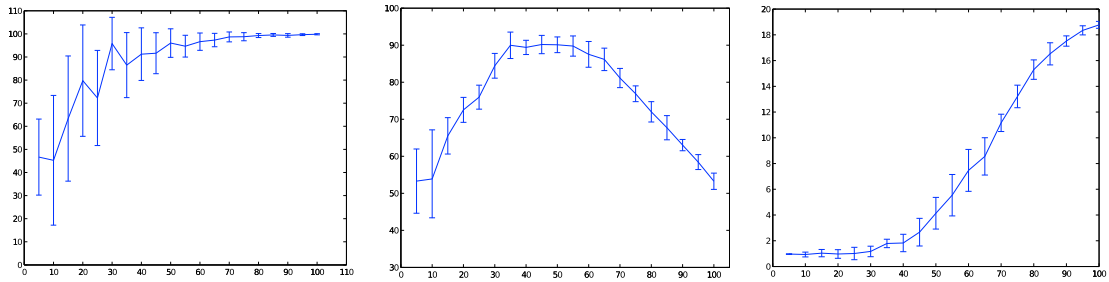
We applied our articulated-shape matching method (Fig. 4.2) to find dense-correspondences between pairs of 3-D articulated shapes from different data-sets, described as voxel-sets or meshed surfaces. First, we evaluate the proposed EM algorithm under controlled *synthetic* examples of rigid voxel-sets under point *permutations, rigid rotations and noise* (Sec. 5.3.1). Second, we show results of the algorithm using the out-of-sample method (Sec. 4.2.7) to initialize the EM, for a *real sequence* under temporal smoothness assumption (Sec. 5.3.2). Third, we demonstrate the performance for *three challenging cases* of voxel-set matching in Sec. 5.3.3, considering widely-separated poses and using the eigenfunction-histogram matching scheme (Sec. 4.2.8). Finally, the same algorithm is applied to match *meshed surfaces* of several articulated objects in widely-separated poses (Sec. 5.3.4). We provide a quantitative score for pairs of meshes for which *ground-truth* is available. We also analyze results for different poses of the Tosca database [4] of shapes, showing that our method is able to match objects within the same category (*intra-class*), but also across categories (*inter-class*), when similar topologies are considered [5].

5.3.1 Matching objects under rigid transformations

In a first series of experiments, we tested the performance of the presented EM scheme for matching a single pose of an articulated object under synthetically generated rigid transformations. To create a pair of shape-graphs to match, we took an initial voxel-set (~ 1300 voxels) and applied

4. http://tosca.cs.technion.ac.il/data.html
5. The code used to generate the results is available online at http://specmatch.gforge.inria.fr/

to its coordinates a random rotation [6], followed by a random permutation (which only modifies the order in which the voxels appear); this yields a second voxel-set. We then embedded the two resulting shape-graphs using LLE over an increasing number of dimensions $\mathcal{D}$, to finally perform the registration of the two point-sets in the embedding space with EM (Algorithm 5). The experiment was repeated for 100 transformations and over 20 voxel-sets of a sequence. The correspondences were computed using the *one-to-one* matching scheme and compared against ground truth. The performance is evaluated in terms of a matching score measuring the percentage of points for which the correct correspondence was found. The resultant matching score for $\mathcal{D}$ varying between 5 and 100, is illustrated in Fig. 5.1-a), where the matching score is shown to increase with the number of dimensions. The error bars represent the results over the 100 transformations and the 20 trials.



a) rotations + permutations b) adding Gaussian noise c) distortion(ψ)

Figure 5.1: a-b) Matching score (vertical axis) as a function of the embedding space dimension $\mathcal{D}$. a) For rotations and permutations. b) also adding Gaussian noise. c) distortion(ψ) corresponding to b). A clear inverse correlation with the matching score can be noticed. Average and standard deviation over 100 transformations and 20 repeated trials are shown.

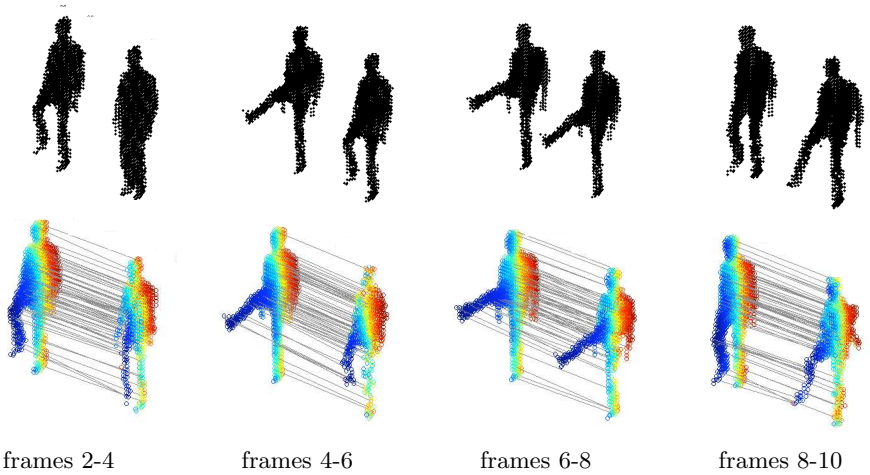
We performed a similar experiment by applying permutations as before and then adding Gaussian noise (with variance comparable to the voxel-grid size) to the coordinates of the voxels. The percentage of correct matches is shown in Fig. 5.1-b). This time, the score increases with the dimension only in the beginning. The maximum is attained at around $30 \leq \mathcal{D} \leq 50$, and the score subsequently declines for higher dimensions. The reason for this behavior is the distortion induced by the embedding functions. This effect is also illustrated in Fig. 5.1-c), where we measure the maximum absolute change of distances between a point $\boldsymbol{x} \in \mathcal{X}$ and its neighbors $\boldsymbol{x}_j$, $(i,j) \in \mathcal{E}(\mathcal{G})$, when a mapping ψ is applied. Analytically,

$$\text{distortion}(\psi) = \frac{1}{\text{avg}(\mathcal{N}(\mathcal{X}))} \max_{i=1,\dots,N} \max_{(i,j)\in\mathcal{E}(\mathcal{G}_\mathcal{X})} |\text{dist}(\boldsymbol{x}_i - \boldsymbol{x}_j) - \text{dist}(\boldsymbol{y}_{\psi(i)} - \boldsymbol{y}_{\psi(j)})|, \tag{5.16}$$

where $\boldsymbol{y}_{\psi(i)}$ indicates the correspondence of point $\boldsymbol{x}_i$ in the $\boldsymbol{y}$ set. The measure is normalized by the average size of the neighborhood, denoted here $\text{avg}(\mathcal{N}(\mathcal{X}))$. Thus, an ideal correspondence map should lie close to one. Fig. 5.1-c) plots distortion(ψ) showing the obvious increase of the distance with dimensionality, but additionally illustrating the correlation with the decrease of the performance

6. Obtained by sampling a 4-D vector from a uniform distribution, which we then interpret as the Rodrigues angle-vector representation of a rotation.

5.3.2 Voxel sequence under temporal smoothness



frames 2-4 frames 4-6 frames 6-8 frames 8-10

Figure 5.2: Matching results for temporally close poses coming from a real sequence of voxel-sets ("kick"). Top: Four pairs of poses extracted from the sequence. Bottom: The related correspondences (ψ) rendered as similar colors for corresponding parts of the body (a subset of the correspondences is also plot as lines).

In a second series of experiments, we used a sequence of voxel-sets generated from the 3-D reconstruction of a scene with a multi-camera system. Fig. 5.2-top shows sample poses coming from a sequence of a person performing a kick motion ($\sim$ 1500 voxels). We use temporally close poses (separated by 2 frames at 15fps) so that we can assume temporal smoothness, and therefore initialize the registration step through the out-of-sample method described in Sec. 4.2.7. Fig. 5.2-bottom shows the corresponding registration results. Correspondences are represented in color. Additionally, we measure distortion(ψ) (Eq. 5.16) for different values of $\mathcal{D}$, the results are shown in Fig. 5.3. The curves indicate that the chosen dimension should be lower than 25 in order to avoid problems linked to distortion.

5.3.3 Widely-separated poses: Three challenging cases

In this section, we demonstrate the performance of our approach on challenging cases for three objects: a wooden mannequin, a person, and a hand; all captured using a multiple-camera system and a silhouette-based voxel reconstruction algorithm.

In the first experiment, we match two widely-separated poses of an articulated object (the wooden mannequin in Fig. 5.4), one of which contains a self-contact (hands touching). We compute the Laplacian embedding of the shape-graphs (Fig. 5.5) and select a common eigenspace from the resulting eigenfunctions by eigenfunction-histogram matching as described in Sec. 4.2.8 (The histogram matching example shown Fig. 4.12 corresponds to these two poses). In this experiment the dimension of the embedding space after eigenfunction selection was $\mathcal{D} = 5$. Notice how the contact between the two hands alters the topology of the embedded shape (Fig. 5.5); without the eigenfunction matching stage, i.e. using only the eigenvalues to order the eigenfunctions, the point registration in the $\mathcal{D}$-dimensional space is difficult and the EM algorithm easily falls in a local min-

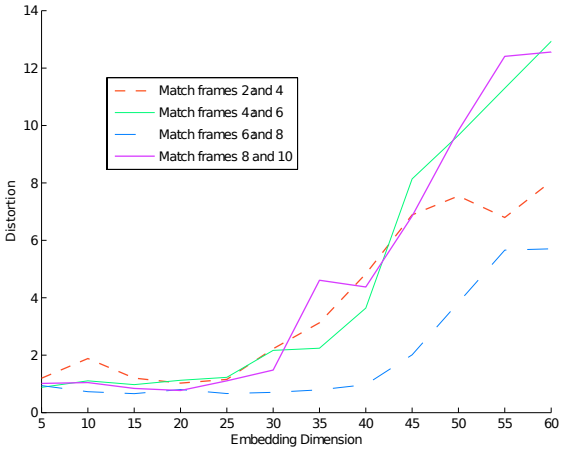
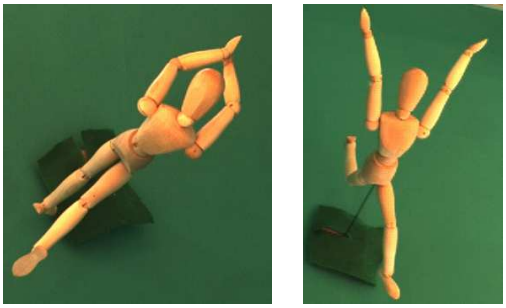


Figure 5.3: The distortion(ψ) for different values of $\mathcal{D}$ for the sample frames in Fig. 5.2.

imum. In contrast, the assignment after intialization with common eigenspace alignment/selection is correct and leads to a good final solution (see 5.7).

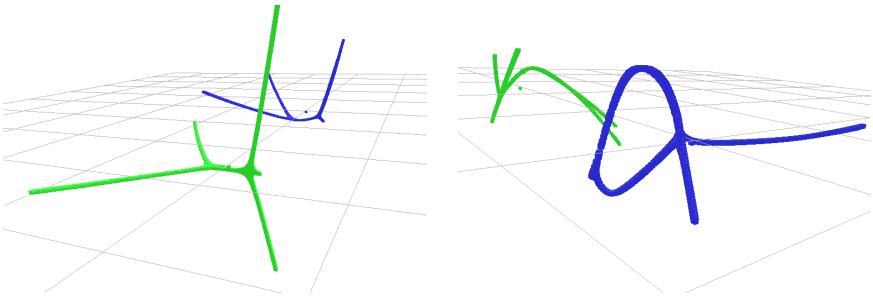


a) Pose 1 b) Pose 2

Figure 5.4: Two poses of a wooden mannequin. In a) the hands touch each other introducing important topological differences between the two associated shape-graphs.

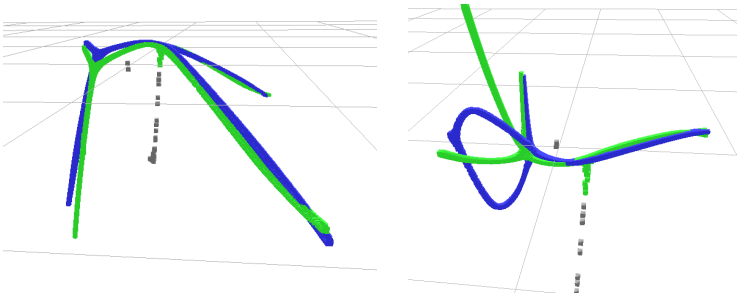
After the initialization, we compute the point-registration under a $\mathcal{D} \times \mathcal{D}$ orthogonal transformation using the EM algorithm described in Sec. 5.2. The algorithm is initialized by provision of an initial orthogonal transformation $\mathbf{R}$ and a covariance matrix $\mathbf{\Sigma}$. The eigenfunction alignment scheme of Sec. 4.2.8 serves as an initialization for $\mathbf{R}$. We use an isotropic covariance $\sigma^2\mathbf{I}$, where σ is chosen sufficiently large to allow evolution of the algorithm and $\mathbf{I}$ is the $\mathcal{D} \times \mathcal{D}$ identity matrix. Here, the radius σ of the initial covariance is approximately 15 voxel-width.

Fig. 5.6 shows the final alignment of the two embedded shapes of Fig. 5.5. Even though the hands' self-contact induces important differences in the graph topology, and thus in the embedded shape-graphs, there is still a significant set of points which preserves the same structure in both poses. Our graph-matching algorithm is capable of recovering the common sub-set and successfully finding the correspondences between the two point-sets, despite the presence of a large number of



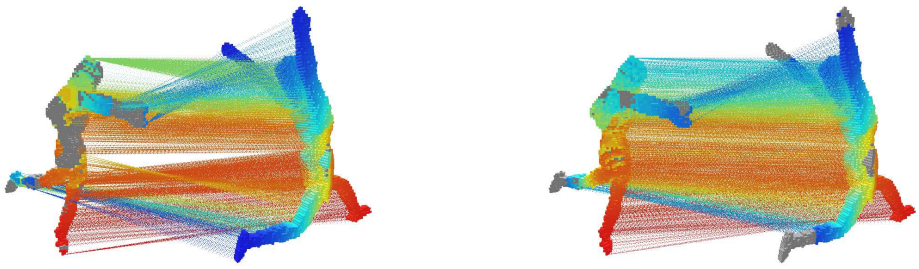
a) Eigenfunctions 1-2-3 b) Eigenfunction 1-4-5

Figure 5.5: Initial embedding: Embeddings of the two poses of the wooden mannequin in Fig. 5.4 (blue: pose 1, green: pose 2). The embeddings are represented as their three dimensional projections on two different subspaces. a) subspace spanned by eigenfunctions 1-2-3, b) subspace 1-4-5.



a) Eigenfunctions 1-2-3 b) Eigenfunction 1-4-5

Figure 5.6: Embedding result after common eigenspace selection and EM. Only the $\mathcal{D}$ retained eigenfunctions are shown.



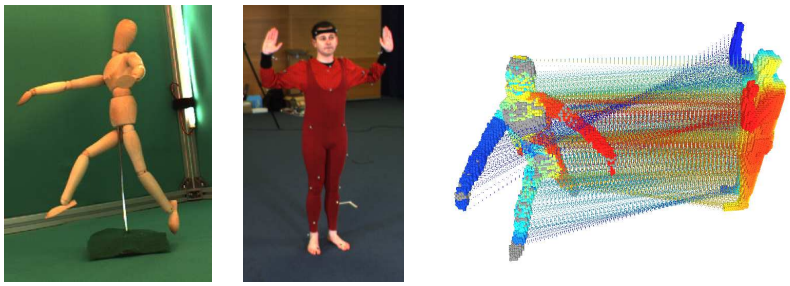
a) After eigenfunction alignment (EM initialization) b) After EM convergence

Figure 5.7: Dense match between the two poses of the wooden mannequin in Fig. 5.4. a) Assignment found after selecting a common eigenspaces with the eigenfunction-histograms matching scheme. b) Final result after convergence of EM. Points in gray illustrate outliers.

unmatchable points. This performance is due both to the properties of the Laplacian embedding and to the outlier rejection mechanism introduced in Sec. 5.2, which rejects these unmatched points

and prevents them from influencing the correctness of the orthogonal alignment.

Finally, Fig. 5.7 shows the registered sets of voxels after the eigenfunction alignment and EM. It is interesting to see how the initial registration, based on aligning the eigenfunctions of the Laplacian embedding, is already capable of providing a good assignment for the voxels in the limbs. This has a simple justification in terms of spectral clustering, since each eigenfunction corresponds to a well-identified group of voxels (c.f. Chap. 6).



a) Shape 1 b) Shape 2 c) Match between a) and b)

Figure 5.8: Matching the wooden mannequin to the 3-D model of a person is possible in our framework, as long as two shapes have a similar topology.

The second experiment in this section illustrates the capability of the proposed algorithm to match shapes from *different data-sets*. Fig. 5.8 shows the correspondences between the wooden mannequin and a 3-D model of a person. Even though the relative dimension and shape of the limbs in each object are different, our method delivers correct correspondences due to the similarity in their topology.

In the third experiment, we show the matching between two different poses of a hand Fig. 5.9. The bending of the little finger creates a different type of self-contact. The algorithm solves for most of the matches and in particular finds appropriate correspondences for the small finger (which is bent). The unmatched region in the palm reflects the change in the connectivity between the palm and the finger, which alters the embedded shapes.

a) Pose 1 b) Pose 2 c) Match between a) and b)

Figure 5.9: Matching two different poses of a hand showing bending and a different type of self-contact. Points in the extended finger are matched to points in the bent finger. The unmatched region in the palm reflects the local change of the connectivity.

Table. 5.3.3 shows additional information about the three experiments above. Notice that most of the points are matched in a few iterations of the EM algorithm. Discrepancies in the graph are reflected in the detected outliers and unmatched nodes.

Example	EM iterations	Number of voxels	Outliers+ Unmatched
Mannequin	6	12667 – 13848	472
Manneq-person	8	11636 – 12267	388
Hand	4	13594 – 13317	39

Table 5.1: Three challenging experiments

5.3.4 Widely-separated poses of articulated objects: Meshes

In this section, we illustrate the performance of the proposed method for registering articulated shapes represented as meshes. Here again, we combine the Laplacian-based embedding, together with eigenfunction selection/alignment of Sec. 4.2.8 and the EM algorithm proposed in Sec. 5.2.

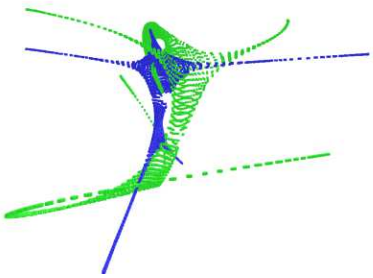
In the following experiments we first show the effectiveness of the eigenfunction selection and matching scheme in the case of meshed-surfaces. Then, we provide quantitative measures of the performance of the algorithm in terms of *correct matching percentage* for a synthetic sequence of meshes for which ground truth associations are available. Third, we illustrate results for *intra-class* mesh matching, *i.e.* for two poses of the same object, for shapes from the publicly available Tosca database. Finally, we show the capability of the method to perform *inter-class* matching, *i.e.* between meshes representing two different even though topologically similar shapes.

As we have shown in the case of voxel-sets, the eigenfunction selection/alignment scheme of Sec. 4.2.8 is also crucial to find correct correspondences between meshed surfaces. Fig. 5.10 illustrates how much this weighs on the final result. When the bottom $\mathcal{D}$ eigenvectors are selected based on eigenvalue ordering to define the embedding space for both shapes, the point-sets in the embedding space cannot be properly aligned, yielding mostly uncorrect associations (Fig. 5.10-a). If only similar eigenfunctions are used to build these embeddings, using their alignment as the initialization for the EM algorithm produces significantly improved results (Fig. 5.10-b).

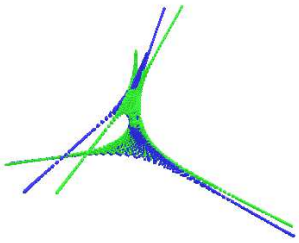
5.3.4.1 Performance score on mesh sequences with ground truth.

We apply our articulated-shape matching algorithm with one-to-one correspondences, to a synthetic sequence generated by deforming a mesh along time, according to an underlying articulated motion. Both topology and number of vertices are kept constant along the sequence; thus, ground truth associations between mesh vertices can be directly obtained. We compute a performance score based on the percentage of correct correspondences. Fig. 5.11 shows a selection of matching results for pairs of poses for the synthetic dance sequence.

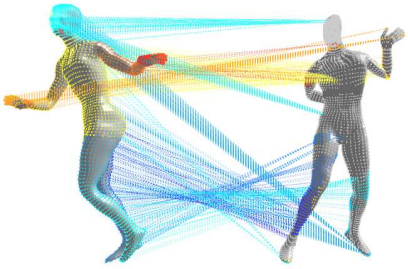
We can appreciate how widely-separated poses are correctly matched in all these situations, where any form of 3-D rigid point-registration would fail (notice the crossing legs of Fig. 5.11-a), or the difference between crossed and extended arms in Fig. 5.11-c). Table. 5.2 shows the percentage of correct correspondences for different pairs of meshes in the sequence. The percentage is computed for an increasing tolerance radius r on the surface. For $r = 0$, the score measures the percentage of exact correct correspondences. For higher values of r, the score also counts as correct all the



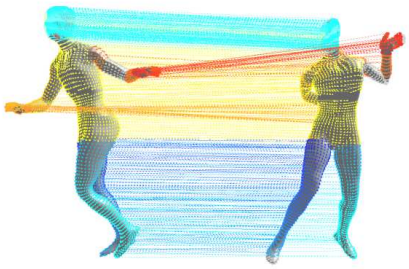
a) Embedded shape-graphs based on eigenvalue-ordering



b) Embedded shape-graph after eigenfunction selection



c) Correspondences after EM initialized with a)



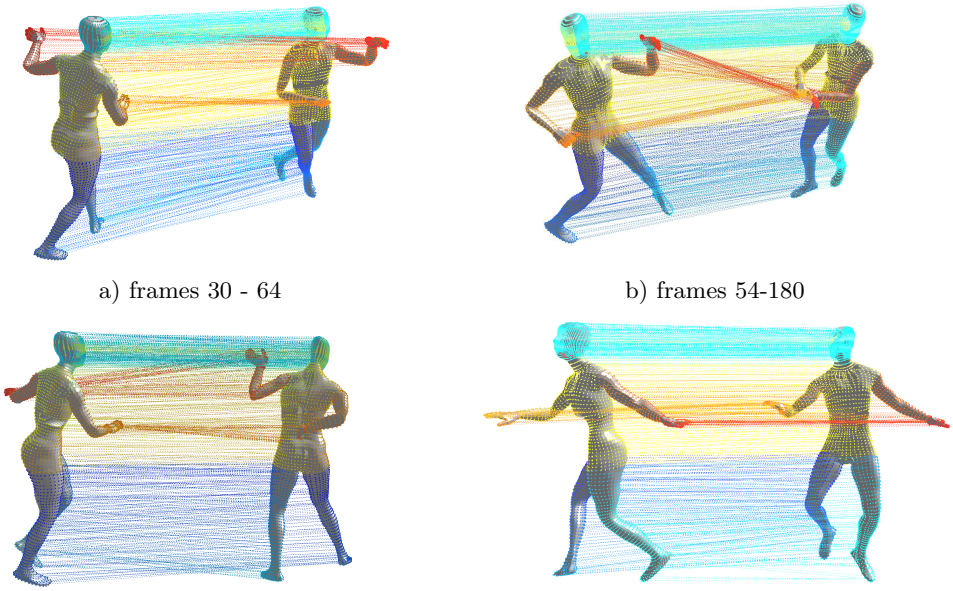
d) Correspondences after EM initialized with b)

Figure 5.10: Advantage of our algorithm against standard spectral graph matching. a): Alignment of embedded shapes (top) and point registration (bottom) based on eigenvalue ordering. b): The same based on the method described in this document.

correspondences that fall within a geodesic distance of 1 (computed on the shape-graph) from the ground-truth matching node. The radius r is considered in the geodesic sense, *i.e.* it is defined by a path of r edges. We refer to the "geodesic" area on the surface enclosed by a given radius r as the"r-ring". The values in Table. 5.2 indicate the good performance of the algorithm, above 80% in the worst case for a $r = 0$ tolerance and increase above 90% for a $r = 5$ tolerance. If we scale the sequence to human proportions, vertices are separated in average 3 cm, and the 5-ring tolerance gives matches within 15 cm. Such tolerance is acceptable considering the surface deforms non-rigidly to follow the underlying articulated motion.

Frames	r = 0	r = 1	r = 2	r = 4	r = 5
30 - 64	83.32	86.19	87.20	87.24	90.26
90 -95	92.31	92.40	92.41	92.63	93.23
69 -141	96.63	96.67	96.69	96.86	97.22
54-180	99.13	99.54	99.70	99.78	99.82

Table 5.2: Matching scores measured for different pairs of poses in the "dancer" sequence obtained by comparison with ground truth. The score describes the percentage of correct matches within a variable r-ring.



a) frames 30 - 64 b) frames 54-180

c) frames 69-141 d) frames 90-95

Figure 5.11: Correspondence maps found after applying the eigenfunction-histogram alignment/selection to initialize the EM algorithm, for different pairs of poses of a synthetic dance sequence.

5.3.4.2 Intra-class and Inter-class correspondences

In this section, we illustrate the results of applying our approach to pairs of articulated objects from the "Tosca" data-set. Two types of experiments are performed. In the first, we match different pairs of articulated poses of the same object, we refer to these as *intra-class* correspondences. In the second experiment, we match two objects that are either partially similar or have a common topology (*inter-class* correspondences).

To quantitatively assess the quality of the correspondences found by our algorithm in absence of ground truth, we propose to measure the smoothness of the correspondence map. We would like to verify that the correspondences in ψ map every edge in the first graph $(i,j) \in \mathcal{G}_{\mathcal{X}}$ to nearby locations in $\mathcal{G}_{\mathcal{Y}}$. Once again, we use the concept of r-rings to give a measure of distance which is independent of the sampling and size of the object. In practice, we evaluate the smoothness score per node. The correspondence of a node $i \in \mathcal{G}_{\mathcal{X}}$ is considered to be smooth if all of its immediate neighbors $(i,j) \in \mathcal{E}(\mathcal{G}_{\mathcal{X}})$ are mapped to nodes in the second graph $\psi(i)$ and $\psi(j)$ which are at most separated by an r shortest path distance, *i.e.* $\text{path}_{\mathcal{G}_{\mathcal{Y}}}(\psi(i),\psi(j)) < r$. Formally,

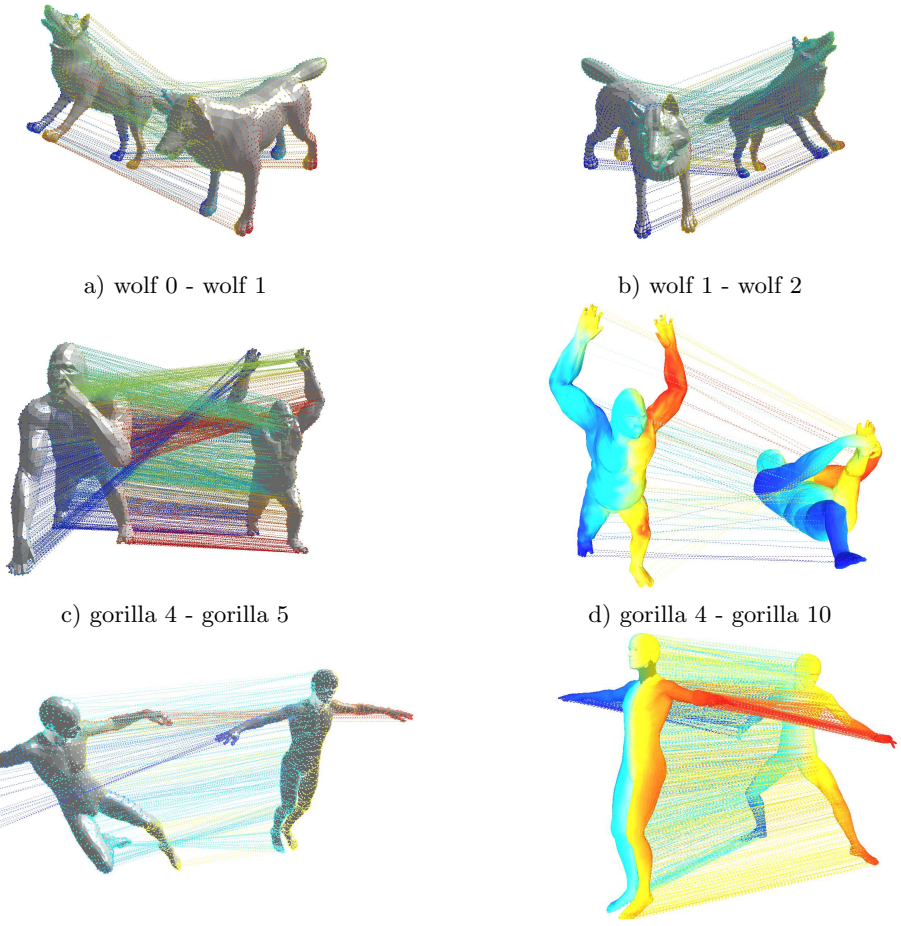
$$\text{smooth}(i,r) = \begin{cases} 1 & \text{if } \text{path}_{\mathcal{G}_{\mathcal{Y}}}(\psi(i),\psi(j)) < r \quad \forall (i,j) \in \mathcal{E}(\mathcal{G}(\mathcal{X})), \\ 0 & \text{otherwise.} \end{cases} \tag{5.17}$$

The score for the map ψ gives the percentage of smoothly mapped nodes. Notice that the score is not symmetric and should not be interpreted as a percentage of correct correspondences[7]. By

7. A correct map needs to be smooth but smoothness does not guarantee that correspondences between geometric

definition, it is a measure of the plausibility of the correspondence map, but it cannot measure its correctness since the ground truth is unknown.

Results for several examples of the *intra-class* matches are shown in Fig. 5.12. Colored lines illustrate the correspondences. The corresponding matching scores are reported in Table. 5.3, attesting the good quality of the matches. The maximum number of iterations for EM is 5 for meshes with ~ 2000 nodes.



a) wolf 0 - wolf 1

b) wolf 1 - wolf 2

c) gorilla 4 - gorilla 5

d) gorilla 4 - gorilla 10

e) david 0 - david 4

f) david 0 - david 10

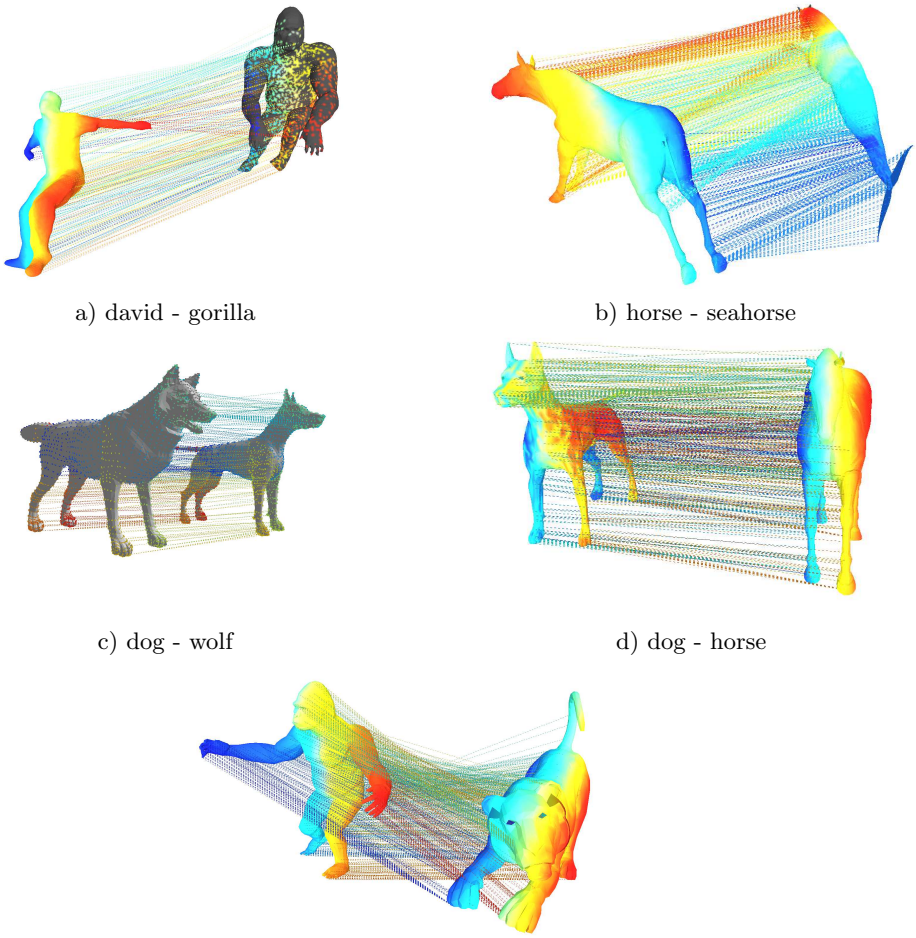
Figure 5.12: Examples of matching of the nodes of two meshes that represent the surface of different 3-D objects from the Tosca data-base. The algorithm is able to correctly match corresponding body-parts even under a dramatic pose transition. Colored lines illustrate the correspondences. Additionally, in d) and f) the facets of the mesh are also colored accordingly.

Finally, Fig. 5.13 shows the algorithm performs well even when matching meshes representing very different shapes. This is the case not only when the pose of the two objects is similar, or

features of the shape are obtained.

	r = 1	r = 2	r = 3	r = 4	r = 5	r = 6
wolf0-wolf1	71.09	77.60	81.60	85.10	88.19	90.88
wolf1-wolf2	67.73	74.62	79.10	83.25	86.75	89.80
gorilla4-gorilla5	92.24	93.11	93.43	93.45	93.49	93.50
gorilla4-gorilla10	91.72	92.38	93.02	93.24	93.38	93.45
david0-david4	62.24	62.43	67.52	70.18	72.04	73.31
david0-david10	72.27	75.30	78.25	79.04	81.22	83.37

Table 5.3: Map smoothness scores for the intra-class mesh pairs of Fig. 5.12.



a) david - gorilla

b) horse - seahorse

c) dog - wolf

d) dog - horse

e) gorilla - lioness

Figure 5.13: Registration results for pairs of objects belonging to different classes (from the Tosca data-base). The correspondences were obtained with the many-to-one approach with $\beta = 5$.

when the topology of the graphs is the more or less the same (*e.g.* a gorilla and a lion have each four extremities), but even when the topology is different (*e.g.* a match between a horse and a

"seahorse" where the tail matches a leg). When correctly initialized, the method delivers the most sensible match between body-parts. Meshes in this experiment have ~ 2000 vertices. The maximum number of iterations of the EM algorithm was 15. Given the discrepancies, the number of *one-to-one* correspondences is relatively low, *e.g.* 135 for gorilla-lioness (2046 and 3401 vertices) or 313 for horse-seahorse (2046 vertices). However, using the *many-to-one* scheme ($\beta = 5$) the number of matches is rises to 3266 for *gorilla-lioness* and to 1598 for *horse-seahorse*. Related smoothness scores are reported in Table. 5.3.4.2.

	r = 0	r = 1	r = 2	r = 3	r = 4	r = 5
david - gorilla	15.05	24.46	29.12	32.26	34.91	36.10
horse - seahorse	32.08	53.24	61.63	66.36	69.13	71.04
dog - wolf	45.50	64.38	72.71	76.79	79.31	80.01
dog - horse	44.16	59.36	64.96	67.57	69.48	72.16
gorilla - lioness	26.32	38.46	45.29	49.72	53.19	56.35

Table 5.4: Matching scores for the pairs of surface meshes belonging to different classes of objects reported in Fig. 5.13.

5.4 Conclusions

This chapter concludes the description of the method for establishing dense correspondences between the shape-graph representation associated with two articulated objects outlined in Fig. 4.2. We addressed the problem using non-linear spectral embedding, common eigenspace selection and unsupervised point registration. We formally introduced the use of the Laplacian matrix for embedding in the context of graph matching, and we show that the formulation as an inexact graph-matching problem leads to a point registration problem under the group of orthogonal transformations in the embedding space.

We provided an analysis of the matching problem whenever the number of nodes in the graph is large, i.e. of the order of 10^3. In particular we call attention to the fact that the eigenvalues of a large sparse Laplacian cannot be reliably ordered. We propose two alternatives to eigenvalue ordering, using either out-of-sample extensions or eigenfunction histogram matching. The point registration that results from eigenfunction selection/alignment algorithms serves as initialization for the EM algorithm, which is subsequently used to refine the registration.

The proposed EM algorithm leads to a general purpose unsupervised robust point registration method. The algorithm can deal with discrepancies between the two sets of points, by incorporating a uniform component in the mixture model. At convergence, the EM algorithm assigns a locally optimal "mean" observation to each cluster center, from which one-to-one or many-to-one correspondences can be obtained.

The good performance of the algorithm, shown over different types of data, is the result of several choices. First, the definition of a *local* graph-connectivity allows the treatment of the otherwise difficult cases, related to self-contacts and topology changes. These cannot be handled when completely connected graphs are defined since a self-contact would imply changes in all the pairwise distances. Furthermore, locality gives rise to very sparse Laplacian matrices and thus to an efficient calculation of the eigendecomposition (we use ARPACK [Lehoucq et al., 1998]). In addition, the initialization provided by the common eigenspace selection is usually good enough for EM to converge in a few

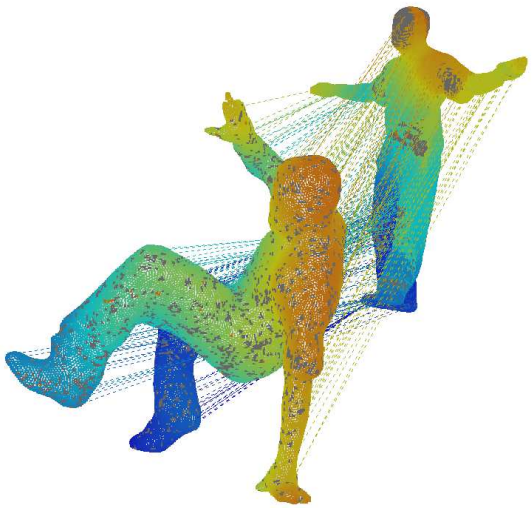


Figure 5.14: Reconstructed surfaces of a hip-hop dancer from a multiple camera system (Data courtesy of Adrian Hilton). The two meshes come from two different sequence and have 31600 and 34916 vertices respectively. The dimension of the embedding is 5 after eigenfunction selection/alignmet. As a result, 2186 one-to-one and 32730 many-to-one correspondences were found in 4 iterations of the EM algorithm.

iterations. As a consequence, the procedure is time-efficient, and that whole matching procedure of large voxel-sets takes only around 10 seconds for a data-set of the order of 10000 nodes.

We have shown that the proposed approach can deal both with voxel-sets and meshed surfaces. So it is general, in the sense that can be applied to other type of *structured* data, *e.g.* laser scans. When using our method with data-captured from multiple-camera reconstruction, one should favor voxel-sets over meshed surfaces. This is because the voxel representation leads to more stable embeddings and expresses the full volumetric information of the shape. Furthermore, the fact that the voxel-representations are regularly sampled contributes to the convergence of the Laplacian embedding towards a "geometrically-aware" eigenbasis. Additionally, the statistical properties of the eigenfunction-histograms also improve, leading to better initializations. Indeed, in sequences where large deformations are present (*e.g.* Fig. 5.14) an automatic initialization becomes difficult.

Chapter 6

Articulated Motion and Shape Segmentation

Contents

6.1 Introduction

As described in the introduction of Chap. 4, a large variety of approaches have been proposed in the literature addressing how capture and model the motion of articulated objects. In the previous chapters, Chap. 4 and Chap. 5, we have proposed a solution to the problem of registering wide-separated poses of 3-D articulated objects, such as those reconstructed from a multiple-camera calibrated system. In this chapter we will focus in the complementary problem of *segmenting* the parts of the articulated object according to *motion* and *shape* cues over time. The approaches presented here formulate the *motion and shape segmentation* as a clustering problem, where non-overlapping groups are sought according to a similarity measures. Appropriate similarities are defined by searching for invariances preserved under the type of applied transformations. In the case of articulated objects the straightforward invariance is the preservation of distances between points attached to the rigid segments, or equivalently, the local preservation of the shape. Using

the similarities defined in this manner we state respectively the *motion-segmentation* and *shape-segmentation* as clustering problems. We then solve these problems by means of the spectral methods introduced in Chap. 2 and Chap. 3 applied to clustering.

The goal of *motion segmentation* is to partition a collection of features into non-overlapping groups, each of which follows a coherent, usually rigid, motion. Most of the approaches for *multi-body motion-segmentation* focus on recovering 3-D motion from monocular videos, by tracking 2D features and relying on the principle that the resultant trajectories [Morris et al., 2001] live in a low-dimensional subspace, with dimensionality proportional to the number of rigid objects in the scene. In order to identify the subspace that represents the motion of each object similarities between trajectories are measured, and used to cluster the features. A common similarity measure used in this context is the *Shape Interaction* matrix (SIM) [Costeira and Kanade, 1998]. The SIM measures the similarity in terms of the dot product between trajectories. Weiss *et al.* [Weiss, 1999] showed that by interpretating the SIM as a similarity matrix, the algorithm derived by Costeira was equivalent to the spectral clustering of [Scott and Longuet-Higgins, 1991b]. Later, the use of the SIM was extended to other *clustering* algorithms [Gruber and Weiss, 2004, Feng and Perona, 1998, Inoue and Urahama', 2001, Wang and Culverhouse, 2003, Park et al., 2004]. However, the SIM-based similarities, assume independent motions and thus fail under rotations or correlated motion *i.e.* articulated motion. To overcome this limitation, Yan *et al.* [Yan and Pollefeys, 2006] proposed to first, estimate the subspaces locally and then, to cluster them based on a similarity that measures the angle between subspaces. However, local subspaces are computed based on spatial proximity, which is inappropriate in the boundaries of rigid parts or when uncorrelated trajectories come close.

In general, defining similarities directly from 2-D trajectories is difficult given the ambiguities inherent to 2-D velocity fields. Instead, we consider solving the problem in 3-D domain using multiple-views, as suggested in [Vidal et al., 2005, Tuzel et al., 2005]. In [Vidal et al., 2005], Vidal [Vidal et al., 2005]*et al.* propose the use of the multi-body epipolar constraint; on the other side, Meer *et al.* [Tuzel et al., 2005] perform clustering in the 3-D parameter space. In contrast, the first part of this chapter shows an application of the scene-flow method (explained in Chap. 7) to solve the motion-segmentation problem. We show that concentrating the efforts on the acquisition of scene-flow trajectories (multiple-frame scene-centered point trajectories), it is possible to define a pairwise similarity that directly measures the rigidity between trajectories. Finally, standard spectral clustering methods based on this similarity are used to recover the groups of features that move coherently.

The second part of the chapter deals with articulated body segmentation based on geometric properties of the shape (instead of appearance). Our primary goal is to segment articulated bodies *consistently over time*. Specifically, we target the segmentation of articulated bodies in sequences of 3-D reconstructions.

In the Computer Graphics and the Shape Analysis communities, shape cues, *e.g.* curvature, topology,*etc.*, have been extensively studied, permitting the segmentation of the objects according to their shape. Some of these methods are capable of generating consistent segmentations over isometric transformations (which preserve intrinsic distances) [Liu and Zhang, 2004, 2007, Rustamov, 2007, de Goes et al., 2008, Reuter, 2009]. However, these approaches focus on single-pose segmentation and tend to fail when the isometry assumption is not verified. In the case of sequences of articulated objects observed and reconstructed from multiple-camera systems, deformations, noise, and topological changes due to self-contacts prevent single-pose algorithms to deliver a consistent segmentation over time. It is therefore necessary to explicitly consider the temporal dimension to ensure consistency.

To solve the time-consistent shape-segmentation problem, we rely the shape-preserving characteristics of articulated objects and non-linear embedding methods, as introduced in Chap. 3. In the context of 3-D articulated objects observed by multiple cameras, recent approaches [Chu et al., 2003, Sundaresan and Chellappa, 2006] have used similar concepts to recover the pose of the object under motion. A critical issue in these methods is the presence of topological ambiguities raised by self-contacts (c.f. Fig. 3.2), as noticed by Sundaresan and Chellappa [Sundaresan and Chellappa, 2006]. In [Sundaresan and Chellappa, 2006] the problem is solved by means of an graphical model defined *a-priori* . In order to avoid the need of *a-priori* information, we propose to propagate clusters in the embedding space over time. This allows us to handle *any type* of topological change for *any type* of articulated object.

In summary, in this chapter we address the problem of segmenting parts of articulated objects in motion observed from a multiple-camera system over time. To give an unsupervised solution we focus on the two invariants that characterize the family of articulated objects, namely, their piecewise-rigid *motion* and the preservation *shape*. In a first approach Sec. 6.2 we analyze how to segment the parts of the objects in the scene by initially tracking features in 3-D, with a scene-flow method that will be detailed in Chap. 7, and then clustering feature trajectories that follow similar motion patterns. We will refer to this problem as 3-D *motion segmentation*. In the second approach, Sec. 6.3 we study how to coherently segment shapes based on the pose-invariance of the spectral representation introduced in Chap. 3 explicitly considering the problem over time. Both methods rely on the application of spectral theory presented in Chap. 2 to the problem of clustering, as detailed in the following section.

6.2 Motion Segmentation

The first part of this chapter focuses on *segmenting* the motion of an articulated or multiple rigid objects observed from a multiple-camera system. A set of 3-D interest points are initially reconstructed in 3-D, for example using an stereo matching algorithm. The features are then tracked in 3-D by means of the *scene-flow* algorithm that will be described in Chap. 7. The result is a set of trajectories describing the 3-D motion of the tracked features in time.

We formulate the articulated or multi-body motion segmentation as a clustering problem. Our objective is to group the trajectories of features undergoing the same rigid motion by defining an appropriate pairwise similarity measure. We use the *rigidity constraint*[1] to define a similarity measure between pairs of trajectories. The major difficulties with scene-flow data are are the possible drift over time and the variable duration of the trajectories, as features may be detected or lost at any frame. As a consequence, trajectories can not be directly represented in a vector space. In practice, we use the variance of the pairwise Euclidian distance, measured over the temporal intersection of trajectories, to define our similarity measure. Given the similarity, we use an standard spectral clustering method [Ng et al., 2002] to cluster the trajectories. As mentioned in the introduction of this chapter, our method differs from previous motion segmentation approaches using spectral mappings in both the type of data used to cluster and the measure of similarity. We illustrate the application of our scene-flow method (Chap. 7) to motion-segmentation, with results for independently moving objects and for articulated objects such as humans. In the following sections we define the proposed similarity measure and describe in detail how to use spectral clustering to solve the motion-segmentation problem.

1. According to the *rigidity constraint*, two points on a rigid object maintain a constant Euclidean distance.

6.2.1 Similarity between scene-flow trajectories

Let N be the number of feature points $\{p_1, p_2, \ldots, p_N\}$ detected on the surface of the objects and tracked during at-most F frames. The scene-flow coordinates are stacked to build a representation of the 3-D trajectory $\boldsymbol{s}_i$ of each feature along the sequence.

Let $\mathcal{S}$ be the set of N trajectories $\mathcal{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_N\}$ in a time sequence of F frames. The content of the i^{th} trajectory at a particular time frame t, $\boldsymbol{s}_i^{(t)} \in I\!R^3$, corresponds to the 3-D global coordinates of the point $p_i^{(t)}$ at frame t. Given the sparse nature of the scene-flow method, features can be added or lost at any time frame. Therefore, a binary *activity mask*, $\boldsymbol{m}_i = \{m_i^{(1)}, \ldots m_i^{(F)}\}$, with $m_i^{(t)} \in \{0, 1\}$, is attached to each trajectory to indicate the frames where trajectory is effectively tracked, as shown in Fig. 6.1. We define the *life-span* of a feature as the total number of frames during which a feature is active.

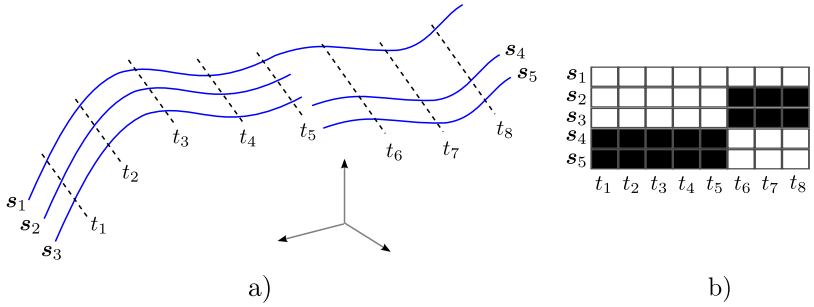


Figure 6.1: a) Five scene-flow trajectories $\boldsymbol{s}_1$, $\boldsymbol{s}2$, $\boldsymbol{s}3$, $\boldsymbol{s}4$ and $\boldsymbol{s}5$. b) Binary mask indicating the time frames in which each trajectory is active (white: active, black: inactive). The distance between non overlapping masks (*e.g.* $\boldsymbol{s}_2$ and $\boldsymbol{s}_4$) is set to ∞.

The motion segmentation problem is formulated as finding optimal cuts on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where each trajectory $\boldsymbol{s}_i$ is represented by a node $i \in \{1, \ldots, N\}$. Initially a completely connected graph is defined, *i.e.* every node in the graph is connected to every other through $\mathcal{E}(\mathcal{G})$. The weights $\mathcal{W}$ are defined to measure the *rigidity* between two trajectories. First, for each pair of trajectories $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, we calculate the Euclidean distance between the position of the features at each time t, *i.e.* $e^{(t)} = \| \boldsymbol{s}_i^{(t)} - \boldsymbol{s}_j^{(t)} \|$. The result is a distance vector $\boldsymbol{e}$ of length F. Given that the distance between two points following the same rigid motion is preserved, we examine the variance of $\boldsymbol{e}$ over the sequence to determine whether $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ are points in the surface of the same rigid object. Formally:

$$e^{(t)}(i,j) = \begin{cases} \| \boldsymbol{s}_i^{(t)} - \boldsymbol{s}_j^{(t)} \| & \text{if } m_i^{(t)} m_j^{(t)} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

$$\text{dist}(i,j) = \begin{cases} var(\boldsymbol{e}) & \text{if } \sum_{t=1}^{F} m_i^{(t)} m_j^{(t)} \neq 0 \\ \infty & \text{otherwise} \end{cases} \tag{6.2}$$

The distance $e^{(t)}(i,j)$ is only calculated for active entries ($m_i^{(t)} = 1$), and between intersecting trajectories (Eq. 6.1). The symmetric adjacency matrix $\mathbf{W}$ associated to the graph $\mathcal{G}$ is obtained

from $e^{(t)}(i,j)$ (c.f. Eq. 3.7). The entries w_{ij} of $\mathbf{W}$ are computed as:

$$w(i,j) = \exp\left(\frac{-\text{dist}(i,j)^2}{2\sigma^2}\right) \tag{6.3}$$

where, σ is a scale parameter that can be estimated from the data statistics, as explained in Sec. 6.2.2.

6.2.2 Clustering scene-flow trajectories with the spectral methods

To cluster the trajectories based on the similarities defined above, we use an standard spectral clustering method, based on the algorithm proposed by Ng *et al.* [Ng et al., 2002] to carry out the segmentation of our 3-D trajectories. This algorithm looks for multiple clusters simultaneously, by first embedding the data-points using the spectral analysis of the similarity matrix, and then performing a k-means clustering in the embedding space. The steps of the algorithm are summarized in Algorithm 6.

Algorithm 6 Motion segmentation based on scene-flow trajectories

1. Detect and track 3-D features over the sequence (c.f. Chap. 7).
2. Calculate all the $\text{dist}(i,j)$ for each pair of trajectories (Eq. 6.2).
3. Estimate the scale parameter σ.
4. Perform the eigen-decomposition of the generalized system $\mathbf{W}\boldsymbol{v} = \lambda\mathbf{D}v$ .This is equivalent to the decomposition of the *random-walk Laplacian* $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ in equation Eq. 2.16.
5. Select the $\mathcal{D}$ eigenvectors corresponding to the greatest eigenvalues and pile them in the columns of a matrix, which we denote here by $\mathbf{V}^{\mathcal{D}}$.
6. Normalize the rows of matrix $\mathbf{V}^{\mathcal{D}}$ to obtain $\overline{\mathbf{V}}^{\mathcal{D}}$.
7. Apply k-means clustering to the rows of matrix$\overline{\mathbf{V}}^{\mathcal{D}}$.

Motion from articulated objects can be corrupted by noise and have different degrees of correlation, therefore it is important to find the right scale parameter σ that weights the distances while converting them to similarities. Fischer *et al.* [Fischer and Poland, 2005] and Zelnik *et al.* [Zelnik-Manor and Perona, 2005] have proposed local scaling for adapting the scale to a neighborhood. We use the [Zelnik-Manor and Perona, 2005] definition, where a local scale is build from $\sigma_{ij} = \sigma_i\sigma_j$; here $\sigma_i = \text{dist}_4(i,j)$ and $j \in \mathcal{N}(i)$ are the nearest neighbors of the trajectory i. When using this definition, an additional step is required to recover the symmetry. The algorithm is able to uncover rigid motion and articulated motion as shown in the examples below.

6.2.3 Examples of motion-segmentation on scene-flow data

In this section we illustrate the application of the scene-flow method to motion-segmentation. We use video sequences from a calibrated multiple-camera (6-8 cameras) system observing a scene where rigid or articulated objects move. In the first sequence we observe two persons moving in the scene (one walking and one waving the arms) during 1000 frames at 15fps. After clustering and reordering the rows according to the clusters, the similarity matrix (see Fig. 6.2) reveals two large clusters and several correlated partitions within. In Fig. 6.3, we illustrate some the trajectories

colored according to the obtained labeling. Since we can not describe rigid motion from less than three correspondences, we classify groups of less than 3 trajectories as outliers.

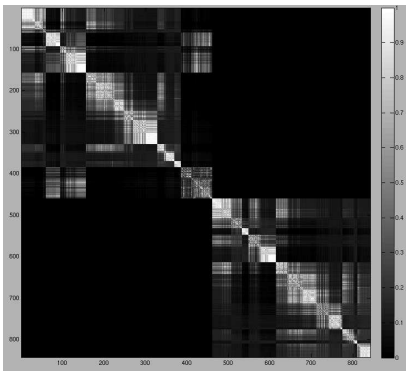


Figure 6.2: Reordered imilarity matrix for the sequence in Fig. 6.3

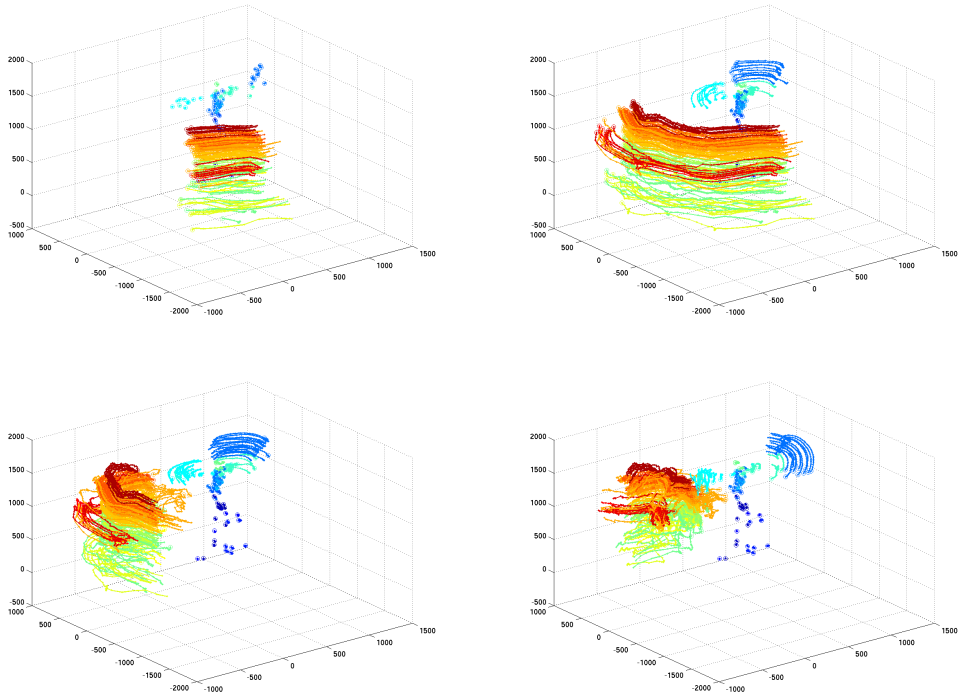
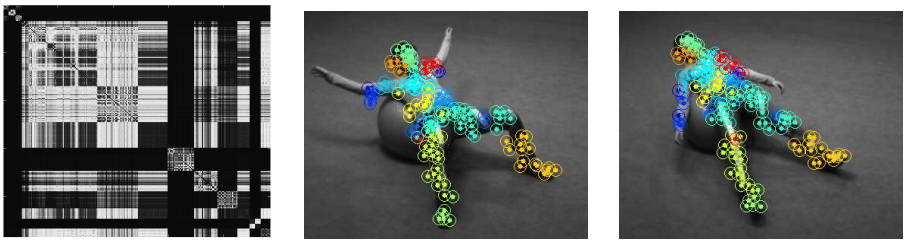


Figure 6.3: Sequence with two humans in the scene (1000 frames at 15fps). Different colors correspond to different labels. Despite the high correlation between body parts, they are correctly clustered.

The second example shows articulated motion of one person exercising on pilates ball. Fig. 6.5 illustrate examples of the clusters obtained with our method. Fig. 6.4 shows the corresponding similarity measure and the features labeled with colors by projecting the 3-D trajectories, on two instant frames of the sequence.



a) b) c)

Figure 6.4: Results of the clustering for a person. a) Similarity matrix. b) and c) Projections of the estimated 3-D trajectories at an instant frame on one of the cameras. Colors correspond to the motion segmentation results of analysing 308 trajectories during 35 frames at 15fps with 15% of missing data. 17 clusters and 66 outliers are found.

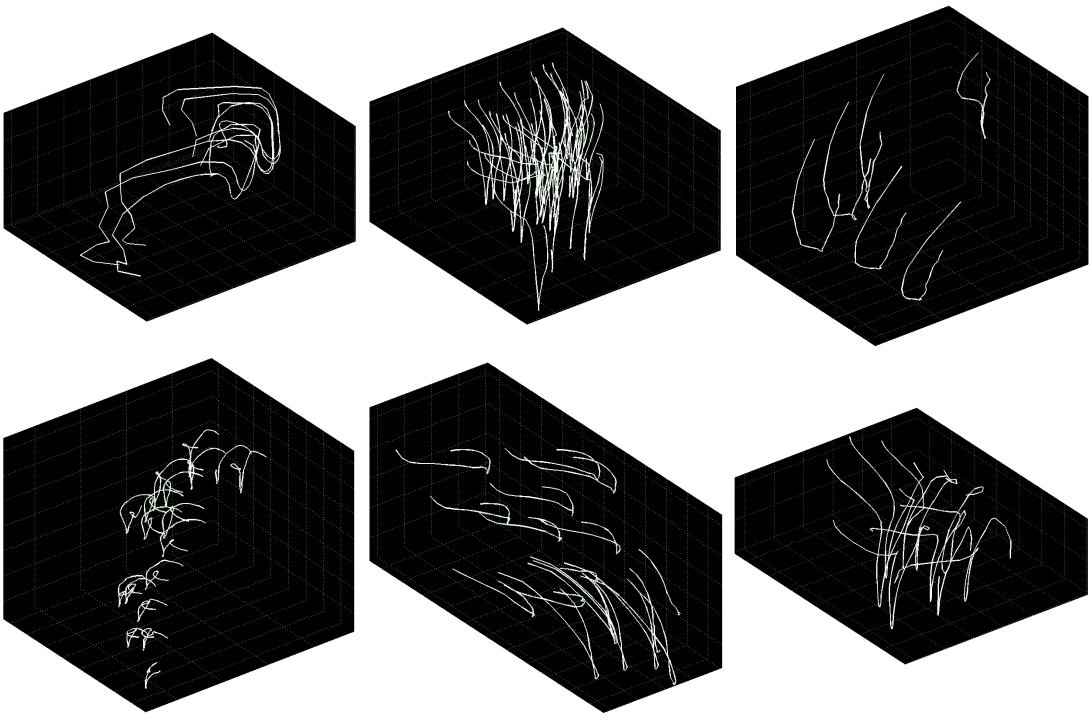


Figure 6.5: 6 of the 17 clusters obtained for the "ball" sequence.

Our results demonstrate the feasibility of using scene-flow trajectories for motion-segmenting of articulated objects, by defining a similarity that directly measures rigidity and using spectral

methods to cluster the trajectories. However, the performance of the algorithm depends on the tracking results. Although tracking in 3-D solves the ambiguities linked to tracking in 2-D images, the surface of the object being tracked needs to be textured to get reliable trajectories. Additionally, the tracking result is also depends on visibility, and may not recover enough features to recover parts involved in complex articulated motion affected for example by self-occlusions. In the following section we present a solution based only on geometry, thus suitable for objects that do not have enough texture, or for which the texture is simply not available.

6.3 Consistent shape segmentation over time

In this section we describe an unsupervised spectral approach to segment 3-D articulated bodies *consistently along time*. We exploit the geometric properties of the shape-graph embedding described in Chap. 3 and explicitly handle topological ambiguities occurring as the parts of the object deform or get in contact with each other over time (c.f. Fig. 3.2).

Our approach is based on three main ideas. First, we apply the principles of spectral clustering to segment the shape by clustering the eigenfunctions of the shape-graph. Since these eigenfunctions relate to the protrusions, they lead to a shape-dependent segmentation. Second, we provide an analysis on the way non-linear spectral embedding methods, such as Laplacian embedding and LLE, influence the shape of the point-set in the embedding space. In particular we notice that the orthogonality constraint imposed on the eigenfunctions, effectively reduces the dimension by representing the global topology of the graph with a reduced almost 1-D embedding. A convenient way to capture clusters in this space is therefore to look for collinear clusters. Third, instead of relying only on a frame-by-frame segmentation, we propagate the initial labels along the sequence by combining the stability of the embedding (result of its local-isometry preservation) and assuming temporal smoothness.

Recent attempts to extend nonlinear reduction to spatio-temporal data [Jenkins and Matarić, 2004, Lin et al., 2006] rely on enforcing temporal relationships when embedding time sequences. This is feasible for sparse sets such as the trajectories of tracked features presented in the previous section, but becomes computationally expensive when dealing with dense shape representations. We propose instead a mechanism to enforce temporal consistency of segments obtained by finding collinear clusters in the embedded space, which are expected to be stable under articulated motion, and propagating them over time.

We analyze local non-linear spectral embedding methods, such as the Laplacian embedding and LLE, in the context of clustering for unsupervised shape-segmentation. As discussed in Sec. 3.2.2.3 and Sec. 3.2.2.4, these methods find an embedding of a data-set by optimizing over the preservation of the local neighborhoods. As a consequence, shape protrusions as high-curvature regions of the surface are also preserved. Moreover, the orthogonality constraint acts as a force stretching the protrusions and making them wider separated and lower dimensional.

A novel scheme for unsupervised body-part segmentation along time sequences is thus proposed in which 3-D shapes are clustered after embedding. Clusters are propagated in time, and merged or split in an unsupervised fashion to accommodate changes of the body topology. Comparisons of our method, on synthetic and real data with ground truth, are performed both against direct segmentation in 3-D using EM, and against clustering in the embedding space obtained by a *geodesic-based* non-linear embedding method (Isomap). Robustness and the effects of topology transitions are discussed.

In the following section we analyze of the benefits of non-linear embedding methods for clustering articulated objects and we detail different algorithms that compose the temporally consistent shape-segmentation approach. Experiments in Sec. 6.3.3 assess the performance of our algorithm and other competing methods, on both synthetic and challenging real data, by comparison with ground truth labeling. In particular, we analyze the way the different methods cope with topology transitions, and study the robustness of our proposed algorithm.

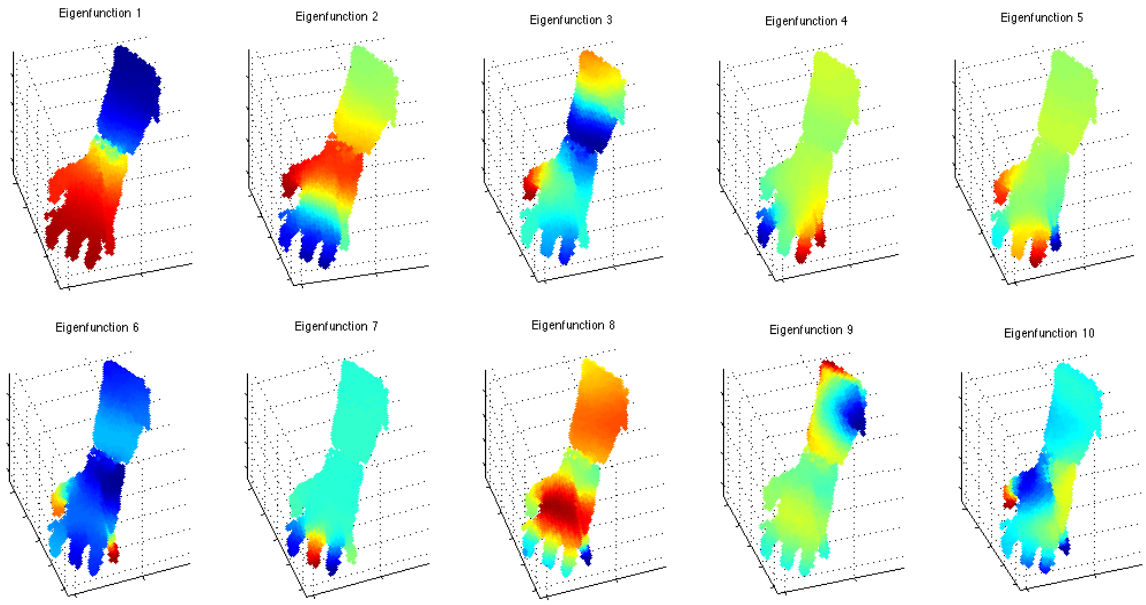


Figure 6.6: Laplacian eigenfunctions are associated with symmetries and protrusions of their domain. For each eigenfunction, the nodal-sets are represented as voxels with identical colors.

6.3.1 Analysis of local non-linear spectral embedding methods

According to the graph representation for shapes introduced in Sec. 3.2.1, the shape of an object can be represented by a graph $\mathcal{G} = (\mathcal{E}, \mathcal{V}, \mathcal{W})$, where vertices $\mathcal{V}(\mathcal{G})$ stand for samples $(\mathcal{X})$ on the surface or the volume of the object $(\mathcal{M}_{\mathcal{X}})$, and edges $\mathcal{E}(\mathcal{G})$ and weights $\mathcal{W}(\mathcal{G})$ describe the local geometry of the shape. As discussed in Sec. 2.3, the *Laplacian* can be seen as an operator on functions $\boldsymbol{f}$ defined on the $\mathcal{G}$. By analogy with the continuous eigenfunctions of the Laplace-Beltrami operator $\triangle_{\mathcal{M}}$, and under the required convergence conditions, the eigenfunctions of Laplacian $\mathcal{L}$ can be seen as stationary functions, forming a natural "base" for functions on the graph [Levy, 2006] (c.f. Sec. 2.3). The stationary property suggests that zero-level sets (*nodal-sets*) are closely related to protrusions and symmetries of the underlying manifold (Fig. 6.6). We explore now, three of the interesting geometric properties of the Laplacian and the LLE embedding methods.

6.3.1.1 Number of protrusions and local isometry

Gauss' "Theorema Egregium" states that the curvature of a surface can be determined entirely by measuring angles and distances on the surface. The "Gaussian curvature" or product $\kappa = \kappa_1 \kappa_2$ of the principal curvatures κ_1, κ_2 (the minimum and maximum curvature of all curves passing through the point) is invariant under local isometry. By definition, both Laplacian embedding and LLE optimize the neighborhood preservation by either enforcing connected nodes to be mapped to nearby locations (as in the case of the Laplacian embedding), or by directly preserving the affine reconstruction weights (as in LLE). When applying one of these algorithms to create an spectral

representation of a shape-graph, the weights are constructed on the basis of the geometry of the shape related to a distance either in the Euclidean space or on the surface of an object. It follows from the neighborhood preservation that distances are also preserved (up to a local scale). As protrusions in the original 3-D shape are obviously associated with high curvature regions of the delimiting surface (in a human body, think of feet or hands) they are also conserved after Laplace Embedding and LLE embedding (also by using Hessian eigenmaps, [D.Donoho and Grimes, 2003]).

6.3.1.2 The orthogonality and covariance constraints

The conservation of the number of protrusions is an effect of the objective function (which preserves local isometry) of the Laplacian and LLE embedding methods, other desirable features that help detecting and clustering shape protrusions are consequences of the orthogonality constraint imposed on the eigenfunctions.

As described in Sec. 3.2.2.3 and Sec. 3.2.2.4, the embedding function, $\Phi = \{\boldsymbol{f}_1, \cdots \boldsymbol{f}_N\}$, of the Laplacian and LLE methods are obtained by solving in each case a constrained optimization problem, respectively stated in Eq. 3.15 and Eq. 3.17. In both cases, the sought functions are constrained to be orthogonal which allows us to find the solution of the embedding as an eigenvalue problem. Apart from reducing the search space of possible embedding functions, the orthogonality constraint has an important effect on the shape of the point-sets in the embedding space ($\Phi(\mathcal{X}) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$). The orthogonality constraint enforces a unit covariance on the points in the embedding space[2]:

$$\frac{1}{N}\sum_{i=1}^{N}\Phi(\boldsymbol{X}_i)\Phi(\boldsymbol{X}_i)^\top = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^\top = \mathbf{I}_{N\times N}.$$

This outer product defines the covariance of the points $\chi = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ in the embedding space. The overall effect produced in the case of an articulated shape is a constraint that forces clusters (in this case limbs) to be separated into orthogonal eigenfunctions. However, since limbs are all attached to the core of the body and all the local neighborhods are enforced to be preserved, the result is a tradeoff between the objective function and the constraints,*i.e.* between the orthogonality and the neighborhood preservation, which gives rise to the line-shape of the limbs in the embedding space.

6.3.2 Unsupervised spectral segmentation

As mentioned before, our goal is to segment shape-graphs representing articulated bodies consistently over entire sequences, relying on the properties of the Laplacian and LLE embedding methods.

2. The orthogonality constraint $\boldsymbol{f}_i^\top \boldsymbol{f}_j = 0$ can be expressed in matrix form:

$$\frac{1}{N}\begin{bmatrix} \boldsymbol{f}_1^\top\boldsymbol{f}_1 & \boldsymbol{f}_1^\top\boldsymbol{f}_2 & \boldsymbol{f}_1^\top\boldsymbol{f}_3 & \cdots \\ \boldsymbol{f}_2^\top\boldsymbol{f}_1 & \boldsymbol{f}_2^\top\boldsymbol{f}_2 & \boldsymbol{f}_2^\top\boldsymbol{f}_3 & \cdots \\ \boldsymbol{f}_3^\top\boldsymbol{f}_1 & \boldsymbol{f}_3^\top\boldsymbol{f}_2 & \boldsymbol{f}_3^\top\boldsymbol{f}_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} \boldsymbol{f}_1(i)\boldsymbol{f}_1(i) & \boldsymbol{f}_1(i)\boldsymbol{f}_2(i) & \boldsymbol{f}_1(i)\boldsymbol{f}_3(i) & \cdots \\ \boldsymbol{f}_2(i)\boldsymbol{f}_1(i) & \boldsymbol{f}_2(i)\boldsymbol{f}_2(i) & \boldsymbol{f}_2(i)\boldsymbol{f}_3(i) & \cdots \\ \boldsymbol{f}_3(i)\boldsymbol{f}_1(i) & \boldsymbol{f}_3(i)\boldsymbol{f}_2(i) & \boldsymbol{f}_3(i)\boldsymbol{f}_3(i) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \mathbf{I}_{N\times N}.$$

where we have expressed the dot product explicitly as a sum over the N elements of $\boldsymbol{f}$, and taken the sum out of the matrix. The previous result is equivalent to:

$$\frac{1}{N}\sum_{i=1}^{N}[\boldsymbol{f}_1(i)\ldots\boldsymbol{f}(N)][\boldsymbol{f}_1(i)\ldots\boldsymbol{f}_N(i)]^\top = \frac{1}{N}\sum_{i=1}^{N}\Phi(\boldsymbol{X}_i)\Phi(\boldsymbol{X}_i)^\top = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^\top = \mathbf{I}_{N\times N}.$$

The result of the discussion in Sec. 6.3.1.1 and Sec. 6.3.1.2 is that after embedding, the protrusions of shape-graph are preserved in the embedding space, while their separation is increased and their intrinsic dimensionality reduced. Additionally, we know that different poses of the original articulated shape are mapped to similar point-sets in the embedding space (c.f. Chap. 4). Relying on these elements we propose an *unsupervised*, time-consistent, method for protrusion segmentation of 3-D shapes where no *a-priori* model (not even a weak topological, or graphical one, as in [Sundaresan and Chellappa, 2006]) is assumed, and the number of clusters themselves is inherently determined by the lower dimensional structure of the embedded points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$.

In order to take direct advantage of the low-dimensional (almost 1-D) shape of the protrusions in the embedding space, we replaces the standard k-means or ncuts algortihms in the embedding space by a *K-wise clustering* approach [Agarwal et al., 2005]. This method allows us to explicitly define high order (beyond pairwise) similarity measures between points.

Given a sequence of 3-D shapes the proposed approach for coherent-shape segmentation over time consists of the following steps. First, embedding the shape-graph using Laplacian embedding or LLE method. Then, the resultant point-set is clustered using the K clustering method in the embedding space. Next, the results of the segmentation are *propagated* along time by extracting the centroids of the clusters at the current frame, and using them as seeds for the segmentation in the next frame. Finally, to ensure that changes of topology in the shape-graph are handled, the number of clusters estimated in an automatic way and the seed-propagation is controlled by merge/and split operations. The overall approach is illustrated in Fig. 6.7. In the following sections we describe in detail the different stages of the proposed method.

6.3.2.1 K-wise clustering in the embedding space

After embedding the shape-graph, the protrusions are well approximated by lines in the embedding space. In order to segment them accordingly, we require an algorithm that explicitly seeks for clusters formed by sets of roughly collinear points. Therefore, we start by defining a similarity function that measures collinearity. An appropriate measure is given by the area of the triangle defined by *triads* of points (as shown in Fig. 6.8). In higher dimensions, an equivalent measure is the volume of the polyhedron defined by the convex-hull sub-set of 3 or more points. As opposed to pairwise similarities, these measures are defined over triads or larger sets of points, so pairwise-based clustering methods can not be directly applied. To get around the problem we first model the similarities with a hyper-graph, and then, approximate the hyper-graph with an ordinary graph where classical methods are applied.

High-order relationships between points can be modeled with a hyper-graph $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H}, \mathcal{W}_\mathcal{H})$. In our setting, each vertex of the hyper-graph $\mathcal{V}_\mathcal{H}(\mathcal{H})$ represents one of the embedded points in the set $\chi = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, while hyper-edges $\mathcal{E}_\mathcal{H}(\mathcal{H})$ connect K-tuples of nodes (sub-sets of K nodes). Each hyper-edge $\boldsymbol{z} \in \mathcal{E}_\mathcal{H}(\mathcal{H})$ is defined by a K-tuple $\boldsymbol{z} = \{z_1, \ldots z_K\}$, where $\boldsymbol{z} \in (\mathcal{V}_\mathcal{H})^K$. The elements z_j correspond to the indices of the vertices linked by the hyperedge ($z_j \in \mathcal{N}$ and $1 \leq z_j \leq N$). Finally, the weights assigned to each hyper-edge $\mathcal{W}_\mathcal{H}(\mathcal{H})$ are obtained with the similarity function $h(\boldsymbol{z})$ defined over the hyper-edges ($h : \boldsymbol{z} \mapsto I\!R^+$).

We use the *clique averaging* method proposed in [Agarwal et al., 2005], to approximate the hyper-graph $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H}, \mathcal{W}_\mathcal{H})$ with a an ordinary graph $\widetilde{\mathcal{H}} = (\mathcal{V}_{\widetilde{\mathcal{H}}}, \mathcal{E}_{\widetilde{\mathcal{H}}}, \mathcal{W}_{\widetilde{\mathcal{H}}})$ with pairwise edges. Only hyper-edges need to be approximated, thus $\mathcal{V}_{\widetilde{\mathcal{H}}} = \mathcal{V}_\mathcal{H}$. To create the connectivity in the approximate graph, one edge is created between every two nodes originally linked by a hyper-edge in $\boldsymbol{z} \in \mathcal{E}_\mathcal{H}$. The weights on the pairwise edges are obtained by solving an optimization problem,

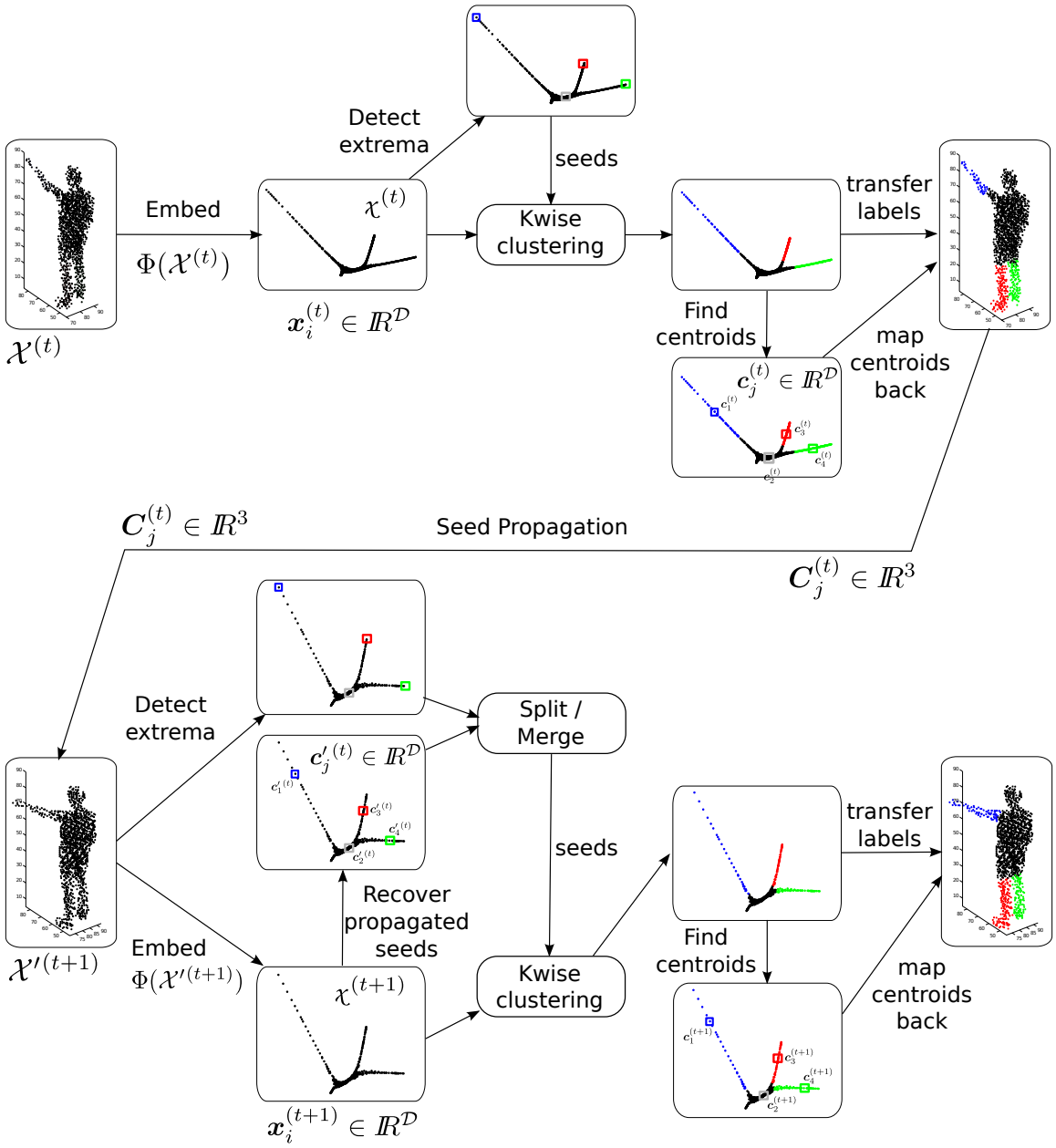


Figure 6.7: Overall description of the temporal consistent segmentation algorithm for two time steps. Top: Initialization. Bottom: Normal iteration of the algorithm at at time $t > 0$.

under the assumption that the weight of a hyper-edge can be computed as the arithmetic mean

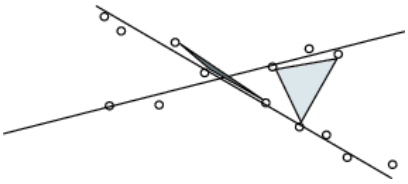


Figure 6.8: Measuring the similarities as the collinearity between triads of points. The area of the triangle defined by three points is an indicator of how well the points are approximated by a line. When the points are close to collinear the area tends to zero.

of the edge weights, *i.e.* ideally $\mathcal{W}_{\mathcal{H}}(\boldsymbol{z}) = \lambda \sum_{i,j \in \boldsymbol{z}} \mathcal{W}_{\widetilde{\mathcal{H}}}(i,j)$, with λ a constant. The resultant constrained least squares optimization can be solved with an efficient iterative method [3].

Finally, the approximated graph $\widetilde{\mathcal{H}}$ is clustered using an standard spectral clustering, with k-means as a final step [Ng et al., 2002]. As a result, the K-wise algorithm yields a segmentation in the embedding space that can be mapped back to the original 3-D space (Fig. 6.10) using the one-to-one trivial correspondence between the indices of the initial $\mathcal{X}$ and embedded χ sets. The steps of the procedure are summarized in Algorithm 7.

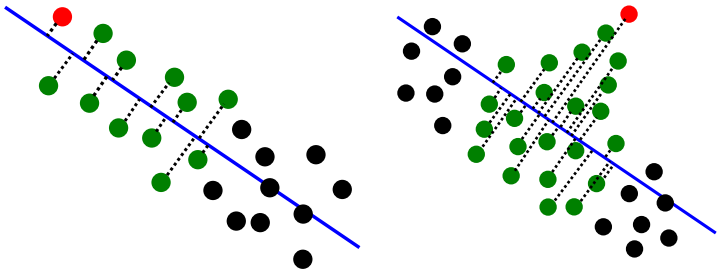
Algorithm 7 K-wise clustering in embedding space.

1. Build the affinity hyper-graph $\mathcal{H} = (\mathcal{V}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}}, \mathcal{W}_{\mathcal{H}})$:
   - Use the vertex of the hypergraph $\mathcal{V}_{\mathcal{H}}(\mathcal{H})$ to represent the embedded point-set $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$
   - Form hyper-edges with $\mathcal{D}$ elements in case of a $\mathcal{D}$-dimensional embedding space.
   - Use the volume of the convex-hull defined by each $\mathcal{D}$-tuple as the high-order similarity measure.
2. Apply K-wise clustering [Agarwal et al., 2005]:
   - *Clique averaging*: approximate $\mathcal{H}$ with $\widetilde{\mathcal{H}}$ by solving the constrained least squares optimization problem that enforces the weight of each hyper-edge to be the arithmetic mean of the weights of the edges incident on it: $\mathcal{W}_{\mathcal{H}}(z) = \sum_{i,j \in z} \mathcal{W}(i,j)$.
   - Do clustering on the approximated graph $\widetilde{\mathcal{H}}$ using an standard spectral method [Ng et al., 2002].

6.3.2.2 Boundary detection and number of clusters.

Algorithm 7 yields a shape-dependent segmentation of the embedded point-set χ but requires as input the number of sought clusters n. Given the embedding properties discussed in Sec. 6.3.1, especially the preservation of the protrusions of the shape, one can find the "correct" number of clusters by detecting *protrusion boundaries* [Rosman et al., 2004]. We devise a simple algorithm to detect the extrema of protrusions in the embedding space. We then use the total number of detected extrema plus one as the number of clusters (denoted by n). The additional cluster will serve to separate the core of the articulated object.

3. We use the `lsqlin` solver in MATLAB Optimization Toolbox, as suggested by [Agarwal et al., 2005]



a) Extrema detected. b) Extrema not detected.

Figure 6.9: Conditions for detecting protrusion extrema in the embedding space. In both figures the current point being analyzed is in red and its neighbors in green. The line that best fits the neighbors is drawn in blue. The neighbors are projected on the line. a) If the projection of the current point is one of the two outermost points on the line, the point is detected as a protrusion extrema. b) the algorithm naturally discards small protrusions.

The extrema detection algorithm is illustrated in Fig. 6.9. Each point in χ is tested to decide whether or not it is an extrema in three steps. First, we find the nearest neighbors (plotted in green on Fig. 6.9) within a threshold distance. Then, the neighbors are projected on to the line that best fits the neighbors(in blue). An embedded point is a protrusion termination iff all its neighbors, projected on this line, lay on one side of the point's own projection (Fig. 6.9-a). A protrusion termination is not detected when the projection has neighbors on both sides (Fig. 6.9-b).

6.3.2.3 Temporal consistency and seed propagation

In the previous sections we have discussed how to obtain a shape-dependent segmentation of an object. However, our main interest is to ensure the consistency of such segmentation over time, along a sequence of an articulated object in motion. Temporal consistency implies that shape-graphs are decomposed into similar segments at every time frame, *i.e.* segments that move coherently and preserve their shape. Here, we propose a method for temporal consistency based on the temporal smoothness assumption and a merging splitting strategy to handle topological changes.

The algorithm is based on the propagation of the cluster centroids along the sequence and cluster centroids $\boldsymbol{c}^{(t)}$ at time t are used to generate initial seeds for clustering at time $t+1$. The propagation is achieved by first, mapping the centroids back to the 3-D space where the samples of the shape live to obtain a set $\boldsymbol{C}^{(t)}$. At time $t+1$ the 3-D centroids from the previous segmentation $\boldsymbol{C}^{(t)}$ are embedded together with the new sample-set $\mathcal{X}(t+1)$. In the new embedding space, the mapped centroids $\boldsymbol{c'}^{(t)}$ are used as seeds for the new clustering step. The propagation is illustrated in Fig. 6.10 and detailed in Algorithm 8.

6.3.2.4 Topology changes and merging/splitting

Local non-linear embedding methods such as the Laplacian embedding and LLE methods are less sensitive to topological changes of the object in motion than *global* geodesic-based embedding methods like Isomap (Fig. 6.17-a-3), given that computations are only based on local neighborhoods.

Nevertheless, topological changes still have an effect on the embedded point-set. In fact, in an unsupervised context where no prior knowledge on the body structure is available, it is not possible

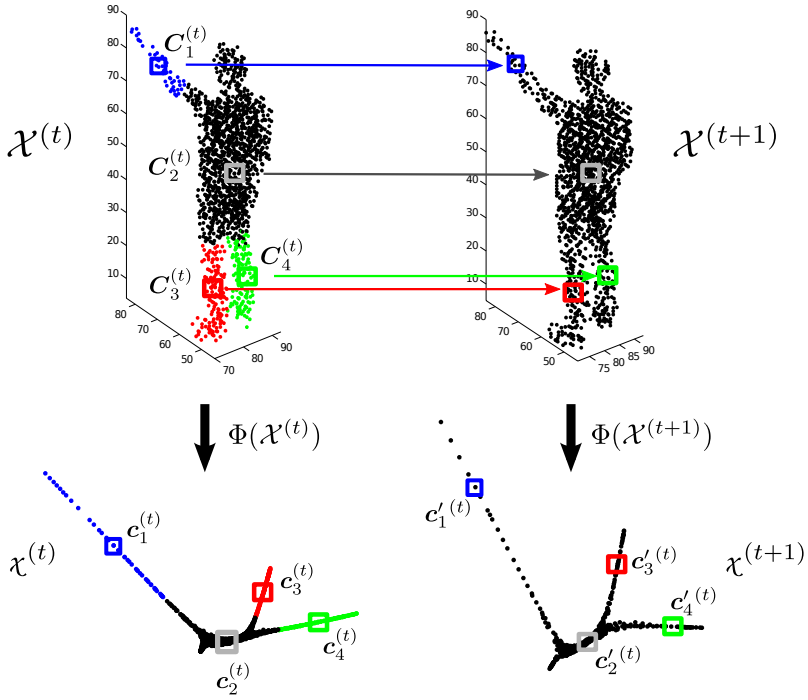


Figure 6.10: Seed propagation for time-consistent clustering in the embedding space. The centroids $\boldsymbol{c}^{(t)}$ at time t are mapped back to obtain the corresponding 3-D coordinates $\boldsymbol{C}^{(t)}$ and added to the 3-D sample-set $\mathcal{X}^{(t+1)}$ at time $t+1$. The resultant embedded centroids $\boldsymbol{c}'^{(t)}$ serve as seeds for clustering the new embedded point-set $\chi^{(t)}$

to distinguish the self-contacts from ordinary cases. Hence, there is no reason to separate adjacent (touching) parts. Instead, it is more sensible to fit the number of clusters to the topology as it changes along time. Therefore, we use he extrema detection algorithm, not only for initialization but also as a tool for implementing the necessary cluster updates, both in number and location, when such a change occurs. Algorithm 9 details the procedure.

To summarize the overall approach illustrated in Fig. 6.7 is described in Algorithm 10.

Algorithm 8 Temporal consistency and seed propagation algorithm

1. Segment the embedded point-set $\chi^{(t)} = \{\boldsymbol{x}_1^{(t)}, \ldots, \boldsymbol{x}_{N^{(t)}}^{(t)}\}$ at time t using K-wise clustering (c.f. Algorithm 7) with $K = D$.
2. Find the cluster centers $\boldsymbol{c}_j^{(t)}$, with $1 \leq j \leq n^{(t)}$, in the embedding space.
3. For each $\boldsymbol{c}_j^{(t)}$, search for the closest nearest neighbor $\overline{\boldsymbol{c}}_j^{(t)}$ belonging to the embedded point-set $\chi^{(t)}$.
4. Map $\overline{\boldsymbol{c}}_j^{(t)}$ back to the original shape to recover its corresponding 3-D coordinates $\boldsymbol{C}_j^{(t)}$.
3. At time $t+1$, augment the sample-set $\mathcal{X}^{(t+1)}$ with the 3-D centroids from time t, yielding the compound data-set:
$$\mathcal{X}'^{(t+1)} = \mathcal{X}^{(t+1)} \cup \{\boldsymbol{C}_j^{(t)}\}, \text{with } j = 1, ..., n^{(t)};$$
4. Embed $\mathcal{X}'(t+1)$ using Laplacian embedding or LLE to obtain:
$$\chi'^{(t+1)} = \chi(t+1) \cup \{\boldsymbol{c}_j'^{(t)}\}, \text{with } j = 1, ..., n^{(t)} ;$$
5. Use the propagated centroids $\boldsymbol{c}_j'^{(t)}$ to initialize the clustering of the new embedded point-set $\chi^{(t+1)}$.

Algorithm 9 Merge and split algorithm

1. At each time instant t detect all extrema of the embedded point-set $\chi^{(t)}$.
   - if $t = 0$ the detected branch terminations are used as seeds for k-wise clustering.
   - Otherwise ($t > 0$), standard k-means is performed on $\chi^{(t)}$ using extrema as seeds, yielding a rough partition of the embedded point-set into distinct branches.
2. Propagated seeds $\boldsymbol{c}_j'^{(t)}$ in the same rough partition are merged (when previously separated body-parts get too close to be distinguished).
3. For each partition of $\chi^{(t)}$ not containing any old seed a new seed is defined as the related branch termination (when previously indistinguishable body-part becomes well separated).

Algorithm 10 Unsupervised Temporal Coherent Shape Segmentation (TLLE)

At each instant t process the current data-set $\mathcal{X}^{(t)} = \{\boldsymbol{X}_i^{(t)}, \ldots, \boldsymbol{X}_{N^{(t)}}^{(t)}\}$ as follows:

1. Map $\mathcal{X}^{(t)}$ or $\mathcal{X}'^{(t)}$ to an embedding space of dimension $\mathcal{D}$ using the Laplacian embedding or the LLE algorithms:

$$\begin{cases} \text{embed } \mathcal{X}^{(t)} & \text{if } t = 0, \\ \text{embed } \mathcal{X}'^{(t)} = \mathcal{X}^{(t)} \cup \{\boldsymbol{c}_j^{(t-1)}\}, \text{ with } 1 \leq j \leq n^{(t-1)} & \text{if } t > 0. \end{cases}$$

   The embedding yields:

$$\begin{cases} \chi^{(t)} = \Phi(\mathcal{X}^{(t)}) = \{\boldsymbol{x}_i^{(t)}, \ldots, \boldsymbol{x}_{N^{(t)}}^{(t)}\} & \text{if } t = 0, \\ \chi^{(t)} = \Phi(\mathcal{X}'^{(t)}) = \{\boldsymbol{x}_i^{(t)}, \ldots, \boldsymbol{x}_{N^{(t)}}^{(t)}\} \cup \{\boldsymbol{c}_j'^{(t-1)}\} & \text{if } t > 0. \end{cases}$$

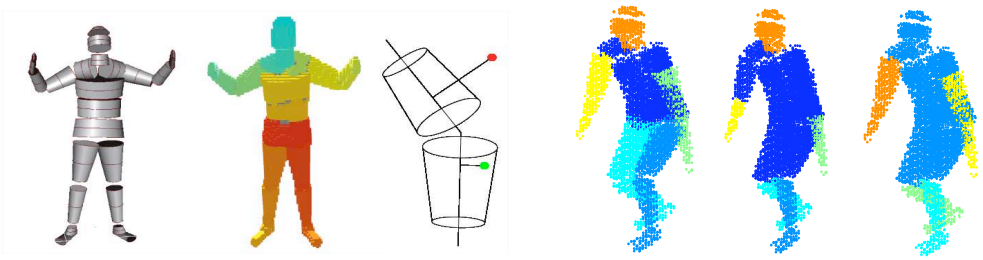
2. Detect all branch extrema of $\chi^{(t)}$: the natural number of clusters $n^{(t)}$ for time t is then set to the number of line-shaped clusters plus one for the core of the object.
3. Cluster the embedded points $\chi^{(t)}$ into $n^{(t)}$ groups by $\mathcal{D}$-wise clustering starting from $n^{(t)}$ seeds:

$$\begin{cases} \text{-Use all extrema as seeds} & \text{if } t = 0, \\ \text{-Derive seeds by split/merge from centroids } \{\boldsymbol{c}'^{(t)}\} \text{ and extrema Algorithm 9} & \text{if } t > 0. \end{cases}$$

4. Find the new set of centroids $\{\boldsymbol{c}_j^{(t)}\}$, with $1 \leq j \leq n^{(t)}$;
5. The labeling of the embedded points induces a segmentation in the original 3-D shape
6. All cluster centroids $\{\boldsymbol{c}_j^{(t)}\}$ are re-mapped to 3-D, the corresponding 3-D centroids $\{\boldsymbol{C}_j^{(t)}\}$ are added to the new data-set $\mathcal{X}(t+1)$ at time $t+1$.

6.3.3 Experiments

We tested our proposed methodology, on synthetic and real data, in order to obtain both qualitative and quantitative assessments of its performance. In the case of synthetic sequences, we used as ground truth the labels automatically generated in a virtual setup (Fig. 6.11-b). An articulated model and motion capture data were used to generate voxel sequences with ground truth.



a)GT model b)GT labels c)GT voxels d)GT seg1 e)GT seg2 f)GT seg3

Figure 6.11: a) Synthetic model to generate labels. b) A different label is asigned to each part; here labels are illustrated with different colors. c) To generate the ground truth, voxels are labeled according to their proximity to the part of the model. d,e,f) Three different ground truth segmentations.

For sequences where *ground truth* is available, we propose three performance indicators:

Coarsening: We consider that a given cluster labeling is valid when it does not oversegment any of the rigid parts that compose the articulated object. The *coarsening score* compares how close the clusters resemble the (rigid) segments of a ground-truth model in this sense. To each segment of the ground-truth model (Fig. 6.11-b), we associate the (unsupervised) cluster that best represents the segment, *i.e.* the cluster containing the largest set of points in common. The difference is evaluated as the percentage of points that belong to the ground truth cluster but were not included in the cluster representing the segment (false-negatives). If all clusters correspond to the ground-truth segments, the percentage is 0, and the coarsening score is 1.

Segmentation: The *segmentation* score compares the unsupervised clusters with respect to three different "natural" *a priori* segmentations of the body. The score is similar to the coarsening one, but taking the labels from the segments in (Fig. 6.11-d,e,f).

Time consistency: Taking as reference the segmentation at time $t = 0$ for each body part, the *time-consistency* score evaluates the drift of the segmentation along time. For each $t > 0$, and for each cluster, we measure the similarity between the current label distribution and the initial one. The consistency score tends to one when the similarity measure is constant in time.

For *real-sequences*, with no ground truth available, we perform comparisons with two methods that also propagate seeds over time to ensure time consistency. In the first comparison method, clustering is performed in 3-D on the original data-set (*e.g.* voxel-set) using EM. We denote this algorithm Dynamical EM (DEM). In the second method for comparison, clustering is done with k-means in the embedding generated by Isomap [J. B. Tenenbaum and Langford, 2000] instead of the Laplacian embedding or LLE.

In the DEM approach, the probability density $p(\mathcal{X})$ of the sample-set $\mathcal{X}$ is modeled with a mixture of Gaussians, where each Gaussian component $p_i(\mathcal{X}) \sim \mathcal{N}(\mu_j, \Sigma_j)$ is treated as a cluster

(c.f. Chap. 5):

$$p(\chi) = \sum_{j}^{M} \pi_j p_j(\chi), \qquad \sum_{j} \pi_j = 1. \tag{6.4}$$

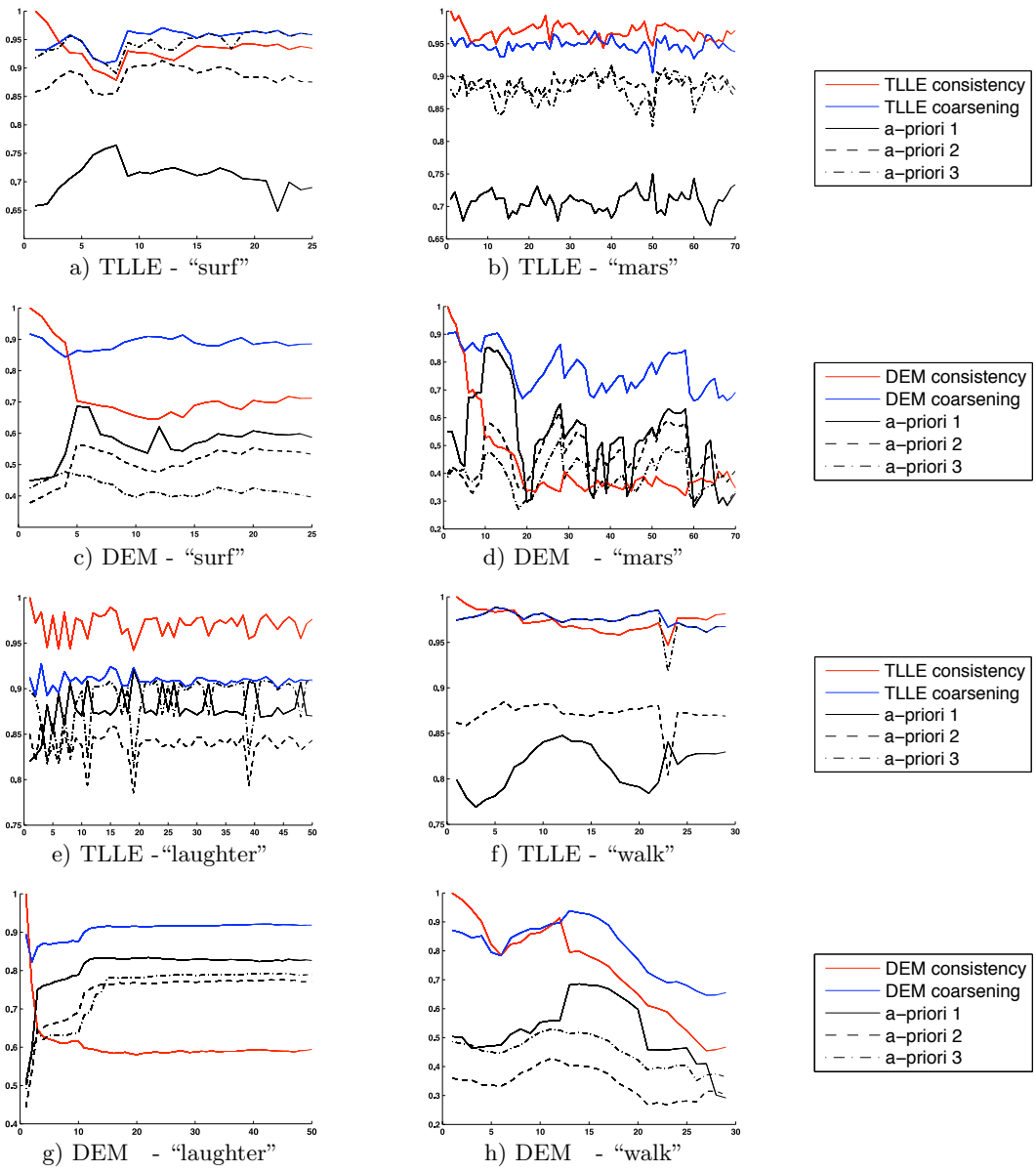
The parameters of the Gaussians and the mixing weights are estimated through the EM algorithm [Dempster et al., 1977], similar to Algorithm 5.

6.3.3.1 Synthetic data

(Fig. 6.12-c,d,g,h) We perform a set of experiments on several synthetic voxel-set sequences generated by simulating human body motion with a kinematic model and motion-capture data [4]. We use LLE for embedding, and thus refer to our method as the temporal LLE (TLLE) clustering. Additionally, we perform the segmentation using the Dynamic EM method. We compute the coarsening, segmentation and temporal consistency scores in both cases. Fig. 6.12 shows graphs comparing the two method for 4 different synthetic sequences.

Our unsupervised segmentation approach TLLE (Fig. 6.12-a,b,e,f) achieves a very good temporal consistency, as witnessed by the consistency score (red curves) between 95 and 100% at all times, even for fairly long sequences. In all cases, the boundary between clusters normally lies in correspondence to rigid segments as demonstrated by the coarsening score (blue curves). The *a priori* segmentation scores (different black curves) seem to favor Fig. 6.11-e partition, i.e. it tends to detect the outermost rigid links instead of entire protrusions (e.g. forearms and tibias instead of arms and legs). This result is explained by the low density (gaps) between the articulations of the synthetic model. The results for the dynamic EM clustering (Fig. 6.12-c,d,g,h) show that the absolute segmentation performance (black curves) is consistently and considerably worse than that of TLLE. Furthermore, the obtained temporal consistency is very low (red curves), indicating that the drifting of the clusters inside the shape along with the motion. As a result, the clusters obtained with DEM usually span different distinct body-parts. This phenomenon is clear in Fig. 6.13-b-2, where the irregular trajectories of the obtained EM clusters for a sub-sequence of "walk" are visually rendered.

4. The actual motion-capture sequences where obtained by applying a model-based tracking method to multiple-view video sequences, as described in [Knossow et al., 2008]



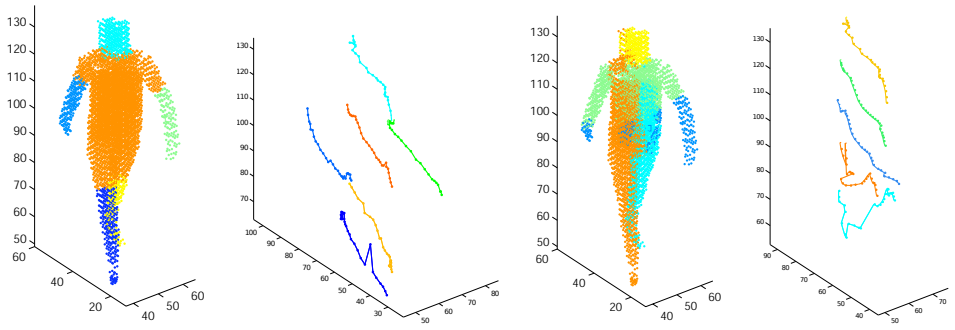
a) TLLE - "surf" b) TLLE - "mars"

c) DEM - "surf" d) DEM - "mars"

e) TLLE -"laughter" f) TLLE - "walk"

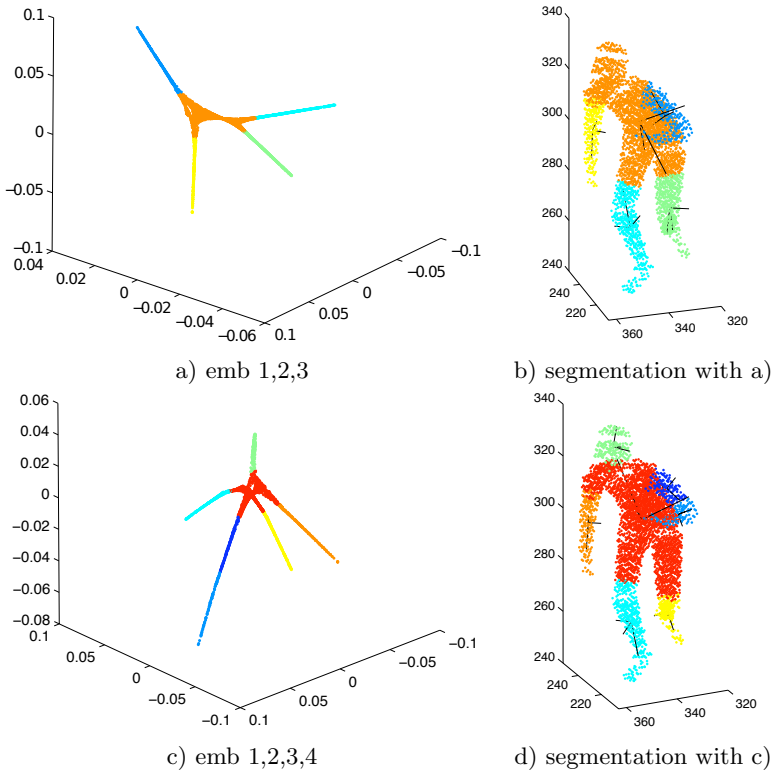
g) DEM - "laughter" h) DEM - "walk"

Figure 6.12: Segmentation scores both for our segmentation algorithm (TLLE a,b,e,f) and for the dynamic EM clustering (DEM c,d,g,h), in comparison with ground truth provided for a number of synthetic sequences ("surf" , "mars" , "laughter" , "walk") with different length, from 25 to 70 frames. Red curves plot the consistency score, blue ones, the coarsening score. Solid, dashdot and dashed lines represent the scores respectively associated with the three *a priori* segmentations depicted on Fig. 6.11-d,e,f.



a) TLLE - seg. b) TLLE - traj. c) DEM - seg. d) DEM - traj.

Figure 6.13: Segmentation results and centroid trajectories for (a,b) our algorithm and (c,d) EM clustering ("walk").



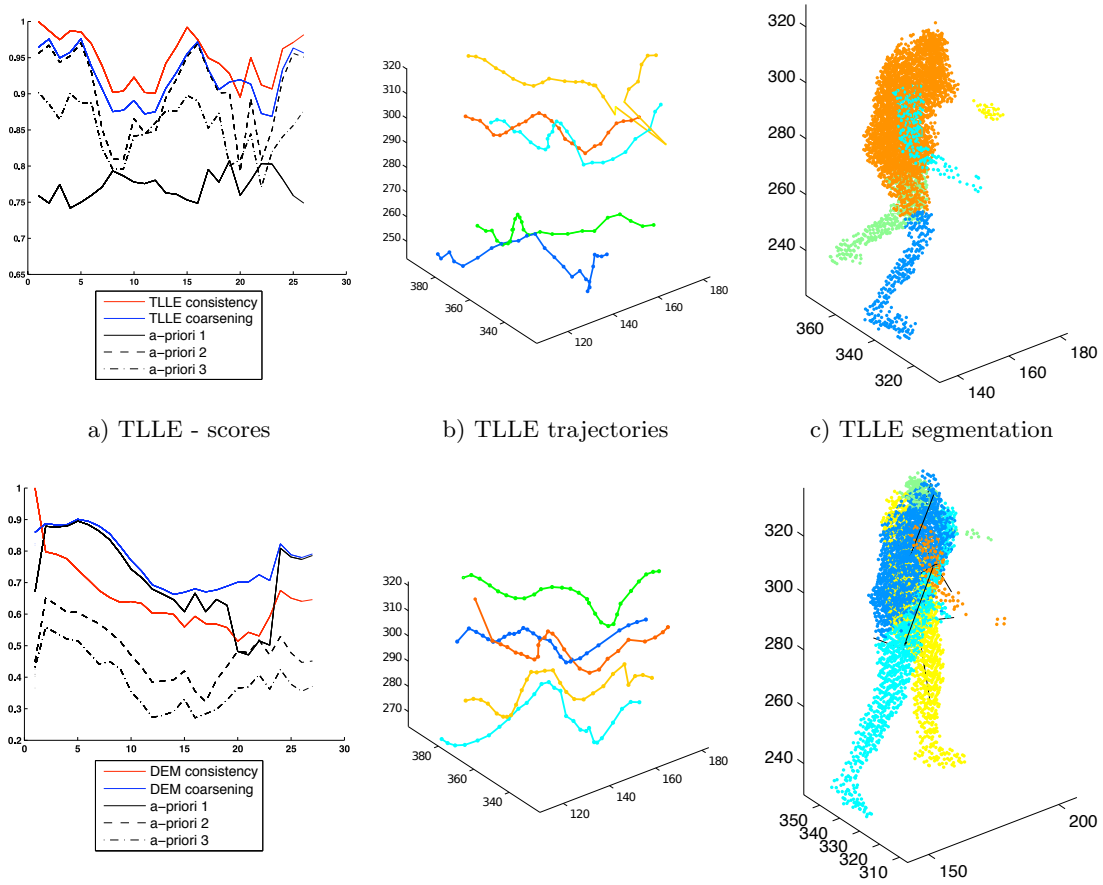
a) emb 1,2,3 b) segmentation with a)

c) emb 1,2,3,4 d) segmentation with c)

Figure 6.14: Different eigenfunctions capture different aspects of the shape geometry. a), b) Segmented embedded point-set and corresponding segmentation in 3-D obtained by selecting eigenvectors 1, 2, and 3. c), d) Corresponding results for eigenvectors 2, 3, and 4.

6.3.3.2 Real data

We also measured performances on a large number of real-world, high-resolution voxel-set sequences generated from images captured with our multiple-camera acquisition system. For sequences for which motion-capture data is available, we quantitatively measure the performance of all competing methods.



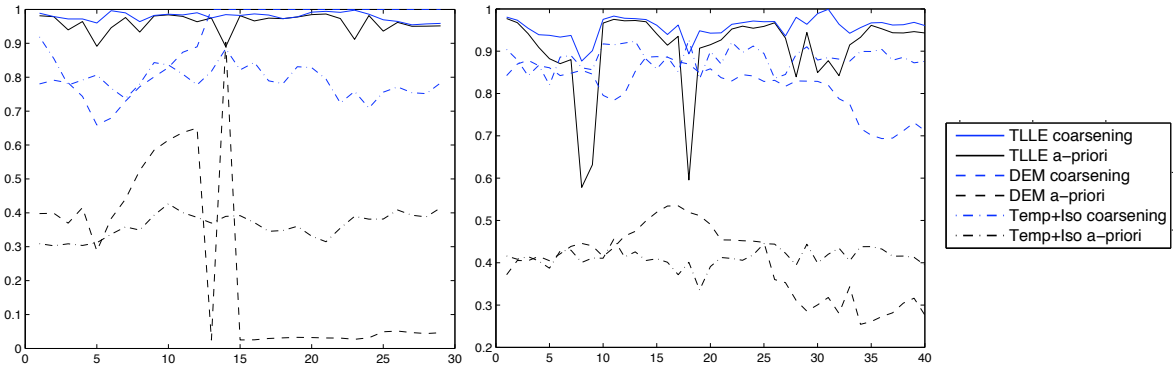
a) TLLE - scores b) TLLE trajectories c) TLLE segmentation

d) DEM scores e) DEM trajectories f) DEM segmentation

Figure 6.15: Segmentation scores (a,d) and centroid trajectories (b,e) obtained for the real sequence "Mars", from $t = 1$ to $t = 27$. Top row show our results. Bottom row shows the dynamic EM clustering results. The segmentation corresponding to the critical frame $t = 20$ of the sequence is shown (c,f).

Voxel-set sequences captured through an acquisition system suffer from noisy and/or missing data (either in the form of holes, or as disconnected components). As a consequence, sections or entire parts of the object are missing during certain periods of time (see Fig. 6.15-c,f). Fig. 6.15 illustrates how, unlike EM and for appropriate values of the parameters (we used $\mathcal{D} = 4, k = 25$ for this experiment), scores are still very high, showing the resilience of the method to unreliable data capture. Drops in the scores (mirrored by a brief sudden glitch in clusters trajectories, Fig. 6.15-b,e)

correspond to frames in which entire limbs are missing from the reconstruction. Fig. 6.16 shows the scores obtained by all competing methods over two additional challenging sequences of real voxel-sets. Our method exhibits robustness to data of very poor quality, easily outperforming at the same time both DEM or clustering in a geodesic-based embedding space. At times where glitches due to extremely corrupted data occur, ($t = 8, t = 17$, right) the topology adaptation algorithm brings back the segmentation on track.



a) "erwan" b) "space-surf'

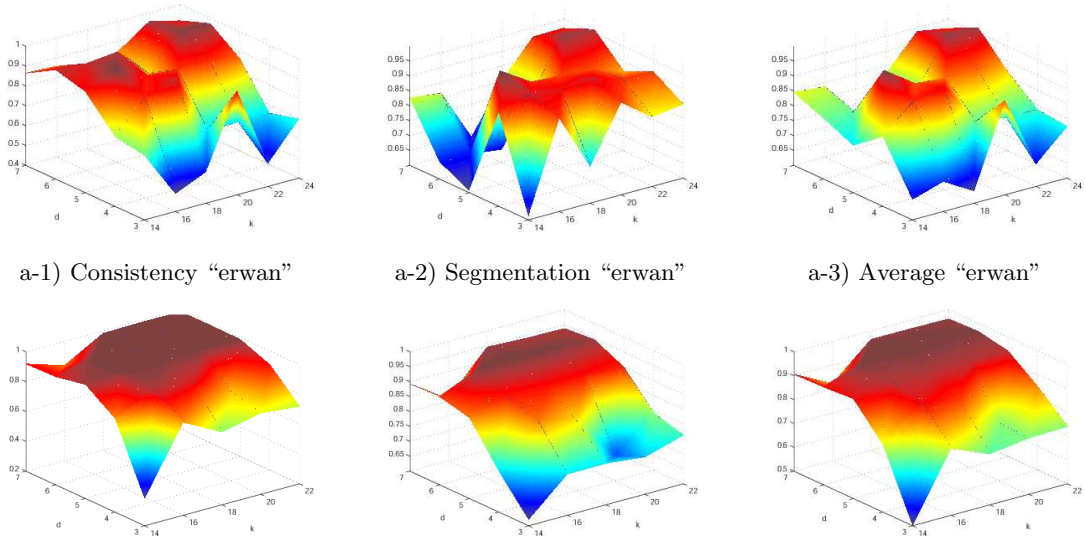
Figure 6.16: Segmentation scores for two real sequences, and for all competing methods. Only the coarsening (blue) and a-priori (segmentation depicted on Fig. 6.11-e) scores are plotted. Solid lines: our method; dashed lines: dynamic EM. Dashdot lines: cluster propagation in Isomap space.

6.3.3.3 Robustness with respect to the parameters

The proposed segmentation methodology relies on the properties of the local non-linear spectral embedding methods (c.f. Section Sec. 6.3.1), namely the Laplacian eigenmaps and LLE methods. These algorithms depend on two parameters: the size of the neighborhood (determined by k or ϵ) and the dimension $\mathcal{D}$ of the embedding space. In order to choose the optimal size of the neighborhood we use the heuristic presented in Sec. 3.2.1.2 (c.f. Fig. 3.5).

As illustrated in Fig. 6.6, peaks on the eigenfunctions are associated with protrusions on the shape, and extrema in the embedding space. Therefore, the embedding dimension $\mathcal{D}$ has a direct impact on the number of clusters n detected by our extrema-detection algorithm. For example, in Fig. 6.14-b,c, eigenfunctions 1,2,3 determine an embedding where the algorithm is unable to segment the head, which is correctly clustered instead when selecting eigenfunctions 2,3,4 (Fig. 6.14-d,e). Evidently, we do not know the number of protrusions in advance, and selecting a big $\mathcal{D}$ leads to higher-dimensional point-sets, which in turn affect the computational performance of the algorithm (notably of the k-wise algorithm). Fortunately, there is usually a reasonable range for the parameters, in which the performance of the algorithm can be expected to be good.

It is important to assess the sensitivity of the algorithm to the main parameters k and $\mathcal{D}$ (or the list of indices of the selected eigenvectors). Fig. 6.17 quantitatively illustrates how the segmentation scores vary when different values of the parameters $k, \mathcal{D}$ are used to compute the embedding (assuming for sake of simplicity that we select the first $\mathcal{D}$ eigenfunctions). The stability of both consistency in time and quality of the segmentation in a large region of the parameter space



a-1) Consistency "erwan" a-2) Segmentation "erwan" a-3) Average "erwan"

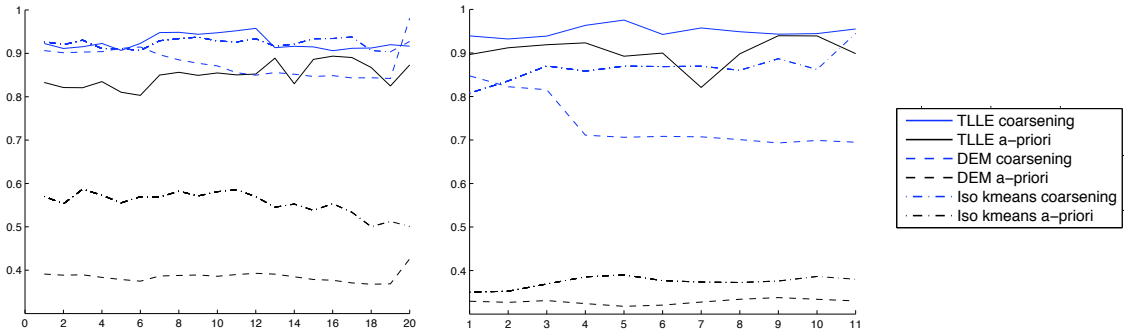
b-1) Consistency "space-surf' b-2) Segmentation "space-surf' b-3) Average "space-surf'

Figure 6.17: Consistency (a-1,b-1), segmentation (a-2,b-2) and the average of consistency and segmentation (a-3,b-3) scores obtained over two example synthetic sequences ("erwan" (a) and "space-surf' (b)) for different values of the parameters $k = 14 : 2 : 22$ (on the abscissa) and $\mathcal{D} = 3 : 7$ (on the ordinate) of the algorithm. The best performances are achieved for a wide range of the parameters.

speaks of the robustness of the approach.

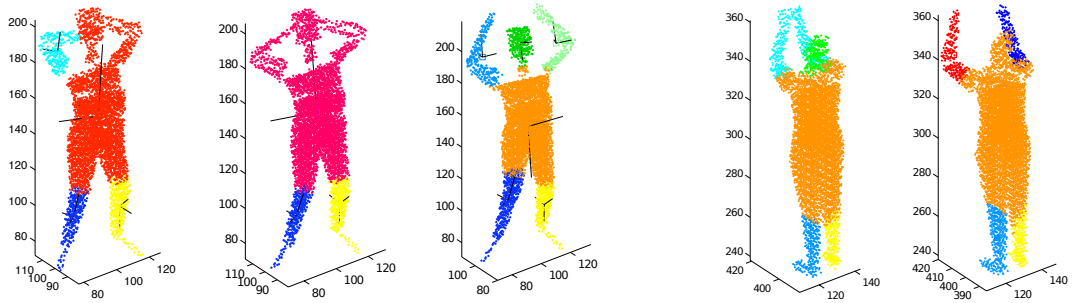
6.3.3.4 Robustness to topology changes.

Despite the stability of the Laplacian and LLE methods to topological changes, instants in which di?erent parts of the articulated body come to contact still a?ect the shape of the embedded point-set χ. However, these events have even more dramatic consequences on embedding methods based on measuring global pairwise geodesic distances, since self-contacts create new paths affecting the distance between all pairs of points in the original sample-set (c.f. Fig. 3.2). Fig. 6.18 compares the segmentation scores of methods based on local and global distances in situations in which the topology of the body changes. Propagating clusters in the the embedding space generated by LLE exhibits superior results and robustness, as the algorithm smoothly adapts to topology changes owing to the properties of the embedded point-set. Examples of the transitions are shown in Fig. 6.19. In the first sequence ("wake up" Fig. 6.19-a), a new cluster is created when the two arms touch the head. When they separate again from the body three new clusters (one for the body and one for each arm) reflect the new topology of the graph. Similar in the second example ("clap" Fig. 6.19-a), when arms come together and separate.



a) "walk" b) "clap"

Figure 6.18: Measuring the relative performances of TLLE, DEM, and embedding methods based on global geodesic distances geodesics (represented by Isomap) for sequences affected by topology changes. Propagated clusters over time are rendered with the same color.



a-1) a-2) a-3) b-1) b-2)

Figure 6.19: Examples of how our segmentation algorithm copes with topology transitions(**a**-2,3,4-"wake up" , **b**-1,2-"clap").

6.4 Conclusions

In this chapter we have presented two methods that address different aspects of the segmentation of articulated shapes. The first performs motion segmentation of sparse scene-flow trajectories. The second finds consistent clusters over time by relying on shape (geometric) cues.

The motion segmentation, relies first on appearance for extracting an sparse representation of the objects in the scene in terms of features and their motion in 3-D (as it will be described in the next chapter). Thanks to this sparse representation we are able to segment groups of trajectories according to their motion by defining a similarity that measures the rigidness between pairs of trajectories and using a standard spectral clustering method. Unlike spatial, speed, or SIM related measures, our definition of similarity is directly defined in 3-D which allows it to handles rotations. We show the advantages of using multiple views to solve the ambiguities related to the motion-segmentation in the monocular set-up (optical-flow and tracking in 2-D). In this case, the effort that we put in the acquisition is reflected in motion-segmentation algorithm that can be solved using a simple clustering algorithm. The main disadvantages of using such application of the scene flow

is first is sparseness and second, the fact that surfaces require to be textured.

In the second approach to articulated segmentation, we address the problem from a geometrical point of view, to take into account the shape of the object while ensuring consistent segmentations over time. Here we directly address the inconveniences of the previous method by further constraining the environment.

In this novel dynamic segmentation scheme, moving articulated bodies are clustered in an embedding space, and clusters propagated in time to ensure temporal consistency. By exploiting some desirable characteristic of Laplacian Embedding and LLE, we estimate the optimal number of clusters in order to merge/split clusters in correspondence of topology transitions. We compared the performance of the algorithm versus direct EM clustering in 3-D, k-means clustering in Isomap space, and ground truth labeling provided through motion capture. The proposed unsupervised segmentation algorithm can be seen as a building block of a wider motion analysis framework, as it provides a coherent body-part segmentation along a sequence. For example, we can fit ellipsoids to

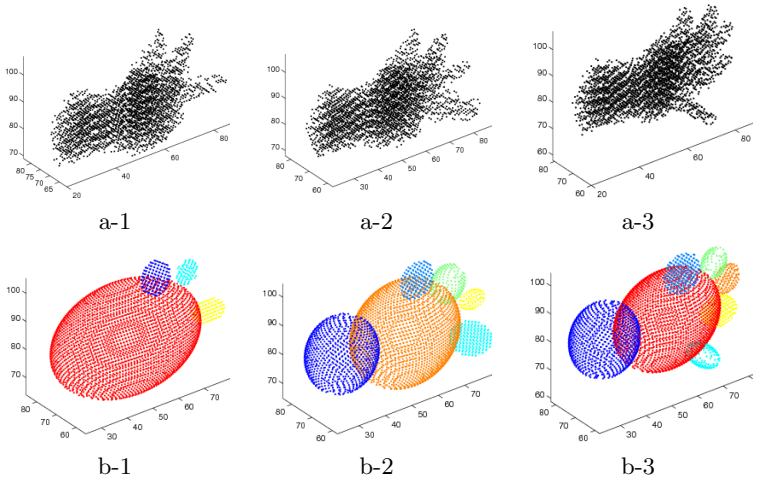


Figure 6.20: **a**-1,2,3**)** A sequence of voxel-sets capturing a counting hand in an augmented reality environment. **b**-1,2,3**)** Corresponding rough articulated model fitting based on clusters.

the segmented protrusions by aligning the moments or principal axes: Fig. 6.20 shows the resulting rough model fit to a sequence of voxel-sets representing a counting hand.

Chapter 7

Scene Flow

Contents

7.1 Introduction

Scene flow was introduced by Vedula et al. [Vedula et al., 1999, 2005] as the 3-D vector field, defined on each point on every surface in the 3-D scene, which represents the motion of these points

between two time frames. Since the optical flow is simply a projection of the scene flow onto a camera image plane, the most obvious way to compute scene flow is to reconstruct it from the optical flow measured in one [Vedula et al., 2005] or more cameras [Vedula et al., 2005, Zhang and Kambhamettu, 2000], possibly helped by a dense stereo reconstruction [Li and Sclaroff, 2005]. In these cases, the difficulty consists in constructing a scene flow which is compatible with several observed optical flows which may bring contradictory informations. Another approach is to work in the scene domain, and to track 3-D scene points or surface elements (surfels) instead of 2-D image points. The most notable work in this area is perhaps the one of Carceroni and Kutulakos [Carcernoi and Kutulakos, 2002]. They model the scene as a set of surfels, each surfel being described by its shape (an oriented planar patch), reflectance (a texture for the albedo and the two specular coefficients of a Phong model), bump map (which models local surface curvature), and motion (modeled as a 3-D affine transform). They show that, given camera parameters and knowing the position of light sources, they can recover the surfel parameter by successive optimizations performed on subsets of the parameters, so that the surfels maximize photo-consistency. The resulting inter-frame 3-D motion field is computed at sampled positions in a known 3-D volume, and the method doesn't try to follow surfels over several frames. The way the method deals with self occlusions is by reconstruction the whole scene so that occlusions can be recovered explicitly. Overall, though the method and the results are impressive, it requires a well-controlled lighting and acquisition setup, and because of the high number of surfel parameters to optimize, it is limited to the recovery of the scene flow in a limited volume. Dellaert et al. [Dellaert et al., 1998] also propose a surfel tracking method, but their method is more focused on extracting a super-resolved surfel texture, and it is limited to one camera. Pons et al. [Pons et al., 2005] propose a two-step approach, in which they solve alternatively for 3-D reconstruction and scene flow using a variational method. However, their method handles the visibility of each scene point from the cameras using the global 3-D reconstruction, which implies that the whole 3-D scene has to be reconstructed without any missing parts.

We propose two novel methods to compute scene flow which work in the scene domain (as in [Carcernoi and Kutulakos, 2002]), but rather than sampling the scene volume and extracting the scene flow at each position in that volume, we propose an approach inspired by classical 2-D tracking, transposed in 3-D. Good tracking candidates are first detected from the original images, the initial surfel pose (i.e. translation and rotation) and texture parameters can be reconstructed from at least two images, and then each 3-D point or surfel is tracked over the longest possible time sequence using a multi-camera extension of the Lucas-Kanade algorithm [Lucas and Kanade, 1981, Baker and Matthews, 2004]. At each time frame and for each surfel, we extract the translation components from all the images where it is visible using a pyramidal approach, and then we can compute the full pose parameters (rotation and translation). The surfel visibility is updated between every two time frames, from the surfel orientation with respect to each camera, and from the correlation between the surfel texture and the images where it projects. The result is a set of trajectories across time, where each 3-D point or surfel can become visible or invisible in each camera at each time frame. Rather than computing a dense scene flow at each time frame as in previous approaches [Vedula et al., 2005, Carcernoi and Kutulakos, 2002], we directly get a set of 3-D trajectories over an interval of time, and scene flow at a given time is obtained by deriving these trajectories. Obviously, further scene motion analysis will be easier to compute from full 3-D trajectories than from instantaneous motion vector fields which need to be integrated over time: trajectories obtained by integrating scene flow may drift from the actual scene motion. In the case of surfel tracking, we make sure that the tracked surfel does not drift from the physical point on the surface by monitoring the intrinsic texture of each surfel. The rest of the chapter is organized as follows: we first describe the general

framework for a multi-camera extension of the Lucas-Kanade tracking method, and we then derive it into two particular methods: Tracking of 3-D points, and tracking of 3-D surfels. We then discuss about the initialization of 3-D points and surfels, which is based on the tracking method itself. Finally, we present results of scene flow and trajectories computation on real scenes and present conclusions and further work on these methods.

7.2 Multi-camera extension of Lucas-Kanade

The Lucas-Kanade method is a classical method to compute optical flow or to track 2-D points in a video sequence, and a reference publication on this subject is the study by Baker and Matthews [Baker and Matthews, 2004], from whom we borrowed most of the notation used in this chapter. They give four formulations of this problem (forward or backward, additive of compositional), from which we use the forward additive method, for reasons which will be explained in the conclusion. In the Lucas-Kanade problem, each tracked feature is described by a vector of parameters p and a texture template $T(\boldsymbol{x})$, where x is a texture point. Given feature parameters $\boldsymbol{p}$, the warp function $\mathbf{W}(x;\boldsymbol{p})$ maps each point x in the texture template T to a point in the image I. The method optimizes the feature parameters in order to minimize an energy formed by the squared differences between the texture template and the image:

$$\sum_x [I(\mathbf{W}(x;p)) - T(\boldsymbol{x})]^2 \tag{7.1}$$

In our case, since there are several cameras, the appearance of the feature may be different in each camera n, consequently there may be different texture templates T_n attached to each warp $\mathbf{W}_n$. A feature may be more or less visible in each camera, which can be expressed by a positive weight v_n, which we call *visibility*. The energy becomes:

$$\sum_n v_n \sum_x [I_n(\mathbf{W}_n(x;p)) - T_n(\boldsymbol{x})]^2. \tag{7.2}$$

The Lucas-Kanade algorithm supposes that an estimate of $\boldsymbol{p}$ is known, so that, at each optimization step, the goal is to find $\Delta\boldsymbol{p}$ which (approximately) minimizes:

$$\sum_n v_n \sum_x [I_n(\mathbf{W}_n(x;\boldsymbol{p}+\Delta\boldsymbol{p})) - T_n(\boldsymbol{x})]^2. \tag{7.3}$$

Using the first order Taylor expansion of Eq. 7.2 to expand Eq. 7.3 gives:

$$\sum_n v_n \sum_x [I_n(\mathbf{W}_n(x;\boldsymbol{p})) + \nabla I_n \frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T_n(\boldsymbol{x})]^2, \tag{7.4}$$

and its partial derivative with respect to $\Delta\boldsymbol{p}$ is:

$$2\sum_n v_n \sum_x \left[\nabla I_n \frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}}\right]^\top \left[I_n(\mathbf{W}_n(x;\boldsymbol{p})) + \nabla I_n \frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T_n(\boldsymbol{x})\right]. \tag{7.5}$$

At the minimum, this partial derivative must be zero, which leads to the following expression for the parameters update:

$$\Delta\boldsymbol{p} = \mathbf{H}^{-1} \sum_n v_n \sum_x \left[\nabla I_n \frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}}\right]^\top \left[T_n(\boldsymbol{x}) - I_n(\mathbf{W}_n(x;\boldsymbol{p}))\right], \tag{7.6}$$

where $\mathbf{H}$ is the Gauss-Newton approximation of the Hessian matrix:

$$\mathbf{H} = \sum_n v_n \sum_x \left[\nabla I_n \frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I_n \frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}} \right]. \tag{7.7}$$

The parameters are then updated ($\boldsymbol{p} \leftarrow \Delta\boldsymbol{p} + \boldsymbol{p}$), and the procedure is iterated until convergence (usually given by $||\Delta\boldsymbol{p}|| < \epsilon$). The next two sections specialize this multi-camera extension of Lucas-Kanade for tracking 3-D points and 3-D surfels using several cameras.

7.3 Tracking 3-D points using several cameras

7.3.1 Lucas-Kanade Iteration

When the tracked features are 3-D points, the parameters vector is simply made out of the world coordinates of this point, $\boldsymbol{p} = (X, Y, Z)$. In order to compute the parameters update Eq. 7.6, we need to define the following ingredients: The texture templates $T_n(\boldsymbol{x})$, the warps $\mathbf{W}_n$, and the Jacobian of each warp $\frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}}$. The templates $T_n(\boldsymbol{x})$ are square windows extracted from the set of previous images centered around the sub-pixel projection in each image of the 3-D position at the previous time frame (bilinear interpolation is used for re-sampling the images). Recall that the warps $\mathbf{W}_n$ are 2-D functionals that map template coordinates to image coordinates in image n. In this case, each warp $\mathbf{W}_n$ is the translation in image n by the 2-D coordinates of the projection of $\boldsymbol{p} = (X, Y, Z)$ in that image (we suppose in the rest of this chapter that the projection is perspective and that nonlinear distortion was removed from the images):

$$\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}) = \boldsymbol{x} + \mathbf{P}_n(\boldsymbol{p}), \tag{7.8}$$

where the coordinates $(x; y)$ of the projection can be written from the projection matrix $\widetilde{\mathbf{P}}_n = (p_{ij})_{3\times 4}$ of camera n using homogeneous coordinates,

$$(sx, sy, s) = \widetilde{\mathbf{P}}_n \tilde{\boldsymbol{p}}, \text{ with } \tilde{\boldsymbol{p}} = (X, Y, Z, 1). \tag{7.9}$$

If we write the projection function P_n without using homogeneous coordinates, we can compute the Jacobian of the warp in image n:

$$\frac{\partial \mathbf{W}_n}{\partial \boldsymbol{p}} = \frac{1}{s} \begin{pmatrix} p_{11} - x p_{31} & p_{12} - x p_{32} & p_{13} - x p_{33} \\ p_{21} - y p_{31} & p_{22} - y p_{32} & p_{23} - y p_{33} \end{pmatrix}, \tag{7.10}$$

and the parameters update (Eq. 7.6) is then computed from the Hessian (Eq. 7.7). Of course, given the expression of the Jacobian, the Hessian $\mathbf{H}$ will be of rank 2 (thus non-invertible) if there is only one camera in which the point is visible (i.e. $\exists n_0,\ v_{n0} > 0$ and $\forall n \neq n_0; v_n = 0$); this is reasonable, since the 3-D position of a point cannot be recovered from only one camera.

7.3.2 Computing the visibility

Usually, the appearance of a 3-D point changes slowly, but its visibility may change abruptly, especially when it becomes occluded by another part of the scene. If only two cameras are used, each point has to be seen in both cameras, so we can only assume $\forall n,\ v_n = 1$. If the point is visible in 3 cameras or more, the energy (Eq. 7.2) can be considered as a weighted least-squares problem with

outliers, where the visibility values v_n are weights which can be obtained using a robust estimator such as Huber's M-estimator [Huber, 1981]. First of all, one must estimate the robust scale of our least squares problem, i.e. the standard deviation of the residuals if outliers were discarded, supposing the remaining residuals follow a normal distribution. This can be done using the MAD (median absolute deviation) scale estimator: $\sigma = 1.4826\,\mathrm{med}_n\{|Y_n - \mathrm{med}_m Y_m|\}$, where med denotes the median, and the Y_n are the signed residuals. However, in our situation, the residuals in the energy (Eq. 7.2) are grouped per image, and the resulting grouped residuals are positive, so that we cannot use the above expression. We therefore have to simplify the MAD scale estimator to the following expression:

$$\sigma = 1.4826\,\{\mathrm{med}_n|Y_n|\}, \tag{7.11}$$

$$\text{with } Y_n = \sqrt{\sum_x [I_n(\mathbf{W}_n(\boldsymbol{x},\boldsymbol{p}) - T_n(\boldsymbol{x}))]^2}. \tag{7.12}$$

Once we have the scale, the visibility values can be estimated using the Huber function:

$$v_n = 1, \text{ if } |Y_n| \leq \sigma\tau, \text{ and } v_n \frac{\sigma\tau}{|Y_n|}, \text{ if } |Y_n \geq \sigma\tau| \tag{7.13}$$

where the 95% asymptotic efficiency on the standard normal distribution is obtained with the tuning constant $\tau = 1.345$. Note that this assures $v_n = 1$ for at least 2 cameras.

These visibility values are re-estimated at the beginning of each iteration, before computing the Hessian (Eq. 7.7) and the parameters update (Eq. 7.6), leading to a 3-D point tracker which is more robust to visibility changes and occlusions. Using a robust estimator in the case where the number of measurements is at most equal to the number of cameras may not be fully justified, however it still seems to be an appropriate way to handle automatically these visibility changes when little is known about the scene geometry, or when the scene surface is not smooth enough to use surfel tracking, as shown by the experiments.

7.3.3 Dropping lost points

If, during an iteration, the condition number of $\mathbf{H}$ (i.e. the ratio of the smallest singular value to the largest singular value) is very small, the parameters update will probably be wrong, so we consider the point is not trackable anymore and deactivate it.

The main problem with the 3-D point tracker is that nothing ensures that the tracked point remains on the surface scene: there can be a drift coming from small 3-D errors accumulated over time. One way to prevent this is to setup two of the cameras as a stereo pair with a small-enough baseline (we will see in Sec. 7.5 that this is also useful for 3-D point initialization): if, after complete optimization, a 3-D point is visible (e.g. $v_n > 0 : 8$) in both cameras, then the cross correlation between windows centered at each projection in these cameras should be above some threshold (0.8 in our experiments), else the 3-D point is considered lost and the track is cut before the current frame.

7.3.4 Pyramidal implementation

The main limitation of the Lucas-Kanade method is that, depending on the window size and the texture template, it may only be capable of tracking motions in the images of about one pixel

between two time frames. In order to track larger motion, a pyramidal implementation based on the 2-D tracker by Bouguet [Bouguet, 2000] is employed, which first estimates the 3-D motion at a coarse resolution (typically 1/8 of the original image size), and then improves it at each finer scale (1/4, 1/2, and 1). When switching from one scale to the other, the parameters vector is unchanged, but the projection matrices must be appropriately resized for the corresponding image resolution.

7.4 Tracking 3-D surfels

7.4.1 Definitions

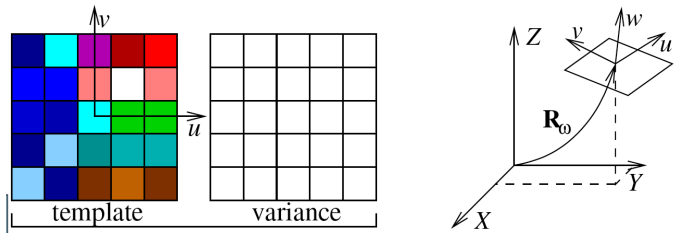


Figure 7.1: A surfel is defined by its *appearance* (a texture template and its variance) and *pose* (a position and rotation in 3-D)

A 3-D surfel is defined as a small planar square region in 3-D space with a *pose* and an *appearance* (Fig. 7.1). The *appearance* can be represented as a texture $T(\boldsymbol{x})$, $x \in [-s, s] \times [-s, s]$, and the resolution (r_u, r_v) of this texture which gives the size of a texture pixel in world units. The resolution is chosen so that the projected surfel resolution is about the same as the resolution of the images. Each surfel has a reference frame (u, v, w) attached to it, where u and v are aligned with the texture image axes, and w is aligned with the normal to the surfel. The appearance of a surfel can be extracted at the first time frame it appears, using the inverse of the warp $\mathbf{W}_n$ in each camera where it is visible. However, since the warp involves a perspective projection, the first time frame may not be the best one to get the best appearance. For that reason, the texture template is updated at each time frame, as we will see Sec. 7.4.4, and the estimated variance of the intensity at each texture pixel $P(\boldsymbol{x})$ is kept together with the texture template $T(\boldsymbol{x})$.

The *pose* of a surfel has 6 degrees of freedom, and can be represented by the position (X, Y, Z) of the center of the surfel, and three parameters for the rotation from the world reference frame to the surfel reference frame (we consider only surfels with a rigid motion, although this could be extended to more complicated local deformations such as an affine motion). Since the function that maps the rotation parameters to the actual rotation should not have singularities, we use a rotation vector $\omega = (\omega_X, \omega_Y, \omega_Z)$ to parameterize the rotation, and the rotation matrix can be computed using Rodrigues' formula.

Since the appearance of a surfel usually changes slowly across time, it is updated in a separate procedure, and the parameters vector only contains the pose parameters:

$$\boldsymbol{p} = (X, Y, Z, \omega_X, \omega_Y, \omega_Z) \tag{7.14}$$

7.4.2 Surfel Tracking

In the energy (Eq. 7.2), we considered the general case were there could be one texture template per camera. In the case of surfel tracking, there is only one texture template T for all the cameras: $T_n(\boldsymbol{x}) = T(\boldsymbol{x})$. The warp from the surfel texture template to camera n is a homography caused by perspective projection, composed by the following successive transforms: scaling up texture pixel units to world units, a translation and rotation corresponding to the surfel pose, and finally a projection in camera n. Let $\mathbf{W}_n(u, v)$ be the warp in Euclidean coordinates, and let $\widetilde{\mathbf{W}}_n$ be the warp that applies to the homogeneous texture coordinates vector $(u, v, 1)$. It can be represented by the following 3×3 matrix:

$$\widetilde{\mathbf{W}}_n = \widetilde{\mathbf{P}}_n \begin{pmatrix} r_{11} & r_{12} & X \\ r_{21} & r_{22} & Y \\ r_{31} & r_{32} & Z \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_u & 0 & 0 \\ 0 & r_v & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.15}$$

where the middle matrix is formed from the two first columns of the rotation matrix $\mathbf{R}_\omega = (r_{ij})$ and the translation vector (X, Y, Z) corresponding to the surfel pose. Let $(a\, b\, c)^\top = \widetilde{\mathbf{W}}_n (u\, v\, 1)^\top =$

$$\begin{pmatrix} p_{11}r_u r_{11}+p_{12}r_u r_{21}+p_{13}r_u r_{31} & p_{11}r_v r_{12}+p_{12}r_v r_{22}+p_{13}r_v r_{32} & p_{11}X+p_{12}Y+p_{13}Z+p_{14} \\ p_{21}r_u r_{11}+p_{22}r_u r_{21}+p_{23}r_u r_{31} & p_{21}r_v r_{12}+p_{22}r_v r_{22}+p_{23}r_v r_{32} & p_{21}X+p_{22}Y+p_{23}Z+p_{14} \\ p_{31}r_u r_{11}+p_{32}r_u r_{21}+p_{33}r_u r_{31} & p_{31}r_v r_{12}+p_{32}r_v r_{22}+p_{33}r_v r_{32} & p_{31}X+p_{32}Y+p_{33}Z+p_{14} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \tag{7.16}$$

Then, the expression for the Jacobian of the texture warp can simply be obtained by differentiation of the warp $\mathbf{W}_n(u, v) = (a/c, b/c)$ with respect to the surfel parameters $\boldsymbol{p}$, *i.e.*,

$$\frac{\partial \mathbf{W}_n(u,v)}{\partial \boldsymbol{p}} = \begin{pmatrix} \frac{1}{c}\frac{\partial a}{\partial \boldsymbol{p}} - \frac{a}{c}\frac{\partial c}{\partial \boldsymbol{p}} \\ \\ \frac{1}{c}\frac{\partial b}{\partial \boldsymbol{p}} - \frac{b}{c}\frac{\partial c}{\partial \boldsymbol{p}} \end{pmatrix}. \tag{7.17}$$

The derivatives of $(a, b, c)^\top$ in Eq. 7.17 with respect to the translation parameters, X, Y, Z are:

$$\frac{\partial (a\, b\, c)^\top}{\partial X} = \begin{pmatrix} p_{11} \\ p_{21} \\ p_{31} \end{pmatrix}, \qquad \frac{\partial (a\, b\, c)^\top}{\partial Y} = \begin{pmatrix} p_{12} \\ p_{22} \\ p_{32} \end{pmatrix}, \qquad \frac{\partial (a\, b\, c)^\top}{\partial Z} = \begin{pmatrix} p_{13} \\ p_{23} \\ p_{33} \end{pmatrix} \tag{7.18}$$

The derivatives of $(a\, b\, c)^\top$ with respect to the rotation parameters follow from the Rodrigues formula, which relates the entries of the matrix $\mathbf{R}_\omega$ to the vector $\omega = (\omega_X, \omega_Y, \omega_Z)^\top$:

$$\mathbf{R}_\omega = \begin{pmatrix} \cos\theta + \hat{\omega}_X^2(1-\cos\theta) & \hat{\omega}_X\hat{\omega}_Y(1-\cos\theta) - \hat{\omega}_Z\sin\theta & \hat{\omega}_Y\sin\theta + \hat{\omega}_X\hat{\omega}_Z(1-\cos\theta) \\ \hat{\omega}_Z\sin\theta + \hat{\omega}_X\hat{\omega}_Y(1-\cos\theta) & \cos\theta + \hat{\omega}_Y^2(1-\cos\theta) & -\hat{\omega}_X\sin\theta + \hat{\omega}_Y\hat{\omega}_Z(1-\cos\theta) \\ -\hat{\omega}_Y\sin\theta + \hat{\omega}_X\hat{\omega}_Z(1-\cos\theta) & \hat{\omega}_X\sin\theta + \hat{\omega}_Y\hat{\omega}_Z(1-\cos\theta) & \cos\theta + \hat{\omega}_Z^2(1-\cos\theta) \end{pmatrix}, \tag{7.19}$$

where the rotation angle is $\theta = ||\omega||$, and the unitary rotation axis $\hat{\omega} = (\hat{\omega}_X, \hat{\omega}_Y, \hat{\omega}_Z)^\top$ is determined by $\hat{\omega}_X = \frac{\omega_X}{\theta}$, $\hat{\omega}_Y = \frac{\omega_Y}{\theta}$, $\hat{\omega}_Z = \frac{\omega_Z}{\theta}$. Using Eq. 7.19, the remaining partial derivatives of the warp

jacobien (Eq. 7.17) are computed as follows:

$$\frac{\partial (a\,b\,c)^\top}{\partial \omega_X} = \theta \begin{pmatrix} -2p_{11}r_u\cos\theta\hat{\omega}_X + p_{12}r_u(1-\cos\theta)\hat{\omega}_Y + p_{13}r_u(1-\cos\theta)\hat{\omega}_Z & p_{11}r_v(1-\cos\theta)\hat{\omega}_Y + p_{13}r_v\sin\theta \\ -2p_{21}r_u\cos\theta\hat{\omega}_X + p_{22}r_u(1-\cos\theta)\hat{\omega}_Y + p_{23}r_u(1-\cos\theta)\hat{\omega}_Z & p_{21}r_v(1-\cos\theta)\hat{\omega}_Y + p_{23}r_v\sin\theta \\ -2p_{31}r_u\cos\theta\hat{\omega}_X + p_{32}r_u(1-\cos\theta)\hat{\omega}_Y + p_{33}r_u(1-\cos\theta)\hat{\omega}_Z & p_{31}r_v(1-\cos\theta)\hat{\omega}_Y + p_{33}r_v\sin\theta \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{7.20}$$

$$\frac{\partial (a\,b\,c)^\top}{\partial \omega_Y} = \theta \begin{pmatrix} p_{12}r_u(1-\cos\theta)\hat{\omega}_X - p_{13}r_u\sin\theta & p_{11}r_v(1-\cos\theta)\hat{\omega}_X - 2p_{12}r_v\cos\theta\hat{\omega}_Y + p_{13}r_v(1-\cos\theta)\hat{\omega}_Z \\ p_{22}r_u(1-\cos\theta)\hat{\omega}_X - p_{23}r_u\sin\theta & p_{21}r_v(1-\cos\theta)\hat{\omega}_X - 2p_{22}r_v\cos\theta\hat{\omega}_Y + p_{23}r_v(1-\cos\theta)\hat{\omega}_Z \\ p_{32}r_u(1-\cos\theta)\hat{\omega}_X - p_{33}r_u\sin\theta & p_{31}r_v(1-\cos\theta)\hat{\omega}_X - 2p_{32}r_v\cos\theta\hat{\omega}_Y + p_{33}r_v(1-\cos\theta)\hat{\omega}_Z \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{7.21}$$

$$\frac{\partial (a\,b\,c)^\top}{\partial \omega_Z} = \theta \begin{pmatrix} p_{12}r_u\sin\theta - p_{13}r_u(1-\cos\theta)\hat{\omega}_X & -p_{11}r_v\sin\theta + p_{13}r_v(1-\cos\theta)\hat{\omega}_Y \\ p_{22}r_u\sin\theta - p_{23}r_u(1-\cos\theta)\hat{\omega}_X & -p_{21}r_v\sin\theta + p_{23}r_v(1-\cos\theta)\hat{\omega}_Y \\ p_{32}r_u\sin\theta - p_{33}r_u(1-\cos\theta)\hat{\omega}_X & -p_{31}r_v\sin\theta + p_{33}r_v(1-\cos\theta)\hat{\omega}_Y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{7.22}$$

Again, if the condition number of $\mathbf{H}$ (the ratio of its smallest singular value to its largest singular value) is very small (e.g. less than 10^{-8}), we consider the surfel untrackable and deactivate it. Note that whereas 3-D point tracking needs at least two cameras, surfel tracking can work with only one camera, as described by Dellaert et al. [Dellaert et al., 1998], but some surfel configurations are degenerate (i.e. $\det \mathbf{H} = 0$), in particular when the surfel plane passes through the optical center.

7.4.3 Computing the visibility

We can use the same method as with 3-D point tracking to compute the visibility of the surfel in each camera, i.e. apply robust estimation to the least squares problem of Eq. 7.2. However, in the case of surfel tracking, we can also use a more geometric approach, by using the surface of the warped surfel as the visibility. One method to compute this surface is to use an approximation of the projective warp by the tangent affine warp at the surfel center (this approximation is valid as long as the surfel projects to a small area in the camera image). If the projective warp in homogeneous coordinates (Eq. 7.15) is written: $\widetilde{\mathbf{W}}_n = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$, then the tangent affine warp at $(0,0)$ is given by a first order Taylor expansion of $\mathbf{W}_n$:

$$\mathbf{W}_n^{\text{affine}} = \begin{pmatrix} \frac{ai-cg}{i^2} & \frac{bi-ch}{i^2} & \frac{c}{i} \\ \frac{di-fg}{i^2} & \frac{ei-fh}{i^2} & \frac{f}{i} \end{pmatrix}, \tag{7.23}$$

and the surface of the projected surfel is proportional to the determinant of the left 2×2 sub-matrix of $\mathbf{W}_n^{\text{affine}}$, or zero if this determinant is negative:

$$v_n = \max\left(0, \det \begin{pmatrix} \frac{ai-cg}{i^2} & \frac{bi-ch}{i^2} \\ \frac{di-fg}{i^2} & \frac{ei-fh}{i^2} \end{pmatrix}\right). \tag{7.24}$$

Nevertheless, this geometric approach cannot handle surfel occlusion by other parts of the scene. To detect occlusions, we compute the cross-correlation between the surfel texture template and the

warped image in each camera. If the result is under some threshold (e.g. 0.8), the surfel's visibility in this camera is set to zero. Of course, occluded surfels can only be detected after pose optimization, so this step has to take place after pose estimation at each time frame. The geometric visibility computation described above could be done at each iteration of the optimization, but since the surfel orientation usually moves slowly between two time frames it can also be done once the pose estimation is complete.

7.4.4 Texture template update

The texture template is extracted at the first time frame from the camera where it is the most visible, using the inverse of the warp in that image. Since we cannot perform occlusion detection without first having the texture template, it is better to extract it from one of the camera images used for surfel initialization (Sec. 7.5). Since the images from the first time frame contain intrinsic noise, and the pose and visibility of the surfel vary over the sequence, the texture extracted at other time frames may bring more information about the texture template, and we can incorporate this information by updating the texture template $T(\boldsymbol{x})$ and its variance $P(\boldsymbol{x})$. We propose to use a discrete Kalman Filter (DKF) for that purpose, which is a simplified version of what was proposed by Dellaert et al. [Dellaert et al., 1998] to build a super-resolved texture map. Basically, the DKF needs several ingredients: A state equation, a measurement equation, the state parameters covariance, and the measurement noise covariance. The state equation is trivial, since the state is the texture template, and it is supposed to be constant over time. The measurement equation is trivial too: we measure the state itself, by inverse warping of each image where the surfel is visible, and the measurement noise covariance is supposed to be diagonal (which is disputable, since neighboring pixels will most certainly have correlated values after applying the inverse warp, because of re-sampling). The DKF update equations can then be written for each camera n and at each template pixel x independently:

$$K = P(\boldsymbol{x})/(P(\boldsymbol{x}) + V_n(\boldsymbol{x})) \tag{7.25}$$

$$T(\boldsymbol{x}) \Leftarrow T(\boldsymbol{x}) + K(I_n(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T_n(\boldsymbol{x})) \tag{7.26}$$

$$P(\boldsymbol{x}) \Leftarrow P(\boldsymbol{x})(1-K), \tag{7.27}$$

where $V_n(\boldsymbol{x})$ is the measurement variance, K is the Kalman gain, and $P(\boldsymbol{x})$ is the texture (and state) variance. The state variance and measurement variances should be expressed in squared intensities, but using an arbitrary multiple of the intensity will not change the update equations: multiplying $P(\boldsymbol{x})$ and V_n in the equations above does not change the value of $T(\boldsymbol{x})$. Therefore, we can express V_n in any unit, $P(\boldsymbol{x})$ will be in the same unit. Intuitively, the measurement noise should decrease when the visibility increases, and $\lim_{v_n \mapsto 0^+} V_n = +\infty$, which led us to an empirical expression for measurement noise variance: $V_n = \frac{1}{v_n^2}$. The DKF equations are used after pose and visibility update, to update every template pixel using all cameras where it is visible.

7.4.4.1 Dropping lost surfels.

Wrong surfels can be detected from the condition number of $\mathbf{H}$ (Sec. 7.3.3), or when the visibility of a surfel is zero in all cameras. In these cases, the surfel is considered lost and the track is cut before the current frame.

7.4.4.2 Handling large motion.

Of course, surfel tracking suffers from the same problems as 3-D point tracking when large motion occur in the images. However, since surfel tracking needs very precise intensity measurements to get the surfel orientation, a pyramidal implementation (Sec. 7.3.4) will probably give wrong results. We propose instead to bootstrap the surfel tracking at each time frame using the 3-D point tracking (Sec. 7.3). This update the surfel position (X, Y, Z), and all six pose parameters are then estimated using surfel tracking. This procedure gave satisfying results in our experiments.

7.5 Good 3-D points or surfels to track

In the classical 2-D point tracking algorithms, the warp function $\mathbf{W}$ is usually a simple image translation, and points are selected in the first image by comparing the eigenvalues of matrix $\mathbf{H}$ (Eq. 7.7) at every image point for that family of warps, and keeping the best candidates [Harris and Stephens, 1988, Shi and Tomasi, 1998]. More recent work extended this to any warp in the image or intensity space [Triggs, 1999], but the case of 3-D tracking has not been studied yet. Of course, the complexity is very different, since we would have to compute the eigenvalues of $\mathbf{H}$ for every point in parameter space, which means every point in 3-D space for point tracking, and every point in 6-D space for surfel tracking.

7.5.1 Initializing 3-D points.

We propose another approach to select points or surfels: let us suppose that some 3-D points can be matched between the images from the first time frame (two of our cameras in the experimental setup have a small baseline for that purpose). Since both 3-D point tracking and surfel tracking start with a 3-D point tracking step, we can compute the Hessian (Eq. 7.7) from the Jacobian of the 3-D warp (Eq. 7.10) at every matched point. We can then select those for which the smallest eigenvalue and/or the condition number of $\mathbf{H}$ are above some threshold. Since the point visibility is not available at this state of the process, only the images used to reconstruct the 3-D point can be considered. If we use a stereo pair to reconstruct the 3-D points, each point is considered visible only in these two images. Since the neighborhood of the matched points is usually similar due to the stereo matching algorithm, the selected points are in fact close to the points found by a classical Harris-Stephens corner detector [Harris and Stephens, 1988]. However, in general, this method can also be used to select 3-D points reconstructed by any reconstruction algorithm, including widebaseline and multi-camera stereo, and give results that are very different from 2-D feature detectors.

7.5.2 Initializing surfels.

If we intend to do 6-D tracking, we also need the local normal to the scene surface, which can be estimated using a method similar to Lucas-Kanade tracking with an affine warp [Devernay and Faugeras, 1998]. The surfel texture template can then be initialized from the image used for its reconstruction where it is most visible, and its texture template variance can be set to a large value. We can then update its visibility (Sec. 7.3.2) and run the surfel tracking method from the first time frame to the first time frame, in order to refine its position and update its texture template. The surfel is then ready to be tracked over the whole sequence.

7.6 Results

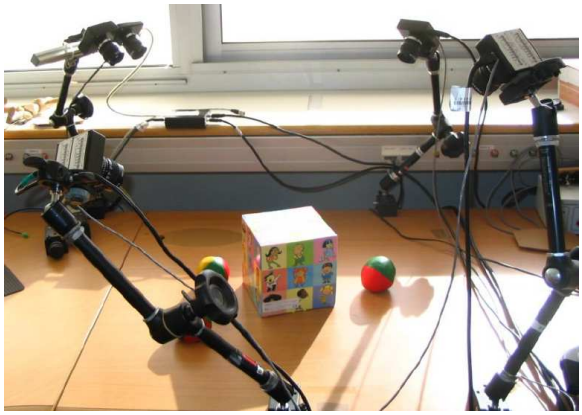
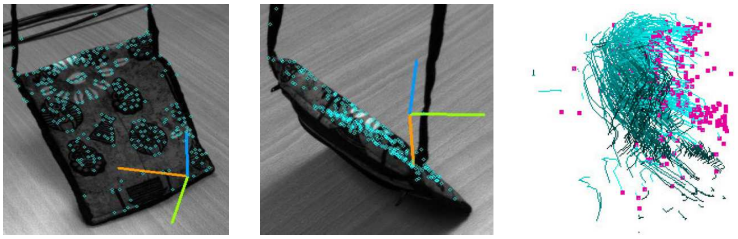


Figure 7.2: The 5-camera setup used for the experiments. The stereo pair (top-left) is used for initialization.

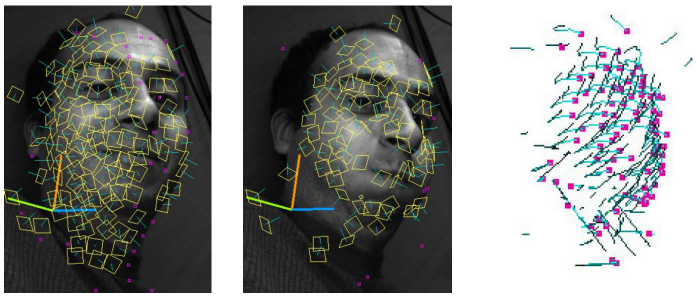
We present some experimental results obtained on a 5-camera system mounted on a desktop (Fig. 7.2). The cameras are 640 × 480 B&W cameras with 6mm lenses, calibrated with a multi-camera calibration method. Two of the cameras have a small baseline and are used for 3-D points and surfel initialization by using a dense stereo reconstruction method. The other cameras are used during tracking only, which explains why the tracked points and surfels in these examples are on parts of the scene that were facing the stereo cameras at the first time frame. This is only due to the initialization method we used in these experiments, and is not a restriction of the tracking methods presented here. No points or surfels are added during the sequences, and the points that are still there at the end of the sequence have been tracked since the very first frame.



(a) (b) (c)

Figure 7.3: 3-D point tracking results (sequence 2): 300 × 300 subimages from cameras 1 (used for initialization) and 5 (showing a visibility problem), and 3-D trajectories.

Sample tracking results are shown Fig. 7.3 and Fig. 7.4, but most of the results are presented as a video material that accompanies this thesis. Several 5-camera sequences are shown for each tracking method (4 sequences for 3-D point tracking, 6 sequences for surfel tracking), and most sequences were tested with both methods. The results are presented both on the original image



(a) (b) (c)

Figure 7.4: Surfel tracking results (sequence 3): 350×480 subimages from camera 1 at time frame 0 and 50, and 3-D trajectories. Invisible surfels are marked by a small square, and visible surfels are drawn in actual size with their normal.

sequences, and as 3-D trajectories which permits a better (though not perfect) visualization of the scene flow. In the 3-D point tracking sequences, all 3-D points are drawn on all camera images, since the visibility v_n is always strictly positive. In the surfel tracking sequences, the full surfel is drawn in a camera image if it is visible (i.e. $v_n > 0$), whereas only the center-point is drawn if the surfel is not visible. The main observation is that whenever surfel tracking fails (i.e. lots of surfels are lost), 3-D point tracking still gives good results. However, when the scene surface is smooth and textured (such as in sequence 1, the "drawing" sequence), surfel tracking gives much better results.

7.6.1 Precision

In order to measure the precision of our method we compared the results to ground truth obtained with a micrometric rotation table (Fig. 7.5). First the rotation axis and plane of the table are calibrated using a grid pattern. Then, objects are placed on the table and tracked with our point algorithm while the table performs one 360 degrees rotation. The error measure is the Euclidean distance between the tracked and predicted points, measured every 10 degrees. Predicted points are obtained by applying the cumulative rotation to the points reconstructed in the first frame. Fig. 7.6 shows the resultant standard deviation in blue, and median absolute deviation, which removes the influence of outliers, in red. The error slowly grows over time and the measured precision is 1.3cm at the end of the rotation. We also illustrate the errors for each coordinate separately. Since the table is aligned with the (x, y) plane, the motion is not significant in the z direction. Thus, errors in z are smaller compared to that of x and y.

7.7 Discussion and perspectives

7.7.1 Which method: 3-D points or surfels?

We presented a multi-camera extension of the Lucas- Kanade algorithm, from which we derived two feature trackers: one that tracks 3-D points parameterized by their world coordinates (X, Y, Z), and one that track planar surface elements (surfels) parameterized by their full 6-D degrees-of-freedom pose $(X, Y, Z, \omega_X, \omega_Y, \omega_Z)$. These two methods are used to recover scene flow

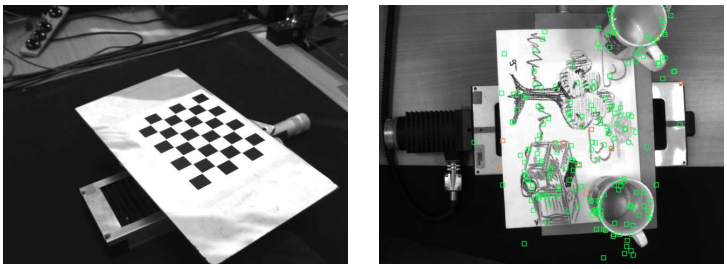


Figure 7.5: Micrometric rotation table used to measure the precision of our approach. The axis of rotation is calibrated using a grid pattern (left). Precision is measured by placing objects on the table and comparing the tracking results against ground truth.

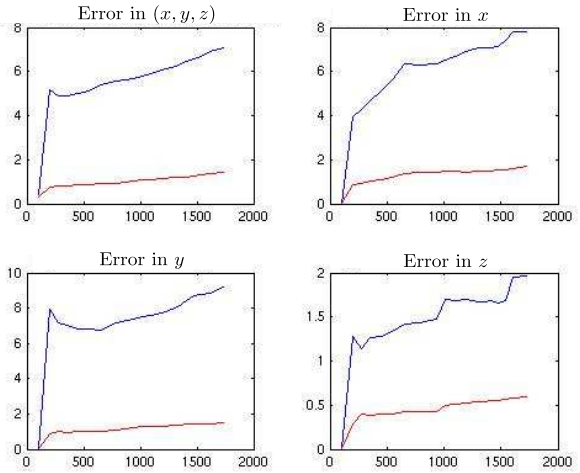


Figure 7.6: Error measurements in cms. The blue curve shows the standard deviation as the sequence evolves. The red curve shows the median absolute deviation.

and trajectories over a period of time. Of course, the latter method brings more information on the scene flow and handles visibility in a more rigorous way, but it still suffers from several weaknesses.

The first problem comes with surfel initialization: the scene surface has to be both locally planar and sufficiently textured in order to be able to extract the surface normal from the initial images. The surfel tracking method also needs enough local texture to be able to follow the surface orientation across time. Overall, these qualities are difficult to find in a natural scene, though they might be met by some parts of the scene (cloth, printed material, textured surfaces such as wood). Besides, surfel tracking is more computationally expensive, because of the complexity of the warps involved in this method, and it could be difficult to track several hundred surfels in real-time.

On the opposite, 3-D point tracking is easy to initialize with any multi-camera reconstruction method, and its computational cost is roughly n times the cost of tracking 2-D points in a single view, where n is the number of cameras. It can easily benefit from highly optimized and robust implementations of the standard 2-D Lucas-Kanade point tracking method [Bouguet, 2000], and

runs in real-time on our 5-camera setup where computations are made on a single 2.8GHz Pentium 4. However, automatic visibility handling by the use of M-estimators is not completely justified, because we are using a robust estimation on a little number of measurements, and it sometimes fails. The method may also suffer from drift problems, where small errors on the 3-D position estimation are accumulated over time, and the 3-D point ends up not being on the surface scene anymore.

A good balance would be to use surfel tracking only at points in the scene where it will give robust results, i.e. on smooth textured surfaces, and to use the 3-D point tracker everywhere else. In fact, if an initial 3-D reconstruction of the scene is available, the 3-D point tracker can be used to recover scene flow at almost every point (i.e. except where the condition number of $\mathbf{H}$ is too small, see Sec. 7.3.3).

7.8 Why the forward additive method?

As noted in Sec. 7.2, among the four possible formulations of the tracking problem [Baker and Matthews, 2004], we picked up the forward additive method, which is the original formulation by Lucas and Kanade [Lucas and Kanade, 1981] and the most straightforward, in which the we optimize a full warp from template space to image space. The other formulations are equivalent, but usually have a lower computational cost. For example, the forward compositional formulation iteratively solves for an incremental warp, and would require the energy (Eq. 7.2) to be rewritten as:

$$\sum_n v_n \sum_x [I_n(\mathbf{W}_n(\mathbf{W}_n(\boldsymbol{x};\Delta\boldsymbol{p});\boldsymbol{p})) - T_n(\boldsymbol{x})]^2 \tag{7.28}$$

The gain in terms of computational cost is due to the fact that the Jacobian of the warp only has to be computed at $\boldsymbol{p} = 0$ and can be precomputed once for all (the *inverse compositional* formulation also enables pre-computation of the Hessian (Eq. 7.7)). However, it requires that the set of warps contains the identity ($\mathbf{W}_n(\boldsymbol{x};0)$ has to be the identity warp), and this condition does not hold in the formulation we use for surfel tracking: the warp could be the identity in one camera, but because of perspective projection of the texture template it cannot be the identity in all cameras for one value of $\boldsymbol{p}$. For this single reason, both the *forward compositional* and *inverse compositional* formulations cannot be used for surfel tracking. The *inverse additive* formulation imposes even further constraints on the set of warps and cannot be used for surfel tracking either.

For the 3-D point tracker, it is possible to use the *compositional formulation* by reworking the warp parameterization so that it contains the identity warp. Nevertheless, the computational cost reduction with respect to the *forward additive* formulation would not be significant: the warp in each image consists only in translations, so that both the Jacobian and the Hessian computations are inexpensive. For the sake of simplicity, we preferred using the *forward additive* formulation for both methods.

7.8.1 Perspectives

Although scene flow was introduced some years ago by Vedula et al. [Vedula et al., 1999, 2005], it still has not gained much attention from the community, and a lot of work is still concentrated on either doing static 3-D reconstructions from multiple cameras, or extracting 2-D optical flow from monocular image sequences. Scene flow is at the crossing of these two techniques, and it should be considered as an essential tool to study motion in 3-D scenes, especially articulated and deformable motion. The main problem probably lies in the fact that reconstructing scene flow

is still a difficult process: Vedula et al. proposed to reconstruct it from optical flow, but optical flow computation is already an ill-posed problem, whereas other attempts (most notably the work of Carceroni and Kutulakos [Carcernoi and Kutulakos, 2002] and Pons et al. [Pons et al., 2005]) involved a complicated optimization framework.

Several problems make scene flow estimation more difficult than optical flow. There is a representation problem: the objects to track are not pixels in the image but small 3-D primitives that have a small footprint in the images. But there is also a visibility problem: these primitives may go from one camera to the other, may become occluded by other parts of the scene, or may disappear completely.

The methods we propose to compute scene flow are based on the much-studied Lucas-Kanade algorithm and its derivatives. Its extension to multiple cameras lead us to two tracking methods: one is capable of tracking 3-D points with automatic handling of the visibility based on robust estimation, and the other tracks surfels (surface elements) and computes the visibility from the geometry of the surfel. They rely on standard least-squares optimization, which can easily be extended to use more robust tracking techniques such as particle filtering. Scene flow can be extracted from as little as two cameras, but any number of cameras can be used without any restriction on their field-of-view, and a complete 3-D reconstruction of the scene is not necessary.

Chapter 8

Conclusions

Contents

8.1 Conclusion

Technological advances in recent years have had a great impact on the way we study the motion of humans and animals. Today, cameras play an important role in most of the motion analysis methods, and optical marker-based systems have become the standard method to capture motion. With the popularization of multiple-camera systems, researchers in Computer Vision try to move towards a more flexible marker-free solution. To this end, a wide range of model-based and learning-based solutions has been proposed, but more general unsupervised solutions remain challenging.

One difficult task in this context is the capture of 3-D articulated motion. Capturing articulated motion is in essence an instance of the "correspondence" problem, where we want to find the location of each limb at each frame in the sequence. In this document we have addressed different aspects of the problem, and provided three different formulations. In the first, we consider correspondences of 3-D features (points or surfels) over-time, which we then cluster in groups of coherent motion. In the second, we study dense graph representations of pairs of objects and find dense correspondences by solving a graph-matching problem. Finally, the third formulation considers correspondence of limbs over time.

Through the methods presented in this document we show the feasibility of capturing motion of articulated objects in an unsupervised manner from a multiple-camera system. Nevertheless, several challenges remain open; for instance handling partial matching in our dense matching, or the enforcement of geometric constraints between features in the scene-flow method.

8.2 Summary of the proposed methods

Each of the methods used to solve the different formulations of the motion-capture correspondence problem, is specific and has advantages and disadvantages over the others, as we see bellow.

The *scene-flow* method represents the objects in the scene as a sparse collection of 3-D features (points or surfels). Each feature is composed of a vector describing its position in the space and an appearance template (one template per camera in the case of point features). The algorithm looks for feature correspondence in time, by minimizing the difference between the warped templates and the current projection of the features on the images. This leads to a non-linear least-squares criteria over their position parameters, which is then solved by a Gauss-Newton algorithm. The algorithm explicitly handles visibility and the rejection of uncertain features. The result is a sparse set of trajectories describing the motion of the objects in the scene. The method is effective for real-time applications, where no prior knowledge on the number or type of objects in the scene. Its main restriction is the need of textured surfaces.

The set of trajectories obtained with the scene-flow algorithm has been used to study articulated-objects, as described in our *motion-segmentation* algorithm. The problem is formulated as the clustering of the trajectories in groups of rigid motion. A solution is found by defining a similarity measure that takes into account the visibility of the features, and using a spectral clustering method. The result is a collection of rigid elements in the scene, and in the case of articulated objects a collection of body-segments.

The *shape-registration* method represents articulated objects as shape-graphs. The correspondence is then formulated as an inexact graph matching problem, based on a spectral relaxation. We have shown that this relaxation is equivalent to finding a spectral representation of the graphs as a set of eigenfunctions. The equivalence allows us to address the problem raised by instabilities in the eigenvalue ordering usually neglected in the literature. As a result, our proposed method solves the inexact graph matching problem in two steps: first, finding a common subspace to register the graphs, and second, registering the resultant point-sets in the embedding space. The common subspace is solved either by an out-of-sample extension when dealing with sequences, or by solving a linear assignment problem between the eigenfunction-histograms. The algorithm is recommended for cases where dense correspondences between the articulated objects are needed. Additionally, it provides a solution for matching widely-separated poses of the object, or for different objects with similar topology. However, it may not be well suited to real-time applications, as it requires the eigen-decomposition of the Laplacian matrix and the registration of the point-sets in the embedding space; both of which are computationally expensive. Finally, the method can deal with structural changes between the graphs, owing to the properties of the Earth Mover's Distance and the outlier rejection in the point-registration. However, these are not enough for solving part-matching, *e.g.* matching only an arm to the entire body, since the difference between their spectral representations has many local-minima.

Finally, the *coherent shape-segmentation* method addresses the consistent correspondence of protrusions in a sequence. The problem is formulated as clustering over time, and solved first, by finding a shape-based segmentation, and then by propagating the cluster seeds over time. The use of the spectral representation in this case guarantees that the clusters always remain related to the protrusions of the object (instead of drifting over time). The method explicitly handles topological changes in the graph by merging and splitting clusters accordingly. A comparison of the characteristics of each method is presented in Table. 1.1.

8.3 Contributions

The goal of this thesis is the unsupervised study of motion-capture and modeling of articulated objects in the context of multiple-camera systems. Several aspects of the related problems were investigated and solved. The main contributions resulting from this investigation are:

- Two novel algorithms to compute sparse scene-flow from multiple-view videos in real-time, based on a multiple-camera extension of the Lucas Kanade algorithm (Sec. 7.3 and Sec. 7.4).
- A spectral representation of the 3-D reconstruction of articulated shapes, the derivation of which brings together concepts from Spectral Graph Theory, non-linear spectral embedding methods and Geometric Processing (Sec. 3.2).
- A method for finding correspondences between different representations of articulated objects (*e.g.* voxel-sets or meshes) in arbitrary poses, coming from the same sequence, from different sequences, or between topologically similar objects (Chap. 4 and Chap. 5).
- A theoretical link between the spectral representation of shapes and spectral graph matching (Sec. 4.2 and Sec. 4.2.5).
- A study of the stability of spectral graph matching and two ways of improving it in the context of articulated shape matching (Sec. 4.2.7 and Sec. 4.2.8).
- A formulation of the point-registration algorithm in the framework of probabilistic clustering and the derivation of an Expectation Maximization algorithm to solve it (Sec. 5.2).
- An algorithm to achieve coherent segmentation of sequences of articulated objects over time (Sec. 6.3).

8.4 Perspectives

The analysis and methods presented in this document show the feasibility of unsupervised methods for articulated-motion capture. Research in the domain is developing fast, but important challenges, such as temporal consistency, are not yet fully solved. Current solutions still rely on high-quality reconstructions and highly controlled environments. Our methods provide practical solutions that can handle situations outside of the assumptions commonly made. For example, as opposed to most of the current state-of-the-art scene-flow methods, which address the recovery of a dense flow field, our method does not rely on a surface that has been reconstructed beforehand. This allows us to recover the scene-flow in real-time and to easily handle multiple objects in the scene. Similarly, many current methods rely on complex photometric models while disregarding the shape information; we have shown that the geometry of the shape provides enough information to register widely-separated poses of the objects.

Regarding the spectral analysis of the geometry of the shape, we have provided a method that finds a common subspace from the eigenfunctions of each shape, instead of simply relying on eigenvalue ordering. Other possibilities could also be interesting; for example, considering statistical properties of the eigenfunctions instead of just histograms. Furthermore other transformations between the eigenfunctions could be considered, instead of only a one-to-one assignment.

One major problem that remains unsolved is partial matching. Towards this end, we believe that the spectral frameworks presented here for matching and segmentation could be unified. Finally, an interesting open question to be addressed is the possibility of formally including the photometric information in the same spectral framework.

Appendix A

Kernel Methods

A.1 Kernel methods

In this appendix we review kernel methods and their relation to non-linear embedding algorithms for representation, manifold learning and dimensionality reduction. Kernel methods rely on a mathematical link between positive definite matrices and continuous kernel functions, similar to the relation between the graph-Laplacian and the continuous Laplacian operators. In most of the non-linear embedding algorithms, positive definite matrices are constructed by measuring pairwise similarities between data-samples. The kernel methods reinterpret these matrices as the evaluation of a continuous kernel function over sample pair. Using the kernel trick, the kernel evaluation becomes a dot product in a high-dimensional Hilbert Space. An embedding in this context consists in finding the map to that high-dimensional space, where the kernel evaluation is equivalent to a dot product. In the following we review the most important results of kernel methods, which serves not only as a complementary view of the non-linear spectral embedding methods, but also as a support for the out-of-sample extrapolation used in Sec. 4.2.7.

Just as in the case of manifold-based embedding methods, the input to kernel methods is a graph, with nodes representing data points and edges weighted by a similarity measure between pairs of nodes. In the kernel formulation, the similarity is assumed to be the result of the evaluation of a kernel function. A *kernel* is a symmetric real-valued function defined on a pair $\boldsymbol{x}, \boldsymbol{x}'$ from a given space $\mathcal{X}$, that is:

$$\mathcal{K} : \mathcal{X} \times \mathcal{X} \mapsto I\!R \quad \text{such that } (\boldsymbol{x}, \boldsymbol{x}') \mapsto \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \tag{A.1}$$

Examples of common kernels are the polynomial and the gaussian kernel:

$$\begin{array}{ccc} \textit{polynomial kernel} & \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}, \boldsymbol{x}' + c)d & \text{for } c \text{ and } d \text{ constants} \\ \textit{gaussian kernel} & \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = exp^{-(\boldsymbol{x}-\boldsymbol{x}')^2/(2\sigma^2)} & \text{with } \sigma \text{ an scale parameter} \end{array}$$

The main interest behind the kernel methods lies on a series of theorems relating a kernel functions to a vector space, usually called the "feature space". The existence of such vector space, facilitates the application of geometric algorithms, since it allows the evaluation of a kernel function $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$ to be computed from the dot product between the two elements $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ in the feature space

and vice-versa. The connection is straightforward when $\mathcal{X} = I\!R^n$ since the canonical product can be used:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle = \sum_i^n \boldsymbol{x}_i \boldsymbol{x}'_i \tag{A.2}$$

When $\mathcal{X}$ is not a dot product space, it is still possible to use kernel methods by means of the *kernel trick*, *i.e.* the result of an ensemble of theorems that determines a method to derive a map Φ, from $\mathcal{X}$ to an appropriate linear space $I\!H$, *i.e.* $\Phi : \mathcal{X} \mapsto I\!H$, such that $\mathcal{K}$ has a dot product representation in $I\!H$:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{I\!H} \tag{A.3}$$

The existence of this map is known as *Reproducing Kernel* property, and it gives rise to the *Reproducing Kernel Hilbert Spaces (RKHS)* in functional analysis. The main property of RKHS is that for a given symmetric positive definite kernel $\mathcal{K}$ over $\mathcal{X} \times \mathcal{X}$, it is always possible to find a Hilbert space $I\!H$ with *reproducing kernel* $\mathcal{K}$, as well as a function $\Phi : \mathcal{X} \mapsto I\!H$ such that Eq. A.3 is verified.

In other words, the *reproducing kernel property* states that, given a kernel function $\mathcal{K}$, there exists a function Φ such that the evaluation of the kernel at points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is equivalent to taking the dot product between $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{x}')$ in some (perhaps unknown) Hilbert Space. In fact, Φ can be thought as a mapping from an input space $\mathcal{X}$ to a generally large (possiblly infinite) feature space $I\!H$ ($\Phi : \mathcal{X} \mapsto I\!H$), where we can compute dot products simply by computing $\mathcal{K}$. A practical result of this property is the so-called *kernel trick*, which consists in replacing dot products by the evaluations of a kernel. By the same principle, taking a kernel product is actually equivalent to mapping the inputs to $I\!H$ and then taking the dot product in the new feature space.

The kernel trick is commonly used to transform any algorithm that solely depends on the dot product between two vectors by replacing it with the kernel function, in such way that the dot products need not to be calculated explicitly. This may be practical, for example, for infinite-dimensional spaces. As a consequence, if the kernel $\mathcal{K}$ is a nonlinear similarity measure, geometric algorithms dependent on the dot product space in $I\!H$ can be performed by using the maps $\Phi(\boldsymbol{x})$. Finally, an important subset of RKHS' are the reproducing kernel Hilbert spaces associated to continuous kernels. These spaces have several applications in machine learning, such as in manifold learning methods, support vector machines and Gaussian processes.

In practice, to exploit the relation in Eq. A.3 for the manifold learning application, an algorithm is needed to define the Hilbert Space from the kernel itself, or in other words, to find $\Phi : \mathcal{X} \mapsto I\!H$. Although, it is possible to explicitly construct a unique $I\!H$ from a kernel, up to an isomorphism; the converse relation (finding Φ) is not completely determined. Indeed, Φ is not unique, and at least two possible constructions are possible, discussed in the following sections. In brief, the first approach, uses $I\!H_K$ as the feature space and defines $\Phi(\boldsymbol{x}) = \mathcal{K}(\boldsymbol{x}, .)$. By the reproducing property of the kernel, the $I\!H$ is defined by the dot product:

$$\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{I\!H\,K} = \langle \mathcal{K}(\boldsymbol{x}, .), \mathcal{K}(\boldsymbol{x}', .) \rangle_{I\!H\,K} = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'). \tag{A.4}$$

The second possibility is to use L_2 as the feature space and rely on the *Mercer-Hilbert-Schmidt theorem* to define Φ as a function of the eigenfunctions ϕ_i and eigenvalues λ_i of $\mathcal{K}$. The later choice permits the interpretation of spectral non-linear embedding methods as kernel methods. The two options are detailed in Sec. A.1.1 and Sec. A.1.2 respectively, following their description in [Daumé III, 2004].

A.1.1 Kernels from Hilbert Spaces

One of the two possible methods to find Φ is to start from a Hilbert Space $\mathbb{H}$ and then try to calculate the dot product in $\mathbb{H}$ by the evaluation of some kernel function. If such kernel actually exists, it is called the *reproducing kernel of* $\mathbb{H}$. Relying on functional analysis and in particular on the *Riesz theorem* (see Theorem 3 below) it is possible to find the equivalence between the evaluation of a certain type of functionals and the dot product of the functional with a given vector.

Theorem 3 (Riesz Theorem or Representer Theorem) *Given a Hilbert space $\mathbb{H}$ of functions from $\mathcal{X}$ to $\mathbb{R}$, for some measurable $\mathcal{X}$. If Γ is a bounded linear functional on $\mathbb{H}$, then there is a unique vector $\boldsymbol{u}$ in $\mathbb{H}$ such that:*

$$\Gamma f = \langle f, \boldsymbol{u} \rangle_{\mathbb{H}} \forall f \in \mathbb{H}. \tag{A.5}$$

According to Theorem 3, applying a functional Γ to a function $f \in \mathbb{H}$ is equivalent to evaluating the dot product of the function f and a given vector $\boldsymbol{u}$. Here, $\mathbb{H}$ is chosen to be a RKHS, and Γ is designed to be the *dirac evaluation functional*, $\delta_{\boldsymbol{x}}$:

$$\delta_{\boldsymbol{x}} f = f(\boldsymbol{x}). \tag{A.6}$$

Since for $\mathbb{H}$ and $\delta_{\boldsymbol{x}}$ the conditions for the Riesz theorem hold, there must exist a unique vector, denoted here $\boldsymbol{\mathcal{K}}_{\boldsymbol{x}} \in \mathbb{H}$, such that:

$$\delta_{\boldsymbol{x}} f = f(\boldsymbol{x}) = \langle f, \boldsymbol{\mathcal{K}}_{\boldsymbol{x}} \rangle_{\mathbb{H}}. \tag{A.7}$$

If now $\boldsymbol{\mathcal{K}}_{\boldsymbol{x}}$ is defined to be equivalent to the evaluation of the kernel function $\mathcal{K}_x : \boldsymbol{x}' \mapsto \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$ defined on $\mathcal{X} \times \mathcal{X}$, then:

$$\begin{aligned} \delta_{\boldsymbol{x}} f &= f(\boldsymbol{x}) && \text{(A.8)} \\ &= \langle f, \boldsymbol{\mathcal{K}}_{\boldsymbol{x}} \rangle_{\mathbb{H}} && \text{(A.9)} \\ &= \langle f, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \rangle_{\mathbb{H}} && \text{(A.10)} \end{aligned}$$

Defining the kernel function $\mathcal{K}_x$ as above leads to the *reproducing kernel* for the Hilbert space $\mathbb{H}$. The space $\mathbb{H}$ is guaranteed to be entirely determined by $\mathcal{K}$, because the Riesz representation theorem assures that for every $\boldsymbol{x} \in \mathcal{X}$, the element $\boldsymbol{\mathcal{K}}_{\boldsymbol{x}}$ satisfying $f(\boldsymbol{x}) = \langle f, \boldsymbol{\mathcal{K}}_{\boldsymbol{x}} \rangle_{\mathbb{H}}$ is unique.

Furthermore, due to the symmetry property of kernels, f in the equation above can also be replaced by a vector $\boldsymbol{\mathcal{K}}_{\boldsymbol{x}'}$ and its corresponding kernel function $\mathcal{K}_{x'}$, such that $\langle \mathcal{K}_x, \mathcal{K}_{x'} \rangle_{\mathbb{H}} = \langle \mathcal{K}_{x'}, \mathcal{K}_x \rangle_{\mathbb{H}}$. As before, $\boldsymbol{\mathcal{K}}_{\boldsymbol{x}}$ and $\boldsymbol{\mathcal{K}}_{\boldsymbol{x}'}$ are the unique representatives of δ_x and $\delta_{x'}$. Replacing the kernels, the dot product in $\mathbb{H}$ can be written in terms of the functions $\mathcal{K}_x$ and $\mathcal{K}_{x'}$ as follows:

$$\langle \mathcal{K}_x, \mathcal{K}_{x'} \rangle_{\mathbb{H}} = \langle \mathcal{K}(\boldsymbol{x}, \cdot), \mathcal{K}(\boldsymbol{x}', \cdot) \rangle_{\mathbb{H}} \tag{A.11}$$

Finally, if we define $f(\cdot) = \mathcal{K}(\boldsymbol{x}, \cdot)$ and $g(\cdot) = \mathcal{K}(\boldsymbol{x}', \cdot)$, we obtain

$$\begin{aligned} \langle f, \mathcal{K}(\boldsymbol{x}, \cdot) \rangle &= f(\boldsymbol{x}) = \mathcal{K}(\boldsymbol{x}, \cdot) && \text{(A.12)} \\ \langle \mathcal{K}(\boldsymbol{x}', \cdot), g \rangle &= g(\boldsymbol{x}') = \mathcal{K}(\boldsymbol{x}', \cdot). && \text{(A.13)} \end{aligned}$$

Thus, the Reproducing Kernel Property guarantees that

$$\langle \mathcal{K}(\boldsymbol{x}, \cdot), \mathcal{K}(\cdot, \boldsymbol{x}') \rangle = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'). \tag{A.14}$$

Eq. A.14 demonstrates that Eq. A.3 is verified and confirms that $\mathcal{K}$ is a *reproducing kernel*. Notice that all reproducing kernels are positive definite. In fact, any positive definite function is a reproducing kernel for some RKHS.

A.1.2 A Hilbert space from a kernel

The description of the second method to find Φ is decomposed in two stages. First, we need to demonstrate that it is possible to define a Hilbert space from a positive definite kernel, such that it "reproduces" the kernel. Second, the map Φ relating the kernel to its RKHS is derived.

In the section Sec. A.1.1, we defined a kernel function in terms of a reproducing kernel Hilbert space. It follows from the definition of an inner product that a kernel defined in this way is symmetric and positive definite. The *Moore-Aronszajn theorem* proves the opposite, *i.e.* that every symmetric, positive definite kernel defines a unique reproducing kernel Hilbert space.

Theorem 4 (Moore-Aronszajn theorem) *Suppose $\mathcal{K}$ is a symmetric, positive definite (p.d) kernel on a set $\mathcal{X}$. Then there is a unique Hilbert space of functions on $\mathcal{X}$ for which $\mathcal{K}$ is a reproducing kernel.*

Based on Moore-Aronszajn's theorem we know that it is possible to create a RKHS, $\mathbb{H}_{\mathcal{K}}$, from a p.d. kernel, such that $\mathcal{K}$ is its reproducing kernel. In practice, to construct $\mathbb{H}_{\mathcal{K}}$ one needs first to establish a vector space $\mathcal{V}$ from which $\mathbb{H}$ can be formed. Then, a dot product and its induced norm are defined to ensure that $\mathbb{H}$ a Hilbert space. By choosing the appropriate $\mathcal{V}$ and dot product function, $\mathbb{H}$ the conditions needed for $\mathbb{H}$ to be a RKHS are verified as shown below.

Consider the set $\mathcal{S} = \{\boldsymbol{\mathcal{K}}_{\boldsymbol{x}} : \boldsymbol{x} \in \mathcal{X}\}$, where each vector $\boldsymbol{\mathcal{K}}_{\boldsymbol{x}}$ is determined by a function $\mathcal{K}_x$ such that $\mathcal{K}_x(\boldsymbol{x}') = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$, with $\boldsymbol{x}' \in \mathcal{X}$. A suitable vector space $\mathcal{V}$ can be defined by the set of all linear combinations of the elements of $\mathcal{S}$. Doing so, each element of $\mathcal{V}$, can be written as $\sum_i \alpha_i \mathcal{K}_{xi}$ and the dot product on $\mathbb{H}_{\mathcal{K}}$ can be defined by:

$$\langle \mathcal{K}_x, \mathcal{K}_{x'} \rangle_{\mathbb{H}\mathcal{K}} = \Big\langle \sum_i \alpha_i \mathcal{K}_{xi}, \sum_j \beta_j \mathcal{K}_{x'j} \Big\rangle_{\mathcal{X}} \tag{A.15}$$

for some vectors α and β. Due to the reproducing property of $\mathcal{K}$. The dot product in $\mathcal{X}$ may be rewritten as a kernel evaluation:

$$\langle \mathcal{K}_x, \mathcal{K}_{x'} \rangle_{\mathbb{H}\mathcal{K}} = \sum_i \sum_j \alpha_i \beta_j \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}'_j) \tag{A.16}$$

Additionally, for the definition of the dot product operation on $\mathbb{H}_{\mathcal{K}}$ to be correct, it is necessary to ensure two conditions. First, that V is complete, and second, that $\mathcal{K}$ is continuous and does not diverge, namely: $\int\int \mathcal{K}^2(\boldsymbol{x}, \boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}'$. The verification of these conditions concludes the construction of the Hilbert space with reproducing kernel $\mathcal{K}$.

The second stage of the procedure after finding $\mathbb{H}_{\mathcal{K}}$ is to define a map $\Phi : \mathcal{X} \mapsto \mathbb{H}$ such that $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathbb{H}}$. The most straightforward way to define Φ is to choose $\Phi = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x})$ and by the reproducing property of the kernel verify:

$$\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}\mathcal{K}} = \langle \mathcal{K}(\boldsymbol{x}, .), \mathcal{K}(\boldsymbol{x}', .) \rangle_{\mathcal{H}\mathcal{K}} = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \tag{A.17}$$

which satisfies the requirements for Φ. This is the converse of the method presented in Sec. A.1.1.

However, it is also possible to use L_2 as the feature space, by using the eigenfunctions ϕ and eigenvalues λ of $\mathcal{K}$ [1] and defining Φ by:

$$\Phi(\boldsymbol{x}) := \begin{pmatrix} \sqrt{\lambda_1}\phi_1(\boldsymbol{x}) \\ \sqrt{\lambda_2}\phi_2(\boldsymbol{x}) \\ \vdots \end{pmatrix} = \left\langle \sqrt{\lambda_i}\phi_i(\boldsymbol{x}) \right\rangle_{i=0}^{\infty} \tag{A.20}$$

which allows us to build the dot product as:

$$\begin{aligned} \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{L_2} &= \left\langle \begin{pmatrix} \sqrt{\lambda_1}\phi_1(\boldsymbol{x}) \\ \sqrt{\lambda_2}\phi_2(\boldsymbol{x}) \\ \vdots \end{pmatrix}, \begin{pmatrix} \sqrt{\lambda_1}\phi_1(\boldsymbol{x}) \\ \sqrt{\lambda_2}\phi_2(\boldsymbol{x}) \\ \vdots \end{pmatrix} \right\rangle_{L_2} && \text{(A.21)} \\ &= \left\langle \left\langle \sqrt{\lambda_i}\phi_i(\boldsymbol{x}) \right\rangle, \left\langle \sqrt{\lambda_i}\phi_i(\boldsymbol{x}') \right\rangle \right\rangle_{L_2} && \text{(A.22)} \\ &= \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(\boldsymbol{x}) \sqrt{\lambda_i}\phi_i(\boldsymbol{x}') && \text{(A.23)} \\ &= \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x}') && \text{(A.24)} \end{aligned}$$

Similar to the spectral theorem for matrices and vectors (c.f. Sec. 2.2.5), the Mercer-Hilbert-Schidt relates kernels to a linear combination of its eigenfunctions and eigenvalues.

Theorem 5 (Mercer-Hilbert-Schmidt) *This theorem states that if $\mathcal{K}$ is a symmetric positive definite kernel (continuous with a finite trace), then, there exists an infinite sequence of eigenfunctions $\langle \phi_i \rangle_{i=0}^{\infty}$ and eigenvalues λ_i with $\lambda_1 \leq \lambda_2 \leq \ldots$ of $\mathcal{K}$:*

$$\int \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \phi(\boldsymbol{x}) \phi(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x} \geq 0, \tag{A.25}$$

The eigenfunctions form an orthonormal basis such that the kernel $\mathcal{K}$ can be represented as a sum of a convergent sequence of product functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x}') \tag{A.26}$$

Using the Theorem 5 the Eq. A.24 can be finally written as:

$$\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{L_2} = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'), \tag{A.27}$$

which once again verifies de desired properties for Φ. In this case, Φ is called a Mercer Feature Map.

1. Initially introduced in Sec. 3.2.2.6, the concept of eigenfunction is recalled here in the context of kernels. Supposing $\mathcal{K}$ is a kernel, ϕ is an eigenfunction of $\mathcal{K}$ if:

$$\int \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \phi(\boldsymbol{x}') d\boldsymbol{x}' = \lambda \phi(\boldsymbol{x}') \qquad \forall \boldsymbol{x} \tag{A.18}$$

In a dot product notation this corresponds to the equivalent expression:

$$\langle \mathcal{K}(\boldsymbol{x}, .), \phi \rangle = \lambda \phi, \tag{A.19}$$

A.1.3 Relation of the Mercer map with non-linear spectral embedding methods

It is the selection of Φ as the Mercer map which makes the link between kernel methods and non-linear spectral embedding methods. This is easy to verify by comparing Eq. A.27 with the Laplacian-based embedding introduced in Sec. 3.2.2.3.

In fact, as it was the case for Isomap, LLE and the Laplacian method, practical kernel algorithms design the kernel function by interpreting the kernel in terms of a similarity measure. This allows us to represent each point ($\boldsymbol{x} \in \mathcal{X}$) by its similarity with *all* other points ($\mathcal{X} \mapsto I\!R^{\mathcal{X}}$ and $x \mapsto \mathcal{K}(., \boldsymbol{x})$). When only a sample-set of the points is available, the kernel function can be approximated by its value on the samples. This leads to a *Gramm matrix* or *kernel matrix*, denoted by $\mathbf{K}$, where $\mathcal{K}_{ij} := \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and for which positive semi-definiteness holds as required for any admissible kernel:

$$\sum_{i,j} a_i a_j \mathcal{K}_{ij} \geq 0 \tag{A.28}$$

for all finite sequences of points $\boldsymbol{x}_i, \ldots, \boldsymbol{x}_n$ of $\mathcal{X}$ and all choices of real numbers $a_1, \ldots, a_n$. Consequently, most of the similarity matrices used in non-linear spectral embedding methods can be related to a kernel matrix $\mathbf{K}$ [Bengio et al., 2004a].

A.1.4 Kernel-based out-of-sample extrapolations

The use spectral methods for embedding point-sets or graphs onto Hilbert spaces usually involves the eigen-decomposition of the an empirical kernel matrix $\mathbf{K}$ describing some similarity measure between an initial data-set of samples. Since the decomposition is calculated on the basis of all the entries of $\mathbf{K}$, adding or removing a pair line/column of the matrix yields to a different mapping function and thus a different embedding space. This is inconvenient for applications for which not all of the points are available at the same time, for example because incoming data is aggregated over time. New arriving data are usually called out-of-sample points and the problem of relating these to the previously calculated map, the *out-of-sample extension problem*. In the following, the out-of-sample extrapolation problem used in Sec. 4.2.7 is stated formally by using the kernel theory introduced in the previous sections of this appendix.

Consider an initial sample set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots \boldsymbol{x}_n\}$ generated with a density function $p(\boldsymbol{x})$ and a Hilbert space $I\!H_p$ of functions with the following inner product:

$$\langle g, h \rangle_p = \int g(\boldsymbol{x}) h(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} \qquad g, h \in I\!H \tag{A.29}$$

Consider now a kernel function $\mathcal{K}$ associated with a linear operator $\mathcal{K}_p$ in $I\!H_p$, such that:

$$(\mathcal{K}_p f)(\boldsymbol{x}) = \int \mathcal{K}(\boldsymbol{x}, y) f(y) p(y) dy. \tag{A.30}$$

As mentioned in [Bengio et al., 2004b], in most of the cases, the generating density p is unknown. Thus, the above inner product and linear operator are approximated by those defined with the empirical distribution $\tilde{p}$ and its induced empirical Hilbert space $H_{\tilde{p}}$.

Let $\widetilde{\mathcal{K}}(a, b)$ be a kernel function with a discrete spectrum, that gives rise to a symmetric matrix $\widetilde{\mathbf{K}}$ with entries $\widetilde{\mathcal{K}}_{ij} = \widetilde{\mathcal{K}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ upon the initial set $\mathcal{X}$. Let $(\boldsymbol{u}_k, \lambda_k)$ be an (eigenvector,eigenvalue)

pair that solves $\widetilde{\mathbf{K}}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k$. Also let (ϕ_k, λ'_{ik}) be an (eigenfunction,eigenvalue) pair that solves $\widetilde{\mathcal{K}}_{\tilde{p}}\phi_k = \lambda'_k f_k$ with $\tilde{p}$ the empirical distribution over $\mathcal{X}$. Bengio *et al.* [Bengio et al., 2004b] suggest to formulate the problem as a Nÿstrom approximation; the out-of-sample problem becomes that of extrapolating the eigenfunction values from the initial eigenvectors.

Appendix B

Earth-Movers Distance for Histogram Matching

B.1 Earth Mover's Diastance for Histogram Matching

This appendix gives a more detailed description of the method used for eigenfunction histogram matching in Chap. 4, namely the Earth Movers Distance (EMD). The description is based on the paper by Rubner *et al.* [Rubner et al., 1998] where the method was introduced and on the more efficient L_1 reformulation of the EMD proposed by Ling and Okada [Ling and Okada, 2007]. Introduced by Guibas *et al.* [Rubner et al., 1998], the Earth Movers Distance (EMD) defines a consistent measure of distance, or dissimilarity, between two distributions of points in a space for which a *ground distance* is given. It reflects the minimal amount of work that must be performed to transform one distribution into the other by moving the distribution mass.

EMD naturally extends the notion of distance between single elements to distance between sets of elements, or distributions. This is a an interesting property in the context of shape matching problems, since it allows for partial matching. When used to compare distributions that have the same overall mass, the EMD is a true metric. However, because of its cross-bin structure it is less sensitive to fixed binning it can be applied to signatures with different sizes. The EMD has been successfully extended to work with high-dimensional spaces [Andoni et al., 2008] and under transformation sets [Cohen and Guibas, 1999]. In [Cohen and Guibas, 1999], an iterative framework transforms a multi-dimesional distribution into another by minimizing their EMD distance. Although such approach could be applied in our case, first, it does not have a direct connection with the representation since it does not preserve the eigenfunctions and second, it may easily converge to a local minima. Instead, we use the histograms only to select a common eigenspace where the point-set registration is performed by an algorithm appropriately designed for the task (c.f. Chap. 5).

The EMD approach models the problem of matching two distributions as a special case of the transportation problem (a bipartite network flow problem), for which efficient algorithms are available. Let $\{h^k_{\mathcal{X}}\}_{k=\{1,\ldots,\mathcal{D}\}}$ and $\{h^l_{\mathcal{Y}}\}_{l=\{1,\ldots,\mathcal{D}\}\cup\{-1,\ldots,-\mathcal{D}\}}$ be the set of eigenfunction histograms for two shape-graphs $\mathcal{G}_{\mathcal{X}}$ and $\mathcal{G}_{\mathcal{Y}}$, as introduced in Sec. 4.2.8. Because of the invariance properties of similar shapes, we can set a fix number of bins, here B, for both histograms. To account for

changes in scale, the histograms can be normalized to have unitary mass. Conversely, if we plan to deal with partial matching of shapes of the scale, normalization is not advised. Solving both for scale changes and partial matching at the same time may be difficult.

Let the index-set $\mathcal{I} = \{1, \ldots, B\}$ (respectively $\mathcal{J} = \{1, \ldots, B\}$) be defined over the bins of each histogram $h^k_{\mathcal{X}}$ (respectively $h^l_{\mathcal{Y}}$). In order to compare two histograms $h^k_{\mathcal{X}}$ and $h^l_{\mathcal{Y}}$, the bins of the first histogram $(h^k_{\mathcal{X}}(i), i \in \mathcal{I})$ are considered as suppliers and those of the second $(h^l_{\mathcal{Y}}(j), j \in \mathcal{J})$ as consumers. The problem consists in finding the flow $\mathcal{F} = \{f_{ij}\}_{i\in\mathcal{I}, j\in\mathcal{J}}$ that minimizes the cost function $e(h^k_{\mathcal{X}}, h^l_{\mathcal{Y}})$, as follows:

$$\min_{\mathcal{F}=\{f_{ij}\}_{i\in\mathcal{I},j\in\mathcal{J}}} e(h^k_{\mathcal{X}}, h^l_{\mathcal{Y}}) = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} g_{ij} f_{ij} \tag{B.1}$$

where g_{ij} is the *ground distance* cost, which can be any distance between the i_{th} and j_{th} bin locations. The notation here corresponds to the comparison of one-dimensional histograms, as demanded in our application; the generalization to higher-dimensions is straightforward.

The minimization in Eq. B.1 is constrained to verify the following conditions:

$$f_{ij} \geq 0 \qquad i \in \mathcal{I}, j \in \mathcal{J} \tag{B.2}$$

$$\sum_{i\in\mathcal{I}} f_{ij} = h^l_{\mathcal{Y}}(j) \qquad j \in \mathcal{J} \tag{B.3}$$

$$\sum_{j\in\mathcal{J}} f_{ij} \leq h^k_{\mathcal{X}}(i) \qquad i \in \mathcal{I} \tag{B.4}$$

$$\sum_{j\in\mathcal{J}} h^l_{\mathcal{Y}}(j) \leq \sum_{i\in\mathcal{I}} h^k_{\mathcal{X}}(i) \tag{B.5}$$

Eq. B.2 expresses the asymmetric character of the suppliers and consumer histograms, since only the flow from a supplier to a consumer is allowed (and not vice-versa). Eq. B.3 forces consumers to fill up all of their capacities. Eq. B.4 limits the supply that a supplier can send, to its total mass. Finally, Eq. B.5 is a feasibility condition, that constraints the total demand not exceed the total supply.

The conditions above are used to build a constraint matrix, which can be very sparse. To solve for the linear optimization problem, the Transportation Simplex (TS) algorithm is used in [Rubner et al., 1998], this is a modified form of simplex algorithm which reduces the number of operations needed to maintain the constraint matrix, by taking advantage of its special structure. Other possible solutions include interior-point algorithms and incapacitated minimum network-flow, which have similar time-complexities. In a recent work, Ling and Okada [Ling and Okada, 2007] have proposed a reformulation of the EMD distance employing the L_1 (Manhattan) distance as ground distance. The new formulation reduces the number of constraints of the linear program, and therefore, the overall time complexity (from larger than $O(N^3)$ to $O(N^2)$ average). This is because the L_1 (Manhattan) as the ground distance the formulation is simplified and it takes only *integer* values. With the L_1 distance any positive flow can be replaced by a sequence of neighboring flows of distance $= 1$ (n-flows). This is because the L_1 distance forms a shortest path system on the integer lattice in such a way that any flow with distance greater than 1 can be obtained as a sum of edges of distance 1.

$$\text{EMD-}L_1(P, Q) = \min_{G=\{g_{i,j;k,l}:(i,j,k,l)\in\mathcal{J}_1\}} \sum_{\mathcal{J}_1} g_{i,j;k,l} \tag{B.6}$$

Subject to:

$$\sum_{k,l:(i,j,k,l)\in\mathcal{J}_1} (g_{i,j;k,l} - g_{k,l;i,j}) = b_{ij} \quad \forall (i,j) \in \mathcal{I} \tag{B.7}$$

$$g_{i,j;k,l} \geq 0 \quad \forall (i,j,k,l) \in \mathcal{J}_1 \tag{B.8}$$

where b_{ij} is the difference between the two histograms at bin (i,j). The constraint forces the total flow that leaves one node minus the total flow that enters the node to be equal to b_{ij}. The consequences of these simplification are a reduction for the number of unknown variables and the constraints, as well as a big computational speed-up since no distance computation is required (all ground distances are one and computations are integer: $g_{ij} = \sum_{dims} |i-j|$)

To perform the EMD-L_1 computation two algorithms are proposed: an extended transportation algorithm and an efficient tree-based algorithm. In our experiments, we use the code provided by the authors [Ling and Okada, 2007].

Bibliography

A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(1):44–58, 2006. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2006.21.

S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005.

J. K. Aggarwal and Q. Cai. Human motion analysis: a review. *Comput. Vis. Image Underst.*, 73 (3):428–440, 1999. ISSN 1077-3142. doi: http://dx.doi.org/10.1006/cviu.1998.0744.

N. Ahmed, C. Theobalt, C. Rössl, S. Thrun, and H.-P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, 2008.

H. Almohamad and S. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(5):522–525, 1993. ISSN 0162-8828.

A. Andoni, P. Indyk, and R. Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 343–352, 2008.

D. Anguelov, P. Srinivasan, H.-C. Pang, D. Koller, S. Thrun, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Neural Information Processig Systems (NIPS)*, 2004.

AR-tracking. `http://www.ar-tracking.de/Home.10.0.html`.

K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 9(5):698–700, 1987.

X. Bai, H. Yu, and E. R. Hancock. Graph matching using spectral embedding and alignment. In *International Conference on Pattern Recognition*, 2004.

C. T. H. Baker. *The Numerical Treatment of Integral Equations*. Clarendon Press, Oxford, 1977.

S. Baker and I. Matthews. Lucaks-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, 2004.

H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. ISSN 1077-3142.

M. Belkin. *Problems of Learning on Manifolds*. PhD thesis, University of Chicago, 2003.

M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. In *Proceedings of the Annual Conference on Learning Theory*, New York., 2005. Springer.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003.

Y. Bengio, O. Delalleau, N. L. Roux, J. franois Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16, 2004a.

Y. Bengio, J. francois Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Neural Information Processig Systems (NIPS)*, pages 177–184. MIT Press, 2004b.

P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14:239–256, February 1992.

D. Bespalov and A. Shokouf. Local feature extraction using scale-space decomposition. In *In ASME Design Engineering Technical Conferences, Computers and Information in Engineering Conference Conference (DETC 2004-57702). ASME Pres*, 2004.

D. Bespalov, W. C. Regli, and A. Shokoufandeh. Local feature extraction and matching partial objects. *Computer-Aided Design*, 38(9):1020–1037, 2006.

S. biasotti, B. Falcidieno, and M. Spagnuolo. Extended reeb graphs for surface understanding and description. In d. B. G. S. Borgefors G., editor, *In Discrete Geometry for Computer Imagery Conference*, volume 1953 of *Lecture Notes in Computer Science, Springer*, pages 185–197, 2000.

S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno. Sub-part correspondence by structural descriptors of 3d shapes. *Computer-Aided Design*, 38(9):1002–1019, 2006.

N. Biggs. *Algebraic Graph Theory (2nd Edition)*. Cambridge University Press, 1993.

C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

J.-Y. Bouguet. Pyramidal implementation of the lucas-kanade feature tracker. Technical report, Intel Corp., Microprocessor Research Labs, 2000.

H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(9):917–922, 1999.

H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters (PRL)*, 18(8):689–694, August 1997.

E. by H. Bunke and T. Caelli. Special issue on graph matching in pattern recognition and computer vision. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3), 2004.

E. by Sven Dickinson, M. Pelillo, and R. Zabih. Special section on graph algorithms and computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 10(23), 2001.

T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(4):515–519, 2004. ISSN 0162-8828.

T. S. Caetano and T. Caelli. Approximating the problem, not the solution: An alternative view of point set matching. *Pattern Recogn.*, 39(4):552–561, 2006. ISSN 0031-3203.

M. Carcassoni and E. R. Hancock. Correspondence matching with modal clusters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(12):1609–1615, 2003a.

M. Carcassoni and E. R. Hancock. Spectral correspondence for point pattern matching. *Pattern Recognition*, 36:193–204, 2003b.

R. Carcernoi and K. Kutulakos. Multi-view scene capture by surfel sampling: from video streams to non-rigid 3-D motion, shape and reflectance. *International Journal of Computer Vision (IJCV)*, 2002.

J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3):569–577, 2003. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/882262.882309.

J. Choi, A. Szymczak, G. Turk, and I. Esasa. Element-free elastic models for volume fitting and capture. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2006.

K. Choo and D. J. Fleet. People tracking using hybrid monte carlo filtering. *IEEE International Conference on Computer Vision (ICCV)*, 2:321, 2001. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2001.937643.

C. W. Chu, O. C. Jenkins, and M. J. Mataric. Markerless kinematic model and motion capture from volume sequences. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2003.

F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

S. Cohen and L. Guibas. The earth mover"s distance under transformation sets. In *IEEE International Conference on Computer Vision (ICCV)*, volume II, pages 1076–1083, 1999.

R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, 2006.

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.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the vf graph matching algorithm. In *Proceedings of the 10th International Conference on Image Analysis and Processing (ICIAP)*, page 1172, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0040-4.

N. D. Cornea, M. F. Demirci, D. Silver, A. Shokoufandeh, S. J. Dickinson, and P. B. Kantor. 3d object retrieval using many-to-many matching of curve skeletons. In *IEEE International Conference on Shape Modeling and Applications (SMI)*, pages 368–373, Washington, DC, USA, 2005. IEEE Computer Society.

J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1071–1076, 1995.

J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision (IJCV)*, 29(3):159–179, 1998. ISSN 0920-5691.

T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Neural Information Processig Systems (NIPS)*, 2007.

T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 2001.

A. Cross and E. Hancock. Graph matching with a dual-step em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(11):1236–1253, November 1998.

F. Cuzzolin, D. Mateus, D. Knossow, E. Boyer, and R. Horaud. Coherent laplacian 3-d protrusion segmentation. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 719–726, Anchorage, USA, 2008.

F. Dagognet. *Etienne-Jules Marey: A Passion for the Trace*. Zone Books, 1992.

D.Aldous and J. Fill:. *Reversible Markov Chains and Random Walks on Graphs*. http://www.stat.berkeley.edu/users/aldous/RWG/book.html, (monograph in preparation), 2002.

B. Dariush. Human motion analysis for biomechanics and biomedicine. *Machine Vision and Applications*, 14(4):202–205, September 2003.

H. Daumé III. From zero to reproducing kernel hilbert spaces in twelve pages or less. Available at http://www.isi.edu/~hdaume/docs/daume04rkhs.ps, February 2004.

D.Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 100(10):5591–5596, 2003.

E. de Aguiar, C. Theobalt, S. Thrun, and H.-P. Seidel. Automatic Conversion of Mesh Animations into Skeleton-based Animations. In *Eurographics*, volume 27-2, Hersonissos, Crete, Greece, 4 2008.

F. de Goes, S. Goldenstein, and L. Velho. A hierarchical segmentation of articulated bodies. In *Eurographics Symposium on Geometry Processing (SGP)*, volume 27(5), pages 1349–1356, 2008.

F. Dellaert, C. Thorpe, and S. Thrun. Super-resolved texture tracking of planar surface patches. In *IEEE International Conference on Intelligent Robotic Systems*, 1998.

M. F. Demirci, A. Shokouf, S. Dickinson, Y. Keselman, and L. Bretzner. Many-to-many feature matching using spherical coding of directed graphs. In *European Conference on Computer Vision (ECCV)*, pages 332–335, 2004.

M. F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, and S. Dickinson. Object recognition as many-to-many feature matching. *International Journal of Computer Vision (IJCV)*, 69(2): 203–222, 2006. ISSN 0920-5691.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc.*, 39:1–38, 1977.

F. Devernay and O. Faugeras. Computing differential properties of 3-D shapes from stereoscopic images without 3-D models. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 1998.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, pages 98–105. John Wiley and Sons, 1973.

A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(10):1285–1295, 2003.

W. Feng and Z.-Q. Liu. Spectral multiplicity tolerant inexact graph matching. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 2, pages 1196–1201, 2006.

X. Feng and P. Perona. Scene segmentation from 3d motion. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, page 225, 1998. ISBN 0-8186-8497-6.

I. Fischer and J. Poland. Amplifying the block matrix structure for spectral clustering. In *Machine Learning Conf. Belgium and Netherlands*, 2005.

M. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

A. Frank. On Kuhn's Hungarian method: A tribute from Hungary. *Naval Research Logistics*, 52 (1):2–5, 2005.

M. C. S. from Vicon. `http://www.vicon.com/`.

R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25(1):130–150, 2006. ISSN 0730-0301.

D. M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999. ISSN 1077-3142. doi: http://dx.doi.org/10.1006/cviu.1998.0716.

D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, page 73, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7258-7.

N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Eurographics Symposium on Geometry Processing (SGP)*, pages 197–206, 2005.

E. Gine and V. Koltchinskii. Empirical graph laplacian approximation of laplace-beltrami operators: large sample results. In *Proceedings of the 4th International Conference on High Dimensional Probability*, 2005.

S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(4):377–388, 1996. ISSN 0162-8828.

G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

K. Grauman and T. Darrell. Fast contour matching using approximate earth mover distance. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2004.

K. Grauman and T. Darrell. The pyramid match kernel pmk : Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision (ICCV)*, volume II, pages 1458–1465, 2005.

A. Gruber and Y. Weiss. Multibody factorization with uncertainty and missing data using the EM algorithm. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 707–714, 2004.

L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 11(9):1074–1085, Sep 1992. ISSN 0278-0070. doi: 10.1109/43.159993.

J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first international conference on Machine learning (ICML)*, page 47, New York, NY, USA, 2004. ACM.

C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conf.*, 1988.

M. Hein. Uniform convergence of adaptive graph-based regularization. In *Proceedings of the Annual Conference on Learning Theory*, pages 50–64, New York, 2006. Springer.

M. Hein, J. yves Audibert, and U. V. Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. In *Proceedings of the Annual Conference on Learning Theory*, pages 470–485. Springer, 2005.

R. A. Horn and C. A. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, UK, 1985.

P. J. Huber. *Robust Statistics*. John Wiley and Sons, 1981.

P. Indyk and N. Thaper. Fast image retrieval via embeddings. In *Proc. Third Workshop Statistical and Computational Theories of Vision, 2003.*, 2003.

K. Inoue and K. Urahama'. Separation of multiple objects in motion images by clustering'. In *IEEE International Conference on Computer Vision (ICCV)*, 2001.

V. D. S. J. B. Tenenbaum and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

V. Jain and H. Zhang. A spectral approach to shape-based retrieval of articulated 3d models. *Computer-Aided Design*, 39(5):398–407, 2007. ISSN 0010-4485.

V. Jain and H. Zhang. Robust 3d shape correspondence in the spectral domain. In *In Proc. of Shape Modeling International*, page 11829, 2006.

O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning (ICML)*, page 56, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-5.

A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(5):433–449, 1999.

M. Kac. Can one hear the shape of a drum? In *American Mathematical Monthly*, pages 1–23., 1966.

T. Kato. *Perturbation Theory for Linear Operators*. Springer Verlag, 1995.

D. Knossow, R. Ronfard, and R. Horaud. Human motion tracking with a kinematic parameterization of extremal contours. *International Journal of Computer Vision (IJCV)*, 79(3), September 2008.

P. Koehl. Protein structure similarities. *Current Opinion in Structural Biology*, 11(3):348–353, June 2001.

R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pages 315–322, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7.

S. Kullback. *Information Theory and Statistics*. Courier Dover Publications, 1997.

K.Varanasi, A. Zaharescu, and E. Boyer. Temporal surface tracking using mesh evolution verify. In *European Conference on Computer Vision (ECCV)*, 2008.

S. Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, Yale University, 2003.

S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(9):1393–1403, 2006. ISSN 0162-8828.

R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK Users Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, 1998. ISBN ISBN 978-0898714074.

M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1482 – 1489, October 2005.

B. Levy. Laplace-Beltrami eigenfunctions: Towards an algorithm that understands geometry. In *In Proc. of Shape Modeling International*, 2006.

F. Leymarie and B. Kimia. Computation of the shock scaffold for unorganized point clouds in 3d. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 821–827, 2003.

R. Li and S. Sclaroff. Multi-scale 3D scene-flow from binocular stereo sequences. In *IEEE Workshop on Motion y Video Computing, (WACV/WMVC)*, 2005.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145151, 1991.

R. Lin, C.-B. Liu, M.-H. Yang, N. Ahuja, and S. Levinson. Learning Nonlinear Manifolds from Time Series. In *European Conference on Computer Vision (ECCV)*, 2006.

H. Ling and D. W. Jacobs. Using the inner-distance for classification of articulated shapes. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 719–726, Washington, DC, USA, 2005. IEEE Computer Society.

H. Ling and K. Okada. Diffusion distance for histogram comparison. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2006.

H. Ling and K. Okada. An efficient emd for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2007.

R. Liu and H. Zhang. Segmentation of 3d meshes through spectral clustering. *Computer Graphics and Applications, Pacific Conference on*, 0:298–305, 2004. ISSN 1550-4085.

R. Liu and H. Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Special Issue of Eurographics 2007)*, 26(3):385–394, 2007.

J. Lladós, E. Martí, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(10):1137–1143, 2001.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60:91–110, 2004.

B. D. Lucas and T. Kanade. An interactive image registration technique with an application to setereo vision. In *International Joint Conference on Artificial Intelligence*, 1981.

B. Luo and E. R. Hancock. Matching point-sets using procrustes alignment and the em algorithm. In *British Machine Vision Conference (BMVC)*, 1999.

B. Luo and E. R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 3(10), 2001.

U. V. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

Y. S. M. Hilaga, T. Kohmura, and T. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *ACM SIGGRAPH*, pages 203–212, 2001.

D. Mateus, F. Cuzzolin, R. Horaud, and E. Boyer. Articulated shape matching using locally linear embedding y orthogonal alignment. In *IEEE ICCV Workshop on Non-rigid Registration yTracking through Learning (ICCV 2007 -NRTL workshop)*, Rio de Janeiro, Brazil, 2007.

D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, Anchorage, USA, 2008.

T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. *Computer Vision and Image Understanding*, 96(3):393–434, 2004. ISSN 1077-3142.

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* Wiley, New York, 1997.

M. Meila and J. Shi. A random walks view of spectral segmentation. In *Artificial Intelligence and Statistics*, 2001.

C. Ménier, E. Boyer, and B. Raffin. 3-D skeleton-based body pose recovery. In *International Symposium on 3D Data Processing, Visualization and Transmission*, Chapel Hill (USA), june 2006.

M. Meyer, M. Desbrun, P. Schroder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Proceedings of VisMath*, 2002.

I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Articulated body posture estimation from multi-camera voxel data. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 1:455, 2001. ISSN 1063-6919.

D. D. Morris, K. Kanatani, and T. Kanade. Gauge fixing for accurate 3d estimation. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2:343, 2001. ISSN 1063-6919.

M.R.Garey and D. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Freeman, 1979.

E. Muybridge. *The human figure in motion : an electro-photographic investigation of consecutive phases of muscular actions*. Chapman, London, 1955.

A. Myronenko, X. Song, and M. Carreira-Perpinan. Non-rigid point set registration: Coherent point drift. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Neural Information Processig Systems (NIPS)*, pages 1009–1016. MIT Press, Cambridge, MA, 2007.

B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Neural Information Processig Systems (NIPS)*, pages 955–962. MIT Press, 2005.

B. Nadler, S. Lafon, R.Coifman, and I. Kevrekidis. Diffusion maps - a probabilistic interpretation for spectral embedding and clustering algorithms. *Lecture Notes in Computational Science and Engineering*, 58:238–260, 2007.

M. Neuhaus and H. Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852–1863, 2006.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processig Systems (NIPS)*, 2002.

M. Niethammer, M. Reuter, F.-E. Wolter, S. Bouix, N. Peinecke, M.-S. Koo, and M. Shenton. Global medical shape analysis using the laplace-beltrami spectrum. In *International Conference on Medical Image Computing and Computer Assisted Interventions*, pages 850–857. Springer, 2007.

J. Park, H. Zha, and R. Kasturi. Spectral clustering for robust motion segmentation. In *European Conference on Computer Vision (ECCV)*, pages 390–401, 2004.

M. Pavan and M. Pelillo. Dominant sets and hierarchical clustering. In *IEEE International Conference on Computer Vision (ICCV)*, pages 362–369, 2003.

U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, pages 15–36, 1993.

M. Polito and P. Perona. Grouping and dimensionality reduction by locally linear embedding. In *Neural Information Processig Systems (NIPS)*, pages 1255–1262. MIT Press, 2002.

J. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005.

J. Puzicha, J. M. Buhmann, Y. Rubner, and C. Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *IEEE International Conference on Computer Vision (ICCV)*, page 1165, Washington, DC, USA, 1999. IEEE Computer Society.

H. Qiu and E. Hancock. Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(11):1873–1890, November 2007.

G. Reeb. Sur les points singuliérs d'une forme de pfaff complètement intégrable ou d'une fonction numèrique. *Comptes Rendu Hebdomadaires des Séances de l'Académie des Sciences*, 222:847–849, 1946.

M. Reuter. Hierarchical shape segmentation and registration via topological features of laplace-beltrami eigenfunctions. *International Journal of Computer Vision (IJCV)*, 2009.

M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace-beltrami spectra as shape-dna of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

L. Reveret, L. Favreau, C. Depraz, and M. Cani. Morphable model of quadrupeds skeletons for animating 3d animals. In *ACM SIGGRAPH*, Los Angeles, USA, July 2005.

A. Robles-Kelly and E. R. Hancock. Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(3):365–378, 2005. ISSN 0162-8828.

G. Rosman, A. Bronstein, M. Bronstein, and R. Kimmel. Manifold analysis by topologically constrained isometric embedding. *International Journal of Applied Mathematics and Computer Sciences*, 1(3):117–123, 2004.

D. Ross, D. Tarlow, , and R. Zemel. Unsupervised learning of skeletons from motion. In *European Conference on Computer Vision (ECCV)*, 2008.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *IEEE International Conference on Computer Vision (ICCV)*, pages 59–66, 1998.

Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision (IJCV)*, 2000.

R. M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Eurographics Symposium on Geometry Processing (SGP)*, pages 225–233, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. ISBN 978-3-905673-46-3.

J. M. S. Belongie and J. Puzicha. Shape matching and object recognition using shape context. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(4):509–522, 2002.

L. K. Saul, S. T. Roweis, and Y. Singer. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. *Spectral methods for dimensionality reduction In O. Chapelle, B. Schoelkopf, and A. Zien (eds.), Semisupervised Learning*. MIT Press: Cambridge, MA., 2006.

C. Schellewald and C. Schnorr. Probabilistic subgraph matching based on convex relaxation. In *Lecture Notes in Computer Science*, volume 3757, pages 171–186, 2005.

D. C. Schmidt and L. E. Druffel. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *J. ACM*, 23(3):433–445, 1976. ISSN 0004-5411.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, A. Smola, L. Bottou, C. Burges, H. Bultho, K. Gegenfurtner, and P. H. Ner. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

B. Schölkopf, A. Smola, and K.Müller. Kernel principal component analysi. *Advances in Kernel Methods-Support Vector Learning*, pages 327–352, 1999.

S. Sclaroff and A. Pentl. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17:545–561, 1995.

S. Sclaroff and A. Pentland. A modal framework for correspondence and description. In *IEEE International Conference on Computer Vision (ICCV)*, pages 308–313, 1993.

S. Sclaroff and A. P. Pentland. Physically-based combinations of views: Representing rigid and nonrigid motion. In *In Proc. IEEE Workshop on Nonrigid and Articulated Motion*, pages 158–164. IEEE Press, 1994.

G. Scott and C. Longuet-Higgins. An algorithm for associating the features of two images. *Biological Sciences*, 244:21–26, 1991a.

G. Scott and C. Longuet-Higgins. An algorithm for associating the features of two images. *Biological Sciences*, 244:21–26, 1991b.

L. Shapiro and J.M.Brady. Feature-based correspondence an eigenvector approach. In *Image and Vision Computing*, volume 10, pages 283–288, 1992.

J. Shawe-taylor, R. Holloway, and C. K. I. Williams. The stability of kernel principal components analysis and its relation to the process eigenspectrum. In *Neural Information Processig Systems (NIPS)*. MIT Press, 2003.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22:888–905, 2000.

J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 1998.

G. Snedecor and W. Cochran. *Statistical Methods*. Iowa State University Press, 1989.

J. E. Solem and F. Kahl. Surface reconstruction from the projection of points, curves and contours. *International Symposium on 3D Data Processing, Visualization and Transmission*, 0:301–307, 2004. doi: http://doi.ieeecomputersociety.org/10.1109/TDPVT.2004.1335214.

O. Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4): 789–807, 2006a.

O. Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4): 789–807, 2006b.

O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. ISBN 978-3-905673-46-3.

J. Starck and A. Hilton. Spherical matching for temporal correspondence of non-rigid surfaces. In *IEEE International Conference on Computer Vision (ICCV)*. Starck, J. and Hilton, A., 2005.

J. Starck and A. Hilton. Correspondence labelling for wide-tiemframe free-form surface matching. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. In *SIGGRAPH*, pages 399–405, New York, NY, USA, 2004. ACM.

J. Sun, L. J. Guibas, and M. Ovsjanikov. Global intrinsic symmetries of shapes. In *Eurographics Symposium on Geometry Processing (SGP)*, 2008.

H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *IEEE International Conference on Shape Modeling and Applications (SMI)*, pages 30–139, 2003.

A. Sundaresan and R. Chellappa. Segmentation and probabilistic registration of articulated body models. In *International Conference on Pattern Recognition*, 2006.

A. Sundaresan and R. Chellappa. Model driven segmentation of articulating humans in laplacian eigenspace. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(10): 1771–1785, 2008.

J. sbastien Franco and E. Boyer. Exact polyhedral visual hulls. In *British Machine Vision Conference (BMVC)*, pages 329–338, 2003.

G. Taubin. A signal processing approach to fair surface design. In *ACM SIGGRAPH*, pages 351–358, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: http://doi.acm.org/10.1145/218380.218473.

P. Torr. Solving markov random fields using semi definite programming. In *Artificial Intelligence and Statistics*, 2003.

B. Triggs. Detecting keypoints with stable position, orientation and scale under illumination changes. In *European Conference on Computer Vision (ECCV)*, 1999.

Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. In *European Conference on Computer Vision (ECCV)*, volume 3023/2004, pages 558–569, 2004.

O. Tuzel, R. Subbarao, and P. Meer. Simultaneous multiple 3d motion estimation via mode finding on lie groups. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976. ISSN 0004-5411.

S. Umeyama. An eigen decomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 10:695–703, 1988.

B. Vallet and B. Lévy. Manifold harmonics. *Computer Graphics Forum (Proceedings Eurographics)*, 2008.

B. J. van Wyk and M. A. van Wyk. Kronecker product graph matching. *Pattern Recognition*, 36 (9):2019–2030, 2003.

S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three dimensional scene flow. In *IEEE International Conference on Computer Vision (ICCV)*, 1999.

S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2005.

D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical Report uw-cse-03-05-01, University of Washington, 2003.

R. Vidal, Y. Ma, S. Soatto, and S. Sastry. Two-view multibody structure from motion. *International Journal of Computer Vision (IJCV)*, 2005.

U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In *Neural Information Processig Systems (NIPS)*, volume 17, pages 857–864. MIT Press, 2005.

H. Wang and P. Culverhouse. Robust motion segmentation by spectral clustering. In *British Machine Vision Conference (BMVC)*, 2003.

H. F. Wang and E. R. Hancock. Correspondence matching using kernel principal components analysis and label consistency constraints. *Pattern Recogn.*, 39(6):1012–1025, 2006.

M. Wardetzky, S. Mathur, F. Klberer, and E. Grinspun. Discrete laplace operators: No free lunch. In *Eurographics Symposium on Geometry Processing (SGP)*, pages 33–37, 2007.

Y. Weiss. Segmentation using eigenvectors: a unifying view. In *IEEE International Conference on Computer Vision (ICCV)*, pages 975–982, 1999.

C. K. I. Williams. On a connection between kernel pca and metric multidimensional scaling. In *Neural Information Processig Systems (NIPS)*, pages 675–681. MIT Press, 2001.

C. K. I. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the twenty-first international conference on Machine learning (ICML)*, pages 1159–1166. Morgan Kaufmann, 2000.

J. Yan and M. Pollefeys. A factorization-based approach to articulated motion recovery. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 1203, June 2005.

J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision (ECCV)*, 2006.

R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

R. Zayer, C. Rssl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. In *Eurographics*, 2005.

L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Neural Information Processig Systems (NIPS)*, 2005.

J. Zhang, K. Siddiqi, D. Macrini, A. Shokouf, and S. Dickinson. Retrieving articulated 3-d models using medial surfaces and their graph spectra. In *Proc. of Int. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 285–300, 2005.

Y. Zhang and C. Kambhamettu. Integrated 3-d scene flow and structure recovery from multiview image sequences. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2000.

Z. Zhang and M. I. Jordan. Multiway spectral clustering: A maximum margin perspective. *Statistical Science*, 23:383–403, 2008.

Y. Zheng and D. Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):643, 2006. ISSN 0162-8828.

K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph.*, 24(3):496–503, 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073219.

M. Ziegler and V. Brattka. A computable spectral theorem. In *CCA '00: Selected Papers from the 4th International Workshop on Computability and Complexity in Analysis*, pages 378–388, London, UK, 2001. Springer-Verlag. ISBN 3-540-42197-1.