UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

LOCAL FEATURES FOR SHAPE MATCHING AND RETRIEVAL

TESIS PARA OPTAR AL GRADO DE DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

IVÁN ANSELMO SIPIRÁN MENDOZA

PROFESOR GUÍA:
BENJAMÍN BUSTOS CÁRDENAS

MIEMBROS DE LA COMISIÓN:
NANCY HITSCHFELD KAHLER
MARÍA CECILIA RIVARA ZÚÑIGA
MICHAEL BRONSTEIN

Este trabajo ha sido parcialmente financiado por CONICYT (Chile) a través de una Beca Doctoral y por el proyecto FONDECYT (Chile) Nr. 1110111.

SANTIAGO DE CHILE
ENERO 2014

Resumen

Los modelos tridimensionales son útiles para representar objetos reales en el mundo digital. Su uso se encuentra en muchas aplicaciones tales como medicina, ingeniería, seguridad y otros. Recientemente, la introducción de dispositivos de captura baratos ha incrementado el interés por este tipo de información, generando una gran cantidad de modelos disponibles en diferentes lugares. Por lo tanto, es imperativo proveer algorithmos efectivos y eficientes para procesar y analizar datos 3D.

La evaluación de similitud de modelos 3D es una tarea importante que puede ser útil para procesos de alto nivel tales como recuperación por contenido y reconocimiento. Estos procesos requieren atención ya que los repositorios 3D están en constante crecimiento y es cada vez más necesario realizar búsquedas basadas en contenido. Más aún, la comparación de modelos 3D es una tarea desafiante debido a la dificultad de representar adecuadamente un modelo 3D para una subsecuente comparación.

En esta tesis, abordamos el problema de matching y recuperación por contenido en modelos 3D. Nosotros proponemos nuevas representaciones para modelos 3D basándonos en la habilidad de detectar características locales robustas sobre mallas. De esta forma, este trabajo se enfoca primero en la detección efectiva y eficiente de puntos de interés en mallas 3D. Luego, nosotros proveemos representaciones efectivas para lidiar con problemas como recuperación genérica, recuperación de formas no rígidas y correspondencias entre mallas. Para la recuperación basada en contenido, nosotros desarrollamos representaciones basadas en grupos de puntos de interés. Estas representaciones permiten aprovechar el poder de representación de las características locales mientras reducimos la cantidad de información necesaria. Para computar correspondencias, desarrollamos una representación jerárquica la cual conlleva una descomposición recursiva de un modelo 3D en regiones y puntos de interés. Esta representación es útil para decrementar el error de localización de las correspondencias en formas no rígidas mientras se reduce considerablemente el tiempo de procesamiento.

Nuestros experimentos muestran que nuestros métodos para detectar estructuras locales en mallas son robustos a diferentes transformaciones. Además, nosotros presentamos una detallada evaluación de nuestras representaciones para recuperación genérica por contenido, recuperación de formas no rígidas y correspondencias en mallas. Desde nuestros resultados, es posible concluir que el uso de características locales puede mejorar el proceso de evaluar la similitud entre modelos 3D. Y más aún, nuestras representaciones pueden ayudar a mejorar simultáneamente la efectividad y la eficiencia de la recuperación de modelos 3D basada en contenido y de la búsqueda de correspondencias.

Abstract

Three-dimensional shapes are useful to represent real objects in the digital world. Their use encompasses a wealth of applications in diverse fields such as medicine, engineering, security, and so on. Recently, the introduction of cheap scanning devices has increased the interest for this kind of information, generating a large amount of models available in several sources. It is therefore imperative to provide effective and efficient algorithms for processing and analyzing 3D data.

The evaluation of similarity of 3D shapes is a basic and important task which can be useful for high-level processes such as retrieval and recognition. These processes need attention as 3D repositories are constantly growing and it is increasingly necessary to provide content-based searches. Moreover, the comparison of 3D models is a challenging task due to the difficulty of adequately represent a shape for a proper comparison.

In this thesis, we address the problem of shape matching and retrieval. We propose new representations for 3D shapes based on the ability of detecting robust local features on meshes. In this way, our work is first focused on the effective and efficient detection of keypoints in 3D shapes. Then, we provide effective shape representations to tackle problems such as generic shape retrieval, non-rigid shape retrieval, and shape matching. For shape retrieval, we develop representations based on groups of keypoints. These representations allow us to take advantage of the representational power of local features while reducing the amount of information needed for the representation. For shape matching, we develop a hierarchical representation which contains a recursive decomposition of a 3D shape in regions and keypoints. This representation is useful to decrease the localization error of correspondences in non-rigid shapes while reducing considerably the processing time.

Our experiments show that our methods to detect local structures in meshes are robust to several transformations. In addition, we present a comprehensive evaluation of our representations for generic shape retrieval, non-rigid shape retrieval and shape matching. From our results, it is possible to conclude that the use of local features can enhance the process of assessing the similarity between 3D shapes. And moreover, our representations may help to improve both the effectiveness and efficiency of 3D shape retrieval and matching.

To my beloved wife, who gives me the inspiration every day.

Acknowledgements

I would like to offer my thanks to all the people who contributed in the realization of this thesis and supported me during the last five years.

I would like to thank to Benjamin Bustos, my thesis advidor, for his continuous guidance during the development of the thesis. He always encouraged me to go ahead and taught me that research done with passion is doubly rewarding. Undoubtedly, this thesis would not have been possible without his advise.

I also would like to thank to the DCC community: professors, administrative staff and colleagues. In particular, I would like to express my gratitude to Angelica Aguirre, Sandra Gaez, Prof. Claudio Gutiérrez, and Prof. Eric Tanter for all the priceless help I received.

I want to give thanks to my friends Teresa Bracamonte, Daniel Moreno, Violeta Chang, José Saavedra, Alcides Quispe and Carlos Bedregal. They made my life easier with their friendship. They also helped me to deal with being away from home and my family. They were my family for five years.

I must offer my thanks to my family: my father Anselmo, my sister María Ynés and her family and my brother José Antonio and his family. Their love and constant support was very important to conclude this work. The biggest acknowledgement is to my mother, Susana. Although she is not in this world, her love and words still rule my steps. Also, I want to thank to my wife's family. In particular, I would like to thank to Rebeca Alfaro, Vicente Alfaro and Jesús Cárdenas for all the love they gave me since I met them.

Last but not least, a special acknowledgement goes for my beloved wife, Analí Alfaro. She is my strength and my inspiration everyday of my life. With her comprehension, she has helped me to create the perfect environment in which I enjoy doing research and being the person I want to be.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Three-dimensional shapes are a suitable representation for real objects in several situations. For example, they are used in fields such as medicine, engineering, archeology, and so on. In recent years, we have witnessed increasing interest in computer graphics and computer vision communities for 3D shape retrieval and matching. As a result, a number of techniques and approaches have been proposed for shape representation, object and feature description, feature selection, and matching algorithms. It is also important to note the number of applications that have emerged, such as:

- Geometric 3D comparison for shoe industry [97].
- 3D hippocampi retrieval [67].
- Classification of pollen in 3D volume data sets [116].
- 3D retrieval for museums [49].
- Human ear recognition in 3D [31].
- 3D face recognition [64, 34, 18, 109].
- 3D object classification for craniofacial research [3].
- 3D protein retrieval and classification [151, 107, 108].
- CAD/CAM [152].
- Archeology [59].
- 3D video sequences [57].

More recently, with the introduction of cheap 3D devices, the availability of 3D information in real applications has considerably increased. One of the most crucial problems is how to assess the similarity between two 3D shapes. This problem is challenging because the only available information is the shape. Therefore, it is necessary to propose approaches to find suitable representations for 3D data in order to do an effective comparison.

In this thesis, we address the problem of shape matching and retrieval. Our approach is entirely based on the detection of robust local features. Our premise states that local features convey enough information to characterize a shape, and at the same time it allows us to tackle several problems such as shape perturbations and non-rigid transformations. Furthermore, the use of local features increase the amount of information, so it is also important to propose efficient solutions to the overall problem.

Our work is based on two key aspects. First, the cornerstone of our research work is the robust and efficient detection of keypoints on meshes. Indeed, we propose the Harris 3D method [128] which has proven to be effective in presence of many shape perturbations and transformations [14, 13]. Second, our approaches for matching and retrieval are based on organizing keypoints in high-level structures. We propose two strategies for organizing keypoints: grouping and hierarchies.

The former is based on the observation that Harris 3D keypoints have a similar distribution on the surface of similar shapes. For this reason, we propose to group keypoints according to several criteria depending on the application:

- **Detection of coarser-level local structures.** We propose to perform a geodesic clustering of keypoints in order to find regions with a high agglomeration. We call *key-components* to the detected regions [129, 131]. Interestingly, the key-components exhibit a high overlap in presence of transformations.
- **Data-aware partitioning.** We propose to perform a spatial clustering of keypoints in generic shapes. As a result, we can extract mesh partitions and use them to represent a shape [132]. These representation are used to evaluate the similarity of two shapes through an optimization approach. The data-aware partitioning has been useful to improve the effectiveness of generic shape retrieval.
- **Discriminative signatures for non-rigid shapes.** We propose to use the local features to construct signatures for non-rigid shapes [126]. These signatures allow us to reduce the amount of information, without compromising the representational power of the features. Our signatures, in combination with the Signature Quadratic Form Distance, improve the effectiveness and efficiency of non-rigid shape retrieval in a large-scale scenario.

The latter strategy is based on the ability of decomposing a shape into robust regions. We propose a new representation for non-rigid shapes: *the decomposition tree* [130]. This structure maintains a hierarchy of local structures where the root node is the entire shape and the leaf nodes contains keypoints. The tree structure allows us to guide the matching process by levels, generating correspondences hypotheses and reducing the search space early in the process. In fact, we propose a hierarchical matching algorithm which takes advantage of the tree structure to perform an efficient search of correspondences in non-rigid shapes.

In this thesis, we present a careful description of our proposals to tackle the problem of shape matching and retrieval. In addition, we validate our ideas with a comprehensive experimentation in every aspect of our approaches.

1.1 Thesis Outline

This thesis is organized as follows:

- Chapter 2 presents the related work of our research. The presentation of the literature is divided taking into account the three major topics of this work. Section 2.1 is devoted to present the state of the art in local features detection. Section 2.2 describes the previous works in shape retrieval. Also, we provide a detailed description of methods for generic shape retrieval and non-rigid shape retrieval. Finally, Section 2.3 presents a description of previous approaches for shape matching.
- Chapter 3 describes the background of our work. This chapter is mainly focused on the introduction of basic concepts which are important to fully understand the rest of the thesis and make it self-contained. Section 3.1 is dedicated to the detailed presentation of state-of-the-art descriptors, which will be used in several parts of the thesis. Section 3.2 describes the datasets used for evaluating our methods. Finally, Section 3.3 presents the background on indexing which will be useful for improving the efficiency of a retrieval system.
- Chapter 4 concentrates on our proposals for effective and efficient detection of local structures on 3D shapes. We present two approaches to detect local features in different scales. Section 4.1 describes the *Harris 3D* algorithm to detect keypoints on 3D meshes. Our experiments show that our algorithm is robust against strong transformations such as noise and holes. More interestingly, the detected keypoints exhibit a distinctive distribution on shapes of the same class, so they are useful for detecting local structures in a coarser level. Section 4.2 presents our approach to detect repeatable components on 3D meshes: *the key-components*. Our experiments show that our key-components exhibit a high overlap in presence of transformations. In both cases, our proposals outperform the state of the art.
- Chapter 5 focuses on the effective use of local features in generic shape retrieval. In this chapter, we propose a *data-aware partitioning algorithm* to decompose a shape into parts. The partitioning algorithm is guided by the spatial concentration of keypoints which allow us to divide an object in a data-driven manner. Subsequently, we formulate the problem of assessing the similarity between two shapes as a optimization problem. Our experiments show that our partitioning scheme has a positive impact in the retrieval performance.
- Chapter 6 presents a new approach for characterizing a non-rigid shape from its local information. In this chapter, we propose three approaches to compute signatures to represent a shape. The combination of these signatures with the Signature Quadratic Form Distance allows us to make contributions in the effectiveness and the efficiency of a retrieval system. Our results show that our approach outperforms the state of the art in large-scale non-rigid shape retrieval.
- Chapter 7 addresses the problem of shape matching. We propose a new representation for non-rigid shapes called the *decomposition tree*. This structure contains a hierarchical decomposition of a shape where the keypoints are stored in the lowest level of the hierarchy. Thus, we propose a hierarchical matching algorithm to compute a correspondence set between two shapes represented by their decomposition trees. Our results show that our representation is robust and efficient. Our matching algorithm

improves the state of the art in two important aspects: it reduces the localization error of correspondences and it considerably reduces the matching time.

- Chapter 8 draws the conclusions of our research work and delineates the future research directions.

1.2 Thesis Publications

The ideas presented in this thesis have been published in the following papers and related publications:

Book Chapters

- [27] This book chapter contains an extensive description of the state of the art of shape matching and provides examples of well-established techniques and new approaches for shape retrieval and recognition. In this chapter, we describe open problems and future trends, which were useful to formulate the ideas of this thesis.

Journals

- [128] This paper presents the Harris 3D algorithm to detect interest points on meshes. The article presents a detailed description of our method. In addition, we deeply explore the effect of the algorithm parameters in the performance using the repeatability of keypoints as evaluation criterion. The method and the experimentation of this research are presented in Sec. 4.1.
- [131] This is a journal version of our workshop article about the detection of the salient regions. This article deeply explores the impact of distinctiveness and size in the definition of regions of interest. In this thesis, the content of the journal article can be seen in Sec. 4.2.
- [132] This paper describes our data-aware partitioning algorithm for generic shape retrieval. In this article, we propose to decompose a shape according to spatial groups of keypoints. Also, we propose to use a optimization formulation to assess the similarity between two shapes. Our experiments show that our new representation can enhance the retrieval of shapes. This method is presented in Chapter 5.
- [80] This paper is an extension of a joint report presented in the Shape Retrieval Contest (SHREC'2011). This paper present a rigorous comparison of methods for non-rigid shape retrieval. In this article, we use a very preliminary approach to tackle the problem of non-rigid shape retrieval using local features.

Conference Proceedings

- [127] This paper presents a preliminary study of the Harris 3D method, which was extended in the journal article [128].
- [14] This paper contains a joint report with evaluations of algorithm for local features detection in the context of the Shape Retrieval Contests (SHREC). In this article, our Harris 3D method is evaluated along with state-of-the-art techniques. In this thesis, part of these evaluations are useful to compare Harris 3D with related works in Sec. 4.1.
- [13] This paper also contains a joint report (SHREC) which evaluates local feature detection methods. The dataset used in this report is different to that presented in [14].
- [78] This paper is a joint report (SHREC) containing the evaluation of non-rigid shape retrieval methods. In this paper, we present a preliminary study about the use of local features for non-rigid shape retrieval.
- [125] This paper describes our thesis proposal. The article contains our general ideas about how to address the problems in our research work. The paper received the Best Paper Award in the Doctoral Symposium of the ACM Multimedia Conference 2011.
- [129] This paper delineates a method to detect salient regions on 3D meshes based on the distribution of keypoints. This article presents a preliminary study, which has been complemented in a journal article under review [131]. In this thesis, the method to detect salient regions and a complete experimentation are presented in Sec. 4.2.
- [10] This paper is a joint report (SHREC) in the context of retrieval of abstract shapes. This article present our first experience in the use of the Signature Quadratic Form Distance for shape retrieval.
- [130] This paper presents a new hierarchical representation for non-rigid shapes and a hierarchical matching algorithm to compute point correspondences. The work presented in this paper is the basis of the Chapter 7 in this thesis.

Unpublished Works

- [126] This manuscript describes our approach for non-rigid shape retrieval using the Signature Quadratic Form Distance. In this paper, we propose new characterizations of shapes through signatures extracted from local features. The content and experimentation of this work are presented in Chapter 6.

Chapter 2

Literature Review

This chapter is devoted to present the related work of our research. We divide the presentation of the literature according to the three major subjects of this thesis: local features (Sec. 2.1), shape retrieval (2.2) and shape matching (2.3).

2.1 Local Features

2.1.1 Keypoint Detection

The interest point detection topic emerged in the computer vision community with the aim of reducing the amount of information used in high-level vision tasks. A pioneering work was presented by Harris and Stephens [50], which was the basis for many later works. For readers interested on interest points detectors on images, we recommend the evaluation paper presented by Schmid et al. [119], which contains detailed descriptions and performance evaluation of several proposed methods.

For 3D meshes, several approaches have been proposed, most of which have tried to extend the detectors proposed for images. After the SIFT method proposed by Lowe [86], a number of extensions have been presented which use Difference-of-Gaussians(DoG) as interest point detector. Castellani et al. [28] applied the DoG detector over vertices in scale-space obtained with successive decimations of the original shape. Vertices with high response in its DoG operator are selected as interest points. In the same way, Zou et al. [159] proposed to build a geodesic scale-space, and subsequently to apply DoG detector on that space for detecting interest points on a surface. Also, Zaharescu et al. [154] assumed that the vertices of an 3D object have associated information such as curvature or photometric properties. Defining a discrete Difference-of-Gaussians operator, the authors applied this operator on the function defined by the associated information over a manifold. This approach showed good results in matching of 3D models sequences.

On the other hand, the geometric diffusion theory can be used for detecting interest points on surfaces. The diffusion process reveals interesting characteristics from the intrinsic geometry of a surface which can be exploited to detect outstanding structures. As a 3D surface property related to the diffusion process on a manifold, the Laplace-Beltrami operator has been also used to detect interest points. Hu and Hua [55] defined the geometric energy of a vertex as function of the eigenvalues and eigenvectors of the Laplace-Beltrami spectrum of a given object. Vertices where the energy is a maximum are considered as interest points. In addition, the energy provides the scale where the selected vertices are interesting. The selected interest points were used in a matching task with promising results. On the other hand, Sun et al. [136] defined the Heat Kernel Signature as a temporal domain restriction of the Heat Kernel on a manifold, which is related to the Laplace-Beltrami spectrum. In 3D meshes, each vertex has an associated signature. A vertex is selected as interest point, when for large time values, its signature has a maximum with respect to the neighbor vertices.

Similarly, Zou et al. [160] proposed to build a scale space of the surface geometry via the surface Ricci flow which satisfies a set of desired properties for a multi-scale representation. The authors applied the Ricci flow over a metric of the surface based on edge lengths. The scale space is represented as a matrix with curvature values calculated from the set of diffused metrics. Then, the Laplacian of a vertex is computed using the cotangent schema using the curvature as associated values. A vertex is considered as an interest point if it is an extreme of the Laplacian in the 1-ring neighborhood and neighboring scales.

Also, curvature-based methods have been proposed. Gelfand et al. [44] described an interest point detector based on a new descriptor called the integral volume descriptor. For each vertex, the amount of volume in the intersection of a ball centered in the vertex and the 3D object describes an interesting local measure. The authors showed that this quantity is closely related to the curvature in the vertex. Vertices with uncommon integral values are selected as interest points. Also using curvature in vertices, Ho and Gibbins [53] suggested a measure called the curvedness measure in order to describe the geometric information in a vertex. The curvedness is calculated from the principal curvatures of a vertex. This measure can be calculated in different scales by selecting different neighborhood sizes which are used to fit quadratic patches over which curvatures are computed. Vertices with extremal values in its curvedness, with respect to neighboring points and scales, are selected as interest points.

Differently, Liu et al. [84] proposed a Monte-Carlo strategy to select a random set of points on a surface with each point having the same probability to be chosen. These points were used in partial shape retrieval. The assumption behind this proposal is that the vertices of a shape are samples of the original surface and the tasks that use them can be affected by shape tessellations. Similarly, Shilane and Funkhouser [124] considered random points on a 3D surface, selecting only those points that contribute to improve the retrieval performance. With a training phase, it was possible to assign a predicted distinction value to each selected point in the 3D collection and thus, using that values to assign new ones to points of a new shape.

As another approach, the mesh saliency defined by Lee [75] has proven to be a robust feature to many 3D applications. The process to compute the mesh saliency of a 3D object begins calculating a Gaussian-weighted average of the mean curvature on a surface. Each vertex in an object is thus associated to the difference of such average in different scales, which is the saliency of that vertex. Vertex with the highest saliency can be considered as interest points.

Conformal parameterization has also been used to propose interest points detectors. Methods based on conformal parameterization [56, 96] transform a 3D surface into a 2D parameterization that can be seen as an image. A 3D to 2D mapping is said to be conformal if angles are preserved. Once an image is computed, interest points can be detected on it, and subsequently these are mapped back to the 3D domain.

Finally, Mian et al. [92] related the repeatability of keypoints (extracted from partial views of an object) with a quality measure based upon principal curvatures. More recently, Creusot et al. [33] proposed a machine learning approach for the detection of keypoint on 3D face scans.

Also, several efforts have been done to evaluate the performance of interest point detection methods in different aspects such as repeatability [14, 144, 13] and localization error with respect to human landmarks [37]. Similarly, Yu et al. [153] evaluated the performance of volumetric 3D interest point detectors.

2.1.2 Component Detection

Mesh decomposition is an important analysis tool with applications in computer vision and graphics. The idea is to partition a given mesh in components or regions which can be used in applications. Although there are a lot of approaches for mesh segmentation, we are interested in those methods driven by local features. For a comprehensive study about mesh segmentation techniques, we recommend the survey by Shamir [121].

One of the earliest techniques for feature-driven mesh decomposition was presented by Mortara et al. [95]. This method decomposes a triangular mesh based on a characterization of a vertex using its local curvature. It analyzes the evolution of the curve formed by the intersection of the mesh with a set of spheres with increasing radii. The number of connected components of the curve and the local properties (curvature and length ratio) define a classification for each vertex, which is used to group vertices with similar features. Differently, Huang et al. [60] proposed to decompose a shape based on a modal analysis. Taking the eigen-decomposition of the Hessian of an energy function defined on the mesh, it is possible to define the set of typical deformations of a mesh. Therefore, this method is able to estimate the parts that tend to be rigid and subsequently segment them.

Local features have also been used for mesh segmentation. Agathos et al. [1] propose a mesh segmentation method based on interest points. Given a mesh, the algorithm computes a protrusion function for each vertex, which is defined as the sum of geodesic distances to all vertex on the mesh. Thus, a vertex is selected as interest point if the value of its protrusion function is greater than the mean of geodesic distances between each pair of vertices. The interest points are grouped in order to avoid regions with many interest points. Each interest point is used as seed for computing the mesh segments. Similarly, Katz et al. [65] computed a 3D embedding for a shape and subsequently, the convex hull of the embedding was calculated.

The vertices of the convex hull were considered as keypoints, over which the method computed a set of core components.

Also, Gal and Cohen-Or [41] proposed to represent a 3D object as a set of salient geometric features. Their scheme entirely relies on curvature information over the shape's surface. This method starts by computing the curvature on a set of sampled points. Next, points are sorted according to their curvature values. The algorithm takes points with high curvature and performs a grouping of neighbor points until a good quadratic fitting surface can be found that approximates better the neighborhood. Subsequently, a region-growing algorithm clusters the mesh by adding points to the initial segments according to an empirical measure which involves area, curvature and similarity. The final clusters are called salient geometric features which are used in shape matching.

On the other hand, Hu and Hua [55] proposed to find keypoints using the eigen-decomposition of the Laplace-Beltrami operator of a shape. Each keypoint has a scale which is used to define a local patch, so a mesh is represented as a set of local patches product of the keypoint-based decomposition. After describing each local patch with its Laplace-Beltrami spectrum, they are used in a matching algorithm. On the other hand, Toldo et al. [143] applied a segmentation based on local properties of the mesh, specifically a shape index computed from the principal curvature values. Each segment is described with a histogram of local properties. Finally, a bag of features approach is used for describe the entire shape in order to be used in shape retrieval. Differently, Shapira et al. [122] performed a hierarchical segmentation using a shape diameter function (SDF). Subsequently, each segment is described using several local features such as a normalized histogram of SDF, shape distribution signatures and conformal geometry signatures. The signatures were used in matching and retrieval.

More recently, the decomposition of meshes from spectral functions defined on the surface has been introduced. The idea of these techniques is to take advantage of intrinsic information to define a function on vertices, edges or faces. Subsequently, the defined function is used by a grouping algorithm which provides a segmentation. For instance, the Heat Mapping approach [40] defines a vertex signature which can be interpreted as the average temperature on the surface by applying heat on a vertex. Then, a segmentation process using the k-means algorithm is driven from points with the highest value of heat affinity. Similarly, the Center-Shift method [137] proposes the evaluation of the biharmonic kernel in a point as a vertex function. Next, the algorithm finds a set of termination points which are vertices with maximal weighted mean of the defined function evaluated in local neighborhoods. Finally, a segment refinement is applied to provide the final segmentation.

Likewise, Skraba et al. [134] proposed to assign to each vertex its heat kernel signature evaluated in a fixed time. With these values, the technique applies a persistence-based clustering. This clustering considers to track regions associated with local maxima of the function. On the other hand, Aubry et al. [5] computed a n-dimensional function for each vertex. Each vertex was represented by its wave kernel signature. Next, every descriptor from a training set of shapes is collected in a large n-dimensional point cloud. This points are clustered by a Gaussian mixture algorithm where points in the same cluster correspond to the same mesh segment. This clusters are used for assigning a label to each vertex of a new shape.

In the same direction, the extension of methods from image processing and computer vision has been studied. For instance, Digne et al. [35] extend the maximally stable extremal regions (MSER) to shape decomposition. The method used the concept of vertex-weighted component trees applied to meshes. To accomplish this goal, it was necessary to use the mean curvature as function defined over the mesh. Similarly, Litman et al. [83] also used the MSER framework to detect stable components on meshes. The authors proposed an approach based on diffusion geometry. The algorithm considers the shape as a graph and associates weights to vertices and edges according to the evaluation of a local property (the heat kernel) between vertices and edges. A very interesting work that also uses concepts from image processing is the Variational Mesh Decomposition [157]. It is based on the well-studied Mumford-Shah functional which is common in image segmentation literature. This technique proposes a convex version of the functional which is applied on a face-based multichannel function on the surface. The function is defined from the eigenvectors of a Laplacian matrix defined on dihedral angles for edges.

2.2 Shape Retrieval

2.2.1 Generic Shape Retrieval

The interest in 3D model retrieval has resulted in a large amount of proposed techniques to overcome the problem. One of the most studied approaches is to convert a 3D model into a more convenient representation for comparison, for example feature vectors. Then, the comparison can be done by defining a distance between those representations. For generic shape retrieval, this approach has received attention due to the efficiency of computing distances between vectors. In this section, we provide a brief description of the state of the art related to descriptors for generic shape retrieval and possible combinations to improve the performance. For a comprehensive study, surveys by Bustos et al. [25] and, Tangelder and Veltkamp [140] are an excellent resource.

Classic methods for 3D shape retrieval can be classified in three groups: view-based, histogram-based, and transform-based. This classification is based on how a feature is extracted from the shape. View-based methods transform a 3D shape into a set of 2D views and subsequently we can apply image techniques to describe the obtained views. For example, the Depth Buffer method [146] computes six views corresponding to the six faces of the bounding cube of an object. Each view stores the projected distances from the object to the projection plane. Then, each view is represented by Fourier coefficients and the final vector is the concatenation of the six obtained views. Another example is the PANORAMA descriptor [106], which computes three views taken from the lateral faces of cylinders oriented according with the coordinate axes. Similar to the Depth Buffer, each lateral face encodes the distance from the object to the face. Then, Fourier and Wavelets coefficients are extracted from each view, which form the final descriptor.

Histogram-based methods summarize shape properties in order to use them as features. For instance, Shape Distributions [99] is a method that computes several geometric properties (distances between pairs of surface points, angles between three random surface points, etc). The method consists of sampling a large amount of points on the shape surface and subsequently measuring some property. Each value obtained for the chosen property is accumulated in a histogram. Thus, the histogram represents an approximation of the distribution of the property and it is expected to be distinctive for each object.

Transform-based methods consist of converting the geometric information by using some mathematical transformation prior to the feature extraction. The goal of applying a transformation is to enhance some information which is not evident in the Euclidean space. In particular, in 3D model retrieval, there is an interest for spherical harmonics to extract features from shapes. Vranic proposed the ray-based descriptor [146] by using a spherical function which is able to capture the behavior of the rays starting in the origin and the intersections with the shape. Similarly, Kazhdan et al. [66] used spherical harmonics frequencies along with the Gaussian Euclidean Distance Transform in a volume representation of a shape.

An interesting and new approach is the combination of different descriptors to improve the performance of individual descriptors. The basic idea is that different descriptors could extract complementary features and their combination could lead to improvements. Bustos et al. [24] proposed to dynamically combine several descriptors using a weighting scheme dependent on the query. Similarly, Vranic [147] proposed to combine three descriptors: Silhouette, Ray-based and Depth-Buffer. This combination (which was called DESIRE) improved the performance of the individual descriptors. On the other hand, Papadakis et al. [105] suggested combining 2D and 3D features to improve the performance of retrieval. As a 2D feature, the authors proposed to use the Depth Buffer method and as 3D feature, they proposed to use spherical harmonics transform for spherical functions obtained from the shape.

The aforementioned combination methods consist of somehow combining two or more description methods. However, these techniques still rely on the global shape for the calculation of each descriptor. Recent approaches have considered the combination of global information and part-based information. Li and Johan [77] used global an local radial distances to describe a shape. First, the method computes a radial distance descriptor by uniformly dividing the surface of a sphere containing the object. The division considers bins at different angle intervals and the average distance of each bin to the object is stored on it. Second, the local component of the method consists of uniformly dividing the bounding cube of the object into $N \times N \times N$ cells. For each vertex on the shape, the method computes the minimum distance to the cell centers and the distance is assigned to the vertex. Finally, thirteen views are extracted using the assigned distances as RGB values. The distance between two shapes with this representation is measured pair-wise between global and local descriptions.

Also, Bustos et al. [26] proposed a simple partitioning scheme in order to combine it with global descriptors. Given a shape, the method computes a global descriptor for it. Next, the shape is divided in eight parts according to the eight octants obtained with the coordinate axes in the 3D Euclidean space. Finally, the method computes a descriptor for each part. To measure the distance of global-partial representations, the authors evaluated several weighting schemata where adaptive weighting showed the best performances. Similarly, Schreck et al. [120] also took the octant partition as a basis. Nevertheless, this new technique considered the matching of the parts as a bipartite graph matching problem. In addition, the authors tested the use of different numbers of parts. Interestingly, it was shown that not using all parts (6 or 7 depending on the dataset) outperformed the retrieval performance.

Regarding the use of local features to decompose a 3D shape, several approaches have been proposed for non-rigid and partial shape retrieval. Toldo et al. [143] proposed to apply a spectral clustering to decompose a mesh into regions. Each region was further described with information such as the shape index, radial geodesic distances and normal directions. The final representation was obtained using a multi-level bag-of-features approach. On the other hand, Shapira et al. [122] presented a technique for describing mesh segments. The segmentation is hierarchically performed using SDF histograms [123]. Next, contextual information is used in order to improve the matching between parts. A bipartite graph is used to measure the context-aware distance between two objects. In addition, mesh decomposition is recently being used as an alternative to 3D shape matching and retrieval.

A new interesting trend is the exploration of new similarity criteria. In the previously discussed works, the similarity was guided by the visual aspect. However, functionality and part compatibility are also being considered for performing similarity search. Kim et al. [69] explored the application of fuzzy correspondences to associate a given region of interest with shapes in a large collections. Also, the use of template-based matching [68, 102] have proven to be effective in the exploration of large collections of 3D shapes. Moreover, the use of functionality as similarity criterion has allowed the introduction of methods for the example-based synthesis of new shapes from large collections [158].

2.2.2 Non-rigid Shape Retrieval

In general, existing methods for non-rigid 3D shape retrieval can be roughly classified into algorithms employing local features, topological structures, isometry-invariant global geometric properties, direct shape matching, or canonical forms. The first solution is to measure the dissimilarity between two models based on their local features that are insensitive to isometric transformations. For instance, the well-known Spin Images [62] was utilized in [84], where they described a 3D object as a word histogram by the vector quantization of all local features (Spin Images) extracted on the surface. Ovsjanikov et al. [101] made use of the Heat Kernel Signature (HKS) [136], which is based on the properties of the heat diffusion process on a 3D shape, and designed a spatially-sensitive bag-of-features approach to retrieve non-rigid models in large databases. Ohbuchi et al. [98] proposed a view-based method using salient local features (SIFT [86]). They represented a whole object as a histogram by using bag-of-features for 2D salient local descriptors extracted from a set of depth-buffer views captured uniformly around the object. More recently, Wang et al. [149] presented Intrinsic Spin Images (ISIs) by generalizing the traditional Spin Images from 3D space to N- dimensional intrinsic shape space, in which their ISIs shape descriptors are calculated on MDS embedding representations of original 3D surfaces.

The second solution is to use topological structures to compare deformable 3D objects. For example, Hilaga et al. [52] developed the Topology Matching technique to compute the similarity between two models via the shape matching of their Multiresolutional Reeb

Graphs (MRGs), while Sundar et al. [138] compared 3D objects by applying graph matching techniques to match their skeletons. Better retrieval performance can be obtained [139] by using topological and geometric features together.

For the third category, isometric-invariant global geometric properties (e.g., geodesic distance) are utilized for non-rigid 3D shape retrieval. Reuter et al. [112] suggested using the Laplace-Beltrami spectra of the models, while Jain and Zhang [61] proposed to use eigenvalues of the geodesic distance matrix of a 3D object to generate 3D shape descriptors that are isometry-invariant. Also, Mahmoudi and Sapiro [87] designed six such signatures based on the distributions of intrinsic distances including diffusion distance, geodesic distance, a curvature weighted distance, etc.

Many investigations have also been made trying to measure the exact dissimilarity between non-rigid 3D models. For instance, Mémoli and Sapiro [90] introduced a theoretical framework to directly compare non-rigid 3D shapes based on the Gromov-Hausdorff (GH) distance. Since calculating the exact value of the GH distance is computationally expensive, Mémoli [89] proposed to approximate the GH distance by solving a mass transportation problem, which is a quadratic optimization problem with linear constraints. Bronstein et al. [19] formulated a Generalized Multi-dimensional Scaling method as an optimization approach for local minimization of the GH-type embedding distortion between two triangular meshes. Apparently, an ideal and complete solution for the comparison of two non-rigid shapes is to match them directly. However, due to its high computational complexity, direct shape matching is impractical for real shape retrieval systems that require instant responses.

The utilization of canonical forms is also a promising solution for non-rigid 3D shape retrieval. Indeed, with canonical forms, any shape searching algorithm can be applied for the retrieval of non-rigid models. As we know, excellent performance, in term of both accuracy and efficiency, has been achieved for rigid 3D shape retrieval. Obviously, if it is possible to construct canonical forms with well-preserved features, the problem of non-rigid 3D shape retrieval could be well resolved. The idea of generating canonical forms in 3D domain was initially proposed in [38], where the authors introduced an invariant representation for isometric surfaces by applying MDS embedding to map the original surface to a small dimensional Euclidean space in which geodesic distances can be approximated by Euclidean ones. In [38], three MDS techniques were discussed and compared to construct such 3D canonical forms. To verify the effectiveness of their canonical forms, they [38] computed a moment-based shape descriptor from embedded surfaces and carried out a simple experiment for object classification. More recently, Lian et al. [79] presented a framework for non-rigid 3D shape retrieval based on the combination of Least Square MDS embedding and a visual similarity based approach. Thanks to the utilization of canonical forms, superior performance was obtained in [79] compared to other existing methods. Also, Lian et al. [81] proposed a feature-preserved 3D canonical transformation. This method ensures pose normalization of a non-rigid object while preserving the features of the parts of the model. The embedding was subsequently used for retrieval.

A recent comparison of state-of-the-art techniques for non-rigid shape retrieval can be found in [80].

2.3 Shape Matching

The problem of shape matching considers the searching of reliable correspondences between two shapes. The most common representation for these correspondences is a set of pairs of points. Going back in the literature, maybe the first reference to shape matching is in the registration of sensed data [32]. The registration requires to automatically align two rigid shapes. The Iterative Closest Point (ICP) algorithm [9] is an effective method to the registration of two models. In addition, many efficient variants have been evaluated [117] and it is probably the *de facto* method for performing alignment and registration tasks. However, the ICP-like methods search for a rigid alignment which can be characterized with six parameters. In the case of non-rigid transformations, it is not possible such characterization and therefore we need to address the problem in a different way.

The matching problem for non-rigid shapes can be formulated as a minimum-distortion problem. Given two near-isometric shapes, the goal is to find a correspondence set that minimizes the distortion between the two shapes. Generally, this problem is stated as an optimization problem where solutions arise from the need to resolve this problem effectively and efficiently. A comprehensive survey on shape correspondences and their application domains can be found in [145].

Bronstein et al. [20] proposed to match two shapes by embedding one into another. Shapes were treated as metric spaces using the geodesic distance as metric. They used a generalization of the multi-dimensional scaling method to find the minimum-distortion embedding between two metric spaces. On the other hand, a Möbius voting approach was used by Lipman and Funkhouser [82]. A conformal flattening was applied to shapes, where it was possible to apply a Möbius transform for triplets of points. The deformation energy caused by the triplets in the rest of the points of the mesh allows us to compute a vote for each couple of correspondences. The pairs with high votes were considered as reliable correspondences. Also, Kim et al. [70] proposed to generate a set of conformal maps and subsequently associate confidence ans consistency weights to each map. Then, the authors proposed to find the best blending from the candidate set in order to compute a final correspondence set.

Many methods solve the problem iteratively through a process of correspondence refinement. Ovsjanikov et al. [103] exploited the use of heat kernel maps to find a unique reliable correspondence. Subsequently, an iterative method propagates the correspondence set according to a kernel map optimization method. Also, a RANSAC-like method to find correspondences was proposed in [142]. The algorithm operates over a set of hypothesis from an initial set computed from keypoints. Similarly, Tevs et al. [141] presented a probabilistic approach to plan the iterative searching of correspondences. Huang et al. [58] proposed to find correspondences in context of non-rigid registration. Their method alternate between finding and registering the correspondence set.

Greedy approaches to tackle the optimization problem have been also considered. Zhang et al. [156] selected keypoints using a geodesic approach. Then, the search of correspondences was guided by a best-first algorithm that models the quality of the correspondence set with a geodesic distortion. On the other hand, Zaharescu et al. [154] proposed a method which resembles the matching of correspondences in images. Adaptations of DoG and HoG were used for selecting and describing keypoints on the mesh surface. Next, a simple matching algorithm that takes the best match between descriptors was presented. Also, Sahillioğlu and Yemez [118] stated the correspondence problem as a graph problem. They proposed to find an initial matching over a bipartite graph, and subsequently a greedy approach was responsible of adding and refining new correspondences.

A common approach is to state the problem of finding correspondences as an integer program. Dubrovina and Kimmel [36] suggested to describe a mesh point using the eigenfunctions of the Laplace-Beltrami operator. The matching algorithm was stated as a quadratic integer program which consider a distortion function to optimize. On the other hand, Wang et al. [148] stated the problem as a optimization problem involving a linear term (similarity between local descriptors) and a quadratic term (geometry consistency). The solution was devised with a graph-matching algorithm. More recently, Rodolà et al. [114] formulated the minimum-distortion problem with a quadratic optimization function. The solution was obtained with a game-theoretic strategy and a merging algorithm to get the final correspondence set. Similarly, Rodolà et al. [115] used elastic net constraint to enhance their game-theoretic approach. The solution to the matching problem was expressed as the sparse solution of a quadratic assignment problem with norm-based constraints. Zeng et al. [155] devised a high-order matching algorithm based on the Möbius transform to populate the correspondences from an initial set. They derived a dual-decomposition of the original problem to obtain an integer linear problem.

Although the most common representation of a matching is a set of point-wise correspondences, a recent effort has been done to find a more flexible representation: the functional maps [100]. This representation is a generalization of point-wise maps which relates functions spaces instead of points. In addition, this theory takes advantage of the Laplace-Beltrami bases for computing reliable correspondences in non-rigid shapes. The advantage of this method is that the matching procedure can be cast as a high-dimensional alignment between the bases of two shapes. Also, the inference of a map is a linear procedure which can be solved efficiently. Based on the functional map representation, Kovnatsky et al. [72] formulated a method to compute point-wise correspondences using common approximate eigenbases from multiple shapes. Also, Pokrass et al. [111] used the functional representation of correspondences to model the correspondence problem as a permuted sparse coding method.

Chapter 3

Background

In this chapter, we provide descriptions for the descriptors and datasets used throughout the thesis. More specifically, the presented descriptors are: DESIRE (Sec. 3.1.1), PANORAMA (Sec. 3.1.2), Heat Kernel Signatures (Sec. 3.1.3), and Wave Kernel Signatures (Sec. 3.1.4). Furthermore, the used dataset are: SHREC 2010 Robust Feature Detection and Description (Sec. 3.2.1), and the SHREC 2009 Generic Shape Retrieval (Sec. 3.2.2).

3.1 Descriptors

3.1.1 DESIRE

This is a composite descriptor formed by the combination of three global descriptors: depth-buffer, silhouettes, and spherical harmonics over ray-based functions [147]. Each descriptor is computed independently and subsequently they are concatenated to form the final DESIRE descriptor for a shape. We now present each individual descriptor.

Depth-buffer

The *depth-buffer descriptor* [146] is an image-based descriptor. It computes 2D projections of the 3D model, and then computes its feature vector from the obtained projections. This descriptor considers not only the silhouette of each projection of the 3D model, but also considers the depth information (distance from the clipping plane, where the projection starts, to the 3D model).

The first step of the depth-buffer descriptor computes 2D projections of the 3D model. To accomplish this, the model must be first normalized in pose (by means of PCA analysis, for example), as this descriptor is not inherently invariant to rotations or scaling. Then, the model must be enclosed in a bounding cube. Each face of this cube is divided into $n \times n$ cells (with initial value 0), which will be used to compute the depth-buffers for each 2D projection.

Finally, the 3D model is orthogonally projected to the face of the bounding cube. The value associated to each cell is the normalized orthogonal distance (a value in $[0,1]$) between the face of the bounding cube and the closest point (orthogonally) in the 3D model.

Formally, let w be the width of the bounding cube. If a point p belongs to the surface of the 3D model, its closest orthogonal cell in the face of the bounding cube is c, and p is the closest point in the mesh to c, the associated value of c is

$$value(c) = \frac{w - \delta(c,p)}{w}, \tag{3.1}$$

where $\delta(c,p)$ is the distance from c to p.

This method works well if the 3D model does not contain a significant number of outliers. Otherwise, the faces of the bounding cube may be too far to the actual surface of the 3D model (it will only be close to the few outliers). This will produce that the values of almost all cells in a face of the bounding cube will be similar, except for the outliers, thus affecting the computation of the descriptor.

To avoid this problem, Vranic [146] suggests using a canonical cube that does not necessarily enclose the 3D model. The canonical cube is defined by a parameter $t > 0$, such that the vertices of this cube correspond to $(x,y,z)|x,y,z \in \{-t,t\}$. The part of the 3D model that lies outside the canonical cube is not used for computing the descriptor, thus any outlier point will be effectively ignored.

The values associated to the cells on each face of the bounding box could be directly used as the attributes for the feature vector. This feature vector would have a dimensionality of $6n^2$. However, such a descriptor may lead to poor retrieval effectiveness [146]. Instead, the depth-buffer descriptor transforms the values in the spatial domain to the frequency space. Then, it selects some of the obtained coefficients to form the final descriptor.

The depth-buffer descriptor computes the 2D discrete Fourier transform for each of the depth-buffers. Briefly, the 2D discrete Fourier transform of a sequence of two-dimensional complex number of equal length (n in our case) is defined as

$$F(u,v) = \frac{1}{n}\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} f(x,y)\mathrm{e}^{-2\pi\mathrm{i}(xu+yv)/n} \tag{3.2}$$

where $f(x,y), 0 \leq x,y \leq n-1$ is the value of the cell defined by the tuple (x,y). With this definition, it is easy to recover the original values $f(x,y)$:

$$f(x,y) = \frac{1}{n}\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} F(u,v)\mathrm{e}^{2\pi\mathrm{i}(xu+yv)/n}. \tag{3.3}$$

The presented formula for $F(u,v)$ takes $O(n^4)$ time ($O(n^2)$ operations must be applied for each cell of the $n \times n$ grid), and it must be computed for each face of the bounding cube, thus it is computationally expensive. However, if n is a power of two, the Fast Fourier Transform can be applied to speed the computation of the coefficients, reducing the time complexity to $O(n^2 \log n)$.

Before computing the Fourier coefficients, the value $f(0,0)$ is aligned with the cell $(n/2, n/2)$. In this way, the computed low frequency Fourier coefficients correspond to those located in the middle of the resultant image (pixels with values $F(u,v)$). As the input of the 2D discrete Fourier transform are real values, the obtained coefficients satisfy a symmetry property:

$$F(u,v) = \overline{F(u\frac{1}{2}, v\frac{1}{2})},\ (u+u') \mod n = (v+v') \mod n = 0, \tag{3.4}$$

where $\overline{F(u\frac{1}{2}, v\frac{1}{2})}$ is the complex conjugate of $F(u', v')$.

After computing the Fourier coefficients, the final depth-buffer descriptor is formed as follows. First, one needs to set a parameter value $k \in \mathbb{N}$ $(k < n/2)$. Then, a set of values p and q are computed, such that they hold the inequality

$$|p - n/2| + |q - n/2| \leq k \leq n/2. \tag{3.5}$$

The absolute values of the coefficients $F(p,q)$ corresponds to the attributes of the final feature vector. It follows that the number of considered coefficients is $k^2 + k + 1$. As we must repeat this process for each face of the bounding cube, the final dimensionality of the descriptor is $6(k^2 + k + 1)$. In this thesis, we use $n = 256$ and $k = 5$, which gives a descriptor of dimension 186.

Figure 3.1 shows the depth buffer renderings for a 3D model of a car. The first row of images shows the depth buffers of the 3D model. Darker pixels indicate that the distance between the view plane and the object is smaller than at brighter pixels. The second row shows coefficient magnitudes of the 2D Fourier transform of the six images.

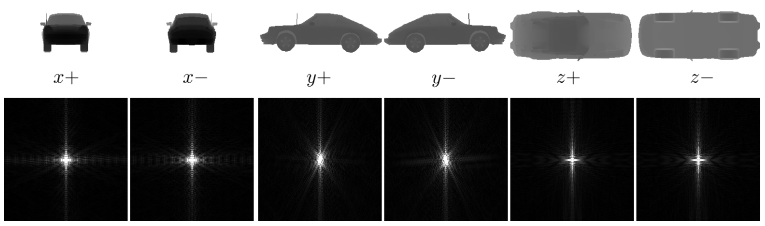


Figure 3.1: Depth-buffer renderings. The top row shows the depth buffers of the 3D model. The bottom row shows their coefficient magnitudes of the 2D Fourier transform. Figure taken from [23].

Silhouettes

The *silhouette-based descriptor* [146] is also an image-based descriptor. At some extent, the process to obtain the descriptor is similar to the depth-buffer processing. More specifically, the model needs to be normalized in pose before computing the silhouettes. Also, the method is based on the computation of orthogonal projections, but in this case only three projections are used (according to the three hyper-planes formed for the three principal axes). Unlike the depth-buffer projections, this method computes orthogonal projections for each hyper-plane, where each pixel in the resulting $n \times n$ image is an attribute of presence of the object in a certain view.

The next step is finding the outer contour. A pixel $c_{\mathrm{i}j}$ belongs to the contour if $c_{\mathrm{i}j} = 0$ (a shape pixel) and at least one of its 4-neighbors belong to the background. The result is a list of pixels that identifies the outer contour C:

$$C = \{c_{\mathrm{i}_0 j_0}, \ldots, c_{\mathrm{i}_{L-1} j_{L-1}}\} \tag{3.6}$$

where $L = |C|$.

Note that the number of points in C could not be constant from object to object, so it is necessary to select a subset K with a fixed number of points. Authors originally proposed to use a sampling in polar coordinates. That is, each contour point $c_p = (\mathrm{i}_p, j_p)$ has a polar representation $\wp(c_p) = (\rho(c_p), \gamma(c_p))$ with $\rho(c_p) \geq 0$ and $-\pi \leq \gamma(c_p) \leq \pi$ such that $c_p = (\mathrm{i}_p, j_p) = O + \rho(c_p)(\cos\gamma(c_p), \sin\gamma(c_p))$, where $O = (n/2, n/2)$. Therefore, the point $k_\mathrm{i} \in K$ is determined by

$$s_\mathrm{i} = \frac{a_{max}}{n}(c_p - O) \mid \rho(c_p) = \max \Psi_\mathrm{i} \mid \Psi_\mathrm{i} = \left\{\rho(c_q) | \gamma(c_q) \approx \frac{2\mathrm{i}\pi}{K}\right\} \tag{3.7}$$

where $0 \leq \mathrm{i} \leq |K| - 1$, a_{max} is the maximum absolute value for x, y, or z coordinates from the 3D object, and n is the resolution for the silhouette image.

Subsequently, the set K can be seen as a sequence. Let f be a function applied to each element in the sequence such that $f_\mathrm{i} = f(s_\mathrm{i})$, the discrete Fourier transform of function f is given by

$$\hat{f}_p = \frac{1}{\sqrt{K}} \sum_{\mathrm{i}=0}^{K-1} f_\mathrm{i} \mathrm{e}^{-j\frac{2\pi}{K}\mathrm{i}\cdot p}, p = 0, \ldots, K-1, \tag{3.8}$$

where j is the imaginary unit and $\hat{f}_p \in \mathbb{C}$ are Fourier coefficients. In addition, the functions f is defined as

$$f_\mathrm{i} = \rho_\mathrm{i}. \tag{3.9}$$

The absolute values for the first k coefficient of each silhouette is used as the final descriptor. Therefore, the final descriptor contains $3k$ values. In this thesis, we use $N = 256$ and $k = 50$.

Ray-based harmonics

The idea behind this descriptor is to represent a 3D object with a function on a sphere [146]. It is expected that the function on the sphere characterizes the geometry of the object. Let S^2 be a sphere of radius 1 with the center at the origin. Let us define the function r

$$\begin{aligned} r :& S^2 \to [0, +\infty) \\ & r(\vec{u}) = \max\{r \geq 0 | r\vec{u} \in I \bigcup \{O\}\} \end{aligned} \tag{3.10}$$

where I is the surface of the input 3D object, and $\vec{u} \in S^2$ is a point in the sphere and whose direction can be represented by a normal vector. The function $r(\vec{u})$ stores the maximum distance from the origin to some point in the 3D object in the direction $\vec{u}$.

The original method consists of sampling $4B^2$ points on the sphere (i.e. $4B^2$ unit vectors) and perform the Spherical Fast Fourier Transform to obtain B^2 complex coefficients. The final descriptor is computed from the absolute values of some Fourier coefficients. In the case of real-valued functions (such as r), the Fourier coefficients has a pyramidal structure because of the symmetry. Then, the idea is to select only some coefficients. Vranic deduced that using k rows from that structure, the final number of coefficient is $k(k + 1)/2$. In this thesis, we use $B = 128$ and $k = 16$.

3.1.2 PANORAMA

PANORAMA [106] is an image-based descriptor, similar to the Depth Buffer descriptor. Hence the first step is to perform a pose normalization of the 3D object. In order to enrich the final descriptor, two methods are jointly used to normalize for rotation: CPCA [146] and NPCA [104]. That is, each 3D object is normalized using both methods, and subsequently each normalized object is processed to obtain a descriptor. The final distance takes into account this fact to combine both descriptions.

The main idea is to project a 3D model onto the lateral face of a cylinder. The original method suggests the use of one cylinder for each principal axis, giving three projections for each object. To obtain a panoramic view, a 3D model is projected to the lateral face of a cylinder of radius R and height $H = 2R$, where $R = 3 \times \mathrm{d}_{mean}$, and where d_{mean} is the mean distance of the model's surface from the centroid.

The next step is to discretize the lateral surface of the cylinder to obtain a set of points $s(\gamma, y)$, where $\gamma \in [0, 2\pi]$ and $y \in [0, H]$. The sampling rate are $2B$ and B, respectively for

γ and y (in this thesis, we use the value recommended by authors $B = 64$). The value γ parameterizes the angle in the plane perpendicular to the cylinder, and the value y parameterizes the height. Each point $s(\gamma, y)$ will be associated to two values regarding the position of the object and the orientation of the object.

Given a point $s(\gamma, y)$, the position of the object is obtained by tracing a ray from the central point of the cross-section in the height y using the direction γ. The position value associated to the point $s(\gamma, y)$ is the furthest distance of a surface point intersected by the ray. In the same way, the orientation value for $s(\gamma, y)$ is $|\cos(\angle(\vec{\gamma}, \vec{N}))|^m$, where $\vec{N}$ is the normal of the face intersected by the ray, in the same way as before. The m^{th} power helps to enhance the contrast of the produced view. Note that in total, this process delivers six panoramic views due to the three axis, with two panoramic views for each one.

The description of the panoramic views is obtained through the Discrete Fourier Transform and the Discrete Wavelet Transform. For each panoramic view, a set of Fourier coefficients are obtained, similar to the application of DFT in the Depth-Buffer descriptor. In addition, given a projection $s(\gamma_u, y_v)$, the Discrete Wavelet Transform is applied as follows

$$W^{\gamma}(j_0, m, n) = \frac{1}{\sqrt{2}B} \cdot \sum_{u=0}^{2B-1} \sum_{v=0}^{B-1} s(\gamma_u, y_v) \cdot \gamma_{j_0,m,n}(u, v), \tag{3.11}$$

$$W^{\psi}(j, m, n) = \frac{1}{\sqrt{2}B} \cdot \sum_{u=0}^{2B-1} \sum_{v=0}^{B-1} s(\gamma_u, y_v) \cdot \psi_{j,m,n}(u, v), \tag{3.12}$$

where $m \in [0, 2B-1]$, $n \in [0, B-1]$, $j \geq j_0$ is the scale of the multi-level DWT, j_0 is the starting scale and $\gamma_{j_0,m,n}(u, v)$, $\psi_{j,m,n}(u, v)$ is the scaling and wavelet functions, respectively. The original proposal suggested to use the Haar and Coiflet filter as basis functions. Next, the mean, standard deviation and skewness are computed for aggregating the coefficient information. In total, the length of the DWT descriptor for a panoramic view is $18 \times \log_2 B + 6$. The final DWT descriptor for an object is obtained by concatenating the six descriptors obtained from the different views.

Finally, the descriptor for an object is composed by the Fourier descriptor and the Wavelet descriptor. To compute the distance between two objects, the Manhattan distance is used for the Fourier descriptor and the Canberra distance for the Wavelet descriptor.

3.1.3 Heat Kernel Signatures

The heat diffusion process over a compact manifold S, possibly with boundary, is governed by the heat equation

$$\Delta_S u(x, t) = -\frac{\partial u(x, t)}{\partial t} \tag{3.13}$$

where Δ_S is the Laplace-Beltrami operator of S and $u(., t)$ is the heat distribution over S in time t.

The fundamental solution of the heat equation is $K_t(x, y)$ called the heat kernel. This represents a solution with a point heat source in x and can be considered as the amount of heat transferred from x to y at time t supposing that the heat source is x. For compact manifolds, the heat kernel can be expressed using the eigenvalues and eigenvectors of the Laplace-Beltrami operator as follows:

$$K_t(x, y) = \sum_{\mathrm{i}=0}^{\infty} \mathrm{e}^{-\lambda_{\mathrm{i}} t} \vec{v}_{\mathrm{i}}(x) \vec{v}_{\mathrm{i}}(y) \tag{3.14}$$

where λ_{i} is the i-th eigenvalue and $\vec{v}_{\mathrm{i}}(\cdot)$ is the i-th eigenvector's entry corresponding to a given point.

Sun et al. [136] formally proved that the heat kernel is isometric invariant, informative(enough, redundant information exists), multi-scale, and stable against perturbations on the surface (it is worth noting that Gebald et al. [43] also introduced the heat kernel signature as a independent result). In addition, restricting the heat kernel to the temporal domain and fixing the spatial variables, we can obtain a representation for each point on the manifold:

$$K_t(x, x) = \sum_{\mathrm{i}=0}^{\infty} \mathrm{e}^{-\lambda_{\mathrm{i}} t} \vec{v}_{\mathrm{i}}(x)^2 \tag{3.15}$$

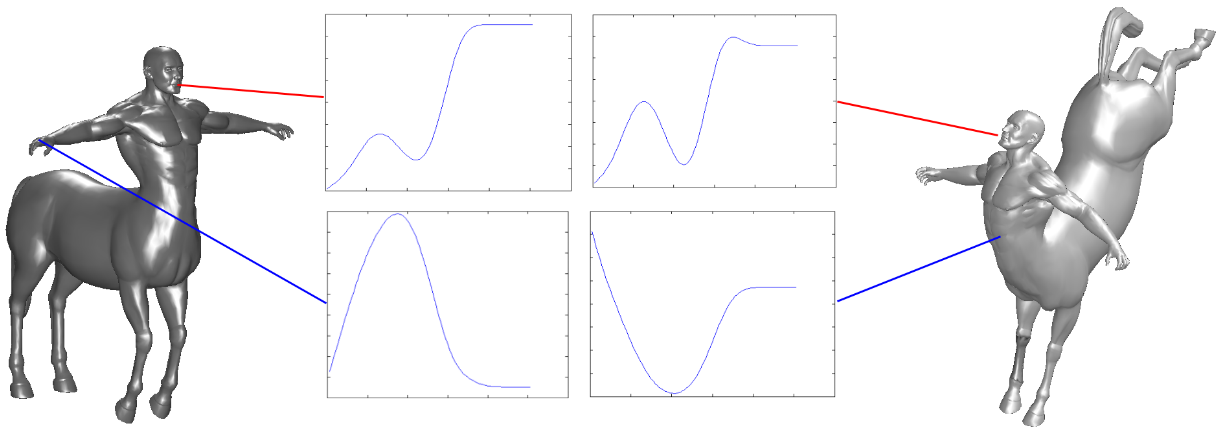


Figure 3.2: Heat kernel signatures calculated on two isometric shapes. At top, signatures in corresponding points look very similar. At bottom, signatures in different points on the mesh differ.

In Fig. 3.2, we show heat kernel signatures for two isometric shapes. Given a shape S, we need to calculate the heat kernel signature for point on S. In practice, the heat kernel signature of a point $x \in S$ is a n-dimensional descriptor vector with each bin corresponding to some value of t:

$$HKS(x) = (K_{t_1}(x,x), \dots, K_{t_n}(x,x)) \tag{3.16}$$

In the experiments of this thesis, we use n = 100, with $t_1 = 4ln10/\lambda_{300}$ and $t_{100} = 4ln10/\lambda_2$. The rest of time values are sampled in the logarithmic scale in the interval $[t_1, t_n]$. In addition, to calculate the discrete Laplace-Beltrami operator, we used the cotangent scheme proposed by Meyer et al. [91]. Also, for this thesis we used the original implementation of HKS[1].

3.1.4 Wave Kernel Signatures

Unlike the Heat Kernel Signatures which are based on the heat equation, the Wave Kernel Signatures [4] are based on the Schrödinger equation

$$\frac{\partial \psi}{\partial t}(x,t) = -\mathrm{i}\Delta\psi(x,t), \tag{3.17}$$

where $\psi(x,t)$ represents the wave function which governs the evolution of a quantum particle on a surface. If the Laplace spectrum of the shape has no repeated values, the wave function is given by

$$\psi_E(x,t) = \sum_{k=0}^{\infty} \mathrm{e}^{\mathrm{i}\lambda_k t}\phi_k(x) f_E(\lambda_k) \tag{3.18}$$

The probability to measure the particle at a point x is $|\psi_E(x,t)|^2$. The Wave Kernel Signature of a point x is then the average probability for infinite times. That is

$$WKS(E,x) = \lim_{T\to\infty} \frac{1}{T}\int_0^T |\psi_E(x,t)|^2 \tag{3.19}$$

Since the functions $\mathrm{e}^{\mathrm{i}\lambda_k t}$ are orthogonal for the L^2 norm, we have

$$WKS(E,x) = \sum_{k=0}^{\infty} \phi_k(x)^2 f_E(\lambda_k)^2 \tag{3.20}$$

In the discrete setting, the first step is to compute the Laplace-Beltrami operator for a mesh and their associated eigenvalues and eigenvectors. Subsequently, the Wave Kernel Signature for a point x is defined as a n-dimensional descriptor as follows

$$WKS(x) = (WKS(\mathrm{e}_1, x), \dots, WKS(\mathrm{e}_n, x)), \tag{3.21}$$

[1]Available in http://www.geomtop.org/software/hks.html.

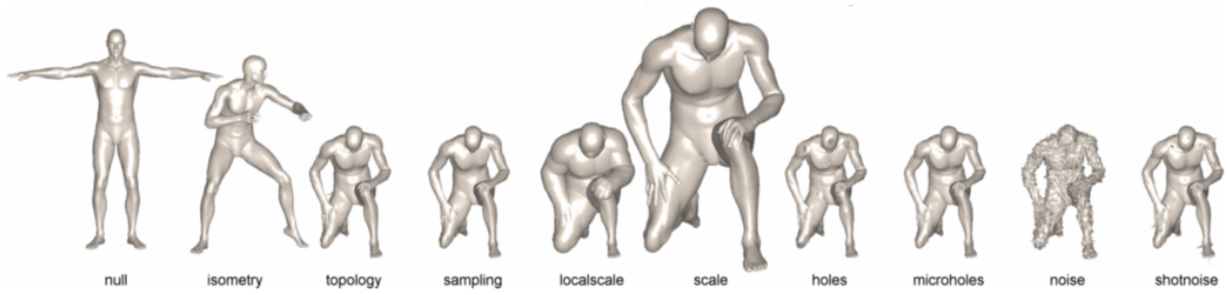


Figure 3.3: Null shape and its transformations. Figure taken from [14].

where $WKS(\mathrm{e_i}, x)$ is evaluated as follows:

$$WKS(\mathrm{e_i}, x) = \frac{\sum_{k=1}^{N} \phi_k^2(x) \mathrm{e}^{\frac{-(\mathrm{e_i} - \log \lambda_k)^2}{2\sigma^2}}}{\sum_{k=1}^{N} \mathrm{e}^{\frac{-(\mathrm{e_i} - \log \lambda_k)^2}{2\sigma^2}}}. \tag{3.22}$$

In addition, in this thesis, we use $n = 100$, $\mathrm{e}_1 = \log(\lambda_1)$ and $\mathrm{e}_{100} = \log(\lambda_{300})$. Also, the remaining values are computed uniformly with an increment of $\delta = (\mathrm{e}_{100} - \mathrm{e}_1)/100$. Finally, the variance σ is set to 6δ. In addition, we use the original implementation of WKS[2].

3.2 Datasets

3.2.1 SHREC 2010 Feature Detection and Description Benchmark

This dataset aims at evaluating the effectiveness of feature detection and description [14]. This dataset is composed of triangular meshes with approximately 10,000 - 50,000 vertices.

The collection consists of three basic shapes (null shapes) from which a set of transformations have been applied. For each null shape, nine transformations were used: isometry (non-rigid transformation), topology, big holes and micro holes, local and global scaling, additive Gaussian noise, shot noise, and down-sampling (less than 20% of the original points). Figure 3.3 shows a null shape and its transformed versions.

Each transformation is performed in five versions. In all transformations, except isometry and scale, each version represents a strength level, so high levels correspond with stronger transformations. The scales used in scaling transformation were 0.5, 0.875, 1.25, 1.625, and 2. For the isometry transformation, each version reflects a non-rigid transformation of the null shape. Therefore, each null shape has 45 transformed shapes and there are 138 shapes in the whole collection.

In addition, for each transformed shape Y in the dataset, there is a groundtruth with dense point correspondences to its corresponding null shape X. The groundtruth is given in the

[2]Available in http://vision.in.tum.de/members/aubry/publications.

form of pair of points $C_0(X,Y) = \{(y'_k, x_k)\}_{k=1}^{|Y|}$ (and same way, $C_0(Y,X)$). This information allows us to evaluate several aspects of local features: repeatability, region overlap, and correspondence localization. In this thesis, we use this dataset for local structure detection (Chapter 4) and non-rigid shape correspondences (Chapter 7). The methodologies to evaluate these tasks are defined when needed.

3.2.2 SHREC 2009 Generic Shape Retrieval

The goal of this dataset is to evaluate generic shape retrieval algorithms [47]. This dataset contains 720 shapes organized in 40 classes with 18 shapes per class. In this thesis, we use this dataset for an application of local features in generic shape retrieval (Chapter 5).

3.2.3 Large-scale Non-rigid Shape Retrieval Dataset

In chapter 6, we are interested in evaluating the effectiveness and efficiency of our shape retrieval approach. So far, most of the datasets have had a limited size, in which is not possible to properly evaluate the efficiency. Therefore, we built a new dataset containing 5604 models. We used the Sumner dataset [135] and the TOSCA dataset [16] (which contain non-rigid shapes) as basis. Then, we collected a large set of unclassified shapes from different datasets such as the Princeton Shape Benchmark [124], the Konstanz benchmark [25], the National Taiwan University benchmark [30]. For the experiments regarding retrieval performances, only models of the Sumner and TOSCA dataset were used as queries. All models were normalized to have unit surface area.

3.2.4 Shape Matching Benchmark

Kim et al. [70] built a dataset for the evaluation of shape matching algorithms. This benchmark collects shapes from three different datasets: TOSCA [16], SCAPE [2], and the Watertight dataset [45]. The benchmark provides a dense ground-truth map which makes possible to evaluate a given correspondence set. In this thesis, we use only the TOSCA part of this dataset since this part contain shapes with distinguishable features. The other parts of the benchmark contain shapes with poor features.

3.3 Indexing

3.3.1 Metric Spaces and Metric Indexing

A metric space is an ordered pair $(\mathbb{U}, \mathrm{d})$, where U is a set and $\mathrm{d} : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}^+$ is a distance function that holds the following properties:

- *Symmetry*: $\forall x, y \in \mathbb{U}$, $\mathrm{d}(x, y) = \mathrm{d}(y, x)$
- *Reflexivity*: $\forall x \in \mathbb{U}$, $\mathrm{d}(x, x) = 0$
- *Strict positiveness*: $\forall x, y \in \mathbb{U}$, $x \neq y \Rightarrow \mathrm{d}(x, y) > 0$
- *Triangle inequality*: $\forall x, y, z \in \mathbb{U}$, $\mathrm{d}(x, z) \leq \mathrm{d}(x, y) + \mathrm{d}(y, z)$

A multimedia database can be considered as a metric space, where the distance d allows us to quantify the dissimilarity between objects. Moreover, it is necessary to formulate searching strategies in order to answer similarity queries. Let $\mathbb{X} \subset \mathbb{U}$ be a set of multimedia objects. There are two typical similarity queries:

- *Range query.* A range query (q, r), $q \in \mathbb{U}$, $r \in \mathbb{R}^+$, is defined as $(q, r) = \{x \in \mathbb{X}, \mathrm{d}(x, q) \leq r\}$.
- *k nearest neighbors queries* (*k-NN*). It reports the k objects from $\mathbb{S}$ closer to q.

The most naive procedure to perform a similarity query is a linear scan. That is computing the distance between the query q and every object in the database. Subsequently, objects are sorted according to the distance and the final result can be computed. Nevertheless, if we assume that the distance d is expensive, linear scan will be slow. Many researches have paid attention to this problem and several proposals have focused in the use of metric access methods (MAMs). A MAM is composed of an index structure and algorithms to perform efficient similarity queries. The cornerstone of these methods is the triangle inequality, which help us to filter out objects during the search, avoiding to compute the whole set of distances as in a linear scan. A survey of metric access methods can be found in [29].

Pivot-based Indexing

The pivot-based indexing is one of the simplest approach to reduce the search complexity [93]. This technique consists of selecting a set of t pivot objects from the collection. The index is composed by a table of pre-computed distances between the pivots and each object in the collection. Thus, a range query (q, r) works as follows: using the triangle inequality it follows for any $x \in \mathbb{X}$ and $1 \leq \mathrm{i} \leq t$ that $\mathrm{d}(q, x) \geq |\mathrm{d}(p_\mathrm{i}, x) - \mathrm{d}(p_\mathrm{i}, q)|$. Therefore, the objects u of interest should satisfy $\mathrm{d}(q, u) \leq r$, so one can exclude objects that satisfy $|\mathrm{d}(p_\mathrm{i}, u) - \mathrm{d}(p_\mathrm{i}, q)| > r$ for some pivot p_i, discarding the direct evaluation of $\mathrm{d}(q, u)$. The algorithm 3.1 shows in detail the algorithm for range queries.

First, the algorithm computes the distances between the query object x and the pivots p_i (lines 1-3). Second, every object in the collection is evaluated, however the pivot exclusion criterion is first evaluated. The algorithm checks if some pivot helps to discard the u_i object (lines 7-12). If it is not possible to discard the object, the algorithm computes the real distance to verify if the range is satisfied (lines 13-17).

The problem with a range query is that many times it is difficult to find a meaningful range radius for a certain collection. A possible solution is to implement a nearest neighbor search, which does not depend on the search radius. In retrieval systems, the k-NN algorithm is commonly used for similarity search. An adaptation of the range query algorithm can be used for k-NN search. Algorithm 3.2 shows the algorithm we use in this thesis.

Algorithm 3.1 Pivot-based Range Query

```
Require: Collection of objects $\mathbb{U}$
Require: Query $x \in \mathbb{X}$
Require: Search radius $r \in \mathbb{R}^+$
Require: Set of pivots $T$ with size $t = |T|$
Require: Index table $\delta$
Ensure: Set of objects $C$

1: for i ← 1 to $t$ do
2:     d$p$(i) ← d($p_{\mathrm{i}}, x$)
3: end for

4: $C \leftarrow \emptyset$
5: for i ← 1 to $|\mathbb{U}|$ do
6:     $flag \leftarrow false$
7:     for $j \leftarrow 1$ to $t$ do
8:         if $|\mathrm{d}p(j) - \delta(\mathrm{i}, j)| > r$ then
9:             $flag \leftarrow true$
10:            $break$
11:        end if
12:    end for
13:    if $notflag$ then
14:        if $\mathrm{d}(u_{\mathrm{i}}, x) \leq r$ then
15:            $C \leftarrow C \cup \{u_{\mathrm{i}}\}$
16:        end if
17:    end if
18: end for
19: Return $C$
```

Algorithm 3.2 Pivot-based k-NN Query

```
Require: Collection of objects $\mathbb{U}$
Require: Query $x \in \mathbb{X}$
Require: Number of neighbors $k$
Require: Set of pivots $T$ with size $t = |T|$
Require: Index table $\delta$
Ensure: Set of objects $C$

1: for i ← 1 to $t$ do
2:     $\mathrm{d}p(\mathrm{i}) \leftarrow \mathrm{d}(p_\mathrm{i}, x)$
3: end for

4: Let $Q$ a priority queue which stores the k-nearest neighbors sorted by distance
5: $Q \leftarrow Q \cup T$
6: Let $mindist$ the greatest distance stored in $Q$
7: for i ← 1 to $|\mathbb{U}|$ do
8:     if $u_\mathrm{i}$ is not pivot then
9:         $flag \leftarrow false$
10:        for $j \leftarrow 1$ to $t$ do
11:            if $|\mathrm{d}p(j) - \delta(\mathrm{i}, j)| > mindist$ then
12:                $flag \leftarrow true$
13:                $break$
14:            end if
15:        end for
16:        if $notflag$ then
17:            if $\mathrm{d}(u_\mathrm{i}, x) \leq mindist$ then
18:                $Q \leftarrow Q \cup \{u_\mathrm{i}\}$
19:                Update $mindist$
20:            end if
21:        end if
22:    end if
23: end for
24: Return $C \leftarrow Q$
```

It is worth noting that the k-NN algorithm is a query search with an adaptive radius *mindist*, which can be adjusted in each iteration. The priority queue allows us to maintain the k-nearest neighbors at every iteration. Finally, the queue stores the final set of nearest neighbors at the end of the algorithm.

Selecting pivots

An important part of the pivot-based approach is the selection of pivots from the collection. Here we describe a simple technique called SSS [110] (Sparse Spatial Selection). The algorithm is simple and it is shown in Algorithm 3.3.

Algorithm 3.3 Sparse Spatial Selection

```
Require: Collection of objects $\mathbb{U}$
Require: Real number $\alpha$
Ensure: Set of pivots $P$
1: Let $M$ be the maximum distance between objects in $\mathbb{U}$

2: $P \leftarrow \{u_1\}$
3: for all $u_i \in \mathbb{U}$ do
4:     if $\forall p \in P$, $d(u_i, p) \geq \alpha \times M$ then
5:         $P \leftarrow P \cup u_i$
6:     end if
7: end for
8: Return $P$
```

The algorithm selects the first object as pivot, and subsequently selects objects far enough from the current set of pivots. The criterion of separability (line 4) ensures that pivots are far from each other. This choice is because is well known that close pivots are not good for the exclusion criterion.

3.3.2 Signature Quadratic Form Distance

Beecks et al. [7] recently introduced the Signature Quadratic Form Distance as a flexible measure for comparing multimedia objects. An interesting advantage of this distance is that an object can be represented with multiple features. In addition, the comparison between two sets of features does not require to have the same number of features in each set.

An object P is represented by a feature set $F = \{f_i\}$, where $f_i \in FS$. FS is a feature space of arbitrary dimension. In addition, let us suppose that F has length K. Furthermore, we need to suppose the existence of a local clustering of $F : C_1, \ldots, C_n$. The *feature signature* S^P is defined as a set of tuples $FS \times R^+$ as follows

$$S^P = \{(c_i^P, w_i^P), i = 1, \ldots, n\} \tag{3.23}$$

where $c_{\mathrm{i}}^P = \frac{\sum_{f \in C_{\mathrm{i}}} f}{|C_{\mathrm{i}}|}$ and $w_{\mathrm{i}}^P = \frac{|C_{\mathrm{i}}|}{K}$ represent the centroid of i-th cluster and a weight, respectively. Note that the size of S^P depends on the local partitioning and it is variable from object to object. Therefore, each feature signature will have a different number of features according to clustering process.

Given two feature signatures $S^P = \{(c_{\mathrm{i}}^P, w_{\mathrm{i}}^P), \mathrm{i} = 1, \ldots, n\}$ and $S^Q = \{(c_j^Q, w_j^Q), j = 1, \ldots, m\}$ and a similarity function $f_S : FS \times FS \to R$, the *Signature Quadratic Form Distance* between S^P and S^Q is defined as

$$SQFD_{f_S}(S^P, S^Q) = \sqrt{(w^P| - w^Q) \cdot A_{f_S} \cdot (w^P| - w^Q)^T} \tag{3.24}$$

where the notation $(w^A|w^B)$ denotes the concatenation of weight vectors. In addition, $A_{f_S} \in \mathbb{R}^{(n+m)\times(n+m)}$ is the similarity matrix defined as

$$a_{\mathrm{i}j} = \begin{cases} f_S(c_{\mathrm{i}}^P, c_j^P) & \text{if } \mathrm{i} \leq n \text{ and } j \leq m \\ f_S(c_{\mathrm{i}-n}^Q, c_j^P) & \text{if } \mathrm{i} > n \text{ and } j \leq m \\ f_S(c_{\mathrm{i}}^P, c_{j-n}^Q) & \text{if } \mathrm{i} \leq n \text{ and } j > m \\ f_S(c_{\mathrm{i}-n}^Q, c_{j-n}^Q) & \text{if } \mathrm{i} > n \text{ and } j > m \end{cases} \tag{3.25}$$

Clearly, the matrix A_S contains the intra-dependence and inter-dependence information from the signatures of both objects. Moreover, the $SQFD$ considers the weights involved in these dependences. Roughly speaking, the weights indicate the importance of each element in the feature signature. Originally, Beecks et al. proposed to use one of the following similarity functions based on a ground dissimilarity function d(for instance $\mathrm{d} = L_2$):

- Minus: $f\_(c_{\mathrm{i}}, c_j) = -\mathrm{d}(c_{\mathrm{i}}, c_j)$
- Gaussian: $f_g(c_{\mathrm{i}}, c_j) = exp(-\alpha \mathrm{d}^2(c_{\mathrm{i}}, c_j))$
- Heuristic: $f_h(c_{\mathrm{i}}, c_j) = \frac{1}{\alpha + \mathrm{d}(c_{\mathrm{i}}, c_j)}$

Chapter 4

Local Structures on 3D Shapes

There are situations where a global representation of a shape could be not suitable in the context of matching and retrieval. For example, let imagine us a 3D object with a non-rigid transformation or maybe an object with missing parts. Clearly in these situations, the assessment of similarity can not rely in global representations since objects may change dramatically in the global sense. Hence, it is necessary to address the representation problem in a more adequate way.

In this chapter, we study alternative representations for 3D shapes based on local structures. The idea is to detect relevant information from a shape in a more local sense and at different levels of granularity. The granularity is associated with the size of the structure, for instance points in finer levels and regions in coarser levels. The goal is to use the local information to represent a shape as a collection of structures, which characterize that information. Later, we show that the local information can be used to complement global representations or be used independently in matching and retrieval tasks. Furthermore, our results allow us to support our premise that local structures convey valuable information to represent a 3D shape.

We propose two types of local structures, which are related to the granularity level: keypoints (at finer levels) and regions of interest (at coarser levels).

A *keypoint* is a point on the shape's surface with a feature that distinguish it from others in its neighborhood. The characterization of a point depends on the application domain, so it is necessary to define a suitable criterion. Here we consider the level of protrusion as a feature for detecting keypoints. Thus, we propose an efficient algorithm to characterize each vertex on a mesh and subsequently decide which vertices are keypoints. Our method is an adaptation of the Harris method to detect interest points in images. This adaptation is not trivial due to the topology-free nature of 3D shapes. In addition, we need to face to transformations which are not usual in images (for instance: topology changes, tessellations, holes, etc). We show that our *Harris 3D* algorithm detects repeatable keypoints and it is robust against several transformations. Moreover, our method is able to detect repeatable features in presence of severe transformations, which makes it a good alternative for real-world applications.

On the other hand, a region of interest is a distinctive patch of a mesh. To illustrate this idea, let imagine us a human shape, which may contain distinctive regions such as face, hands and feet. More specifically, we define a region of interest as a patch where there are a lot of distinctive keypoints. We propose an algorithm to detect key-components guided by our results on keypoint detection. A key-component is a shape's region that contains a high density of keypoints. In other words, we use the structures at a finer level to find structures at a coarser level. Interestingly, the key-components detected with our method also exhibit a high repeatability against transformations.

This chapter is organized as follows. Section 4.1 presents the Harris 3D algorithm to detect keypoints on meshes. In addition, we conduct a comprehensive evaluation in order to show the effectiveness of our method and the robustness against several transformations. Section 4.2 describes our method to detect key-components. We also carry out an exhaustive evaluation to measure the repeatability of the key-components in presence of transformations. All experiments of this chapter were performed on the SHREC'2010 robust feature detection and description benchmark. Finally, Section 4.3 is devoted to discuss our reflections about the detection of local structures.

4.1 Harris 3D: Interest Points Detection on Meshes

The interest point detection on 3D data is a challenging problem for several reasons. First, there is no consensus about the definition of an interest point. A commonly used definition (that we use in this thesis) relates the measure of interest with the level of protrusion of outstanding local structures. So, vertices on smooth or nearly planar sections of a surface will have low interest, as opposite to vertices in regions with uncommon local structure. For instance, in a human-shaped model, an interest point detector should select vertices on the face, hands, and feet. Second, the topology in 3D meshes is arbitrary. That is, a vertex can have an arbitrary number of neighboring vertices. This makes the tasks of selecting a local neighborhood around a vertex harder. In addition, this drawback causes that different tessellations can represent the same locality and therefore, an interest point detector should be able to deal with that. Third, without a well-defined topological structure for meshes, the extent of a locality in which a vertex is an interest point is unknown or difficult to compute. Finally, there is no additional information other than the position of vertices and the connectivity information among them. This fact complicates the process because the level of interest needs to be measured using the available information, which also depends of the topology of the mesh.

In a different scenario, interest point detectors for images have reached an acceptable effectiveness. The reason is that image structure is well-defined and interest points correspond with pixels that represent interesting structures in the scene captured in the image. In that sense, several methods have been proposed for detecting interest points taking into account different scales and transformations. As a result, the range of applications in computer vision that makes use of interest points detection has increased considerably in recent times such as image matching [86], image stitching [22], and human activity recognition [73, 74], just to name a few.

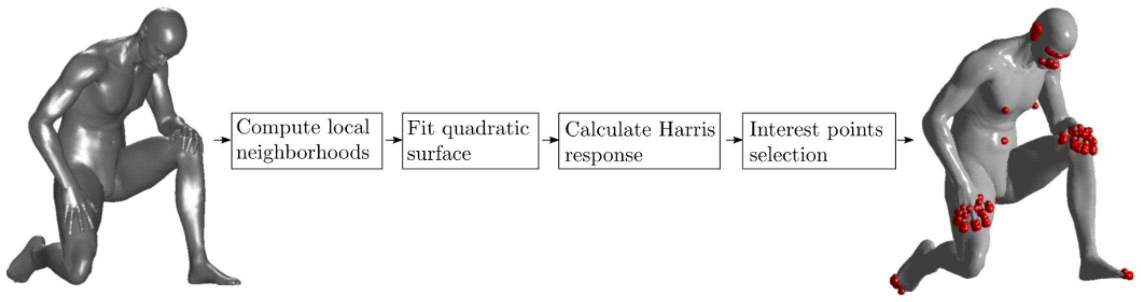

Compute local neighborhoods
Fit quadratic surface
Calculate Harris response
Interest points selection

Figure 4.1: This figure shows the steps we propose to detect interest points in 3D meshes.

However, despite the success of these techniques in the image domain, trying to adapt them to 3D meshes is not a trivial task. Firstly, the adaptation is not direct because 3D meshes structure is very different from images. Secondly, the transformations required to be robust in 3D domains (isometry, topology, change of tessellations, downsampling, etc.) are also different.

In this section, we present an effective and efficient extension of the Harris operator for 3D meshes. We chose the Harris operator for several reasons. First, the computation of the operator is an efficient and simple task. This is an important issue if we want use the interest point detection as a preliminary stage of subsequent process such as shape matching, registration or object recognition. Second, Harris-based methods have been effectively used in a number of applications and they have a high effectiveness as reported in the evaluation reports [119, 94]. Finally, an interesting evidence has been found recently, which greatly favors the Harris interest point detection method. Loog and Lauze [85] recently showed an important connection between the Harris operator and the computational visual attention model of visual perception. Roughly speaking, their model proved that interest points detected by the Harris method have low probability of appearing in other locations in the same image. These reasons encouraged us to investigate an effective extension of the Harris operator for 3D meshes, trying to comply with the robustness to the transformations in 3D domain.

Our method is outlined in Fig. 4.1. Given a 3D mesh, not necessarily manifold, the main process is performed in a vertex-wise manner. The overall process consists of four steps. Firstly, our algorithm determines a local neighborhood around a vertex. The subsequent tasks are performed over this local neighborhood. Secondly, the neighborhood is processed so that it is prepared for a fitting step. We try to fit a quadratic surface to the set of points. This surface is a good representation of the locality and we consider it as an local image. Thirdly, we propose to calculate derivatives using a smoothing over the surface. We can use these derivatives for computing the Harris response for each vertex. Finally, our method selects the final set of interest points.

4.1.1 Interest Points Detection

Harris and Stephens [50] proposed an interest points detector for images. Their method is a popular technique due to its strong invariance to rotation, scale, illumination variation, and image noise [119]. The Harris detector is based on the local autocorrelation function of a signal, which measures the local changes of the signal with patches shifted by a small amount in different directions. The local autocorrelation is defined as:

$$\mathrm{e}(x,y) = \sum_{x_\mathrm{i},y_\mathrm{i}} W(x_\mathrm{i},y_\mathrm{i})[I(x_\mathrm{i}+\triangle x, y_\mathrm{i}+\triangle y) - I(x_\mathrm{i},y_\mathrm{i})]^2 \tag{4.1}$$

where $I(.,.)$ denotes the image function and $(x_\mathrm{i}, y_\mathrm{i})$ are the points in the Gaussian function W centered on (x, y), which defines the neighborhood area in analysis.

Using a Taylor expansion truncated to the first order terms to approximate the shifted image, we obtain:

$$\begin{aligned}\mathrm{e}(x,y) &= \vec{S}\begin{bmatrix} \sum_{x_\mathrm{i},y_\mathrm{i}} W.I_x^2 & \sum_{x_\mathrm{i},y_\mathrm{i}} W.I_x.I_y \\ \sum_{x_\mathrm{i},y_\mathrm{i}} W.I_x.I_y & \sum_{x_\mathrm{i},y_\mathrm{i}} W.I_y^2 \end{bmatrix}\vec{S}^T \\ &= \vec{S}E(x,y)\vec{S}^T\end{aligned} \tag{4.2}$$

where $\vec{S} = [\triangle x \ \triangle y]$ is a shift vector, I_x and I_y denote the partial derivatives in x and y, and along with W are evaluated in $(x_\mathrm{i}, y_\mathrm{i})$ points.

Harris and Stephens proposed to analyze the eigenvalues of matrix E, which contains enough local information related to the neighborhood structure. In addition, to avoid the expensive eigenvalue calculation, they proposed to assign to each pixel in the image the following value:

$$h(x,y) = \mathrm{de}t(E) - k(tr(E))^2 \tag{4.3}$$

with k constant.

The Harris operator has been used in many applications in image processing and computer vision by its simplicity and efficiency. However, the problem with 3D data is that the topology is arbitrary and it is not clear how to calculate the derivatives. To cope this problem, Glomb [46] suggested some approaches. We take this work as a basis for proposing a robust interest points detector on 3D meshes.

Robust Harris Operator on 3D Meshes

Given a vertex of a 3D object, we are interested in calculating the Harris operator value associated with that point. A 3D object is represented as a set of vertices V and a set of faces F with adjacency information between these entities. In addition, our method is not restricted to manifold meshes.

Let v be the analyzed vertex and $V_k(v)$ the neighborhood considering k rings around v. More formally, let us consider an object as a graph $G(V', E')$, where $V' = V$ and E' is the

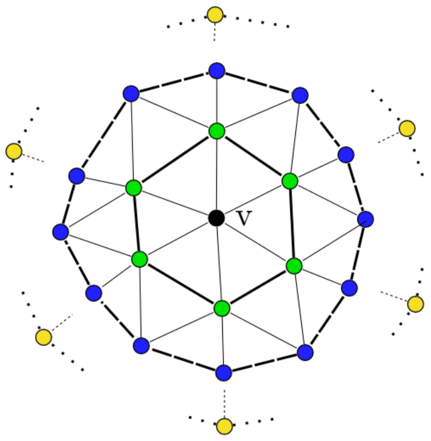


Figure 4.2: Point v and its neighbor rings. Firstly, $V_1(v)$ is composed by vertices connected by strong edges. Secondly, $V_2(v)$ is composed by vertices up to those connected by dashed edges. Finally, $V_k(v)$ is composed by all vertices until those connected by pointed edges.

set of edges obtained from the adjacency information of the object. Given a point $v \in V'$, $V_k(v)$ is defined as

$$V_k(v) = \{w \in V' \text{ such that } |\mathit{shortest\_path}(v, w)| \leq k\}. \tag{4.4}$$

Figure 4.2 shows vertex v (black circle), the first ring around v (path formed by green circles), the second ring (path formed by blue circles), and k-th ring (path formed by yellow circles). All these vertices correspond to the neighborhood $V_k(v)$. The method to calculate k will be explained later in this section.

We calculate the centroid of $V_k(v)$ and translate the set of points so the centroid is in the origin of the 3D coordinate system. Then, we compute the best fitting plane to the translated points. To do so, we apply Principal Component Analysis to the set of points and we choose the eigenvector with the lowest associated eigenvalue as the normal of the fitting plane. In our opinion, applying PCA is a better choice than the least square fitting because the assumption $z = f(x, y)$ does not have a good behavior when the data do not exhibit such functional characteristic.

The set of points is rotated so that the normal of the fitting plane is the z-axis. As we choose the less principal component as normal, the points exhibit a good spread in the XY-plane after rotation and therefore we can only work in XY-plane to calculate the derivatives. As final step before calculating derivatives, we translate the set of points so that the point v is in the origin of the XY-plane. This step will facilitate the further analysis.

To calculate derivatives, we fit a quadratic surface to the set of transformed points. Using least square method, we find a paraboloid of the form:

$$z = f(x, y) = \frac{p_1}{2}x^2 + p_2xy + \frac{p_3}{2}y^2 + p_4x + p_5y + p_6. \tag{4.5}$$

We chose a quadratic surface with only six terms because it represents a paraboloid. That is, it is the best choice if we need a function of two variables with quadratic terms. Adding more terms implies that it is possible to fit a more complex surface. However, more complex surfaces do not have well-defined derivatives in certain points of the domain. In addition, we needed a simple expression in order to apply the derivatives.

As we are interested in derivatives in the point v, one could directly evaluate the derivatives of $f(x, y)$ in the point $(0, 0)$, i.e.:

$$f_x = \left. \frac{\partial f(x,y)}{\partial x} \right|_{x=0} \tag{4.6}$$

$$f_y = \left. \frac{\partial f(x,y)}{\partial y} \right|_{y=0} \tag{4.7}$$

The above expressions should be a good estimate of derivatives. However, these can be influenced by noise. Instead, we propose to apply a Gaussian function as proposed originally by Harris and Stephens [50]. However, a difficulty arises because in the original expression the derivatives are discrete functions and our derivatives are continuous functions. To address this problem, we propose to apply the integration of the derivatives with a continuous Gaussian function as follows:

$$A = \frac{1}{2\sigma^4\pi} \int_{\mathbb{R}^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} . \left(\frac{\partial f(x,y)}{\partial x} \right)^2 \mathrm{d}x\mathrm{d}y \tag{4.8}$$

$$B = \frac{1}{2\sigma^4\pi} \int_{\mathbb{R}^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} . \left(\frac{\partial f(x,y)}{\partial y} \right)^2 \mathrm{d}x\mathrm{d}y \tag{4.9}$$

$$C = \frac{1}{2\sigma^4\pi} \int_{\mathbb{R}^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} . \left(\frac{\partial f(x,y)}{\partial x} \right) \left(\frac{\partial f(x,y)}{\partial y} \right) \mathrm{d}x\mathrm{d}y \tag{4.10}$$

where σ is a constant, which defines the support of the Gaussian function and the factor $1/2\sigma^4\pi$ is a normalization value.

Using calculus, we can reduce the expressions to

$$A = \frac{p_4^2}{\sigma^2} + p_1^2 + p_2^2 \tag{4.11}$$

$$B = \frac{p_5^2}{\sigma^2} + p_2^2 + p_3^2 \tag{4.12}$$

$$C = \frac{p_4 p_5}{\sigma^2} + p_1 p_2 + p_2 p_3 \tag{4.13}$$

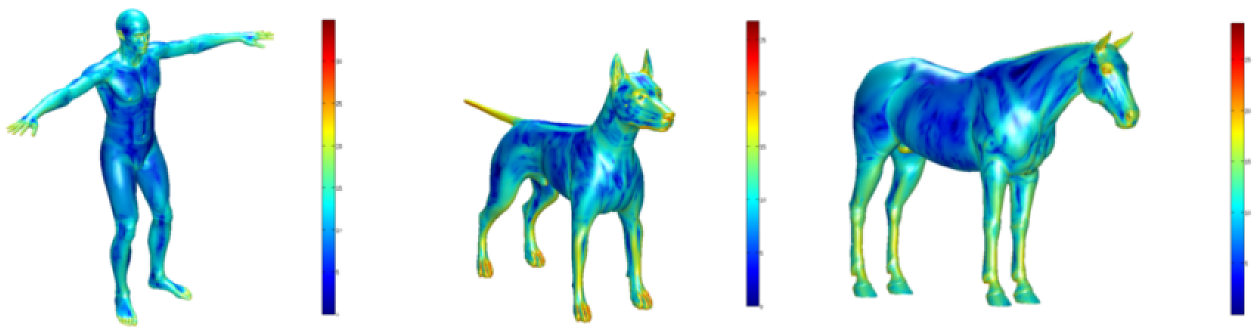


Figure 4.3: Harris 3D operator plotted as a saliency value for each vertex. Note how the high values are present in discriminative regions of the meshes. See in color for better visualization.

Finally, we can formulate the matrix E associated with the point v using the previously calculated values:

$$E = \begin{pmatrix} A & C \\ C & B \end{pmatrix} \tag{4.14}$$

The Harris operator value in the point v is calculated as in Eq. 4.3. This operator can also be seen as a saliency value associated to each vertex in the mesh. It is expected that higher values of the operator correspond to outstanding vertices on the mesh. To exemplify this point, see Fig. 4.3 where the Harris operator is plotted on the surface of some objects. Note the concentration of high operator values in distinctive regions of the meshes.

Evaluation of Integrals

In this section, we provide a detailed evaluation of the integrals for computing values A, B, and C. Note that in the Equations 4.8, 4.9 and 4.10, the integrals are evaluated regarding the derivatives of $f(x, y)$ (Eq. 4.5). Hence, we first calculate the derivatives:

$$\begin{aligned} \frac{\partial f(x,y)}{\partial x} &= p_1x + p_2y + p_4 \\ \frac{\partial f(x,y)}{\partial y} &= p_2x + p_3y + p_5 \end{aligned} \tag{4.15}$$

Now we calculate the products of derivatives:

$$\left(\frac{\partial f(x,y)}{\partial x}\right)^2 = p_1^2x^2 + 2p_1p_2xy + 2p_1p_4x + p_2^2y^2 + 2p_2p_4y + p_4^2 \tag{4.16}$$

$$\left(\frac{\partial f(x,y)}{\partial y}\right)^2 = p_2^2x^2 + 2p_2p_3xy + 2p_2p_5x + p_3^2y^2 + 2p_3p_5y + p_5^2 \tag{4.17}$$

$$\left(\frac{\partial f(x,y)}{\partial x}\right)\left(\frac{\partial f(x,y)}{\partial y}\right) = p_1p_2x^2 + (p_1p_3 + p_2^2)xy + (p_1p_5 + p_2p_4)x \\ + p_2p_3y^2 + (p_2p_5 + p_3p_4)y + p_4p_5 \tag{4.18}$$

Note that the terms x^2, xy, y^2, x, and y are always involved in the products. Therefore, we will only show the evaluation of integral 4.8. Evaluations for 4.9 and 4.10 can be evaluated similarly.

Using the linearity property of integrals, the evaluation of integral 4.8 can be reduced to simpler integrals as follows (we omit the normalization value here to facilitate the evaluation, and we consider it back for the final result):

$$\begin{aligned}\int_{\mathbb{R}^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} \cdot \left(\frac{\partial f(x,y)}{\partial x}\right)^2 \mathrm{d}x\mathrm{d}y =& p_1^2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x^2 \cdot \mathrm{d}x\mathrm{d}y + \\ & 2p_1p_2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot xy \cdot \mathrm{d}x\mathrm{d}y + \\ & 2p_1p_4 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x \cdot \mathrm{d}x\mathrm{d}y + \\ & p_2^2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot y^2 \cdot \mathrm{d}x\mathrm{d}y + \\ & 2p_2p_4 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot y \cdot \mathrm{d}x\mathrm{d}y + \\ & p_4^2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \mathrm{d}x\mathrm{d}y\end{aligned} \tag{4.19}$$

1. **Evaluating** $p_1^2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x^2 \cdot \mathrm{d}x\mathrm{d}y$
   We make a change to polar coordinates to evaluate this integral. We replace $r^2 = x^2+y^2$, $x = r\cos\theta$, $y = r\sin\theta$ and $\mathrm{d}x\mathrm{d}y = r\mathrm{d}r\mathrm{d}\theta$. This changes gives us

$$\begin{aligned}p_1^2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x^2 \cdot \mathrm{d}x\mathrm{d}y &= p_1^2 \int_0^{2\pi} \int_0^{\infty} \mathrm{e}^{-r^2/2\sigma^2} \cdot r^3 \cdot \cos^2\theta \mathrm{d}r\mathrm{d}\theta \\ &= p_1^2 \int_0^{2\pi} \left(\int_0^{\infty} \mathrm{e}^{-r^2/2\sigma^2} \cdot r^3 \mathrm{d}r\right) \cos^2\theta \mathrm{d}\theta\end{aligned} \tag{4.20}$$

The integral in parentheses is evaluated in Appendix A.0.6, hence we replace its value

$$\begin{aligned}
p_1^2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x^2 \cdot \mathrm{d}x\mathrm{d}y &= p_1^2 \int_0^{2\pi} 2\sigma^4 \cos^2\theta \mathrm{d}\theta \\
&= 2\sigma^4 p_1^2 \int_0^{2\pi} \cos^2\theta \mathrm{d}\theta \\
&= \sigma^4 p_1^2 \int_0^{2\pi} (\cos 2\theta + 1)\mathrm{d}\theta \\
&= \sigma^4 p_1^2 \left(\theta + \frac{1}{2}\sin 2\theta\right)\bigg|_0^{2\pi} \\
&= 2\sigma^4 \pi p_1^2
\end{aligned} \tag{4.21}$$

The integral involving the term y^2 in Eq. 4.19 can be evaluated in a similar way.

2. **Evaluating** $2p_1p_2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot xy \cdot \mathrm{d}x\mathrm{d}y$
Similar to the previous integral, we make a change to polar coordinates

$$\begin{aligned}
2p_1p_2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot xy \cdot \mathrm{d}x\mathrm{d}y &= 2p_1p_2 \int_0^{2\pi} \int_0^{\infty} \mathrm{e}^{-r^2/2\sigma^2} \cdot r^3 \cdot \cos\theta \cdot \sin\theta \mathrm{d}r\mathrm{d}\theta \\
&= 2p_1p_2 \int_0^{2\pi} \left(\int_0^{\infty} \mathrm{e}^{-r^2/2\sigma^2} \cdot r^3 \mathrm{d}r\right) \cos\theta \cdot \sin\theta \mathrm{d}\theta
\end{aligned} \tag{4.22}$$

The integral in parentheses is evaluated in Appendix A.0.6, hence we replace its value

$$\begin{aligned}
2p_1p_2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot xy \cdot \mathrm{d}x\mathrm{d}y &= 4p_1p_2\sigma^4 \int_0^{2\pi} \cos\theta \sin\theta \mathrm{d}\theta \\
&= 4p_1p_2\sigma^4 \left(\frac{1}{2}\sin^2\theta\right)\bigg|_0^{2\pi} \\
&= 0
\end{aligned} \tag{4.23}$$

3. **Evaluating** $2p_1p_4 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x \cdot \mathrm{d}x\mathrm{d}y$
In polar coordinates, the integral yields

$$\begin{aligned}
2p_1p_4 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x \cdot \mathrm{d}x\mathrm{d}y &= 2p_1p_4 \int_0^{2\pi} \int_0^{\infty} \mathrm{e}^{-r^2/2\sigma^2} \cdot r^2 \cdot \cos\theta \mathrm{d}r\mathrm{d}\theta \\
&= 2p_1p_4 \int_0^{2\pi} \left(\int_0^{\infty} \mathrm{e}^{-r^2/2\sigma^2} \cdot r^2 \mathrm{d}r\right) \cos\theta \mathrm{d}\theta
\end{aligned} \tag{4.24}$$

The integral in parentheses is evaluated in Appendix A.0.5, hence we replace its value

$$\begin{aligned}
2p_1p_4 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \cdot x \cdot \mathrm{d}x\mathrm{d}y &= 2p_1p_4 \int_0^{2\pi} \frac{\sigma^3\sqrt{2\pi}}{2} \cdot \cos\theta \mathrm{d}\theta \\
&= 2p_1p_4 \left(\frac{\sigma^3\sqrt{2\pi}}{2}\right) \int_0^{2\pi} \cos\theta \mathrm{d}\theta \\
&= 2p_1p_4 \left(\frac{\sigma^3\sqrt{2\pi}}{2}\right) (\sin\theta)|_0^{2\pi} \\
&= 0
\end{aligned} \tag{4.25}$$

The integral involving the term y in Eq. 4.19 can be evaluated in a similar way.

4. **Evaluating** $p_4^2 \int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \mathrm{d}x\mathrm{d}y$
   This integral is evaluated in Appendix A.0.4, and the final result is

$$\int_0^{\infty} \mathrm{e}^{-x/2\sigma^2} \cdot x \cdot \mathrm{d}x = 2\sigma^2\pi p_4^2 \tag{4.26}$$

Finally, summing up the resulting evaluations, we obtain

$$\int_{\mathbb{R}^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} . \left(\frac{\partial f(x,y)}{\partial x}\right)^2 \mathrm{d}x\mathrm{d}y = 2\sigma^4\pi p_1^2 + 2\sigma^4\pi p_2^2 + 2\sigma^2\pi p_4^2 \tag{4.27}$$

and applying the normalization value, the final result is obtained

$$\frac{1}{2\sigma^4\pi} \int_{\mathbb{R}^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} . \left(\frac{\partial f(x,y)}{\partial x}\right)^2 \mathrm{d}x\mathrm{d}y = p_1^2 + p_2^2 + \frac{p_4^2}{\sigma^2}. \tag{4.28}$$

Adaptive Neighborhood Size

Several approaches can be considered to select the number of rings around a point as neighborhood. If the object tessellation is uniform, i.e., almost all triangles in the object have the same size, we can use a constant number of rings to all points, or use the points contained in a ball of radius r and centered in point v. However, in irregular and complex meshes, these methods do not approximate a neighborhood adequately.

To tackle this problem, we propose an adaptive technique. Our method selects a different neighborhood size depending on the tessellation around a point. We consider an object as a graph, similar to the definition of Eq. 4.4. Given a point $v \in V'$, a k-ring around v is the set of points where the length of the shortest path to v is k:

$$ring_k(v) = \{w \in V' \text{ such that } |shortest\_path(v,w)| = k\} \tag{4.29}$$

The distance from a point v to the $ring_k(v)$ is defined as

$$\mathrm{d}_{ring}(v, ring_k(v)) = max_{w \in ring_k(v)} \|v - w\|_2 \tag{4.30}$$

Finally, we define the neighborhood size of a point v as

$$radius_v = \{k \in \mathbb{N} \text{ such that } \mathrm{d}_{ring}(v, ring_k(v)) \geq \delta \text{ and } \mathrm{d}_{ring}(v, ring_{k-1}(v)) < \delta\} \tag{4.31}$$

where δ is a fraction of the diagonal of the object bounding rectangle.

It is important to note that the proposed method always finds a neighborhood to a point, even with complex and irregular tessellations around that point.

In addition, as we provide an approximate extent to each neighborhood, we can use this information to consistently apply the Gaussian function when calculating the Harris response. The extent of the Gaussian is controlled by the parameter σ, which is defined for each vertex as follows

$$\sigma_v = \frac{\delta}{radius_v} \tag{4.32}$$

Therefore, each point has a different support for the applied Gaussian when calculating its operator value and it is consistent with the neighborhood size as well.

Selecting Interest Points

With each vertex associated with its Harris operator value, we propose two ways to select the interest points of a given object. Firstly, we preserve the vertices which are local maximum. To do so, we select a vertex v which holds the following condition:

$$h(v) > h(w), \forall w \in ring_1(v) \tag{4.33}$$

Secondly, we propose two approaches to select the final set of interest points.

- **Select the points with the highest Harris response.** We can pick a constant fraction of interest points depending on the application. In this proposal, we obtain the points with higher saliency and therefore, some portions of the object do not have interest points.
- **Representatives of Interest Points Clusters.** This approach can be used when we want a good distribution of interest points in the object surface. This proposal consists of two steps. First, we sort the pre-selected interest points according to their Harris operator value in decreasing order. Second, we apply Algorithm 4.1 to cluster the sorted points and select the final set of interest points. The selection algorithm

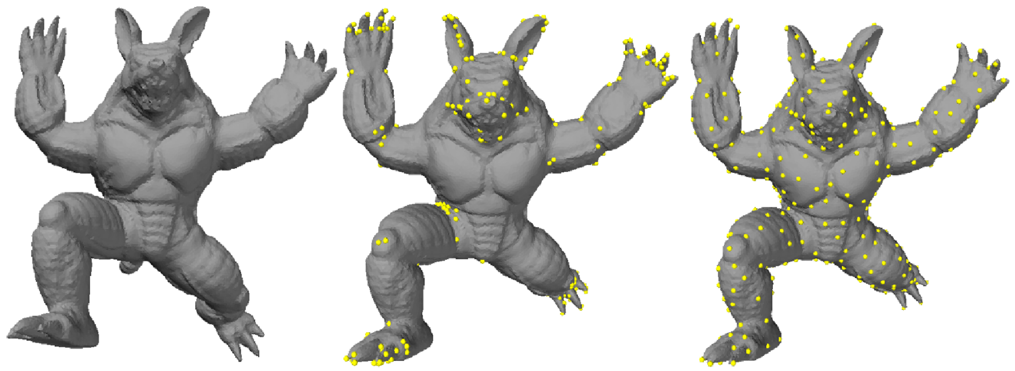


Figure 4.4: Selection options. (A) Armadillo model. (B) Selected points with highest Harris response. (C) Selected points by clustering.

keep a list of chosen points Q which is initially empty. The algorithm scans the set of candidate keypoints according with their highest Harris response. When it finds a new candidate in an area not covered by previous chosen points, the method selects the point and insert it in Q.

Algorithm 4.1 Interest Points Clustering

Require: Set P of pre-selected interest points in decreasing order of Harris operator value
Ensure: Final set of interest points

```
1: Let $Q$ be a set of points
2: $Q \leftarrow \emptyset$
3: for i $\leftarrow$ 1 to $|P|$ do
4:     if $min_{j \in [1,|Q|]} \|P_i - Q_j\|_2 > \rho$ then
5:         $Q \leftarrow Q \cup \{P_i\}$
6:     end if
7: end for
8: Return $Q$
```

The value of ρ can be considered as a fraction of the diagonal of the object bounding rectangle and it has effect in the number of returned interest points.

Figure 4.4 shows the result of the two options to select interest points.

4.1.2 Experimental Evaluation and Discussion

In this section, we show the experimental results of our implementation of the Harris 3D method using a standard benchmark. The presentation of results is divided in two parts. First, we experiment with several aspects and parameters of our method. The objective is to investigate the effect of the parameters on the repeatability of interest points. Second, we compare our method with methods in the state of the art, namely the Heat Kernel Signatures proposed by Sun et al. [136] and the Salient Points detection method proposed by Castellani et al. [28]. Furthermore, we use the SHREC 2010 dataset (Sec. 3.2.1) for our experiments.

Evaluation Methodology

An interest point detection method returns a set of detected points $F(Y) = \{y_k\}_k$ for each shape Y (typically, $|F(Y)| \ll |Y|$). The performance is measured by comparing the interest points computed for transformed shapes and the corresponding null shape.

The quality of the interest points detection was measured using the repeatability criterion. Assuming for each transformed shape Y in the data set the ground-truth dense correspondence to the null shape X to be given in the form of pairs of points $C_0(X, Y) = \{(y'_k, x_k)\}_{k=1}^{|Y|}$ (and same way, $C_0(Y, X)$), an interest point $y_k \in F(Y)$ is said to be repeatable if a geodesic ball of radius R around the corresponding point $x'_k : (x'_k, y_k) \in C_0(X, Y)$ contains an interest point $x_j \in F(X)$. The subset $F_r(Y) \subseteq F(Y)$ of repeatable interest points is given by

$$\begin{aligned} F_{R,X}(Y) = \{y_k \in F(Y) : F(X) \cap B_R(x'_k) \neq \emptyset, \\ (x'_k, y_k) \in C_0(X, Y)\}, \end{aligned} \tag{4.34}$$

where $B_R(x'_k) = \{x \in X : geod(x, x'_k) \leq R\}$ and $geod$ denotes the geodesic distance function in X. The repeatability $rep(Y, X)$ of $F(Y)$ in X is defined as the percentage of interest points from $F(Y)$ that are repeatable,

$$rep(Y, X) = \frac{|F_{R,X}(Y)|}{|F(Y)|}. \tag{4.35}$$

For a transformed shape Y and the corresponding null shape X, the overall feature interest points detection quality was measured as $(rep(Y, X) + rep(X, Y))/2$. The value of $R = 5$ was used in the benchmark. This radius constitutes approximately 1% of the shapes diameter. Interest points without ground-truth correspondence (e.g. in regions in the null shape corresponding to holes in the transformed shape) were ignored.

Analysis of parameter values

In this section, we present the experimental results of our method. We experimented with several aspects and investigated the effect of the parameters on the repeatability of our proposal. Specifically, we evaluated the following aspects:

- **The type and size of local neighborhood.** We tested three options: spatial neighborhood, adaptive neighborhood and ring neighborhood. We are interested in evaluating the effectiveness of each option.

- **The parameter K.** We tested with different values for this parameter in order to investigate its effect in the calculation of Harris response.
- **The type of interest point selection method.**. We only evaluated the method that selects the points with higher response. We intended to figure out the effect of the number of selected interest points in the effectiveness of our algorithm.

It is important to point out that, for all experiments we used a basic configuration of parameters and only the analyzed parameter was changed. Our basic configuration consisted of adaptive local neighborhood with $\delta = 0.01$, $K = 0.04$, and selection of 1% of the number of vertices with higher Harris response as final interest points set. All graphs presented in our results were adequately scaled for a better visualization.

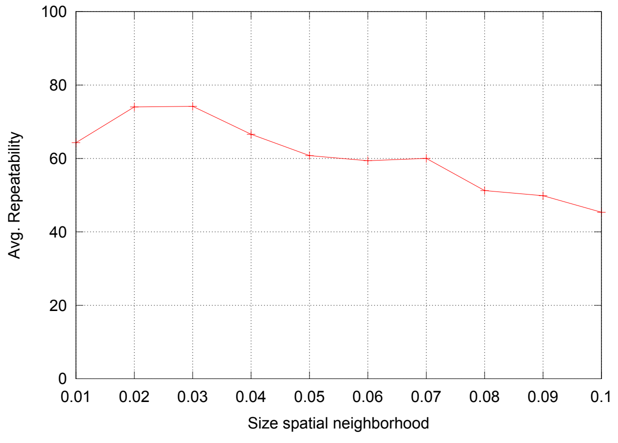
Spatial neighborhoods. Our first experiment consisted in evaluating the repeatability of our method when local neighborhoods were determined as the set of vertices lying inside of a ball centered in the analyzed vertex. We varied the radius of the ball and calculated the average repeatability for all transformations and for all levels. The radii were taken as fraction of the diagonal of the bounding box of the object. It is important to mention that when we use spatial neighborhoods, the σ_v parameter is set to the quarter of the spatial radius. Figure 4.5a shows the results of this experiment.

We varied the fraction of the diagonal in the range [0.01, 0.1][1]. The highest values of repeatability were obtained for 0.02 and 0.03. In order to find the best value, we plot the repeatability curves for these values with respect to the level of transformation (see Fig. 4.5b). Both curves are very similar, with a slight advantage for $fraction = 0.03$.

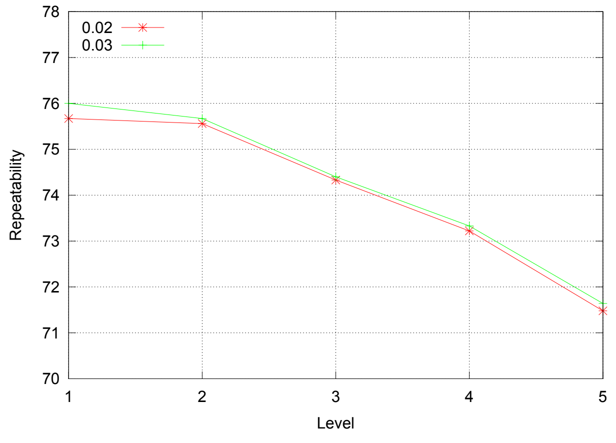
There are two important issues that we should remark. First, the average repeatability decreases with large neighborhoods. This is because large neighborhoods cannot be well fitted by a quadratic surface, and therefore the calculation of derivatives and the Harris response are not robust. Second, as it can be seen in Fig. 4.5b, repeatability always decreases with stronger levels of transformations (except in those where the level does not represent the strength of the transformation). Nevertheless, this behavior was expected.

Table 4.1 shows the repeatability for each transformation and each level for $fraction =$ 0.03. Although there are transformations where, on average, the repeatability is greater than 75% (for example, isometry, topology, holes, micro holes, scale and shot noise), spatial neighborhoods have some problems. On the one hand, vertices belonging to a spatial neighborhood do not necessarily belong to the same local surface around the analyzed vertex. Moreover, the set of vertices could not belong to the same connected component. Having neighborhoods with vertices not belonging to the local surface, the fitting task is not robust, damaging the overall process and therefore the effectiveness. On the other hand, spatial neighborhoods do not ensure a good balance around the analyzed vertex, which is a problem if the models have poor triangulations.

[1]In practice, values below 0.01 represent a very small neighborhood. This does not guarantee to find a surrounding point set for the subsequent computation.



(a)



(b)

Figure 4.5: (a)Repeatability values obtained by selecting spatial neighborhoods. The radii of the ball was chosen as a fraction of the diagonal of the bounding box of the object. Results show average repeatability for the range [0.01, 0.1] for the radii. (b)Average repeatability for spatial neighborhoods with radii 0.02 and 0.03, respectively. The comparison was done by levels of transformation.

Adaptive neighborhoods. Our second experiment consisted in varying the parameter δ with the adaptive local neighborhood approach. Figure 4.6 shows the average repeatability obtained in this experiment. Table 4.2 shows all values of repeatability obtained by the best configuration ($\delta = 0.01$). We tested values smaller than 0.01 for δ; however, the improvement was not significant.

Similarly to the spatial neighborhoods, the average repeatability decreases as the extent of the neighborhoods increases. Large neighborhoods cause an inadequate fitting, affecting the effectiveness of the method. Results show an improvement of more than 15% in almost all entries with respect to those obtained by spatial neighborhoods. This percentage represents approximately 45 repeatable interest points detected in difference, compared to the spatial neighborhood effectiveness. However, although the results improve significantly, there is still visible a rapid fall in the repeatability values for noise transformation, even below those obtained by spatial neighborhoods.

From Table 4.2 we can note some interesting issues. On the one hand, from the transformations where the level represents strength, some of them present only a slight fall of average repeatability between the minimum and maximum levels: topology (0.33%), holes (1.41%) and micro holes (0.08%). On the other hand, from stronger level, the worst performance is obtained by sampling and noise transformations.

The behavior with the sampling transformation is expected because our method relies on selecting a fraction of the number of vertices as interest points. So the number of interest points detected in level 1 is much larger than in level 5. Clearly, the number of repeatable interest points in this transformation cannot be larger than the number of interest points in high levels. Therefore, the repeatability decreases considerably with downsampling. Also, with respect to the noise, the level of distortion of the meshes in stronger levels of this

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	75.77	79.89	78.01	79.45	79.85
Topology	76.25	76.39	76.21	76.19	76.18
Holes	75.56	75.52	75.36	75.34	74.98
Micro holes	76.19	76.17	76.10	75.98	75.92
Scale	82.58	80.01	77.38	74.79	72.29
Local scale	74.33	72.49	69.34	65.46	62.66
Sampling	73.21	71.06	69.35	66.75	59.51
Noise	73.81	73.14	71.78	69.94	67.48
Shot noise	76.32	76.36	76.04	76.09	75.90
Average	76.00	75.67	74.40	73.33	71.64

Table 4.1: Repeatability of our method using spatial neighborhood with $fraction = 0.03$. Average number of detected points: 303.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	93.59	95.14	94.43	95.05	95.18
Topology	93.48	93.43	93.25	93.16	93.15
Holes	92.08	92.04	91.60	91.25	90.67
Micro holes	93.59	93.59	93.56	93.55	93.51
Scale	94.29	93.80	93.42	93.04	92.60
Local scale	93.49	92.89	91.27	88.81	86.55
Sampling	92.23	90.40	87.83	84.52	77.98
Noise	92.33	81.83	72.51	66.75	63.44
Shot noise	93.54	92.60	91.27	89.93	88.20
Average	93.18	91.75	89.90	88.45	86.81

Table 4.2: Repeatability of our method using adaptive neighborhoods with $\delta = 0.01$. Average number of detected points: 303.

transformation is high, causing a considerable deformation in the shapes.. Surely, local neighborhoods are affected considerably and the process of fitting is not robust. We argue that the decrement with respect to spatial neighborhoods is due to the use of the connectivity when collecting the rings around the analyzed vertex. As the noise is applied in the direction of the normal of each vertex, the same parameter δ in different levels of transformations can determine neighborhoods of considerable different sizes. Obviously, for the reasons explained before, the repeatability of our method is affected.

Ring neighborhoods. The third experiment evaluated the repeatability with different numbers of rings selected as neighborhood. It is important to mention that when we use a ring neighborhood, the σ_v parameter is set to a quarter of the maximum graph distance between v and a boundary vertex. Figure 4.7 shows the result for this experiment. Table 4.3 shows all repeatability values.

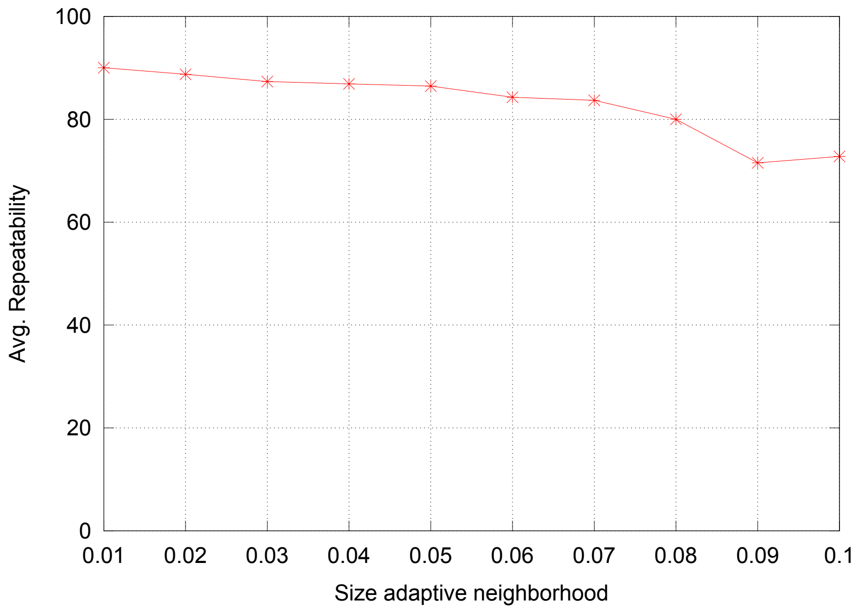


Figure 4.6: Average repeatability for adaptive neighborhoods with several δ values.

From Fig. 4.7, we can observe the same effect on the size of the neighborhood. With larger neighborhoods, average repeatability begins to decrease systematically. The interesting thing about this experiment is to note that a very small neighborhood gave the best results. One reason could be that, with small neighborhoods, the fitting step is robust and thus, derivatives are well calculated. Nevertheless, small neighborhoods are unstable in presence of noise or another distortion transformation. Still, repeatability values for stronger levels of noise improve up to 13% with respect to spatial neighborhoods and up to 15% with respect to adaptive neighborhoods.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	95.86	96.74	96.51	96.78	96.79
Topology	95.92	95.92	95.84	95.75	95.70
Holes	93.60	93.77	93.74	93.83	93.48
Micro holes	95.86	95.86	95.88	95.89	95.93
Scale	96.76	96.54	96.24	95.93	95.39
Local scale	95.92	94.97	93.49	91.35	89.01
Sampling	94.69	93.55	92.19	89.05	80.26
Noise	92.97	92.07	90.82	89.71	88.54
Shot noise	95.99	95.57	94.90	93.62	92.41
Average	95.28	95.00	94.40	93.55	91.95

Table 4.3: Repeatability of our algorithm using one ring neighborhood. Average number of detected points: 303.

Parameter K The parameter K is used in Eq. 4.3 to calculate the Harris response for a given vertex. This parameter needs to be tuned experimentally. We varied the parameter in

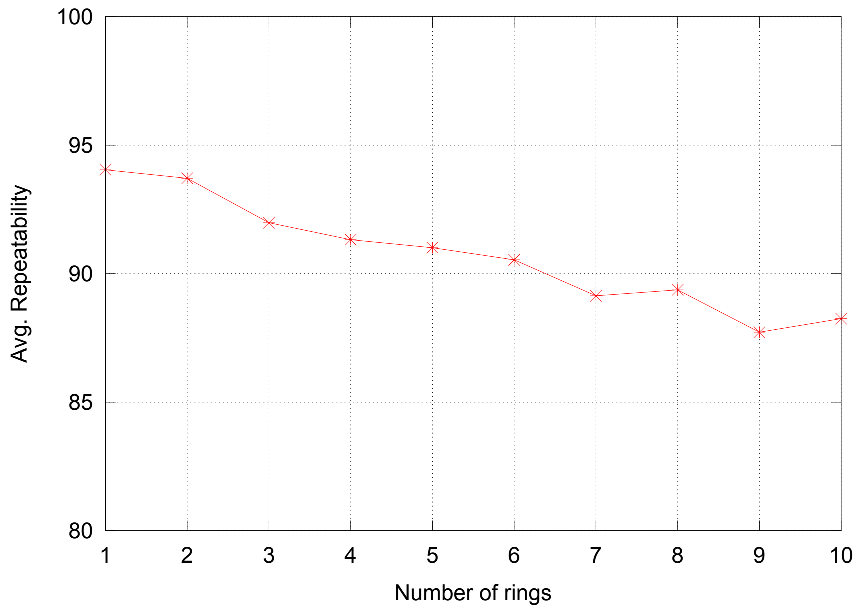


Figure 4.7: Average repeatability with ring neighborhoods sizes taken from the range [1, 10].

the range [0.04, 0.1] and tested the average repeatability. Figure 4.8 shows the results. In our implementation, the best result was for $K = 0.07$. However, the improvement was not significant with respect to our default value $K = 0.04$ (approximately 0.34%, representing almost 2 repeatable interest points of difference).

Interest point selection In Section 4.1.1, we proposed two options for the final selection of interest points. The clustering approach is interesting from the point of view of applications requiring points well distributed over the whole surface. Nevertheless, as the process is based on grouping vertices with high responses, this step is not necessarily robust according to the repeatability criterion. Therefore, we did not consider this approach in our evaluation.

On the other hand, selection of vertices with higher response is interesting because the number of selected vertices is an important issue in applications. For example, in shape matching, we might be interested in only a few points, as the efficiency of matching is closely related to the number of points to be matched. So, we evaluated the effect of reducing the number of selected vertices.

Our method selects a fraction of the number of vertices as interest points, so the smaller is the fraction, the fewer interest points are selected. We varied the fraction in the range [0.001, 0.01]. Figure 4.9 shows our results.

Clearly, the average repeatability increases as the number of interest points is increased. This trend is maintained for values of fractions larger than 0.01. An important aspect to note is that by reducing the number of interest points in half, with respect to the value of fraction 0.01, the average repeatability decreases approximately in 6.34%. This is an advantage because if we need to apply subsequent processes, we can opt for reducing the number of interest points selected, at expense of slightly reducing the robustness of the delivered points.

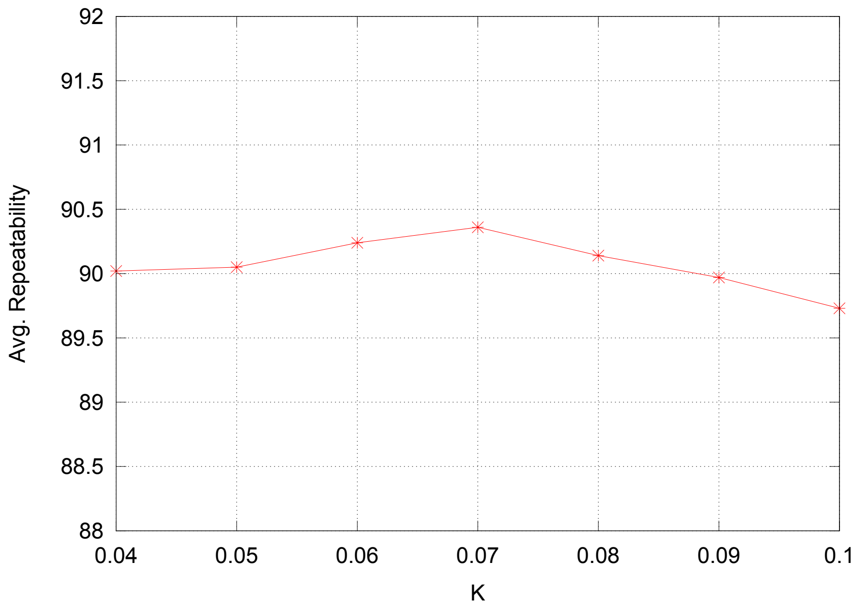


Figure 4.8: Effect of varying K in average repeatability.

The number of interest points will finally depend on the application and the trade-off between robustness and efficiency.

As an additional test, we combine the parameter values with the best effectiveness, obtaining a slight improvement. Table 4.4 shows the repeatability values for all transformations in all levels. Compared to Table 4.3, on average results improve for all levels.

Comparison with other methods

In order to compare our method with the state of the art, we selected two recent methods for detecting interest points on 3D meshes: Heat Kernel local maximum-based feature detector [136] (hereafter, we will refer this method as HKS due to its close relation to the Heat Kernal Signatures) and Salient Points [28]. Next, we specify the configuration used for these methods:

Heat Kernel local maximum-based feature detector (HKS). As this method relies on the eigendecomposition of the Laplace-Beltrami operator on the mesh, we needed to simplify the meshes to approximately 10,000 vertices (we used Garland's method [42]). Interest points were computed on the simplified meshes and mapped back to the original mesh by using the nearest neighbor vertex. We used the value $t = 0.1$ from the total area of the surface to evaluate the Heat Kernel local maximum and 2-ring neighborhood in order to detect interest points. We present three variations:

- HKS1: No pre-processing step was used. We implemented this variant based on the original HKS implementation.[2]

[2]http://www.geomtop.org/sunjian/software/hks.html

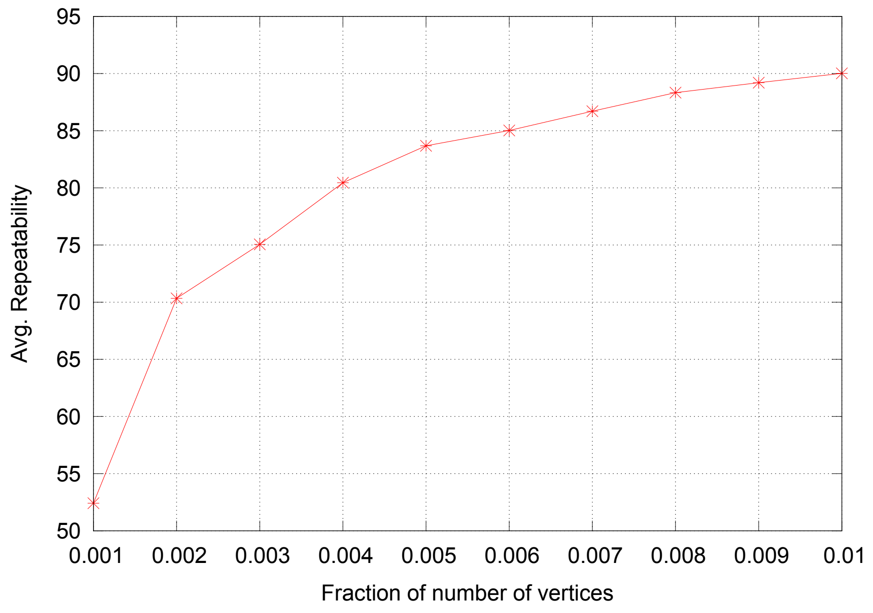


Figure 4.9: Effect of reducing the number of interest points in average repeatability.

- HKS2: The Geomagic sotfware was used for removing non-manifold edges and faces were consistently oriented. Results for this variant were taken from the original report [14].
- HKS3: Filtering using persistent homology was used to discard unstable feature points. Results were also taken from [14].

Salient Points (SP). We chose the best performance of this method from the SHREC feature detection benchmark.

Tables 4.5, 4.6 and 4.7 show the repeatability obtained by the variants of the HKS method. Table 4.8 shows the results of the Salient Points method. All comparison were done with respect to our best results shown in Table 4.4. Our method is represented by H3D.

As can be seen, our method outperformed the HKS1 variant without pre-processing and the Salient Points method. The benefit is consistent in each entry of the table, which is an important result regarding the relevance of the techniques compared. With respect to HKS2 and HKS3, these variants present significant improvements. However, they need well-formed shapes in order to work properly, which affects their efficiency considerably.

Table 4.9 presents the best method for each entry in the repeatability table. It can be seen that Harris 3D method outperforms the rest in stronger levels (4-5). The Heat Kernel based method is predominant in isometry, scale and sampling transformations. In the isometry transformation, the effectiveness of this method (HKS3 variant) is perfect with an average repeatability of 100%. A similar scenario takes place in the scale transformation, where for small scales (levels 1-2-3) the average repeatability is 100%. For the sampling transformation, the average repeatability is near to perfect too.

The reason for the good performance of the Heat Kernel based method in the aforemen-

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	96.01	96.73	96.26	96.60	96.62
Topology	96.01	95.97	95.82	95.73	95.71
Holes	94.62	94.43	94.10	94.01	93.81
Micro holes	96.01	96.01	95.98	95.96	95.95
Scale	97.06	96.89	96.28	95.62	94.94
Local scale	96.24	94.96	93.40	91.26	88.84
Sampling	95.31	93.62	92.08	89.13	80.42
Noise	93.09	92.58	91.59	90.33	88.79
Shot noise	96.03	95.66	95.00	93.83	92.79
Average	95.60	95.21	94.50	93.61	91.99

Table 4.4: Repeatability of our algorithm using combination of values with the best effectiveness. Average number of detected points: 303.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	83.20	87.39	88.44	87.29	87.21
Topology	79.73	81.27	81.49	81.08	80.94
Holes	80.15	77.57	75.86	73.35	71.16
Micro holes	81.02	80.76	80.68	80.43	80.50
Scale	79.99	79.78	79.90	80.18	80.36
Local scale	80.38	80.65	78.84	75.55	72.99
Sampling	82.70	82.23	82.66	82.04	78.99
Noise	75.80	74.55	72.37	69.34	68.23
Shot noise	79.96	81.14	81.15	80.07	78.77
Average	80.33	80.59	80.15	78.82	77.68

Table 4.5: Repeatability of HKS1 feature detection algorithm. Average number of detected points: 35.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	98.08	98.72	98.01	97.88	98.04
Topology	97.44	96.10	92.26	91.22	88.64
Holes	91.48	90.60	86.78	83.73	81.86
Micro holes	98.08	96.69	96.00	95.52	94.87
Scale	99.36	99.36	98.50	97.90	97.68
Local scale	98.08	94.83	90.09	83.05	78.31
Sampling	97.05	97.88	97.39	96.27	92.35
Noise	95.30	92.78	91.67	89.24	87.62
Shot noise	98.08	96.22	93.39	90.45	87.32
Average	96.99	95.91	93.79	91.70	89.63

Table 4.6: Repeatability of HKS2 feature detection algorithm. Average number of detected points: 23.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	100.00	100.00	100.00	100.00	100.00
Topology	94.44	90.38	87.45	88.70	85.76
Holes	80.54	79.00	75.25	72.10	69.99
Micro holes	100.00	100.00	98.15	96.58	95.64
Scale	100.00	100.00	100.00	98.61	97.78
Local scale	97.44	96.79	93.02	87.25	82.90
Sampling	100.00	100.00	100.00	100.00	96.20
Noise	100.00	95.19	93.16	89.37	85.77
Shot noise	100.00	95.30	90.03	82.10	74.38
Average	96.94	95.19	93.01	90.52	87.60

Table 4.7: Repeatability of HKS3 feature detection algorithm. Average number of detected points: 23.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	86.17	87.42	87.24	87.76	88.15
Topology	86.18	85.63	85.58	85.56	85.56
Holes	85.72	85.10	84.34	83.56	82.58
Micro holes	68.52	62.27	57.96	54.75	51.99
Scale	89.80	88.28	86.82	85.14	83.70
Local scale	85.73	84.97	84.48	83.33	82.12
Sampling	85.02	83.15	82.21	79.94	77.61
Noise	87.31	85.43	83.28	81.36	79.40
Shot noise	85.95	84.42	82.77	81.76	81.23
Average	84.49	82.96	81.63	80.35	79.15

Table 4.8: Repeatability of SP algorithm. Average number of detected points: 409.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	HKS3	HKS3	HKS3	HKS3	HKS3
Topology	HKS2	HKS2	H3D	H3D	H3D
Holes	H3D	H3D	H3D	H3D	H3D
Micro holes	HKS3	HKS3	HKS3	HKS3	H3D
Scale	HKS3	HKS3	HKS3	HKS3	HKS3
Local scale	HKS2	HKS3	H3D	H3D	H3D
Sampling	HKS3	HKS3	HKS3	HKS3	HKS3
Noise	HKS3	HKS3	HKS3	H3D	H3D
Shot noise	HKS3	HKS2	H3D	H3D	H3D
Average	HKS3	HKS3	HKS3	H3D	H3D

Table 4.9: Methods with the best performance by transformations and strengths. HKS - Heat Kernel local maximum. H3D - Harris 3D.

tioned transformations is its intrinsic definition. This property allows it to appropriately define a characteristic shape based on the spectrum of the Laplace-Beltrami operator, and therefore it is robust against isometry and rigid transformations such as scaling. On the other hand, it is also robust to different samplings of the input mesh, as the interest points are selected for being points with large values of Heat Kernel Signatures in large times. That is to say, the interest points selection process is robust to different tessellations.

Differently, in transformations that deform the local structure of shapes, Harris 3D obtained the best results. Firstly, there is a total predominance of our method with respect to the holes transformation. Secondly, our method outperforms the rest in most levels (3-5) for topology, local scale and shot noise transformations. Finally, it is interesting that our method is also the best in stronger levels for micro holes and noise transformations. Averages in Table 4.9 represent the majority in each level, so we can observe the predominance of Harris 3D in levels 4 and 5.

Figure 4.10 shows some examples of interest points over a class of shape of the SHREC feature detection benchmark using Harris 3D.

4.2 Key-Components: Regions of Interest on Meshes

A basic and important task is to find interesting structures in representations such as 3D point clouds or meshes. Many proposals have been presented to detect interest points (also called keypoints) on 3D data. Regarding meshes, a keypoint is a point on the mesh with a local outstanding structure. As such, the keypoints represent interesting information at fine scales and thus, they could be sensitive to noise and other transformations. Therefore, it is required to find larger and interesting structures to overcome the problems at fine scales.

In this section, we propose an algorithm to detect features at a coarse level on meshes. Our motivation is that larger structures are more resilient to local changes, while allowing

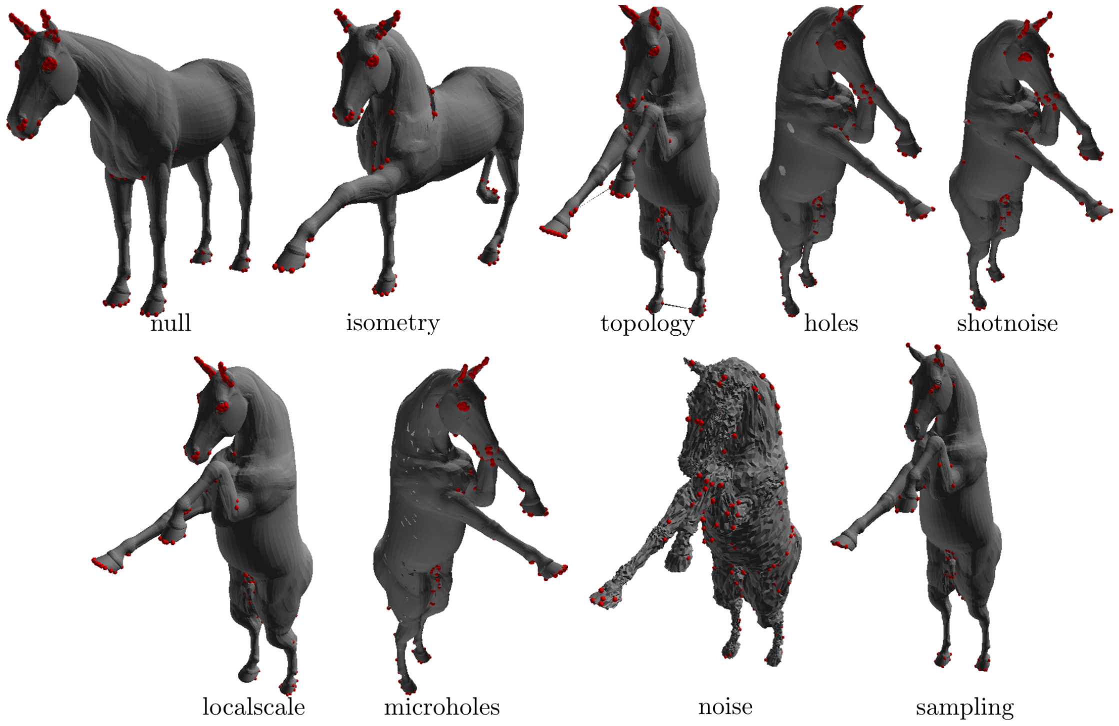


Figure 4.10: Shapes with interest points. Interest points are represented with small red balls.

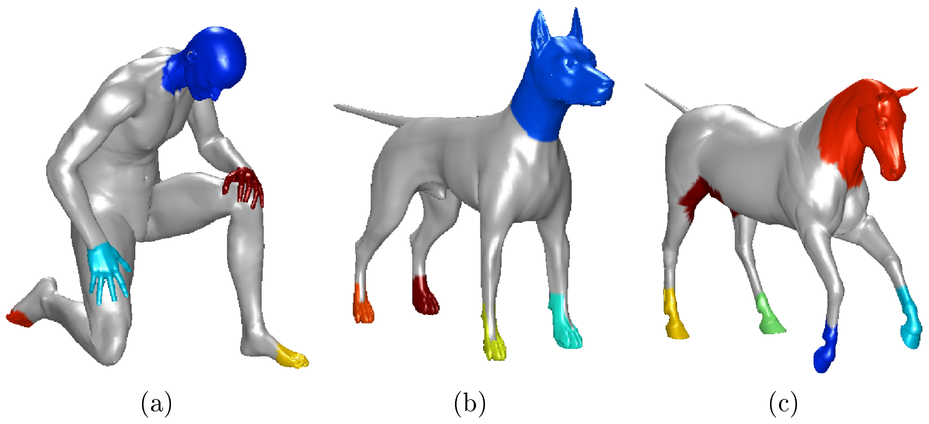


Figure 4.11: Key-components detected on 3D meshes using our method.

us to reduce the amount of information to represent 3D meshes in retrieval and recognition tasks. The idea is to decompose a 3D mesh in a set of components, which should be consistently found in meshes regardless the applied transformation. In addition, the number of components should be much less than the number of keypoints, so using the components in subsequent tasks would be efficient.

We introduce the term *key-component* as a region on a 3D mesh where there are a lot of discriminative local features (see Fig. 4.11). In such way, key-components correspond to regions with high protrusion and they are therefore distinctive for the object. Additionally, the size of the salient region is determined by a clustering algorithm used to find agglomerations of keypoints in a sense of geodesic closeness. Moreover, key-components will be useful to the extent that they are repeatable and robust against several transformations.

Our method is inspired by the cognitive theory of saliency of visual parts [54]. This theory studied the important role of object parts in high level vision tasks. In addition, it exposed the characteristics of parts in order to be considered as salient. To this respect, the theory formulated the existence of three key aspects for parts: the relative size, the protrusion and the strength of the boundary. We present a procedure to detect salient parts or regions on 3D meshes trying to cover the aforementioned aspects. More specifically, our method selects regions with agglomerations of keypoints as key-components, so they are expected to have a high protrusion. Additionally, our results confirm the fact that the size is important to define robust salient regions.

Our method differs from mesh segmentation methods as it computes a non-complete decomposition of a mesh while is aware of the local features present in the components. Even more, if we would like to establish a comparison with image processing tasks, we would say that mesh segmentation is related to image segmentation while our method is more similar to image saliency detection [48]. In other words, we are interested in detecting portions of the mesh which are distinctive enough and robust and repeatable against transformations rather than decomposing the whole mesh.

Our method consists of three steps: keypoint detection, clustering in the geodesic space, and key-component extraction. Our proposal is based on the detection of interest points, which can be effectively used for detecting stable components on meshes. Here, we use the Harris 3D method presented in Sec. 4.1. Figure 4.12 depicts the pipeline of our method to detect key-components.

4.2.1 Clustering in the Geodesic Space

Key-components are those regions on the mesh in which there is a high concentration of local features. One way to measure the concentration is using the geodesic distances between the keypoints, and therefore grouping them according to their closeness in terms of this kind of distance. Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of keypoints previously detected, our goal is to find partitions $S_{\mathrm{i}} \subset S$, $\mathrm{i} = 1 \ldots m$ over the set of keypoints S in order to fulfill the following properties:

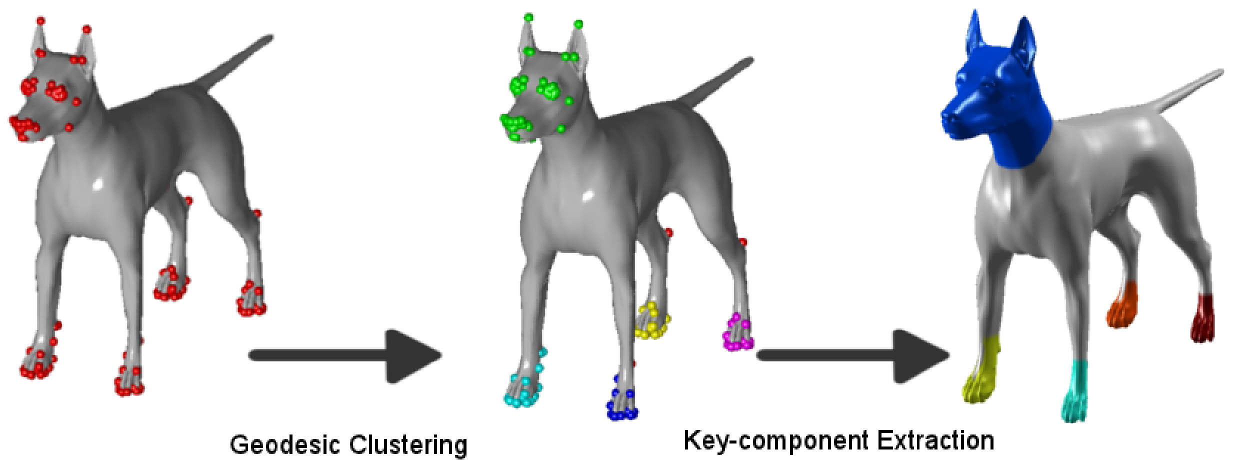


Figure 4.12: The process to detect key-components. First, Harris 3D keypoints are detected on a shape. Second, a clustering in the geodesic space is performed to find concentrations of keypoints. Finally, each geodesic cluster generates a region.

1. $\mathrm{d}_{geod}(x, y) \leq R$, $\forall x, y \in S_{\mathrm{i}}$.
2. $\mathrm{d}_{geod}(x, y) \geq T$, $\forall x \in S_{\mathrm{i}}$ and $\forall y \in S_j$, $\mathrm{i} \neq j$.
3. $|S_{\mathrm{i}}| \geq N, \forall \mathrm{i}$

Property 1 suggests that elements in a subset S_{i} share approximately the same location on the mesh (threshold R controls the proximity permitted). Property 2 states that two subsets S_{i} and S_j cannot be very close to each other (threshold T controls how far two subset should be). Property 3 establishes that each partition should contain a minimum number of keypoints to be considered as a valid partition. In addition, we consider a non-complete partitioning of the initial set S. Obviously, there may be keypoints which meet the two first properties, but not the third. This is because some interest points could be isolated, and therefore they would not belong to any partition. Moreover, isolated keypoints could have been selected due to noise. It is clear that, in order to detect consistent components on meshes, we need to discard isolated keypoints.

In practice, we need to consider a clustering process regarding the geodesic distances between keypoints. In order to accomplish this goal, our method computes a set $P \in \mathbb{R}^2$, in which euclidean distances between elements in P approximately preserve the geodesic distances between elements in S. That is, we need to find the set P such that

$$P = \underset{p_1,\dots,p_n}{\operatorname{argmin}} \sum_{\mathrm{i}<j} (\|p_{\mathrm{i}} - p_j\| - \mathrm{d}_{geod}(s_{\mathrm{i}}, s_j)) \tag{4.36}$$

where each $p_{\mathrm{i}} \in \mathbb{R}^2$ corresponds to the keypoint $s_{\mathrm{i}} \in S$.

This problem is commonly called Multidimensional Scaling [11] and it is used to embed points in one space into another (generally for better visualization). The optimization problem in the Eq. 4.36 is a minimum-distortion problem and can be solved with an iterative method which takes a random sampling in the destination space as starting set P. The

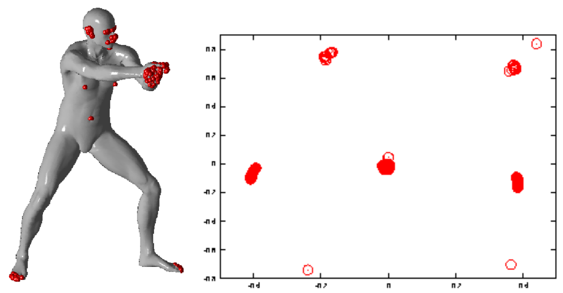

(a) (b)

Figure 4.13: Left: Shape with keypoints. Right: Multi-dimensional scaling of the keypoints.

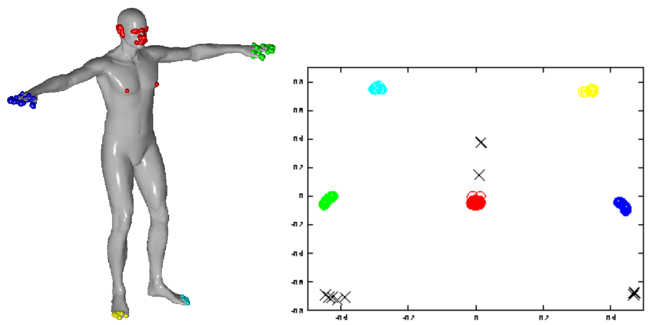

(a) (b)

Figure 4.14: Left: Shape with cluster of keypoints. Right: multi-dimensional scaling of the keypoints. Points represented as crosses do not belong to any cluster.

method used in this work was the SMACOF algorithm [11]. In addition, for approximating the geodesic distances, we used the fast marching method [71]. Figure 4.13 shows the resulting set of 2D points applied on a set of keypoints. Note how the resulting points represent the distribution of keypoints on the mesh.

Next, we apply a clustering algorithm over the set P in order to define the partitioning of S. We proposed a clustering algorithm derived from Leow and Li [76] (See algorithm 4.2). The algorithm iterates over two steps: assignment and update. The assignment step (lines 4-18) performs in point-wise manner. Firstly, the distance to the closest cluster is obtained. If the distance is greater than T (the inter-cluster threshold), we create a new cluster (according to the property 2). Otherwise, if the distance is not greater than R (the intra-cluster threshold), the point p is inserted in the corresponding cluster (according to property 1). After the assignment step, each point belongs to a cluster. Subsequently, the update step (lines 19-26) computes the new centroids for clusters that meet the property 3. Otherwise, clusters with a few points are removed, and their points are inserted back in P to be further processed. Note that the algorithm could converge before the last iteration, however we opt for using a number of iterations as stop criterion. In all experiments presented in Section 4.2.3, we set $Iter = 10$. This value was set empirically from the observation that, on average, the clustering algorithm converges in 6-8 iterations. Figure 4.14 shows the groups of keypoints found using our algorithm.

Algorithm 4.2 Adaptive Clustering

```
Require: Set of points $P$
Require: Inter-cluster distance $T$
Require: Intra-cluster distance $R$
Require: Minimum number of elements per cluster $N$
Require: Number of iterations $Iter$
Ensure: Set of clusters $C = \{C_1, \ldots, C_m\}$

1:  Let $C$ a set of clusters
2:  $C \leftarrow \emptyset$
3:  for $j \leftarrow 1$ to $Iter$ do
4:      for each $p \in P$ do
5:          if $C = \emptyset$ then
6:              $\mathrm{d} = 2T$
7:          else
8:              $C* = \arg\min_{C_\mathrm{i} \in C} \mathrm{di}st(p, C_\mathrm{i})$
9:              d = distance from $p$ to $C*$
10:         end if
11:         if $\mathrm{d} > T$ then
12:             $C_{new} = \{p\}$
13:             $C \leftarrow C \cup C_{new}$
14:             $P \leftarrow P - \{p\}$
15:         else if $\mathrm{d} \leq R$ then
16:             $C* \leftarrow C * \cup \{p\}$
17:             $P \leftarrow P - \{p\}$
18:         end if
19:     end for
20:     for i $\leftarrow 1$ to $|C|$ do
21:         if $|C *| \geq N$ then
22:             Update centroid for $C*$
23:         else
24:             $P \leftarrow P \cup C*$
25:             $C \leftarrow C \backslash \{C*\}$
26:         end if
27:     end for
28: end for
29: Return $C$
```

4.2.2 Key-component Extraction

The starting point to extract mesh components is the set of clusters previously computed. Each cluster will generate a component comprising the region of the mesh where the keypoints are located. Now, we need a criterion to decide how large this region will be. In addition, the selected region should be large enough to include all the keypoints in the cluster.

We start by defining the geodesic center of each cluster. The idea is to determine the point on the mesh which is the center of the distribution of a cluster. This point could be used as the center of the region to be extracted as component. We can take advantage of the transformed set of points P in order to accomplish this goal. The geodesic center of a cluster is a point on the mesh whose mapped version in $\mathbb{R}^2$ is near to the centroid of the cluster of the transformed points. To solve this, we choose the closer point to the centroid in $\mathbb{R}^2$ as the geodesic center. Note that the selected point is only an approximation of the real geodesic center, as our method is selecting a keypoint (finding the real geodesic center is a hard task as we would have had to map every point on the mesh into the $2D$ space, which is impossible in practical terms). Formally, let C_{i} be the set of 2D points corresponding to the set S_{i} of keypoints. The geodesic center of S_{i} is defined as follows:

$$\hat{c}_{\text{i}} = \{s_j \in S_{\text{i}} | c_j = \underset{c \in C_{\text{i}}}{\operatorname{argmin}} \|c - centroid(C_{\text{i}})\|\} \tag{4.37}$$

where $p_j \in \mathbb{R}^2$ corresponds to $s_j \in S$.

Now, we need to define a size for the component. To do that, our method computes the smallest sphere containing every keypoint in a cluster. This is a classic problem in computational geometry and it can be efficiently solved using linear programming [88]. The output of this tasks is a pair $(o_{\text{i}}, r_{\text{i}})$ representing the center and the radius of the sphere enclosing the keypoint in the cluster S_{i}.

Once the geodesic center $\hat{c}_{\text{i}}$ and the sphere $(o_{\text{i}}, r_{\text{i}})$ have been computed, we propose a region growing algorithm on the mesh. Our initial seed is the vertex $\hat{c}_{\text{i}}$ and the constraint for the growing step is imposed by the sphere. Algorithm 4.3 details this procedure. Briefly, the region growing algorithm starts from the geodesic center $\hat{c}_{\text{i}}$ and inserts the neighbor vertices into the queue. Each time a vertex is extracted from the queue, the algorithm verifies if the vertex is a keypoint. If so, the keypoint is deleted from the remaining set. The algorithm finishes when the remaining set is empty, which means that a component has been extracted and it contains the complete set of input keypoints.

It is worth noting that we introduce a scaling factor $\sigma > 1$ for the radius r_{i} (algorithm 4.3, line 15). Thus, we ensure a connectivity between the keypoints in S_{i}. A value greater than 1 would guarantee to find a connected component lying inside the sphere with radius $\sigma \times r_{\text{i}}$. That is, a suitable choice for σ should be in the interval $(1, 2]$. We avoid a value of one in our choice because the patch containing the cluster of keypoints could contain more than one connected component. Indeed, the larger the σ value, the higher the probability that the extracted region is a connected component. On the other hand, the σ value can not be extremely large because it could affect the characterization of the cluster of keypoints. In all

Algorithm 4.3 Key-component Extraction

```
Require: Vertex set $V$
Require: Geodesic center $c_\mathrm{i}$
Require: Cluster of keypoints $S_\mathrm{i}$
Require: Sphere $(o_\mathrm{i}, r_\mathrm{i})$
Require: Scaling radius factor $\sigma$
Ensure: Vertex set $V_R$
Ensure: Face set $F_R$

 1: Let $V_R$ be an empty vertex set
 2: Let $F_R$ be an empty face set
 3: Let $waiting$ be the set of remaining keypoints
 4: Let $visit$ed be a vertex queue
 5: $visit$ed.$enqueue(c_\mathrm{i})$
 6: $waiting \leftarrow S_\mathrm{i}$
 7: while $waiting \neq \emptyset$ and $visit$ed $\neq \emptyset$ do
 8:     $v \leftarrow visit$ed.de$queue()$
 9:     if $v$ is not marked then
10:         $V_R \leftarrow V_R \cup \{v\}$
11:         Mark $v$
12:         $waiting \leftarrow waiting - \{v\}$
13:         for each $w \in v.adjacentVertices()$ do
14:             if $w$ is not marked then
15:                 if $\|w - o_\mathrm{i}\| < \sigma \times r_\mathrm{i}$ then
16:                     $visit$ed.$enqueue(w)$
17:                 end if
18:             end if
19:         end for
20:         $F_R \leftarrow F_R \cup v.adjacentFaces()$
21:     end if
22: end while
23: Unmark vertices
24: Return $F_V$ and $F_R$
```

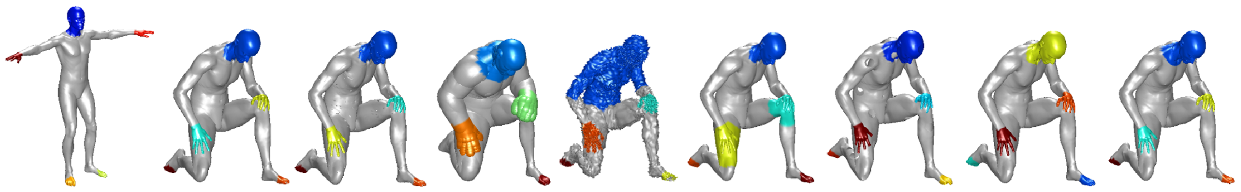


Figure 4.15: Key-components detected on shapes with several transformations. From left to right: null shape, isometry, microholes, local scale, noise, topology, holes, sampling, and shot-noise. Color are arbitrary.

our experiments, we use $\sigma = 1.5$ which allow us to always obtain one connected component in the extraction without compromising the characterization of the keypoint clusters.

Figure 4.15 shows the components detected in several shapes.

4.2.3 Experimental Evaluation and Discussion

In this section, we describe the evaluation criterion used to assess our method, and the experimental results. Our experiments were performed on the SHREC 2010 dataset (Sec. 3.2.1). Also, the models were normalized so the surface area is 1. This facilitated the configuration of the clustering parameters.

Evaluation Criterion

To evaluate our approach, we use the methodology previously used in Litman et al. [83] to determine the repeatability of a decomposition. Our goal is to determine if the mesh components are consistent between a null shape and a transformed shape. Given a null shape X and a transformed mesh Y, the components are represented as $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$, respectively. Using the ground-truth, we compute the corresponding component to each component Y_j in X, which is denoted as X'_j. Then, the component repeatability between X and Y is defined as

$$R(X,Y) = \sum_{j=1}^{m} \max_{1 \leq \mathrm{i} \leq n} O(X_\mathrm{i}, X'_j) \tag{4.38}$$

where the overlap between two components is defined as an area ratio

$$O(X_\mathrm{i}, X'_j) = \frac{A(X_\mathrm{i} \bigcap X'_j)}{A(X_\mathrm{i} \bigcup X'_j)}. \tag{4.39}$$

In addition, we define the repeatability in overlap o as the percentage of components in the entire collection that have overlap greater than o with their corresponding null shape.

Clearly, totally coincident components give a repeatability of 1.

Effect of Key Ingredients

Our method relies on two aspects which are important in the final resulting key-components. First, the number of keypoints could determine the protrusion of the regions and therefore their distinctiveness. Second, the clustering parameters could reveal the importance of the size in the repeatability of key-components. For these reasons, this section is devoted to study the effect of these two aspects in order to find a good parameter configuration.

Regarding the number of keypoints, we test five configurations: three with a fixed number of keypoints (100, 200, and 300) and two with a number that depends of the number of vertices (1% and 1.5%). Furthermore, we consider three configurations for the clustering which correspond to small ($R = 0.05, T = 0.1$), medium ($R = 0.1, T = 0.2$), and large regions ($R = 0.15, T = 0.3$). In addition, we evaluate the minimum number of keypoints needed for a key-component ($N = 10, 20, 30$). It is worth mentioning that values for clustering are empirical, mainly guided by the fact that meshes are normalized to area one. Figure 4.16 shows a plot with the results using all possible configuration as mentioned before. This plot shows the repeatability at overlap 0.8 as an average for every shape in the dataset (including transformations).

There are two aspects which deserve attention. First, there is a notorious predominance for the use of 1% of the number of vertices as keypoints. This responds to the need of balancing the trade-off between quality and quantity of local features. In other words, much more keypoints could contain noisy information, and therefore it could degrade the quality of key-components. In counterpart, much less keypoints could not determine robust regions. This follows from the fact that regions are determined by groups of keypoints and obviously less keypoints could not tend to cluster. A particular case is shown for small regions and many keypoints per region ($R = 0.05, T = 0.1, N = 20$ and $R = 0.05, T = 0.1, N = 30$). In these cases, the use of few keypoints (200 and 100, respectively) shows the best repeatability. We believe that these two configurations exhibit a particular behavior because a few keypoints are grouped in few small regions. As these small regions correspond to the presence of many keypoints, they are distinctive and hence repeatable (although below the repeatability of large regions).

Second, the highest repeatability values correspond to large regions. That is, large regions are more stable to transformations and the probability that large key-components come from perturbed features is low. Moreover, among the different configurations for large regions, there is a trend regarding the expected number of keypoints per region. The greater the number of keypoints allowed to belong to a region, the greater the repeatability of the key-components. This result encourages us to believe that key-components can correspond to regions with high protrusion which are distinctive and repeatable at the same time. This finding is consistent with the theory of saliency of visual parts in terms that key-components are distinctive. Furthermore, there is a visible relation between robustness and repeatability, and the relative size of regions.

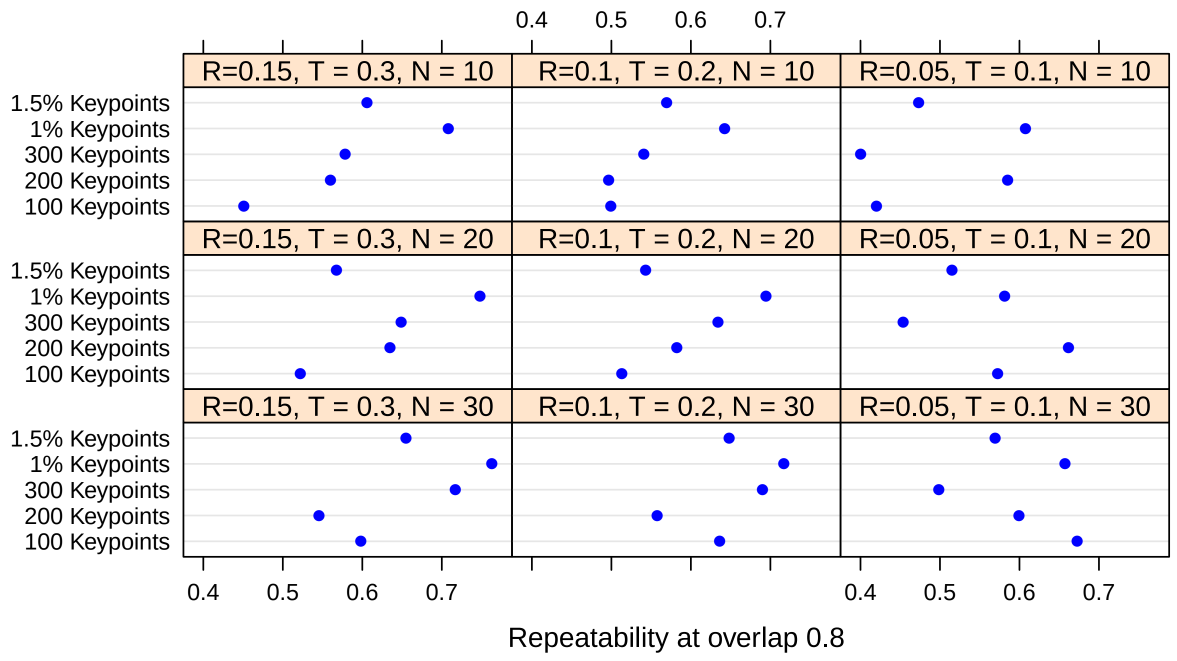


Figure 4.16: This plot shows the average repeatability at overlap 0.8 for different parameter configurations. Columns represent region sizes: large (left), medium (middle), and small (right). Rows represent the minimum number of keypoints allowed in a cluster: $N = 10$ (top), $N = 20$ (middle), and $N = 30$ (bottom). Each block contains the repeatability for five different number of keypoints: three fixed configurations (100, 200, and 300) and two depending on the number of vertices (1% and 1.5%). Each block has the same scale.

To provide a closer look on the behavior of our algorithm against transformations, we chose the three configurations with the highest average repeatability at overlap 0.8 (in addition, we named each configuration to facilitate reading) :

- **KC-1:** # keypoints = 1% number of vertices, $R = 0.15, T = 0.3, N = 30$.
- **KC-2:** # keypoints = 1% number of vertices, $R = 0.15, T = 0.3, N = 20$.
- **KC-3:** # keypoints = 1% number of vertices, $R = 0.1, T = 0.2, N = 30$.

Variant KC-1 determines large regions with a high number of keypoints. Figure 4.17 plots the repeatability of key-components at several overlap values. Most transformations (shotnoise, scale, microholes, isometry, localscale, and noise) obtained a high repeatability (greater than 80%) at overlap 0.8. Even more, four transformations (shotnoise, microholes, scale, and isometry) have a repeatability greater than 90%. This indicates that this variant is very robust for these transformations regardless the transformation strength. Differently, topology, holes and sampling transformations obtained a repeatability below 80%. With respect to the topology transformation, we believe that large key-components degrades the performance because they tend to include the introduced topological noise while the region is extracted. The larger the region, the more likely to merge two parts of the mesh that are not connected in the original mesh. In addition, our approach heavily depends on the computation of geodesic distances which are distorted with this transformation.

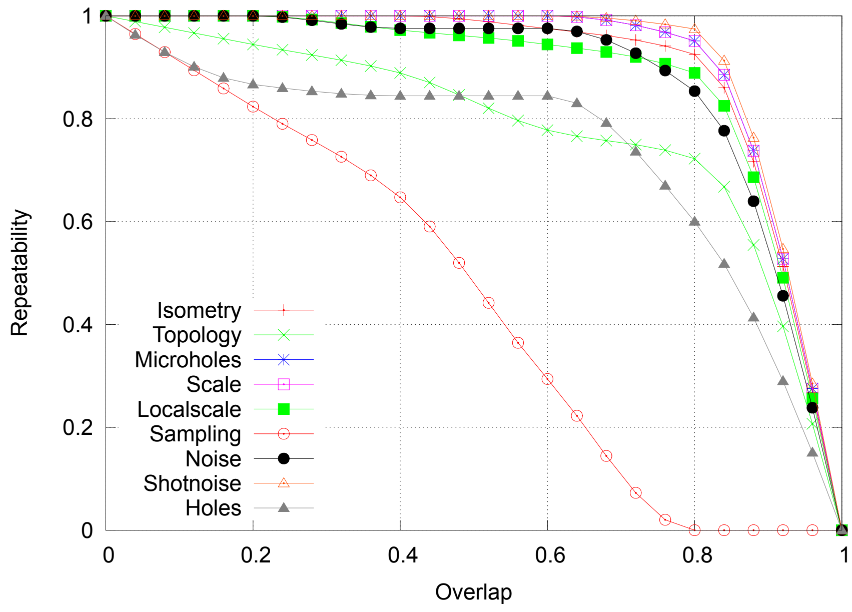


Figure 4.17: Overlap vs. Repeatability plot for the KC-1 variant.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	0.94	0.95	0.85	0.93	0.94
Topology	0.75	0.70	0.84	0.77	0.74
Micro holes	0.93	0.95	0.95	0.95	0.93
Scale	0.95	0.95	0.92	0.93	0.95
Local scale	0.95	0.87	0.93	0.88	0.93
Sampling	0.62	0.38	0.09	0.02	0.00
Noise	0.92	0.91	0.93	0.94	0.86
Shot noise	0.93	0.95	0.96	0.96	0.94
Holes	0.78	0.69	0.77	0.86	0.79
Average	0.86	0.82	0.80	0.80	0.79

Table 4.10: Average overlap values for variant KC-1.

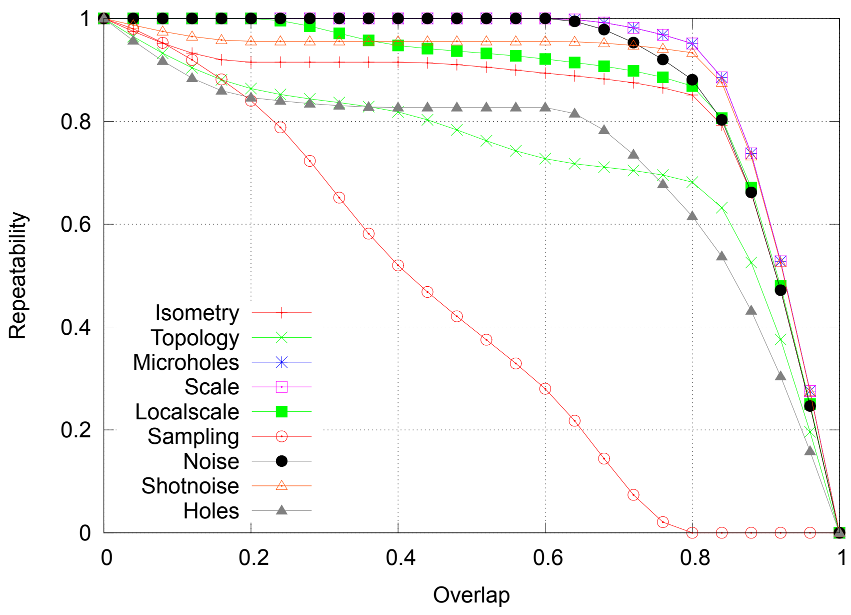


Figure 4.18: Overlap vs. Repeatability plot for the KC-2 variant.

Also, a special case is the sampling transformation whose repeatability value drops to zero. In effect, the down-sampling of meshes is not well handled by our approach due to the dependency of the number of vertices in the keypoint detection stage. In this case, the three variants use a number of keypoints which depends on the number of vertices (specifically 1%). Obviously, when a shape is down-sampled, the number of keypoints is also reduced. This fact could affect the entire process of key-components extraction, which depends on the number of keypoints and the distribution of them over the surface. Despite of this, we believe that the KC-1 variant of our technique is robust and it provides distinctive and repeatable key-components.

As well, Table 4.10 presents the average overlap for each transformations and its strengths. These results support those obtained in Fig. 4.17. The same four transformations with repeatability greater than 90% in overlap 0.8 have high overlap values in most of the strengths.

For the variant KC-2, Fig. 4.18 and Table 4.11 show the results. The difference between KC-1 and KC-2 is minimal. Moreover, there are four transformations (microholes, scale, localscale and noise) where repeatability values are very similar (see Table 4.10 and Table 4.11).

With respect to the variant KC-3, almost all transformations present a drop in their repeatability values at overlap 0.8 with respect to the previous variants (see Fig. 4.19). Nevertheless, it is worth noting that the repeatability for the topology transformation rose above 90%. It also can be seen in Table 4.12, where the overlap values for all the topology strengths were improved with respect to KC-1 and KC-2. We believe that this phenomenon correspond to the fact that medium-size regions are more stable to the topological noise. Furthermore, the overlap values for the highest strengths in localscale and shotnoise were also improved.

In order to compare the three variants KC-1, KC-2 and KC-3, Table 4.13 shows the winner

Transform.	**Strength** 1	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	0.85	0.95	0.78	0.87	0.87
Topology	0.74	0.64	0.78	0.71	0.67
Micro holes	0.93	0.95	0.95	0.95	0.93
Scale	0.95	0.95	0.92	0.93	0.95
Local scale	0.95	0.87	0.93	0.88	0.87
Sampling	0.58	0.41	0.24	0.02	0.00
Noise	0.92	0.91	0.93	0.94	0.91
Shot noise	0.85	0.87	0.95	0.95	0.93
Holes	0.68	0.75	0.70	0.81	0.74
Average	0.83	0.81	0.80	0.78	0.76

Table 4.11: Average overlap values for variant KC-2.

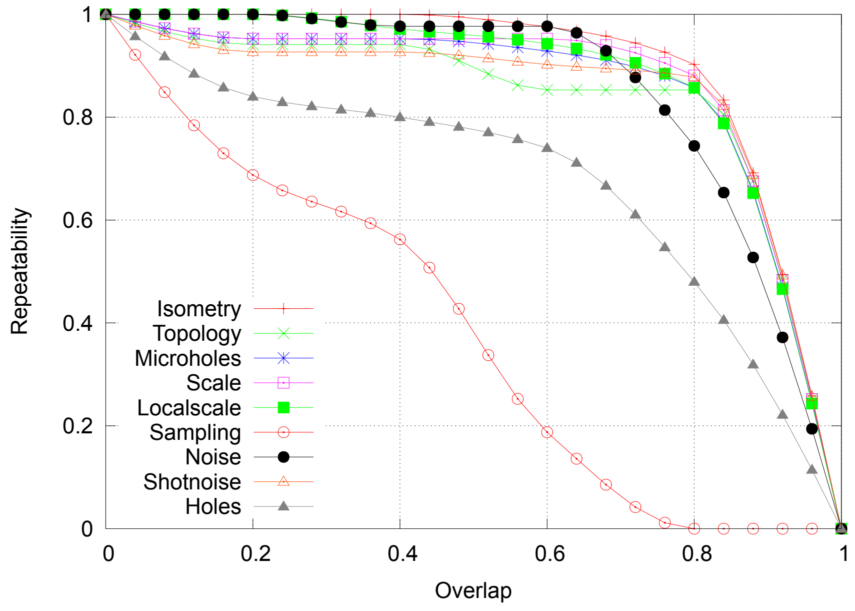


Figure 4.19: Overlap vs. Repeatability plot for the KC-3 variant.

Transform.	**Strength** 1	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	0.91	0.94	0.88	0.94	0.94
Topology	0.87	0.84	0.86	0.79	0.78
Micro holes	0.86	0.92	0.93	0.92	0.86
Scale	0.92	0.93	0.86	0.86	0.92
Local scale	0.95	0.87	0.87	0.91	0.94
Sampling	0.56	0.23	0.07	0.02	0.00
Noise	0.90	0.90	0.92	0.92	0.86
Shot noise	0.86	0.89	0.90	0.89	0.95
Holes	0.72	0.65	0.77	0.83	0.77
Average	0.84	0.80	0.79	0.79	0.78

Table 4.12: Average overlap values for variant KC-3.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	KC-1	KC-1	KC-3	KC-3	KC-1
Topology	KC-3	KC-3	KC-3	KC-3	KC-3
Micro holes	KC-1	KC-1	KC-1	KC-1	KC-1
Scale	KC-1	KC-1	KC-1	KC-1	KC-1
Local scale	KC-1	KC-1	KC-1	KC-3	KC-3
Sampling	KC-1	KC-2	KC-2	KC-1	KC-1
Noise	KC-1	KC-1	KC-1	KC-1	KC-2
Shot noise	KC-1	KC-1	KC-1	KC-1	KC-3
Holes	KC-1	KC-2	KC-1	KC-1	KC-1
Average	KC-1	KC-1	KC-1	KC-1	KC-1

Table 4.13: Comparison of the three evaluated variants: KC-1, KC-2, and KC-3.

configuration for each transformation and their strengths in terms of overlap values. There is a clear predominance of KC-1 in most of the transformations, although KC-3 get the best results for topology and the highest level of localscale. Nevertheless, we believe that KC-1 is the best variant which confirms the robustness and repeatability of large key-components.

Comparison with other methods

So far, there are no methods that explicitly detect regions on 3D surfaces under the motivation of finding robust components. Nevertheless, one method that has more to do with ours is that proposed by Litman et al. [83, 13]. This methods proposed to detect stable components in deformable meshes using a diffusion geometry approach. Unlike our method, the approach of Litman et al. decomposes a mesh into a set of (possibly overlapping) components using a approach similar to the MSER detection in computer vision. In addition, a component could be part of a larger component which entirely contains the first one. Although there is no a clear connection between that method and ours, we propose a variation to detect key-components based on the components provided by the original method. We call this variant *MSER key-components*.

The algorithm we implement is simple and it is described as follows (Fig. 4.20 shows the three stages of our implementation):

- **Detecting MSER's.** We use the original implementation from Litman et al. [83] to detect a set of initial components.
- **Detecting keypoints.** For each mesh, we computed keypoint based on the heat kernel of a vertex. Our implementation follows the method proposed by Sun et al. [136]. Briefly, we evaluated the heat kernel signature for a vertex $HKS_t(x,x)$ in $t = 0.1 \times surface\_area$. Next, we selected a vertex x as keypoint if $HKS_t(x,x) > HKS_t(y,y), \forall y \in N_2(x)$, where $N_2(.)$ is the 2-ring neighborhood of a vertex.
- **Selecting the MSER key-components.** We could have chosen the set of components which contains the detected keypoints. However, since components can overlap,

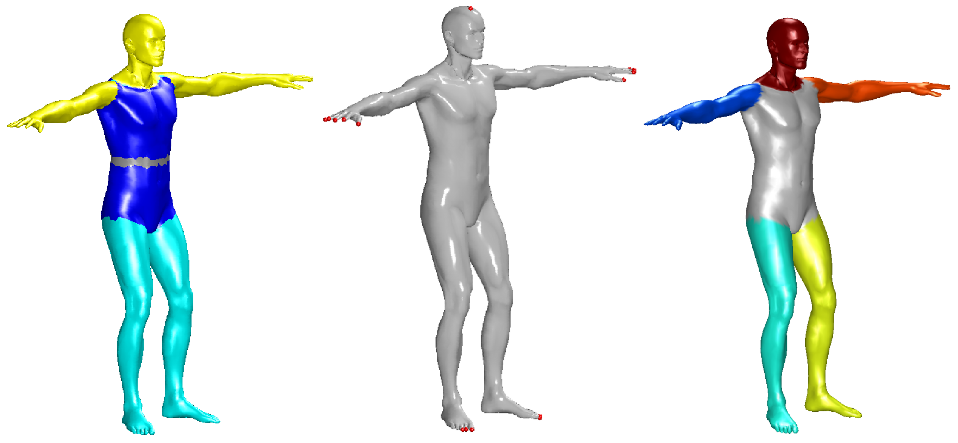


Figure 4.20: The three stages of the MSER key-components detection. At left, components detected with the approach of Litman et al. At middle, HKS keypoints detected. At right, final MSER key-components detected.

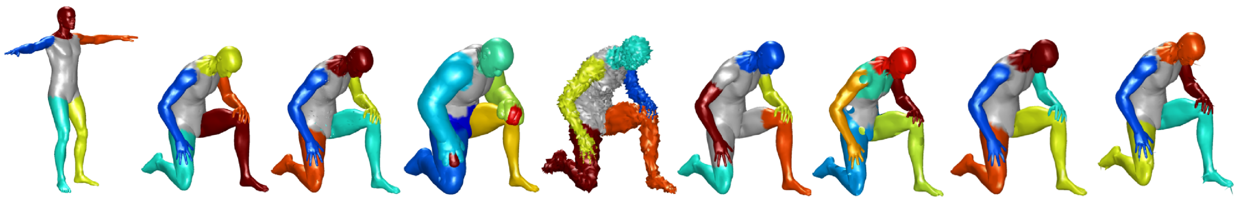


Figure 4.21: MSER key-components detected on shapes with several transformations. From left to right: null shape, isometry, microholes, local scale, noise, topology, holes, sampling, and shot-noise. Colors are arbitrary.

we need to apply one more constraint. In this case, we selected the components with smaller area which cover the entire set of keypoints. Figure 4.21 shows the MSER key-components detected using the proposed variant.

Figure 4.22 and Table 4.14 show the obtained results for the MSER key-component method. Three transformations (isometry, shotnoise and microholes) obtained repeatability values greater than 0.8 at overlap 0.8. In addition, note the improvement in holes and sampling transformation with respect to our method. That is, while our method obtained null repeatability at overlap 0.8, MSER key-components obtained almost 0.6. Despite of the results obtained for these two transformations, our method outperforms MSER key-components in the rest of transformations. Moreover, this can also be seen if we compare our best configuration and the MSER-key-components with respect to the overlap values (see Tables 4.10 and 4.14).

An aspect that also deserves attention is the need of preprocessing in the MSER key-components. The original method for detecting the initial components depends on the definition of edge or vertex weights. These weights are computed using diffusion geometry. This procedure requires to compute a similarity matrix in a vertex-wise manner, and subsequently compute an eigen-decomposition for that matrix. The problem with this approach is that

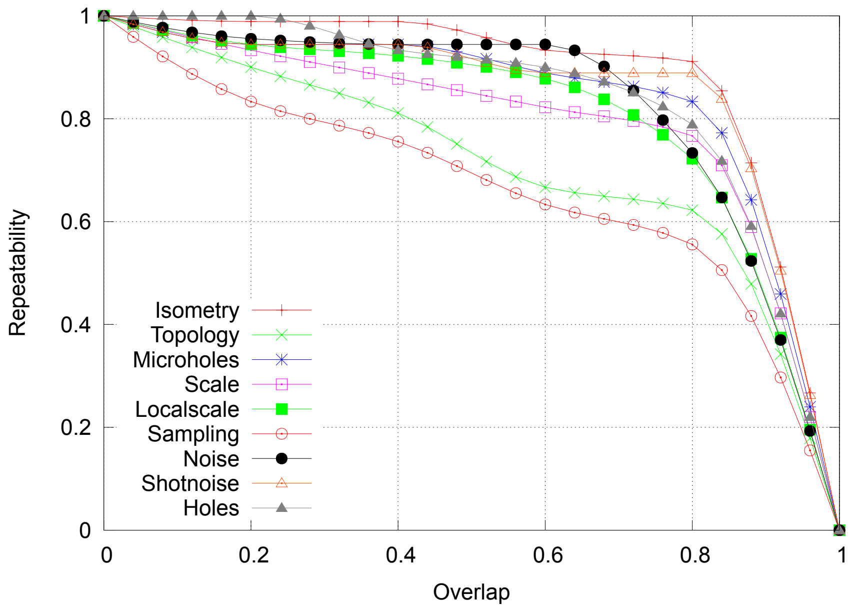


Figure 4.22: Overlap vs. Repeatability for the MSER key-component method.

	Strength				
Transform.	**1**	**$\leq$2**	**$\leq$3**	**$\leq$4**	**$\leq$5**
Isometry	0.83	0.88	0.90	0.87	0.86
Topology	0.77	0.72	0.67	0.52	0.55
Micro holes	0.82	0.81	0.80	0.79	0.78
Scale	0.76	0.84	0.80	0.67	0.58
Local scale	0.77	0.72	0.71	0.64	0.58
Sampling	0.83	0.83	0.80	0.62	0.12
Noise	0.79	0.79	0.83	0.83	0.79
Shot noise	0.83	0.83	0.80	0.80	0.80
Holes	0.79	0.77	0.66	0.63	0.62
Average	0.80	0.80	0.78	0.71	0.63

Table 4.14: Average overlap values for the MSER key-components approach.

if we have large models, the similarity matrix is very large and the eigenproblem could be unmanageable. The method proposed by Litman et al. suggested to simplify a model prior to the components detection step. In contrast, our method does not need that requirement and yet it is efficient. In this respect, the main advantage of our method is that once the keypoints have been detected, the subsequent tasks only work over these reduced number of information. We believe that the fact of relying on the initial keypoint detection step allows us to maintain the whole process efficient. This ability to deal with large meshes can be useful in applications where simplification is not an option since one could lose important details.

4.3 Concluding Remarks

In this chapter, we presented two methods to obtain local structures on 3D meshes. We introduced the Harris 3D algorithm to detect interest points and the key-component detection algorithm to obtain regions of interest. In both cases, we evaluated the robustness of our methods using the SHREC'2010 robust feature detection and description benchmark. The goal of our experiments was to evaluate the repeatability of local structures in presence of transformations in different strength levels.

In the case of keypoint detection, Harris 3D has proven to be effective to detect repeatable features. The keypoint detection task is important due to the ability of reducing the amount of information needed in subsequent processes. Furthermore, our algorithm effectively adapts the well known Harris corner detection for images in order to be used for 3D meshes.

Our method is robust for several reasons. First, the use of a Gaussian function to smooth the derivatives surface contributes to effectively mitigate the effect of local deformations introduced by noise, holes, change of tessellations, and so on. Second, our proposal of adaptive neighborhoods improves considerably the alternative of spatial neighborhoods. In addition, from the results obtained by using ring neighborhoods, our experiments confirm the fact that balanced neighborhoods favor the overall process. This is because, with a balanced set of points, the approximation of derivatives in the analyzed point is better. Finally, along with the task of final selection of interest points, Harris 3D proposes several alternatives for its effective use, making an interesting alternative in applications such as shape matching and registration, just to name a few.

Furthermore, our performed experiments suggest that our method could be used in extreme conditions with good results. This is an important advantage with respect to the state of the art, since it allow us to deal with several kind of shapes and expand the spectrum of possible applications in the future.

In the case of regions of interest, our algorithm to detect key-components showed a good performance in terms of repeatability and overlap. The key-components are suitable for matching and recognition tasks due to their high repeatability obtained in our experiments using a standard benchmark. Interestingly, the proposed method detects consistent components under several transformations such as noise, local scale, holes, and non-rigid transformations. In our opinion, key-components represent an alternative to fine scale features. On the one hand, we showed that key-components are stable to local transformations. On the other hand, the number of key-components is obviously much less than the number of keypoints, so matching algorithms using local features could benefit from our approach.

Also, our experiments showed evidence of the connection between our method and the theory of saliency of visual parts proposed in the cognitive science. First, key-components correspond to regions with high protrusion since they are found from agglomerations of robust and distinctive local features. Moreover, the proposed clustering is responsible for ensuring that local features that belong to some perturbation (noise, holes, etc.) are not considered in the detection of salient regions. Second, there is an intimate relationship between the size of the key-components and the robustness against mesh transformations. Experiments showed that large salient regions (each with a large agglomeration of keypoints) are more repeatable. These two aspects are consistent with the aforementioned theory, so our method can be thought of as a method embodying this theory.

The most important lesson learned in this chapter is that it is possible to obtain robust local representations at different levels of granularity. We will show, in subsequent chapters, that the high repeatability of these local structures indeed allows us to improve the performance in tasks such as retrieval and matching.

Chapter 5

Data-aware Partitioning for Generic Shape Retrieval

In this chapter, we consider the problem of generic shape retrieval. A common approach to facing this problem is to compute an intermediate representation (feature vectors or graphs, for instance) and subsequently defining the similarity of two objects as the similarity of their representations. In this direction, there are methods that exploit the visual similarity, the statistical properties of 3D measures, or the possibility of defining transform functions on the data, just to name a few. However, one of the most critical problems is the semantic gap. That is, the intermediate representation may not be able to capture all the needed information of a shape and therefore the effectiveness of searching may be seriously affected.

A previous study by Bustos et al. [23] showed that some features could well represent certain classes of objects and furthermore, some features could be complimentary in representing a shape. This is because algorithms cover only a part of the possible spectrum of characteristics such as shape, silhouette, or intrinsic properties. Thus, a natural extension of classic approaches was the combination of features for improving the effectiveness of retrieval. Approaches in this direction have been previously presented by Bustos et al. [24], Vranic [147], and Papadakis et al. [105], all of them with promising results. However, the semantic gap is still latent in this approach as any possible combination of features could not represent important characteristics to discriminate between objects.

A more recent approach is the combination of global and part-based information. The idea is to combine features extracted from an entire object with features extracted from parts of an object. Some techniques have been presented so far by Li and Johan [77], Bustos et al. [26], and Schreck et al. [120]. All of these techniques share a common aspect: the part-based features come from a fixed partitioning of the objects. Although it was possible to improve the effectiveness with respect to using only global features, the fixed partitioning limits the possibility of having truly distinctive parts. This opens up a question on how to define a new kind of partitioning dependent on the shape information.

We believe that the use of local features can enhance the use of global features in shape retrieval. That is, we are trying to mitigate the effect of the semantic gap. For instance, a

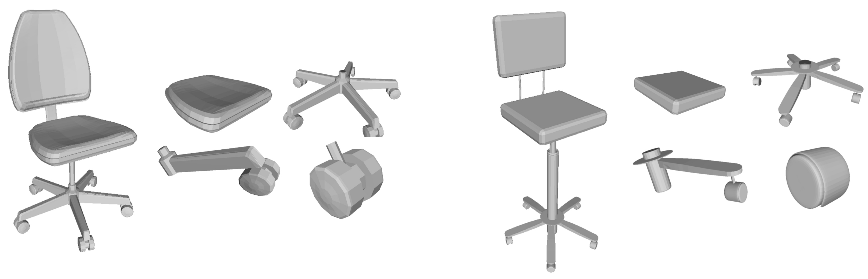


Figure 5.1: Two globally dissimilar chairs. Note that the chair at right is taller than the left one. Nevertheless, it is possible to find similarities between their parts, which can be exploited to improve the similarity measure between the two objects.

common fact is having two objects with different appearance in the same class. Obviously, a global feature could differ in those objects. However, in a local sense, it is still possible to find correspondences between parts, so we can take advantage of this fact to improve the similarity measure (see Fig. 5.1). Therefore, the discriminative power of local features combined with global features could help to improve the effectiveness in the similarity search.

In this chapter, we propose a shape retrieval method using a data-aware partitioning algorithm. Our idea is to exploit the local characteristics of objects to determine discriminative parts. Thus, each object is represented by its global feature and a set of features extracted from parts. The partitioning method relies on finding robust local features (namely keypoints) on the object's surface and subsequently determining the parts where there is a high concentration of keypoints (for instance, a human shape commonly has many features located in hands, feet, and head). Beyond techniques which made use of the bag of features approach to aggregate local descriptors for retrieval, our method is the first attempt in combining global an local features found in a data-adaptive way for generic shape retrieval.

Our approach is a generic, simple framework by which global and local descriptors can be combined in a data-adaptive way. The approach is able to provide on average, an improvement over the retrieval effectiveness of state of the art global descriptors. A careful, systematic analysis of the results is performed to assess in detail the magnitude of the improvement, relating it with global-only methods, and identifying classes of models for which the method is particularly effective. We test our approach using our Harris 3D interest point detector in combination with two robust view-based descriptors. Our approach is flexible in that it can accommodate further (possibly application-specific) object segmentation and description schemes, if needed.

The chapter is organized as follows. Section 5.1 describes our partitioning algorithm based on local features. Section 7.2 is devoted to the matching methods and the definition of our similarity measure. Section 5.3 describes our experiments and presents the discussion of our results. Finally, Section 5.4 presents the conclusions of the chapter.

5.1 Data-aware 3D Partitions

In this section, we present a partition algorithm based on finding groups of discriminative local features. Our method does not guarantee disjoint or complete partitions. However, as it uses interest points detected on the mesh for partitioning, we believe that the resulting fragments are representative enough. Therefore, the partitions can be useful for improving the matching between two 3D models.

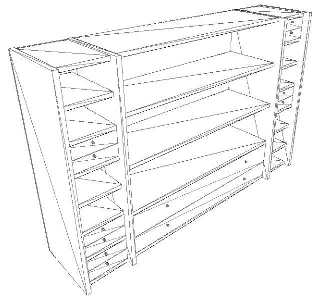
Our method consists of three steps:

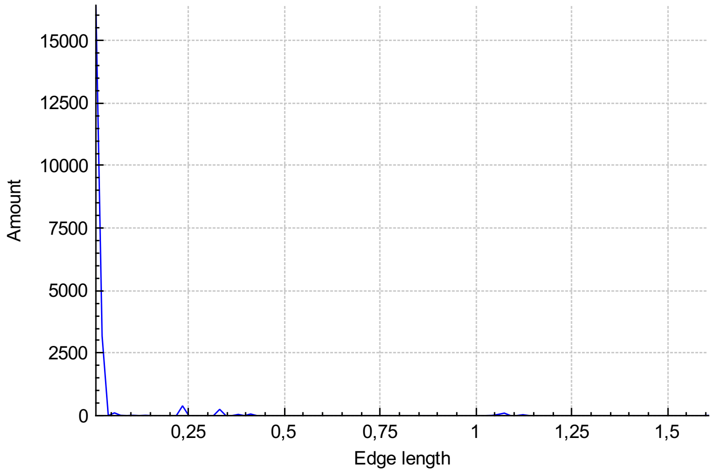
- **Interest point detection.** We aim at selecting a small set of points on the mesh surface. We consider that a vertex is interesting if it has an outstanding geometric structure in comparison with its neighborhood. We use our Harris 3D method to perform the interest point detection. Section 4.1 presents Harris 3D in detail, so we omit this step in this section.
- **Clustering of interest points.** We perform a clustering in order to find groups of interest points under some constraints.
- **Cluster-based partition.** We use the resulting clusters for defining representative partitions for matching.

5.1.1 Control of mesh resolution

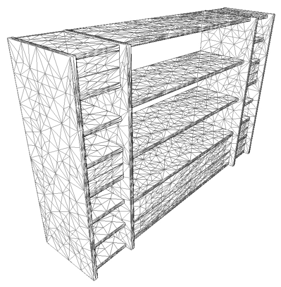
Note that the Harris 3D method depends on local neighborhoods around a vertex. Nevertheless, generic 3D shapes could come from different sources where their primary goal was not the analysis or processing. It is therefore common to find objects with bad triangulations. Moreover, many meshes are optimized for rendering, so regular portions of them are represented by large triangles. It poses a problem for 3D analysis, where meshes with regular triangulations are preferably needed. Therefore, it is necessary to control the size of the neighborhoods prior to the interest point detection. In addition, our goal is to ensure a consistent neighborhood computation along the entire mesh.

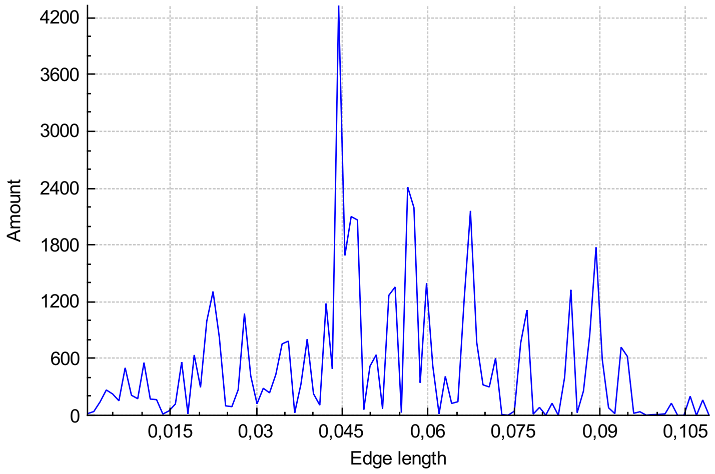
We implement the algorithm for control of mesh resolution proposed by Johnson [63]. This algorithm assumes the spacing between vertices as the resolution to be improved. More specifically, the mesh resolution is the median of the edge length histogram. The goal is to decrease the edge length spread (or variance) of the histogram around a desired resolution. To accomplish this goal, the algorithm performs local operations over the edges which are too large (split operation) or too small (collapse operation). Each edge is associated with a weight which combines its length difference with the desired mesh resolution and the geometric shape change if any operation is performed. Finally, a greedy strategy performs local operations guided by a priority queue defined over the weights. An example of our implementation is shown in Fig. 5.2


(a)



(b)


(c)



(d)

Figure 5.2: Effect of mesh resolution: Shape with bad triangulation (a) and its poorly distributed edge length histogram (b). Shape processed with the mesh resolution algorithm (c) and its improved edge length histogram.

5.1.2 Clusters of Interest Points

Once we have computed the interest points for a mesh, our goal is to use them for extracting representative partitions. The main idea is to find clusters of interest points in the 3D space, so each cluster would define a portion of the mesh which is interesting and distinctive. Basically, we use the adaptive clustering algorithm proposed in Section 4.2 (see Algorithm 4.2). In this case, the clustering algorithm is used to find clusters of keypoints in 3D space.

The adaptive clustering algorithm uses two distance thresholds R, T, and the minimum number of points in a cluster N as parameters. The algorithm scans $Iter$ times the set of points trying to find clusters which hold two criteria: the distance between each point within a cluster to its centroid is not larger than R (intra-cluster constraint), and the distance between cluster's centroids is not smaller than T (inter-cluster constraint). In addition, if after a scan, the number of points inside a cluster is less than N, the cluster is discarded. Also, the points of the small cluster are pushed back in the point collection for the next iteration. If a cluster has more than N points, its centroid is updated.

Note that there may exist points which never hold with the cluster constraints, and those points are simply discarded. We are interested in groups of interest points, because these could be in a representative part of the mesh. Hence the behavior of discarding isolated points is important for our purposes, because those points could be noise and therefore would not represent an important feature of the mesh.

Another important aspect is the flexibility of the adaptive clustering algorithm with respect to the obtained number of clusters. The number of clusters depends on the point distribution and the cluster parameters. This is an advantage because each object would have a different number of clusters depending of their interest points. In this way, each object would have a data-aware flexible representation.

The presented clustering algorithm determines a spatial partitioning of the keypoints where clusters are always circle-shaped. Hence, we also tested the DBSCAN [39] algorithm for clustering which has a different functionality. DBSCAN is a density-based algorithm which is able to detect clusters of arbitrary shapes and it is based on proximity and density concepts. However, after experimentation, we found that it was difficult to correctly define the density thresholds for this algorithm. In addition, in many of our experiments, DBSCAN computed very few clusters per shape, underestimating the representational power of the interest points.

5.1.3 Partitioning and Description

Our partitioning algorithm is quite simple. For each cluster of interest points, we proceed as follows:

- The algorithm computes the smallest 3D sphere containing all points within the cluster. We used a linear programming algorithm for this purpose [88]. The outputs of this step are the center of the sphere and the radius.
- Next, we extract the portion of the mesh lying inside of a 3D sphere formed by the previously computed center and the radius scaled by a factor of δ (we study the effect of δ in Section 5.3). It can be done by scanning the complete set of vertices and verifying which vertex lies inside the sphere. However, this can be computationally expensive for large meshes. Here we use an improved method. We build a kd-tree with all vertices of the mesh, and thus a range search is performed using as query the center of the sphere and the radius. Note that the kd-tree needs to be built only once, and it can be used for each partition by changing the query. Finally, the method builds a new mesh using the set of points inside the sphere and their associated faces.

For description, we compute a global descriptor for the entire model and subsequently compute a global descriptor for each partition. Formally, given an object O, its representation is defined as

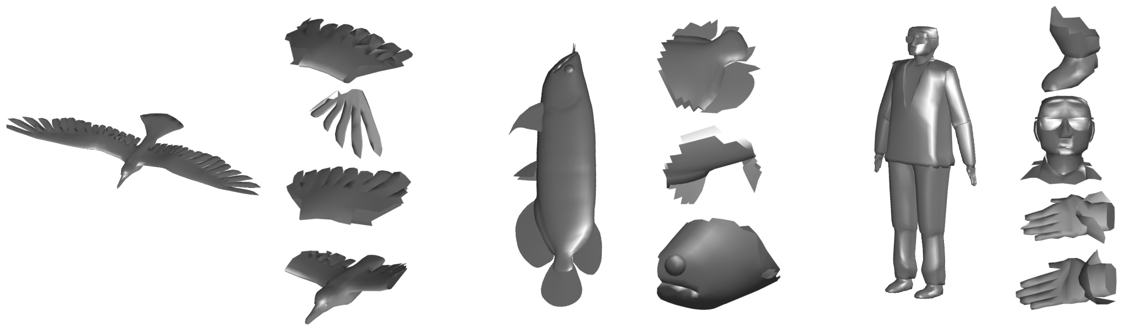


Figure 5.3: Three examples of partitions obtained with our method using the parameter configuration used in Sec. 5.3.2.

$$S_O = \{(s_O, P_O) | s_O \in \mathbb{R}^n \text{ and } P_O = \{p_O^1, p_O^2, \dots, p_O^m\}, p_O^{\text{i}} \in \mathbb{R}^n\},$$

where s_O is a n-dimensional descriptor representing the complete object, P_O is a set of m n-dimensional descriptors representing each partition. Figure 5.3 depicts an example of some partitions obtained with our algorithm.

5.2 Matching

At this point, we need to define a distance between two representations as shown previously. Given two 3D objects O and Q, each with their representations:

$$S_O = \{(s_O, P_O) | s_O \in \mathbb{R}^n \text{ and } P_O = \{p_O^1, p_O^2, \dots, p_O^m\}, p_O^{\text{i}} \in \mathbb{R}^n\} \text{ and}$$

$$S_Q = \{(s_Q, P_Q) | s_Q \in \mathbb{R}^n \text{ and } P_Q = \{p_Q^1, p_Q^2, \dots, p_Q^k\}, p_Q^{\text{i}} \in \mathbb{R}^n\},$$

where O has m partitions, and Q has k partitions. Our goal in this section is to define an appropriate distance $\text{d}(S_O, S_Q)$, which measures the dissimilarity between two objects using their representations. The main problem is how to define a dissimilarity between two sets of descriptors with different lengths.

We consider a linear combination between the global-to-global distance and the partition-based distance. That is

$$\text{d}(S_O, S_Q) = \mu \| s_O - s_Q \|_2 + (1 - \mu)\text{d}(P_O, P_Q), \tag{5.1}$$

where $0 \leq \mu \leq 1$ weights the contribution of the involved terms.

5.2.1 Integer Linear Programming

The matching problem is how to find a correspondence set between two collections of descriptors. Clearly, this problem is difficult because it is not possible (at least not within a reasonable time) to evaluate all possible combinations of correspondences. We will state the problem using a linear programming formulation for searching for a feasible solution.

We define an indicator variable as follows

$$x(\mathrm{i}, j) = \begin{cases} 1, & \text{if } p_O^{\mathrm{i}} \text{ matches } p_Q^j \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Note that $|x| = m \times k$, that is each element in P_O could be matched with each element in P_Q. Let's think of x as a binary string of length $m \times k$. The number of configurations for x is $2^{m \times k}$, which allows us to figure out the complexity of the problem. Obviously, if m and k are large enough, the number of possible matches increases exponentially. Nevertheless, we can add some constraints to the problem. For instance, if p_O^{i} is already matched to p_Q^j, then p_O^{i} should not be matched to any other element in P_Q. Formally, if $x(\mathrm{i}, j) = 1$, then $\sum_j x(\mathrm{i}, j) = 1$, and therefore also $\sum_{\mathrm{i}} x(\mathrm{i}, j) = 1$.

We use the indicator variable x to formulate an objective function as follows

$$f(x) = \sum_{\mathrm{i},j} \|p_O^{\mathrm{i}} - p_Q^j\|_2 . x(\mathrm{i}, j), \tag{5.3}$$

where the goal is to find the optimum x^* which minimizes $f(x)$. Formally,

$$x^* = argmin_x f(x), \tag{5.4}$$

subject to

$$\sum_{\mathrm{i}} x(\mathrm{i}, j') = 1 \text{ and } \sum_j x(\mathrm{i}', j) = 1 \ \forall \mathrm{i}, j$$

Moreover, we can consider the optimum $f(x^*)$ as the dissimilarity function $\mathrm{d}(P_O, P_Q)$. However, the optimum $f(x^*)$ depends on the number of matches, reaching lower values when P_O and P_Q have a few elements. In order to overcome this problem, we normalize the value of the optimum, and we obtain the final dissimilarity measure:

$$\mathrm{d}(P_O, P_Q) = \frac{f(x^*)}{min(|P_O|, |P_Q|)}. \tag{5.5}$$

Note that the normalization in Eq. 5.5 also contributes to maintain the symmetry of the distance. This is an important aspect if one considers indexing the distance for fast searching.

Numerical Aspects

To numerically solve the Eq. 5.4, we define a matrix of distances

$$C(\mathrm{i}, j) = \|p_O^{\mathrm{i}} - p_Q^{j}\|_2, \tag{5.6}$$

where each element of this matrix stores the L_2 distance between descriptors from P_O and P_Q. Thus, the problem of finding x^* in Eq. 5.4 can be stated as a binary linear programming problem

$$\min_x C^T x \text{ such that } \begin{cases} Ax \leq b \\ A_{\mathrm{eq}} x = b_{\mathrm{eq}} \\ x \text{ is binary} \end{cases} \tag{5.7}$$

where C and x are linearized versions of themselves, A and b represent linear inequality constraints, and A_{eq} and b_{eq} represent linear equality constraints. In fact, the constraints $\sum_{\mathrm{i}} x(\mathrm{i}, j) = 1$ and $\sum_j x(\mathrm{i}, j) = 1$ need to be placed in the linear constraints.

The solution for the problem in Eq. 5.7 is given by a branch-and-bound algorithm which tries to solve it using LP-relaxation approaches [150].

5.2.2 Integer Quadratic Programming

The linear programming formulation finds the best set of correspondences only regarding the dissimilarity between descriptors in P_O and P_Q. The problem with this formulation is that it discards the spatial information of the partitions from which the descriptors come. Obviously, our algorithm does not ensure consistency in the spatial sense. In this section, we enrich our previous formulation by adding spatial consistency between descriptors.

Recalling the indicator variable x. If we have two correspondences $x(\mathrm{i}, j) = 1$ and $x(\mathrm{i}', j') = 1$, one can expect that the spatial relationship between fragments i and i′ from shape O is quite similar to the spatial relationship between fragments j and j' from shape Q. Of course, the idea is to minimize the difference between spatial distances of partitions, while maintaining the dissimilarity between descriptors. Therefore our new formulation for Eq. 5.3 is

$$\begin{aligned} f(x) = & \alpha \sum_{\mathrm{i},j,\mathrm{i}',j'} |\mathrm{d}_S^O(\mathrm{i},\mathrm{i}') - \mathrm{d}_S^Q(j,j')| x(\mathrm{i},j) x(\mathrm{i}',j') + \\ & \beta \sum_{\mathrm{i},j} \|p_O^{\mathrm{i}} - p_Q^{j}\|_2 . x(\mathrm{i},j) \end{aligned} \tag{5.8}$$

where $\mathrm{d}_S^O(\mathrm{i},\mathrm{i}')$ is the spatial distance between fragments i and i′ from the object O, α and β are weights to set the contribution of the spatial consistency and the descriptor dissimilarity, respectively. In addition, the new formulation is subject to the same constraints as Eq. 5.4. Finally, we can use the new formulation to find an optimum x^* and therefore we will use the same distance as shown in Eq. 5.5.

Regarding the spatial distances, during the process of finding the partitions, we compute distances between the centers of the spheres which generate the partitions. In this way, the algorithm makes available the spatial information in the matching.

Numerical Aspects

To numerically solve Eq. 5.4 using the objective function in Eq. 5.8, we define a matrix with the distance differences as follows

$$D(\{\mathrm{i},j\},\{\mathrm{i}',j'\}) = |\mathrm{d}_S^O(\mathrm{i},\mathrm{i}') - \mathrm{d}_S^Q(j,j')|, \tag{5.9}$$

where $\{\mathrm{i},j\}$ denotes the linear index of the pair (i,j). Clearly, we need to consider the complete set of spatial relationships between pairs of partitions. The dimension of the matrix D is $mk \times mk$. Thus, the problem of finding x^* in Eq. 5.8 can be stated as a binary quadratic programming problem

$$\min_x \frac{1}{2} x^T D x + C^T x \text{ such that } \begin{cases} Ax \leq b \\ A_{\mathrm{eq}} x = b_{\mathrm{eq}} \\ x \text{ is binary} \end{cases} \tag{5.10}$$

where C and the constraints were defined in Eq. 5.7.

The solution for the problem in Eq. 5.10 is also given by a branch-and-bound algorithm with LP-relaxation, but in this case using a quadratic objective function [8].

5.3 Experimental Evaluation and Discussion

In this section, we present our experiments and results. The section is organized as follows. Section 5.3.1 presents the experimental setup and evaluation measures. Section 5.3.2 presents a study of the contribution of partition matching in the overall method. Section 5.3.3 presents a sensitivity analysis of parameters. Section 5.3.4 discusses the effectiveness of our method in a class-by-class analysis. Section 5.3.5 investigates the correlation between effectiveness and important aspects such as number of vertices and number of parts. Finally, Section 5.3.6 presents results using the PANORAMA descriptor.

5.3.1 Experimental setup

For our experiments, we use the SHREC'2009 generic benchmark (Sec. 3.2.2). To evaluate the retrieval effectiveness of our method, we use common measures in the retrieval community such as mean average precision (MAP), nearest neighbor (NN), first tier (FT) and second tier (ST). Briefly, we describe each measure as follows:

- Mean Average Precision (MAP): Given a query, its average precision is the average of all precision values computed in each relevant object in the retrieved list. Given several queries, the mean average precision is the mean of average precision of each query.
- Nearest Neighbor (NN): Given a query, it is the precision at the first object of the retrieved list.
- First Tier (FT): Given a query, it is the precision when C objects have been retrieved, where C is the number of relevant objects to the query.
- Second Tier (ST): Given a query, it is the precision when 2*C objects have been retrieved, where C is the number of relevant objects to the query.

In the retrieval experiments, each object in the collection is used as query, and subsequently we average the measures for each object to obtain the effectiveness for the entire dataset.

Regarding the descriptors, we tested the DESIRE and PANORAMA descriptors (both described in Sec. 3.1.1 and 3.1.2, respectively). Each time we describe a mesh using these descriptors, the input mesh is normalized in pose (rotation, translation and scale) prior to the description. As DESIRE is faster to compute than PANORAMA, we preferred to use DESIRE for presenting a detailed study of our approach. Subsequently, we use PANORAMA to validate our results.

5.3.2 The role of partition matching

The goal of this section is to show the contribution of the partition matching in the distance computation. Recall the definition of our distance in Eq. 5.1. Our distance is a linear combination between global distance (using the DESIRE descriptor) and partition distance. The contribution of the partition distance in the final distance depends on the parameter μ. So we conducted an experiment to measure the effect of μ in the effectiveness of the proposed distance.

We test different values for μ in the interval $[0, 1]$ and investigate the best value according to the obtained MAP. As our objective is to evaluate only the effect of μ, we fixed the values

Table 5.1: MAP values for different values of μ (values are in [0,100] scale)

μ	**LPM**	**QPM**
0	9.39	5.14
0.1	15.06	6.51
0.2	21.93	8.42
0.3	29.20	11.26
0.4	35.90	14.99
0.5	41.14	19.97
0.6	44.90	26.20
0.7	47.49	33.63
0.8	48.93	41.47
0.9	**49.52**	47.79
1.0	49.10	49.10

for any other parameter (see Section 5.3.3 for a sensitivity analysis about parameters). Next, we show a summarized description of the parameters used in this experiment:

- For the Harris 3D algorithm, we select 200 keypoints for each object.
- For the clustering algorithm, we consider the length of the diagonal of the minimum bounding box of an object ($diag$) to define the spatial parameters: R and T. Thus, $R = 0.1 \times diag$, $T = 0.2 \times diag$, $N = 10$, and $Iter = 10$. Note that the R and T parameters vary for each object. The N and $Iter$ parameters were set empirically.
- The scale factor of the sphere radius in the patch extraction step was set to 1.
- In addition, α and β in Eq. 5.8 are 1.

We compare our two proposals, linear programming matching (LPM) and quadratic programming matching (QPM) with a baseline algorithm (GM), which only uses the global descriptors for retrieval (note that GM is a special case of our proposed distance when $\mu = 1$).

Table 5.1 shows the MAP for several values of μ, using both techniques LPM and QPM. The best result for LPM is obtained in $\mu = 0.9$ with 49.52. This value shows an improvement with respect to using only global descriptors. Note that $\mu = 1.0$ represents the GM baseline approach, as it considers a total contribution of the global descriptor distance. It is worth noting that the best MAP value for LPM is obtained through a large contribution of the global distance. In contrast, the incorporation of geometric consistency in the QPM approach does not seem to contribute to the effectiveness. Nevertheless, the shown MAP values are an average of the entire dataset. This can bring up the fact that it is possible that certain classes exploit the geometric consistency. We dedicate the Section 5.3.4 to study this situation.

We also show a recall-precision plot for the different configurations of μ (see Fig. 5.4). Note the improvement of our method when $\mu = 0.9$ in contrast to other values, even when global matching is used ($\mu = 1$). Moreover, the precision improvement is visible in every recall value, so it confirms the results in Table 5.1.

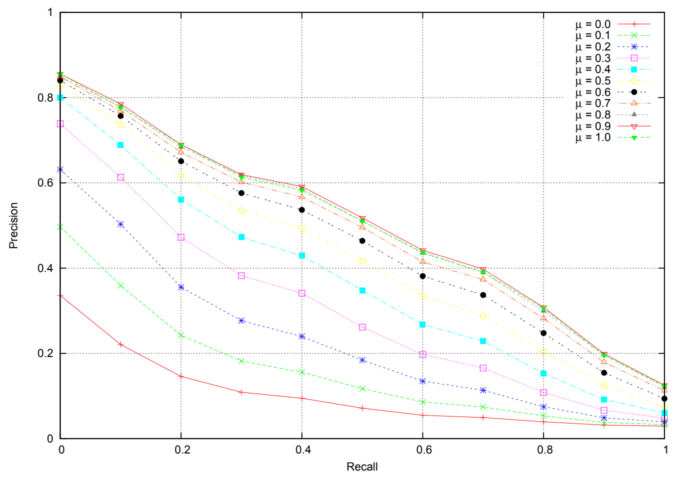


Figure 5.4: Recall - Precision

5.3.3 Sensitivity Analysis

In this section, we evaluate several parameters of our method in order to find the best configuration. We take the finding of the previous section as a starting point . That is, all results presented in this section were computed for $\mu = 0.9$. The parameters to be evaluated are:

- **Keypoint selection:** we evaluate six configurations taking into account a fixed number of keypoints (200, 300, and 400) or a number of keypoints as a ratio of the number of vertices on the mesh (at ratios 1%, 2% and 4%). In the presented results, we use the labels *IP-200*, *IP-300*, *IP-400*, *IP-0.01*, *IP-0.02*, and *IP-0.04*, respectively.
- **Clustering parameters:** we evaluate three configurations regarding the spatial constraints of the clustering algorithm. In our experiments, we use the diagonal of the bounding box of an object (diag) as reference to the clustering parameters. Thus, it is possible to associate the clustering parameters with the size of an object. We define three configurations corresponding to small, medium, and large clusters. The configurations are defined as follows (in the presented results, the diag factor is discarded to facilitate the reading):
  - Small clusters: $R = 0.1 \times \text{di}ag$, $T = 0.2 \times \text{di}ag$.
  - Medium clusters: $R = 0.15 \times \text{di}ag$, $T = 0.3 \times \text{di}ag$.
  - Large clusters: $R = 0.2 \times \text{di}ag$, $T = 0.4 \times \text{di}ag$.
- **Scaling factor for partition:** we evaluate three different scaling factors for the radius of the partitioning sphere. We use 1.0, 1.25, and 1.5.

As we are interested in studying the impact of the parameters in our approach, we present results using all the aforementioned evaluation measures. In addition, we discuss that effect in the Linear Programming Matching (LPM) and Quadratic Programming Matching (QPM) separately.

Figure 5.5 shows the mean average precision for LPM using all possible combinations of parameter configurations. Note that regardless of the used keypoint selection, LPM gives better results when small clusters are used. In addition, the mean average precision decreases when the size of clusters is increased. It means that large partitions are more unstable compared to small partitions. It sounds logical, since small clusters are expected to be compact agglomerations of keypoints, and therefore the region containing them can be considered as a representative partition for an object. In contrast, large clusters allow distant keypoints to belong to the same cluster. As a result, the probability of a keypoint to belong to any cluster is high. Therefore, isolated keypoints (possibly due to noise) could be influencing the generation of poor partitions.

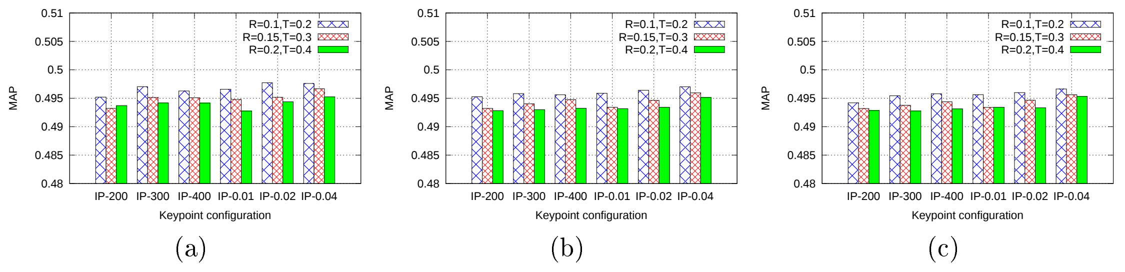


Figure 5.5: Mean average precision (MAP) and sensitivity analysis on our Linear Programming Matching approach. (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

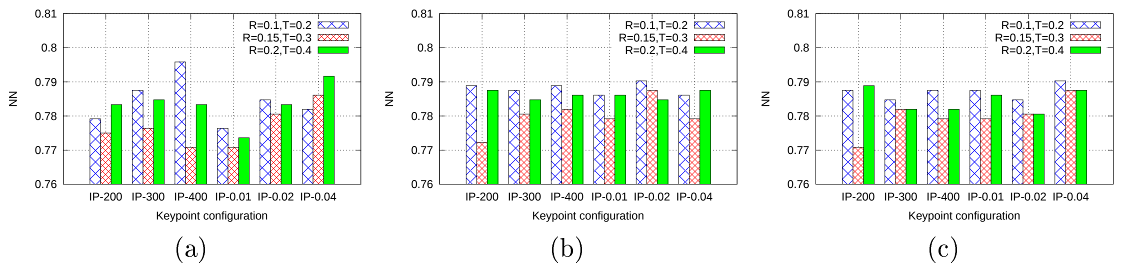


Figure 5.6: Nearest neighbor (NN) and sensitivity analysis on our Linear Programming Matching approach (LPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

Interestingly, the scaling factor for partition is very related to the previous finding. The scaling factor is used to extract the partitions after the clustering algorithm. So if the scaling factor is large, the partition will be large too. Again, by looking at Fig. 5.5, small values of δ give the best results. Therefore, we can confirm that there is a inverse relation between partition size and effectiveness.

Another important point is that, for each plot, the best mean average precision was obtained when we selected a number of keypoints in accordance with the number of vertices. In fact, the highest MAP was 0.4977, obtained with IP-0.02, R = 0.1, T = 0.2, and $\delta = 1.0$. The reason to choose this in contrast to a fixed number of keypoints is evident. With a fixed number of keypoints, it is not possible to guarantee a good representation for a shape. This fact conditions the representativeness power of the keypoints, because it is likely that when we took a fixed number of keypoints per model, this amount can be large for some models and small for others. On the other hand, the adaptive alternative seems to be a good choice taking into account that an object can have an arbitrary number of vertices.

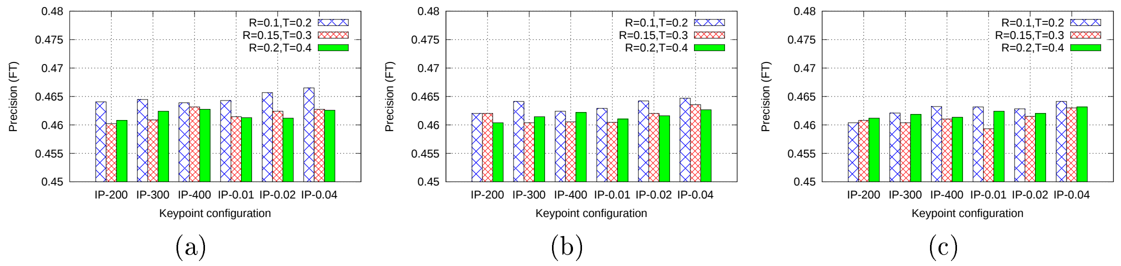


Figure 5.7: First tier (FT) and sensitivity analysis on our Linear Programming Matching approach (LPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

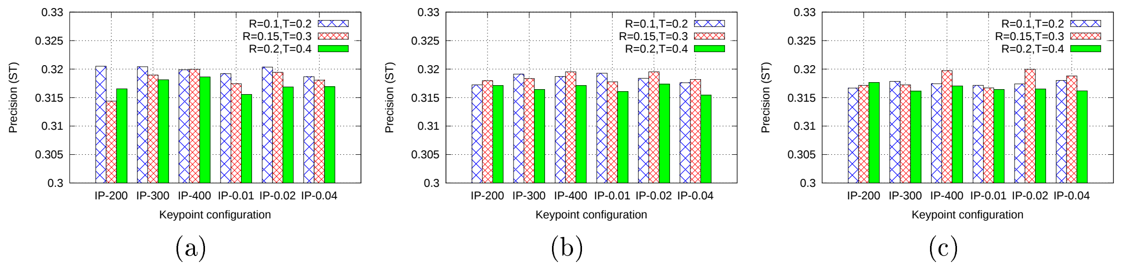


Figure 5.8: Second tier (ST) and sensitivity analysis on our Linear Programming Matching approach (LPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

Also, we present results for nearest neighbor (NN), First Tier (FT), and Second Tier (ST) in Fig. 5.6, 5.7 and 5.8, respectively. The NN measure evaluates the capacity of an algorithm to retrieve a relevant object in the first position of the retrieved list. The highest value 0.7958 was obtained using a fixed number of keypoints (400), small clusters (R=0.1, T = 0.2) and the smallest scaling factor (1.0). On the other hand, our results about FT and ST show a similar behavior as MAP. That is, higher values are obtained when clusters are small, and the scaling factor for a partition is small. The highest value 0.4665 for FT was obtained with IP-0.04, R = 0.1, T = 0.2 and $\delta = 1.0$. Similarly, the highest values 0.320507 was obtained with IP-0.02, R = 0.1, T = 0.2, and $\delta = 1.0$.

Regarding QPM, Fig. 5.9, 5.10, 5.11 and 5.12 show our results for MAP, NN, FT and ST, respectively. Surprisingly, the best MAP scores were obtained with medium size clusters. Moreover, unlike the mean average precision of LPM, large clusters give better results when the number of keypoints depends on the number of vertices. In this connection, the configuration that delivers the largest partitions (R=0.2, T = 0.4 and $\delta = 1.0$) has one of the highest MAP score. Although this situation contrasts too much with the results obtained for LPM, there is a reason for this behavior. The QPM approach depends not only on the similarity between part descriptors, but also on their geometric disposition. In fact, QPM gives the same importance to the similarity between descriptors and their consistency. In our opinion, the geometric consistency is causing this phenomenon. Our reasoning is that larger parts are more consistent in a geometrical sense than smaller parts. For instance, consider two human shapes: one with arms close to body, and other with open arms forming a T.

Small partitions could characterize hands, legs, and head. Differently, large partitions could characterize upper body and lower body. So obviously the geometric consistency of upper body and lower body remains more similar than hands, legs and head in this scenario. This situation is not uncommon in 3D datasets, and the SHREC'2009 dataset is not an exception. Therefore, in our opinion, the results we obtained exhibit the importance of considering the size of partitions as an key aspect to accomplish good effectiveness.

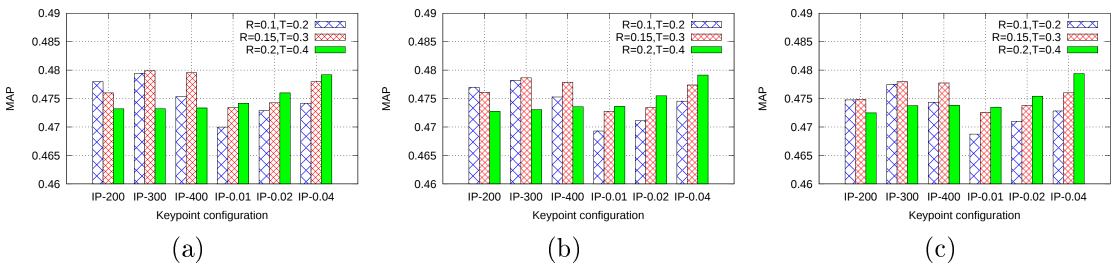


Figure 5.9: Mean average precision (MAP) and sensitivity analysis on our Quadratic Programming Matching approach (QPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

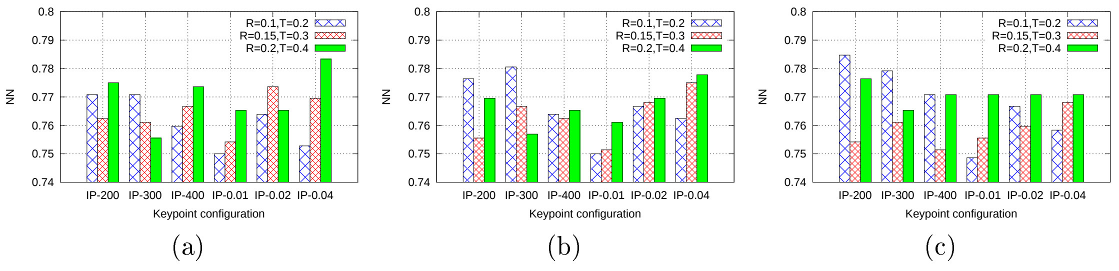


Figure 5.10: Nearest neighbor (NN) and sensitivity analysis on our Quadratic Programming Matching approach (QPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

Similarly to LPM, the First Tier (see Fig. 5.11) and Second Tier (see Fig. 5.12) in QPM exhibit an analogous behavior to its mean average precision. For the First Tier, large clusters give better results against their counterpart. Moreover, the highest FT score 0.4511 is obtained with the largest possible partition (IP-0.04, R = 0.2, T = 0.4, δ=1.5). Likewise, using the Second Tier, the predominant scores are present either when using large clusters or when using a scaling factor greater than 1.0.

5.3.4 Class-by-class Analysis

In this section, we show a more detailed evaluation of our approaches from the point of view of the effectiveness in each class of the dataset. The motivation to perform this evaluation is two-fold. First, all the retrieval measures used in previous experiments are a result of averaging. Average is a good way to condense a series of values. However, it can also hide valuable information in finer levels of analysis. Second, after seeing the results obtained in

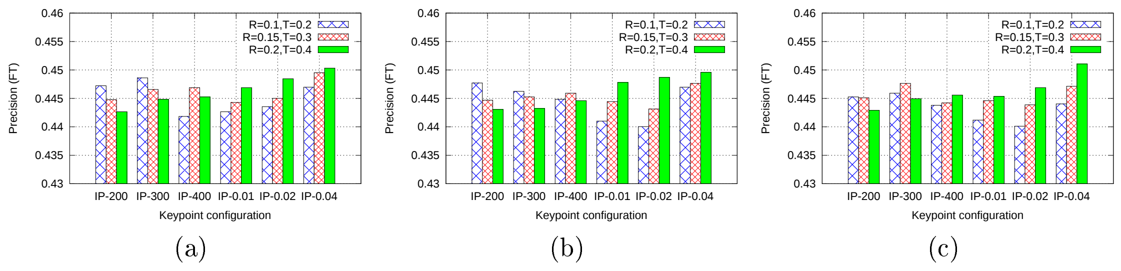


Figure 5.11: First tier (FT) and sensitivity analysis on our Quadratic Programming Matching approach (QPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

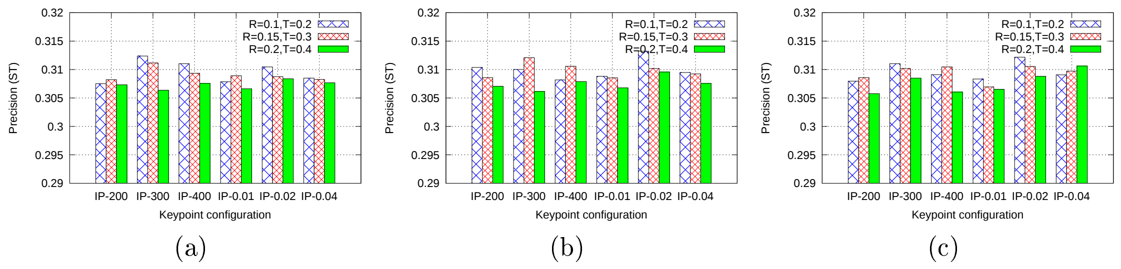
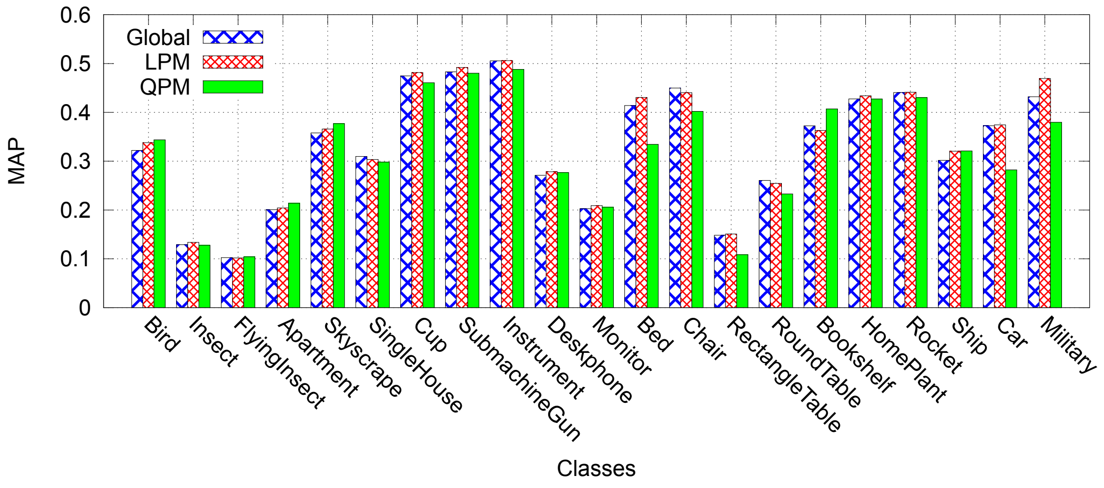


Figure 5.12: Second tier (ST) and sensitivity analysis on our Quadratic Programming Matching approach (QPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

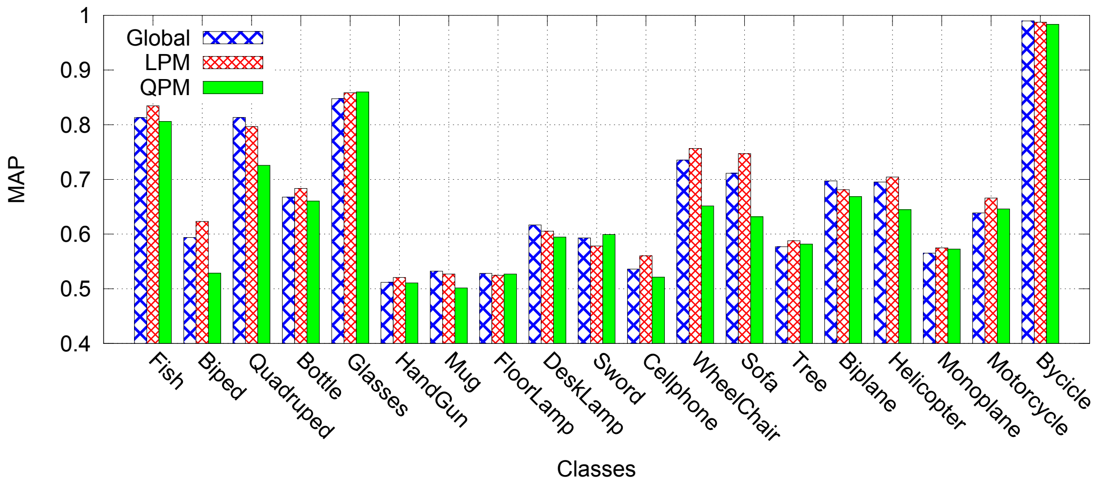
previous sections, our approach can be suitable depending on shape classes. So it is necessary to study the effect of our approach in each class of the dataset. Therefore, this can reveal useful information to decide when to effectively use our proposal. The results of this section were computed using the best combination found in the sensitivity analysis of Section 5.3.3, namely IP-0.02, R = 0.1, T = 0.2 and δ=1.0.

Figure 5.13 shows the mean average precision for each class in the SHREC'09 dataset. We divide the classes into two figures to best visualization. Each figure shows the comparison of Global Matching, Linear Programming Matching and Quadratic Programming Matching for each class as clustered bars. Our method was able to improve the effectiveness in 30 out of 40 classes. Moreover, when the objects within the same class have similar local structures and geometric consistency (such as in bookshelf, bird, apartment and skyscrape), the QPM approach outperforms the global matching and LPM.

Also, note the existence of 10 classes (single house, chair, round table, quadruped, mug, floor lamp, desk lamp, sword, biplane and bicycle) where it was not possible to improve the effectiveness with any of our approaches. However, it is also worth noting that in general, all of these 10 classes share a characteristic: the high variability of objects within the same class not only in a global sense, but also in a local sense. To illustrate this point, let us take as example the class *Chair*, on which our approaches did not improve. In our opinion, it is due to the high variability in the global sense. Moreover, shape parts also have a high variability (see Fig. 5.14). As a result, the keypoints can be concentrated in different parts



(a)



(b)

Figure 5.13: Mean average precision for each class in the SHREC'09 dataset. Plots were scaled for better visualization.

of the models, as each object can contain distinctive local features not repeatable in its class. Therefore, LPM and QPM can not take advantage of the partitioning technique proposed here. Consequently, this can be the cause for the moderate improvement of our method with respect to global matching. However, we believe that similar situations could influence the effectiveness of any algorithm.

The found evidence allows us to state the strengths and limitations of our approach. On the one hand, our method can improve the effectiveness in classes that share local information. That is, when models from the same class have common and similar parts, an improvement is expected. On the other hand, our approaches can not deal with extreme variability of parts between objects within the same class.

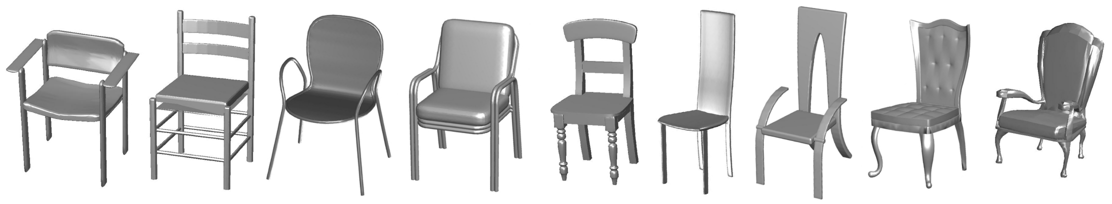


Figure 5.14: Samples of class *Chair*. Note the high variability of parts amongst shapes.

5.3.5 Correlation Analysis

In this section, we investigate the possible relationships between several factors that affect the effectiveness of our proposed methods. To do so, we use a correlation analysis and a statistical significance study among eight variables defined in the following, also introducing the abbreviations for each variable to be used in the analysis.

- Number of partitions (NP)
- Number of vertices (NV)
- MAP for global matching (GM)
- MAP for LPM (LPM)
- MAP for QPM (QPM)
- MAP gain of LPM over GM (G1)
- MAP gain of QPM over GM (G2)
- MAP gain of QPM over LPM (G3)

The three last variables were obtained by computing the difference of MAP scores of the involved methods. To obtain the data in this experiment, we computed the eight values using each model in the collection as a query. Therefore, we obtained eight values for each model, and subsequently we used all that information to compute the correlation matrix shown in Table 5.2. In addition, we computed the p-values for testing the hypothesis of no correlation. So for each correlation value, we have a p-value indicating the statistical significance of that correlation. We assume p-values < 0.05 as significant. The matrix of p-values is shown in Table 5.3.

The information provided by the correlation matrix and the p-values allow us to verify some aspects observed in previous experiments. For instance, there is a high correlation between the number of partitions and the three gain measures, namely G1, G2 and G3. First, the correlation between the number of partitions and G1 (MAP gain of LPM over GM) is positive. So the greater the number of partitions, the higher the improvement of LPM over GM. Second, the correlation between the number of partitions and G2 and G3 (MAP gain of QPM over GM and LPM, respectively) is negative. So it means that QPM benefits from fewer partitions. These two situations are consistent with the findings of Sec. 5.3.3 when we showed that QPM presents better results with large partitions. So now we can conclude that QPM is suitable for matching few large partitions. In contrast, LPM can perform a good matching regardless the geometric consistency, just by considering more small partitions. It is expected that the small partitions belong to distinctive features of the

Table 5.2: Correlation matrix between eight variables: Number of partitions (NP), number of vertices (NV), MAP for GM (GM), MAP for LPM (LPM), MAP for QPM (QPM), MAP gain for LPM over GM (G1), MAP gain for QPM over GM (G2), and MAP gain for QPM over LPM (G3).

Variables	**NP**	**NV**	**GM**	**LPM**	**QPM**	**G1**	**G2**	**G3**
NP	1.0000	0.0323	-0.0131	-0.0015	-0.0696	0.1318	-0.2658	-0.3145
NV	0.0323	1.0000	0.0361	0.0431	0.0361	0.0815	-0.0039	-0.0377
GM	-0.0131	0.0361	1.0000	0.9962	0.9784	0.0149	-0.2119	-0.2132
LPM	-0.0015	0.0431	0.9962	1.0000	0.9776	0.1022	-0.1972	-0.2351
QPM	-0.0696	0.0361	0.9784	0.9776	1.0000	0.0484	-0.0055	-0.0254
G1	0.1318	0.0815	0.0149	0.1022	0.0484	1.0000	0.1566	-0.2625
G2	-0.2658	-0.0039	-0.2119	-0.1972	-0.0055	0.1566	1.0000	0.9119
G3	-0.3145	-0.0377	-0.2132	-0.2351	-0.0254	-0.2625	0.9119	1.0000

Table 5.3: Matrix of p-values for the correlation between eight variables: Number of partitions (NP), number of vertices (NV), MAP for GM (GM), MAP for LPM (LPM), MAP for QPM (QPM), MAP gain for LPM over GM (G1), MAP gain for QPM over GM (G2), and MAP gain for QPM over LPM (G3).

Variables	**NP**	**NV**	**GM**	**LPM**	**QPM**	**G1**	**G2**	**G3**
NP	1.0000	0.3875	0.7262	0.9682	0.0621	0.0004	0.0000	0.0000
NV	0.3875	1.0000	0.3329	0.2484	0.3329	0.0287	0.9160	0.3121
GM	0.7262	0.3329	1.0000	0.0000	0.0000	0.6895	0.0000	0.0000
LPM	0.9682	0.2484	0.0000	1.0000	0.0000	0.0060	0.0000	0.0000
QPM	0.0621	0.3329	0.0000	0.0000	1.0000	0.1949	0.8836	0.4957
G1	0.0004	0.0287	0.6895	0.0060	0.1949	1.0000	0.0000	0.0000
G2	0.0000	0.9160	0.0000	0.0000	0.8836	0.0000	1.0000	0.0000
G3	0.0000	0.3121	0.0000	0.0000	0.4957	0.0000	0.0000	1.0000

objects. In summary, we have shown the importance of the number of partitions and their size in the improvement of the retrieval effectiveness of our technique.

Regarding the number of vertices of the meshes, there is a useful correlation which deserves attention. The number of vertices is highly correlated with the MAP gain of LPM over GM (G1). In other words, as the correlation is positive, we can say that LPM benefits from meshes with a large number of vertices. It reveals a remarkable connection with the previous analysis. With large number of vertices, one can expect meshes with more detail, and hence they can contain rich features. Moreover, if we prefer to select small clusters, the resulting partitions will be distinctive and small. In addition, we can obtain many partitions since the number of keypoints could depend of the number of vertices. Finally, following our previous analysis, LPM obtains better effectiveness when the input is a set of many distinctive partitions as a product of meshes with many vertices.

It is also worth noting the dependency of our approaches (LPM and QPM) on the global matching. This can be evidenced in the high correlation between GM and both LPM and QPM. Obviously, this fact is in accordance with the use of $\mu = 0.9$ which is associated to a

Table 5.4: Results for different values of μ in LPM using PANORAMA (values are in [0,100] scale)

μ	**NN**	**FT**	**ST**	**MAP**
0	42.0833	21.2337	15.3064	20.4128
0.1	60.6944	31.3889	21.8709	31.3174
0.2	74.5833	40.6536	28.6029	42.286
0.3	81.8056	48.268	33.7908	50.9521
0.4	85.4167	54.1912	38.701	57.3621
0.5	88.3333	57.8758	41.393	61.7762
0.6	88.8889	60.3513	43.1822	64.4322
0.7	88.4722	61.585	44.375	65.9167
0.8	88.8889	62.165	44.951	66.6028
0.9	**89.0278**	**62.3366**	**45.3023**	**66.813**
1.0	89.0278	61.9853	44.7917	66.7291

large contribution of the global matching in the final distance computation.

5.3.6 Results with PANORAMA

In this section, we present the results of our method using the PANORAMA descriptor [106]. For this experiment, we used the best parameter configuration as shown in Section 5.3.3. Table 5.4 shows the results by varying the contribution of the part matching (parameter μ) in the LPM technique. Similar to a previous experiment (see Section 5.3.2), we obtained the best results when $\mu = 0.9$. This result validates our argument about the contribution of the partition matching in the effectiveness of generic shape retrieval.

5.4 Concluding Remarks

In this chapter, we presented a shape retrieval method that combines global descriptors and part-based descriptors. We proposed a method for determining data-adaptive partition from meshes. Partitions were derived from agglomerations of distinctive keypoints on shapes. Finally, matching between partitions was stated as a integer program in order to compute correspondences.

From our experiments, it is possible to say that partition matching contributes to improving the retrieval effectiveness. Our method was able to achieve significant improvements in classes with objects containing common distinctive parts. In contrast, there is a limitation when objects within a class do not share common distinctive parts. Therefore, the partition matching degrades the effectiveness of global descriptors instead of improving it. Nevertheless, we believe that our approach partially attenuated this limitation with its ability to determine characteristic partitions.

In our opinion, our method offers new representational capabilities for 3D shapes which have proven to be effective in conjunction with global descriptors. In addition, we found a high correlation between the achieved effectiveness and the partitions provided by our method. Specifically, the number and size of the partitions play an important role for defining an effective similarity measure. This is because these two factors are well related to the quality of partitions (and their distinctiveness) and therefore, they influence the overall performance.

Chapter 6

Non-Rigid Shape Retrieval

A key aspect in multimedia collections is the content-based search. It aims at using the multimedia information itself in order to determine the similarity between two objects. Since associated meta-data is not always available, this searching approach is suitable. Moreover, some effective methods have been presented in the context of image searching, for instance. However, these methods are not directly adaptable to other media such as 3D data because it has its own characteristics and analysis tools.

There are several important issues involving content-based search. First, the goal is to provide good effectiveness when a query is given. That is, one wants to retrieve a ranked list of objects from the collection and expects to have similar objects on the first rankings. Second, the search time should be the small enough to ensure a practical use of the collection. Finally, the evolution of the retrieval system in time is also important. A retrieval system should be scalable and dynamic. These two aspects imply that the system should support a constant growth. This sounds logical, since the amount of multimedia information is growing rapidly.

We are interested in content-based retrieval of 3D shape collections. Our work consists in using local information from shapes in order to describe them. Thus, unlike a global approach where a shape corresponds to a single feature descriptor, a 3D shape is represented by a set of local features. Local features extracted from shapes allow us to deal with several problems such as deformable transformations, for instance. However, it also brings up the problem on how to effectively compare two 3D models represented as a set of local features. So far, many methods have been proposed in order to address the matching of 3D shapes using local information. The most of them have been proposed taking into account only the effectiveness, leaving out other key aspects such as efficiency and scalability.

In this chapter, we present a novel method to represent a 3D shape based on local features. Our method finds clusters of keypoints which correspond to surface regions with an outstanding local structure. Thus, it is possible to reduce the number of features to represent a shape, and therefore the subsequent process can be efficient. In addition, we apply the Signature Quadratic Form Distance (see Section 3.3.2) to compare two set of features. This distance has proven effective in the image search domain [6], and we will show that it can enhance 3D model retrieval.

In our opinion, there are several reasons why the SQFD is suitable for our purposes. First, it is a flexible way to compare two multimedia objects represented by feature sets. Coupled with it, if we are able to reduce the size of the representations, we can also get efficiency. Second, the distance is context-free as it involves just the feature signatures of the models being compared, not the rest of the database. So, it is easy to add new models into the collection because we just need to extract their representations and store them. This behavior would allow us to build a scalable system for dynamic collections. Third, as SQFD treats the object representations as a black-box feature sets, it is easy to redesign or adjust the entire model without the need of reinventing the mechanism of assessing similarity. Hence, SQFD is an universal distance for comparing object representations based on local features.

The contribution of this chapter is three-fold. First, We propose the use of keypoints on meshes in order to make the local features more discriminative. This step enables the reduction of the amount of information used in the matching. In addition, we present a novel method to detect clusters of keypoints on 3D meshes. Thus, each object will be represented by a set of features extracted from regions. Second, We apply the Signature Quadratic Form Distance to compare two models represented by a set of features. Finally, we evaluate our approach and compare it with recent techniques from the state of the art.

6.1 Finding Signatures for 3D Shapes

We present three simple approaches to use local features and the Signature Quadratic Form Distance. The three approaches heavily rely on a clustering process to determine the feature signatures. We use the Algorithm 4.2 for clustering.

6.1.1 Feature signatures on all vertices

In this approach we aim at partitioning the whole feature space of a 3D object to define the feature signatures. Let S be a 3D model with n vertices. We represent S as a set of heat kernel signatures computed for each vertex on the mesh. We call $FS(S)$ to this representation and it is formally defined as:

$$FS(S) = \left\{ \frac{hks(v_{\mathrm{i}})}{\|hks(v_{\mathrm{i}})\|} | v_{\mathrm{i}} \in S, \mathrm{i} = 1, \ldots, n \right\} \tag{6.1}$$

The normalization of the heat kernel signatures help us to properly define the intra-cluster and inter-cluster thresholds for the clustering.

Now, we can apply the clustering algorithm over $FS(S)$ (in the experiments of Sec. 6.2, the clustering parameters are $R = 0.1$, $T = 0.2$, $N = 10$). The clustering computes a

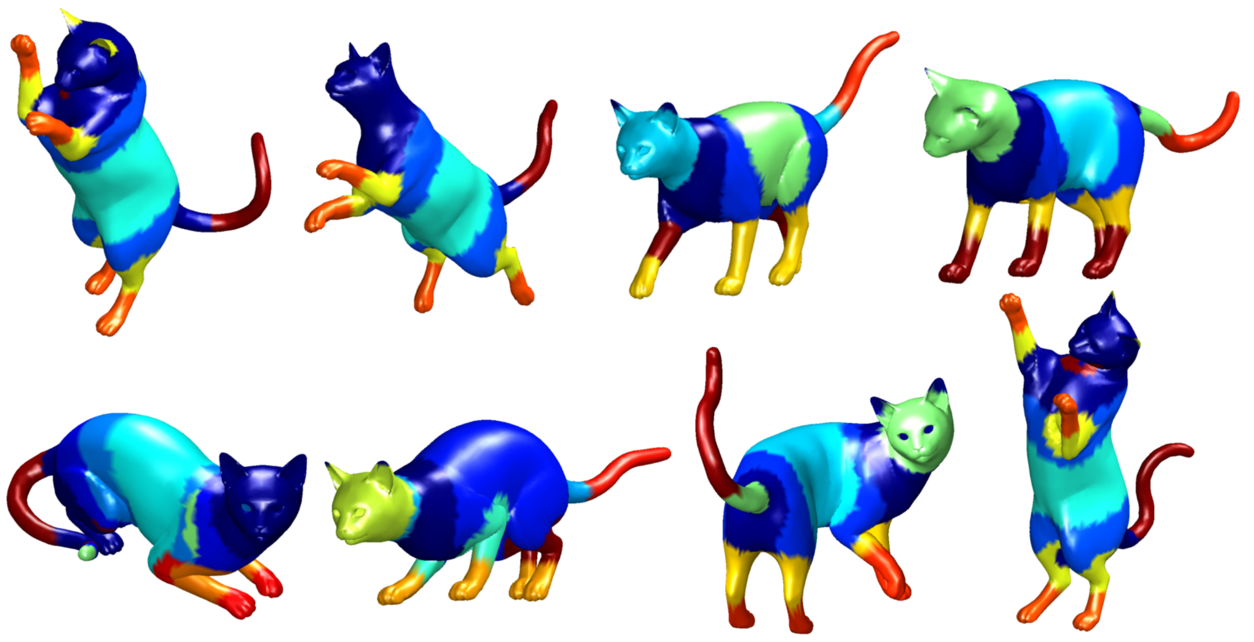


Figure 6.1: Colors represent the clusters using the HKS descriptors of the entire shape. The clusters are consistent despite the non-rigid transformations. Note: the colors are arbitrary and they are only used to show the resulting clusters in a shape.

partitioning not necessarily complete, as there are points which do not belong to any cluster. In this respect, each remaining heat kernel signature is assigned to the nearest cluster. Thus, now we have a complete and disjoint partitioning of the feature space. Finally, we represent S as suggested in Eq. 3.23.

Figure 6.1 is an example of local clustering using the HKS descriptors in shapes within the same class. Note that clusters are consistent and therefore, signatures will be consistent as well.

6.1.2 Feature signatures on keypoints

In this approach we aim at partitioning a sub-set of the feature space of a 3D object in order to define the feature signatures. The idea behind the use of a sub-set is that the whole feature space is not necessarily discriminative. In our opinion, there is a sub-set which is representative for each model. Therefore, our goal is to find that sub-set and partition it, so the feature signatures represents a partitioning of the most discriminative features.

However, the task of finding that sub-set directly from the feature space is difficult. Therefore, at this stage, we apply an interest point detection on meshes using our Harris 3D algorithm (Section 4.1) to select the 1% of the number of vertices as keypoints. In our opinion, detecting keypoints on meshes is a good choice to select the most relevant vertices on the mesh. Subsequently, we can use the heat kernel signatures on the keypoints as the sub-set needed.

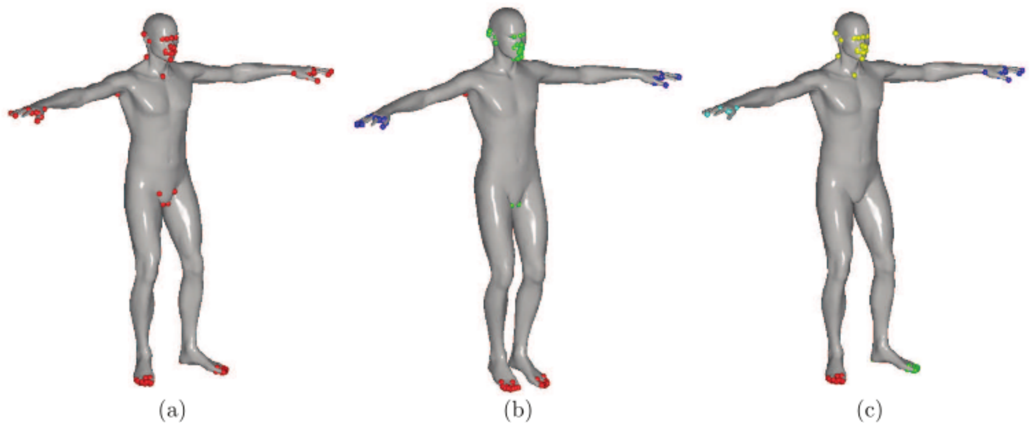


Figure 6.2: Local features in our approach: (a) Harris 3D keypoints, (b) clusters of keypoints based on their HKS, (c) keypoints within region of interest.

The justification to use an keypoint detector is that it allows us to reduce the amount of information for subsequent tasks. In our opinion, there are vertices which are not representative and, in general, their local geometry is found in all models. Therefore, these vertices are not suitable to discriminate between models from different categories. For example, Harris 3D, as described above, can reduce the number of vertices considerably, while maintaining the vertices which are highly discriminative in the shapes.

Once we have a set of keypoints, we need to compute the feature signatures for the 3D models. Let P be a 3D model with n vertices. Let $FS(P)$ be the feature space of P as defined in Eq. 6.1. After the detection step, we have a set of vertices $IP(P) = \{v | v \in P\}$ with m keypoints. So the feature sub-set induced by the set of keypoints is defined as

$$FS_{IP}(P) = \left\{ \frac{hks(v)}{\|hks(v)\|} | v \in IP(P) \right\} \tag{6.2}$$

The final step is to apply the clustering algorithm over the set $FS_{IP}(P)$ (in the experiments of Sec. 6.2, the clustering parameters are $R = 0.1$, $T = 0.2$, $N = 10$). Then, we proceed in a similar way as described in the previous section.

In addition to represent a 3D model with the most discriminative features, the use of keypoints allows us to reduce the computations in two ways. First, we need to compute less heat kernel signatures. Second, the clustering algorithm obviously takes less time. This issue is important because it is possible to improve the effectiveness while improving the efficiency.

Figure 6.2(b) shows an example of clusters of keypoints. Note that keypoints in both hands belong to the same cluster. This happens because the symmetry of the shape. Additionally, this phenomenon reduces the number of signatures considerably (in the figure, there would be only three signatures).

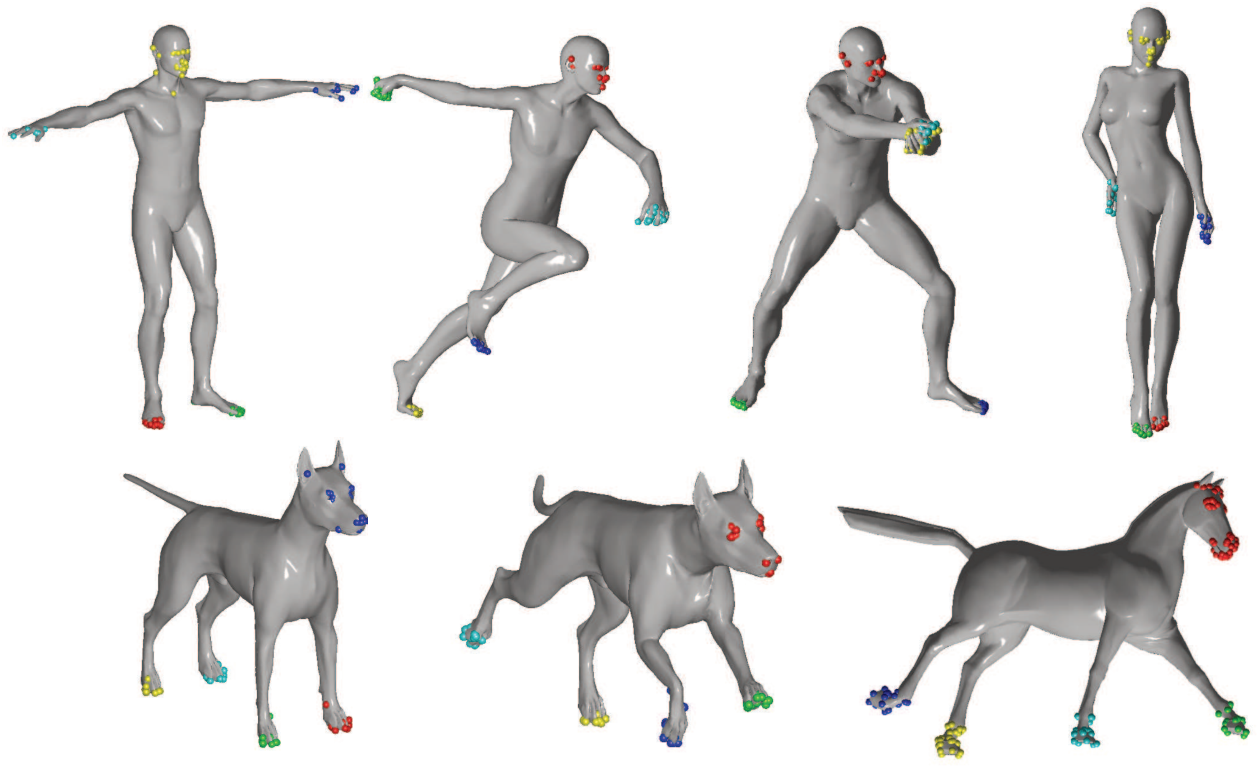


Figure 6.3: Clusters of keypoints detected on several 3D shapes.

6.1.3 Feature signatures on clusters of keypoints

This method is related to the key-components presented in Section 4.2. In this case, we do not perform the clustering in the feature space, but a geodesic clustering of keypoints. This step is performed using the algorithm proposed in Section 4.2.1 to group the keypoints. Our idea is to represent a shape with signatures obtained from the geodesic clusters of a shape. It is important to mention that we use the same parameters as the best configuration found for extracting the key-components (KC-1 variant in Sec. 4.2.3), except by the parameter N which in this case was set to 10.

With the clusters previously detected, now we proceed to describe them. Let R be the set of clusters of length $n = |R|$ from a shape P. Each cluster contains a set of keypoint for which we compute the heat kernel signatures. Therefore, at this stage we have a set of clusters represented by the normalized descriptors of their corresponding keypoints.

We compute a feature signature for a 3D object, where each cluster corresponds with a unique descriptor and a weight as in Eq. 3.23, where the element c_{i}^{P} is the average of the normalized heat kernel signatures in the cluster, and the weight w_{i}^{P} is proportional to the number of elements in R_{i} with respect to the total number of keypoints after clustering.

In this approach, the keypoints which were used to find the key-components (Section 4.2) are used to compute a signature. Interestingly, the clusters of keypoints are discriminative and representatives as can be shown in Fig. 6.3.

6.2 Experimental Evaluation and Discussion

In this section, we present our experiments and results. Section 6.2.1 explores the importance of the parameters in the Signature Quadratic Form Distance. Section 6.2.2 evaluates the indexability of the SQFD in our approach. Section 6.2.3 presents the results concerning to efficiency of our retrieval system. Finally, Section 6.2.4 compares our technique with state-of-the-art methods.

6.2.1 Effectiveness Evaluation

The SQFD is defined taking into account a similarity matrix A_{f_S}. The similarity matrix needs a ground dissimilarity function which needs to be metric. Common examples of dissimilarity functions are the L_p distances. In this section, we evaluate the distance functions L_1, L_2, L_∞. In addition, the transformation of distance values into similarity values depends on a α parameter when the Gaussian or the heuristic similarity function are used. We also evaluate the effect of α in our three proposed approaches. We name our variants as Total (Section 6.1.1), Keypoint (Section 6.1.2), and Cluster (Section 6.1.3) respectively. This will help us to present the results in a more concise way. We use the same measures for effectiveness evaluation as described in Sec. 5.3.1.

Here we use the mean average precision (MAP) to evaluate the effectiveness of our proposals. Figure 6.4 shows the effect of α in the Mean Average Precision using the Gaussian and the heuristic similarity function applied to the variant Total. For the Gaussian function, the best MAP values are obtained using the L_2 distance and small values of α. On the other hand, for the heuristic function, the best MAP values are also obtained using the L_2 distance, but using large values of α. It is important to note that the discriminative power of the SQFD depends on the transformation of descriptor distances into similarity values. More important, the parameter α plays an important role in the distinctiveness between signatures. For example, using the Gaussian function, MAP values drop considerably after $\alpha = 10$. The explanation of this fact is that the slope of the similarity functions flatten in large values of α for the Gaussian function and small values for the heuristic function. Note that a near-to-flat similarity function is not useful because it would not assign very similar values to different ground distance values.

Figures 6.5 and 6.6 show the mean average precision obtained for the variant Keypoint and Cluster, respectively. The behavior is similar to that obtained with the variant Total: small values of α get the best MAP in the Gaussian function, as opposite to the heuristic function which requires large values of α.

Table 6.1 shows a comparison of our variants with the best MAP's obtained. In each variant, the best MAP is obtained using the L_2 distance and the heuristic function with $\alpha = 1000$. In addition, the Total variant obtains the best overall MAP of 0.8497. The effectiveness decreases in variants Keypoint and Cluster, with a slight difference between them.

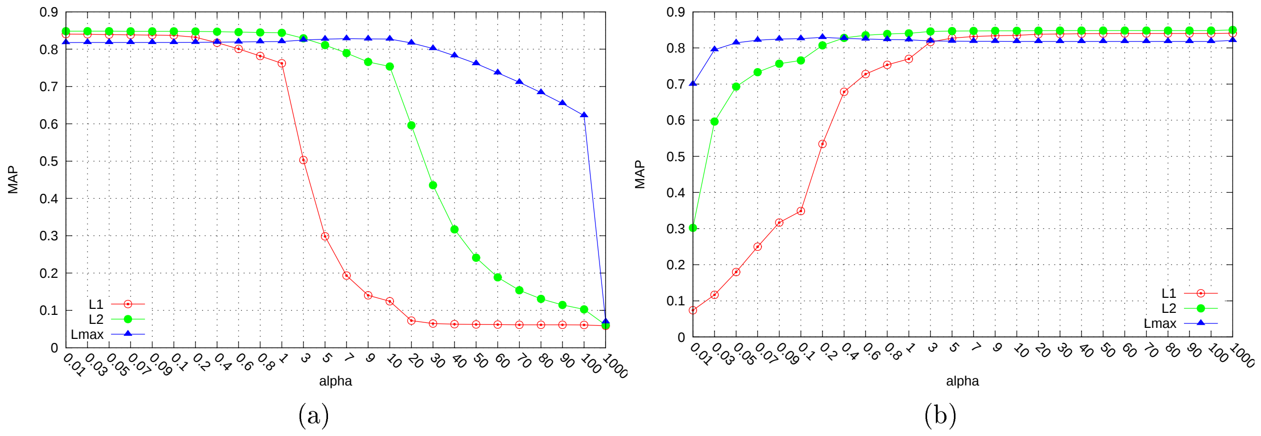


Figure 6.4: Mean average precision for our variant Total. (A) Gaussian similarity function. (B) Heuristic similarity function.

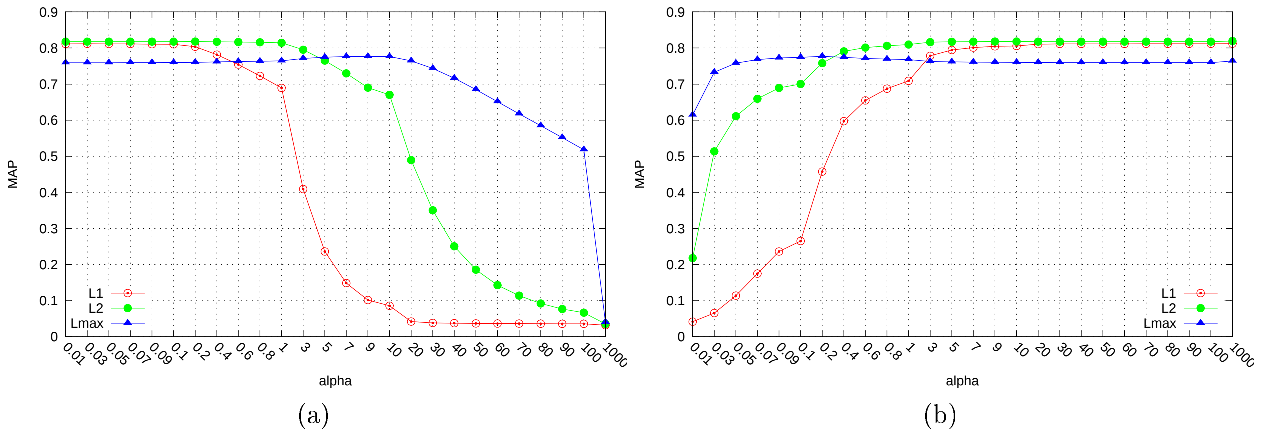


Figure 6.5: Mean average precision for our variant Keypoint. (A) Gaussian similarity function. (B) Heuristic similarity function.

In our opinion, the predominance of the Total variant with respect to the others can be explained by the amount of information used in the process of description. Since the Total variant uses the complete set of features, it allows to include all this information in the description. As the clusters in the feature space seems to be consistent in the presence of non-rigid transformations, the SQFD takes advantage of the intrinsic information of the signatures to perform an effective assessment of the similarity between shapes. In contrast, the selection of local features (keypoints and clusters of keypoints) could decrease the representational power of the signatures. We believe that, although keypoints and clusters of keypoints are robust, some information is missing in order to describe an entire shape. Probably a few keypoints with a bad localization (not repeatable in the class) affect the whole process of description using the signatures. Nevertheless, it is worth noting that the MAP remains above 80% in all our variants, which is an indication of the effectiveness of out three proposals.

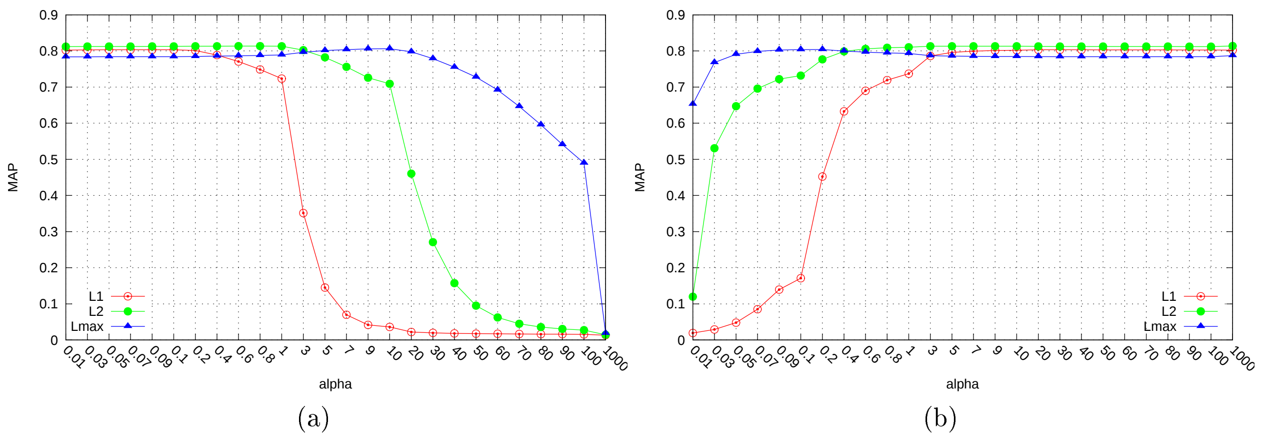


Figure 6.6: Mean average precision for our variant Cluster. (A) Gaussian similarity function. (B) Heuristic similarity function.

6.2.2 Indexability Evaluation

Another important aspect of our evaluation is the indexability of the collection using the SQFD. We computed the intrinsic dimensionality, which will be an indicator of how indexable is the SQFD in our approaches. A low intrinsic dimensionality corresponds to metric spaces where many objects can be discarded from the use of the triangle inequality. Therefore, a low intrinsic dimensionality would imply an efficient algorithm for searching.

To compute the intrinsic dimensionality, we use the proposal of Chavez et al. [29]

$$\rho = \frac{\mu^2}{2\sigma^2} \tag{6.3}$$

where μ and σ^2 are respectively the mean and the variance of the distance histogram of a metric space. In practice, given an object collection, we compute 1×10^6 distances between random objects, and subsequently we compute the mean and the variance.

Figures 6.7, 6.8 and 6.9 show the effect of α in the intrinsic dimensionality using the Gaussian and the heuristic function applied to our three variants.

Interestingly, the lowest values of the intrinsic dimensionality are obtained in values of α where we obtained the best MAP's (that is, small α values for the Gaussian function, and large α values for the heuristic function). This evidence is important because we can choose a good parameter in order to obtain both effectiveness and efficiency at the same time.

In this experiment, it is clear that the Cluster variant obtained the lowest values of intrinsic dimensionality (Fig. 6.9). This can be explained by the specificity of the signatures. Since the Cluster variant is based on a process of geometric selection, the resulting signatures are more specific than in the other variants. This specificity contributes in the distribution of the signatures in the space, allowing a large variance. For instance, the specificity would help to guarantee that signatures of shapes in the same class are similar. In contrast, signatures in

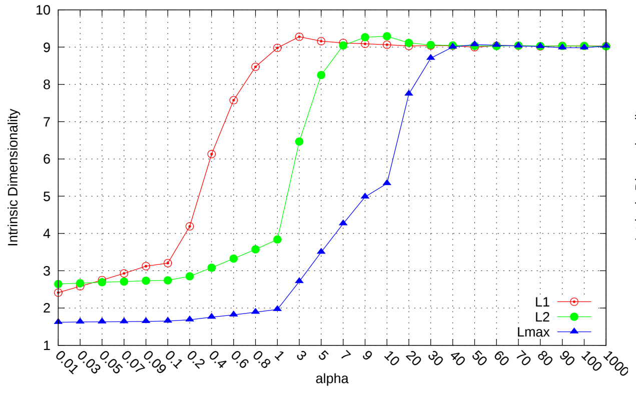
Variant	**Function**	α	**Distance**	**MAP**
Total	Gaussian	0.01	L_1	0.8403
		0.01	L_2	**0.8482**
		9.0	L_∞	0.8275
	heuristic	1000	L_1	0.8407
		1000	L_2	**0.8497**
		0.2	L_∞	0.8291
Keypoint	Gaussian	0.02	L_1	0.8116
		0.02	L_2	**0.8177**
		9.0	L_∞	0.7765
	heuristic	1000	L_1	0.8120
		1000	L_2	**0.8196**
		0.2	L_∞	0.7769
Cluster	Gaussian	0.09	L_1	0.8034
		0.8	L_2	**0.8136**
		10	L_∞	0.8060
	heuristic	30	L_1	0.8034
		1000	L_2	**0.8137**
		0.2	L_∞	0.8040

Table 6.1: Best MAP's for each variant along with their parameter choices.

different classes should be dissimilar enough. Therefore, the distance variance would increase due to the specificity. On the other hand, let us look at the case of the Total variant. In this case, each signature contains a average descriptor which represents the entire cluster. In our opinion, the averaging decreases the specificity of the signatures. Clearly, this fact affects the distance variance because less specificity increase the possibility of having similar signatures in the entire collection. To illustrate this point, we computed the variance of the histogram of distances for the Total variant (with heuristic function, L_2 distance and $\alpha = 1000$) and the Cluster variant (with heuristic function, L_2 distance and $\alpha = 10000$). The parameter configurations were chosen according to the intrinsic dimensionality obtained in Figures 6.7 and 6.9, where the intrinsic dimensionality of the Cluster variant reached the lowest value. The variance for the Total variant was 8.2258×10^{-8} and the variance of the Cluster variant was 2.1871×10^{-6}. This experiment shows our argument about the intrinsic dimensionality and its relation to the specificity of the signatures.

6.2.3 Efficiency Evaluation

The SQFD is an expensive distance, and therefore it is important to provide mechanisms to accelerate the search. When an object is queried, the most naive approach to measure the distance to all object in the collection is linear scan. In this section, we show that we can accelerate the search process considerably by organizing the collection in an index. Given a collection, the procedure to obtain the index is as follows: (1) Choose a set of pivot objects using the SSS algorithm (see Sec. 3.3.1), (2) Compute and store the distances between pivots and object collection (this will be our index, a pivot table). Then, pivot tables can be used


(a)

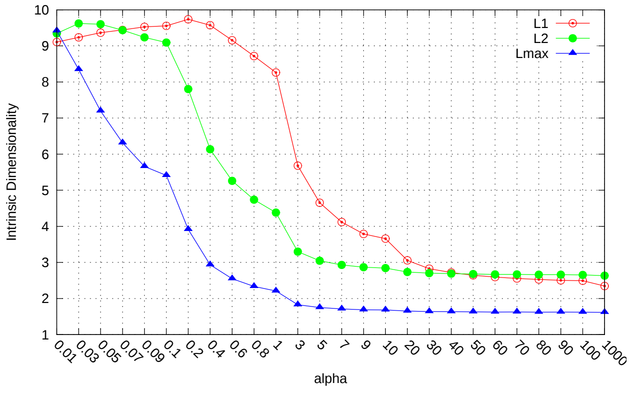
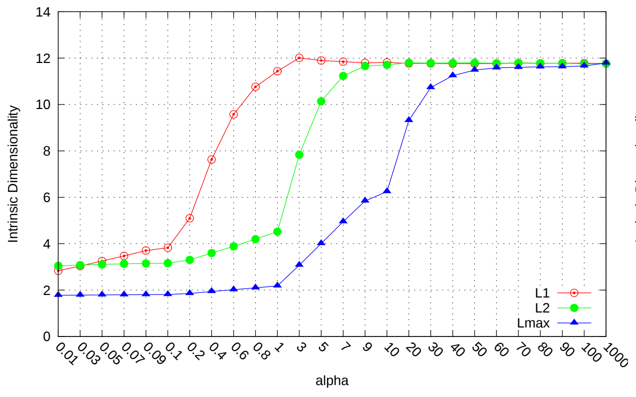

(b)

Figure 6.7: Intrinsic dimensionality for our variant Total. (A) Gaussian similarity function. (B) Heuristic similarity function.


(a)

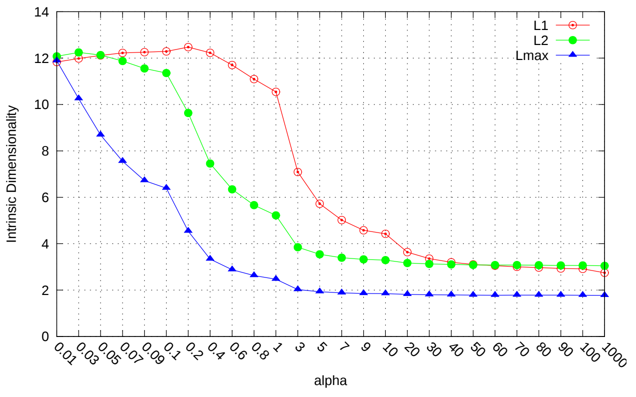

(b)

Figure 6.8: Intrinsic dimensionality for our variant Keypoint. (A) Gaussian similarity function. (B) Heuristic similarity function.

for searching using the algorithm shown in Sec. 3.3.1.

To test the efficiency of our approaches, we used k-NN queries to make the comparisons. For each result in this section, we executed 100 k-NN queries (using the algorithm 3.2 with k = 10, and randomly chosen queries) and then we average the searching time. In comparison, the same set of queries are used to perform linear scans and times are measured accordingly. In order to compare the metric approach to the linear scan, we use the speedup, or the ratio between the linear scan time and the pivot-based time.

Figures 6.10, 6.11 and 6.12 show the speedup for our three variants respectively. A first observation is the direct relationship that exist between intrinsic dimensionality and speedup. The highest values of speedup are obtained for small values of α in the Gaussian function and large values for the heuristic function. In the case of the Total and Cluster variants, the speedup is almost 20x with the Gaussian similarity function. Also, the maximum speedup is obtained with the L_1 distance, which is expected because this distance is cheaper than the

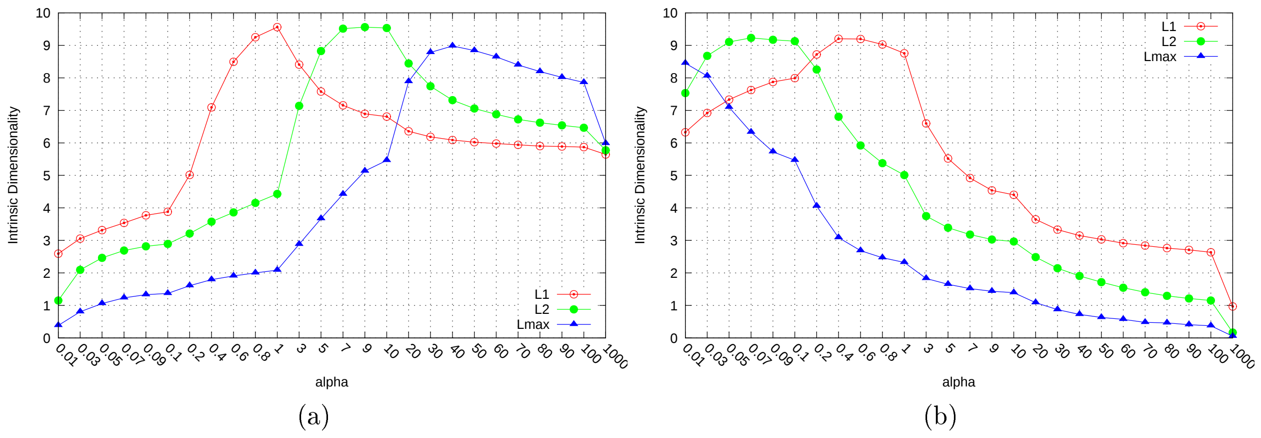


Figure 6.9: Intrinsic dimensionality for our variant Cluster. (A) Gaussian similarity function. (B) Heuristic similarity function.

others.

The most important result of this experiment is obtained with the variant Keypoint(Fig. 6.11). The maximum speedup we obtained is above 22x using the Gaussian similarity function and it represents the best improvement in term of efficiency. Particularly, this variant is efficient because the signatures are smaller than in other variants. Therefore, the evaluation of the SQFD takes less time in average.

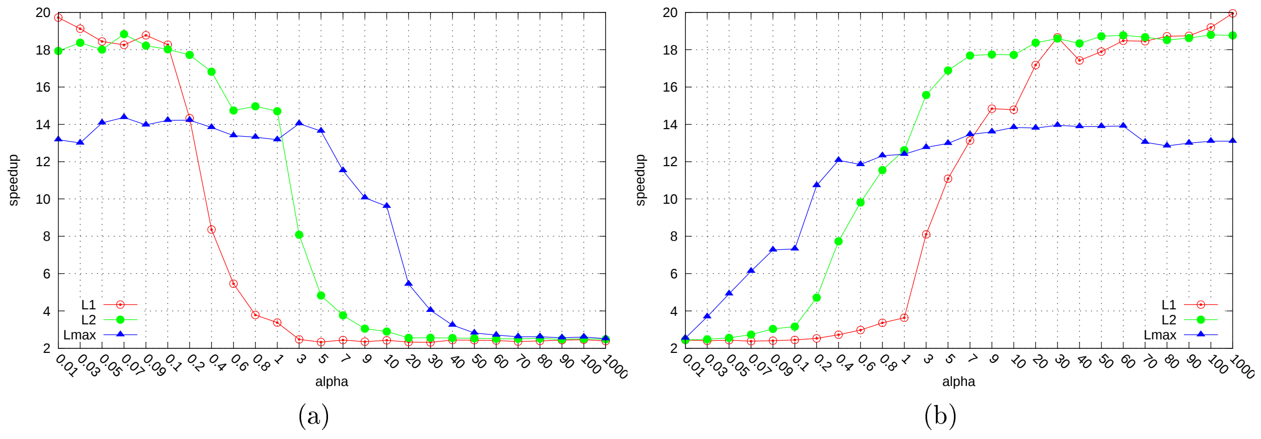
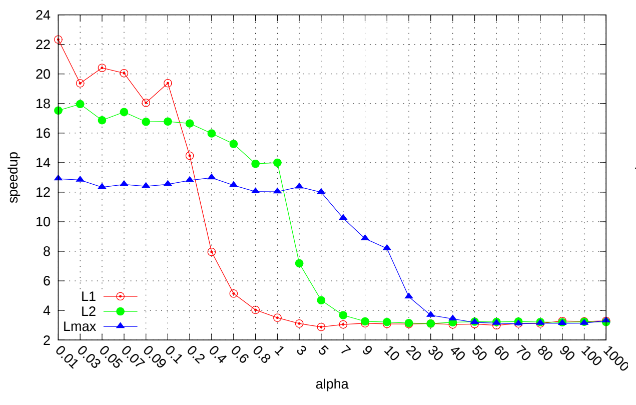


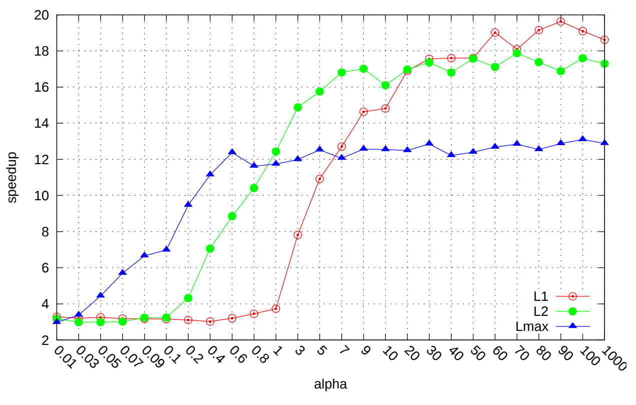
Figure 6.10: Speedup for our variant Total with respect to linear scan. (A) Gaussian similarity function. (B) Heuristic similarity function.

Next, we present results regarding the query time. Table 6.2 shows the average required time for extracting the feature signatures from a query, and subsequently searching the 10 nearest neighbors. In addition, we report the average number of features per signature for each variant. In this experiment, we use the heuristic function with the L_2 distance and $\alpha = 1000$. This parameter combination obtained the best MAP values in Section 6.2.1, and now we want to test how efficient can be our variants in the same scenario.

The scenario of query time is favorable to the Keypoint variant. Obviously, this method

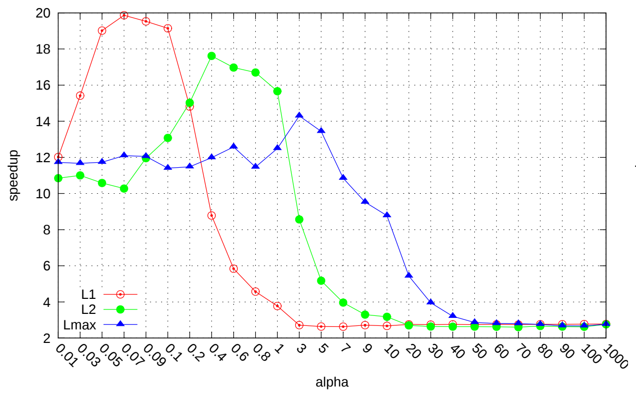


(a)

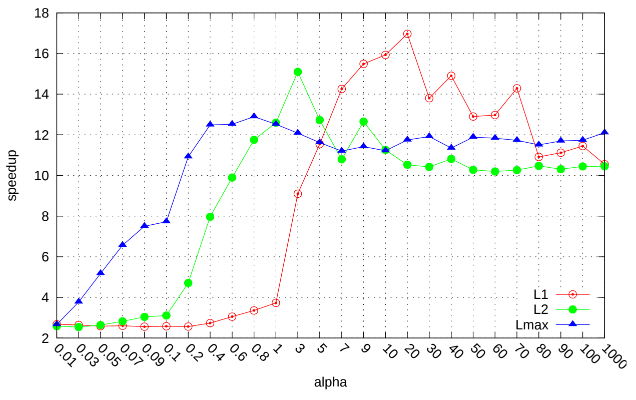


(b)

Figure 6.11: Speedup for our variant Keypoint with respect to linear scan. (A) Gaussian similarity function. (B) Heuristic similarity function.



(a)



(b)

Figure 6.12: Speedup for our variant Cluster with respect to linear scan. (A) Gaussian similarity function. (B) Heuristic similarity function.

considers a reduced number of descriptors in the clustering, so the computation of the feature signatures is fast. In addition, as it contains the least average number of signatures, the SQFD is applied rapidly. In contrast, the Total variant took more time due to the large number of descriptors (one per vertex) used in the clustering. On the other hand, the Cluster variant took a considerable time to compute the signatures. This is because this variant considers additional tasks such as calculation of geodesic distances and multi-dimensional scaling. Therefore, the use of some variant depends on the application and the required efficiency or effectiveness.

6.2.4 Comparison with the State of the Art

Our method was also compared to methods from the state of the art. We chose two representative techniques: Shape Google which uses a Bag-of-Features approach using heat kernel-based descriptors, and ShapeDNA which is an effective global technique for deformable

Method	**Num. Feat.**	**Sig.(sec)**	**10-NN search with index (sec)**
Total	14	0.8738	0.0741
Keypoint	5	0.0055	0.0197
Cluster	9	1.1263	0.0579

Table 6.2: Average number of feature signature and required times for computing the feature signatures and querying.

shapes. Following, we briefly describe each technique and present the parameter configuration.

- **Shape Google.** Bronstein et al. [15] proposed to use a Bag-of-Features approach with heat kernel-based descriptors extracted from a shape. The idea is to represent a shape as a distribution of the occurrence of a visual vocabulary computed by clustering the whole set of descriptors from a collection. Bronstein et al. suggested to use a scale-invariant version of the heat kernel signatures (SI-HKS [21]). In this experiment, we use the same configuration proposed originally by the authors with the exception of the values for τ which were from -25 to 1 with increments of 1/16. For the dictionary, we used a vocabulary of size 48 calculated via k-means clustering.
- **ShapeDNA.** Reuter et al. [113] proposed a descriptor invariant to deformable shapes. The method computes the Laplace-Beltrami operator for a shape, and subsequently, it performs a eigen-decomposition. A few largest eigenvalues are considered as descriptor for the whole shape. In this work, we consider 10 eigenvalues for each shape.

First, we compare the methods in terms of effectiveness using the best configuration presented in Table 6.1. Our method obtains the best results regarding MAP, NN, FT, and ST. In addition, Figure 6.13 shows the precision-recall plot, where we can observe the predominance of our technique over ShapeDNA and ShapeGoogle.

Methods	**MAP**	**NN**	**FT**	**ST**
ShapeDNA	0.7199	0.9039	0.6707	0.4155
ShapeGoogle	0.7676	0.9301	0.6963	0.6530
Total variant	**0.8497**	**0.9476**	**0.7990**	**0.7340**

Table 6.3: Comparison of our method with the state of the art.

The comparison of effectiveness encourages us to think that the use of a local clustering is useful for describing shapes. While Shape Google aggregates the local descriptors through a quantization using a global dictionary, our method entirely considers the information of a shape for describing it. It is possible that the global dictionary attenuates the discriminative power of local descriptors, hence reducing the effectiveness. On the other hand, ShapeDNA only considers the spectrum of a shape for description. Although the final descriptors are simple to compare, it is possible that the ShapeDNA do not convey the needed information to effectively discriminate shapes.

With respect to the query time, Table 6.4 shows the average query time for computing a 10-NN search. For our approaches, we added up the average time of computing the signature with the average time of performing a 10-NN search using a pivot index. As can be

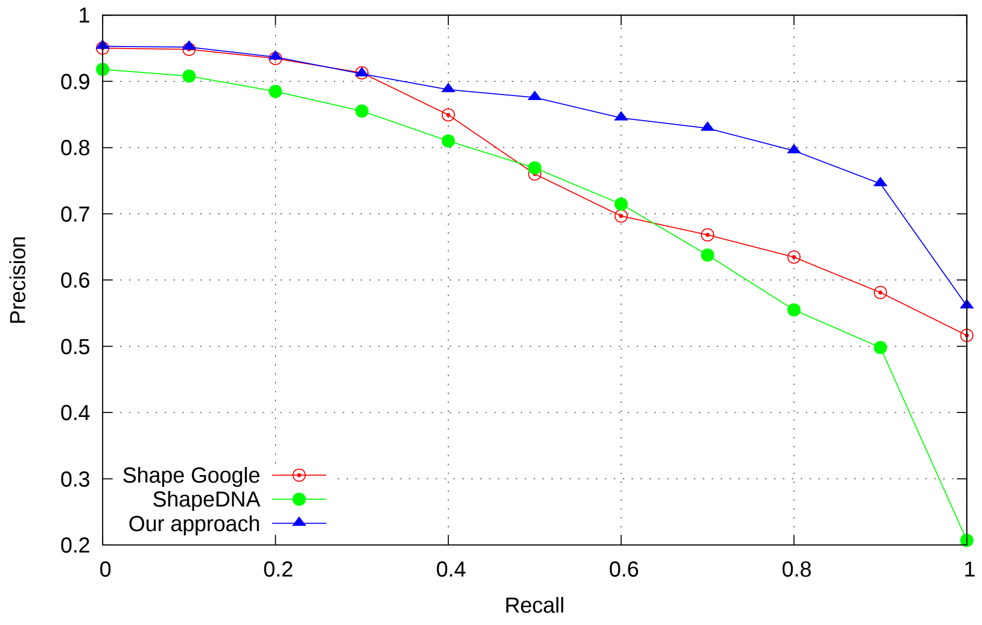


Figure 6.13: Comparison of precision-recall curves for our method and methods from the state of the art.

Method	Query time
ShapeDNA	0.01
Shape Google	0.1330
Total	0.9479
Keypoint	0.0252
Cluster	1.1842

Table 6.4: Query time for each compared method.

seen, ShapeDNA is the fastest method, as it considers only to apply a distance function(for instance, L_2) between descriptors. In the case of Shape Google, most of the time is used to compute the bag of features, and subsequently to perform the distance computations. However, the most important result about query times is that obtained by our Keypoint variant. It is almost as fast as ShapeDNA, but much more effective. Therefore, the Keypoint variant is a good alternative for a retrieval system where efficiency is a matter.

Finally, we claim that our methods can be potentially used for large collections due to the high effectiveness, and their efficiency is good in memory and comparable in time. In addition, our methods allow us to have a dynamic system, where new models can be easily added into the system.

6.3 Concluding remarks

In this chapter, we propose a novel approach for non-rigid shape retrieval using local features. The representation using a local partitioning of the feature space has shown to have a positive impact in retrieval performances. This fact can be noted in the high effectiveness achieved by the Total variant and the good efficiency achieved by the Keypoint variant. Likewise, the Signature Quadratic Form Distance has proven to be effective for assessing the similarity of shapes. In addition, the indexability of the SQFD allows us to improve considerably the efficiency of a retrieval system. Also, the SQFD has enabled the possibility of implementing our method in dynamic collections. The advantage of our approach is that the process of description of a shape is independent of the rest of shapes. This is important if we consider a retrieval system as a dynamic system which is constantly updated.

An important aspect of our results is that, depending of the requirements for a retrieval system, our method offers several characteristics. On the one hand, if we are interested in the retrieval results, the Total variant can be used because its high effectiveness. On the other hand, if we are interested in the query time, the Keypoint variant performed better in this case. Anyway, our approach (in its different variants) performed better than state-of-the-art methods.

Chapter 7

Shape Matching

The problem of finding correspondences in 3D shapes is an important problem in computer vision and computer graphics. In particular, the reliable detection of correspondences in non-rigid shapes has received notable attention in recent years. This problem is inherently difficult due to the complexity of formally characterizing a non-rigid transformation. Additionally, in real-world applications, one would expect shapes containing perturbations such as noise, topological changes, scale, and so on. Therefore, it is imperative to devise robust and efficient techniques to finding reliable correspondences in non-rigid shapes (possibly with perturbations).

The most used approach to tackle this problem implies to find a mapping between sparse sets of surface points (keypoints). This problem commonly is formulated as an optimization problem that involves the matching of local descriptors and some criterion for geometric consistency. Generally, the optimization is carried out through an integer quadratic program.

Due to the combinatorial nature of the correspondence problem, we need to search strategies to efficiently solve the problem. Additionally, we need to take into account the robustness against mesh perturbations. In this chapter, we propose an algorithm to find correspondences in non-rigid shapes based on a hierarchical decomposition. In Chapter 4, we have shown that it is possible to obtain robust decompositions on 3D meshes. Our motivation is the fact that if two shapes are near-isometric, they should have regions which are near-isometric as well. If a shape S is decomposed in a set of regions $S_1, S_2, \ldots, S_n$, then a shape T (near-isometric to S) should also have a partition set $T_1, T_2, \ldots, T_n$, where S_i is near-isometric to T_j. This idea can be applied again on regions recursively.

In light of this observation, we propose an algorithm to build a *decomposition tree* of a given shape. Internal nodes represent regions and leaf nodes represent keypoints. As one traverses the tree in depth, the shape structures are smaller. In addition, we propose a hierarchical matching algorithm which takes two decomposition trees as input and performs in a bread-first manner. This algorithm first matches regions in high levels of the trees and propagates the process until reaching the leaf nodes, where finally the keypoints are matched. The decomposition tree is similar in spirit to the component tree proposed by Litman et al. [83]. The main difference is that our representation is designed to support the subsequent matching process. In addition, in contrast to the component tree (which makes use of diffusion geometry), our method uses the distribution of keypoints on the surface to guide the decomposition process.

In our method, we address two important aspects: efficiency and robustness. Regarding efficiency, our matching algorithm has the ability of reducing the searching space of correspondences by making use of the hierarchical decomposition. Once two regions correspond (because their internal nodes matched), we only look for correspondences in the associated sub-trees. This allows us to discard an important amount of matches early in the process. With respect to the robustness, we take advantage of the provably good properties of diffusion-based descriptors in the context of non-rigid matching to obtain discriminative descriptions for regions and keypoints. In addition, the decomposition process is guided by the distribution of robust keypoints on the shape's surface.

In summary, the contributions of this chapter are two-fold. First, we present a novel representation for non-rigid shapes: the *decomposition tree*. This structure characterizes the hierarchical decomposition process, where the root node contains the original shape and leaf nodes contain keypoints. Second, we develop a matching algorithm that takes advantage of the decomposition trees to find correspondences. Our algorithm is efficient as it allows us to discard matches in early stages of the process.

7.1 The Decomposition Tree

In this section, we present the decomposition algorithm. It has two stages: the pre-processing step and the generation of the decomposition tree. Our approach to decompose a mesh is inspired in the key-components proposed in Chapter 4.

7.1.1 Pre-processing

Given a 3D shape X, we compute a set of keypoints K_X using the Harris 3D algorithm. Subsequently, we calculate the geodesic distances between each pair of keypoints using the fast marching method [71]. These distances will be used to control the geometric consistency of the correspondence set in the matching algorithm.

Since the decomposition method is based on the distribution of keypoints on the shape's surface, we propose a heuristic to discard isolated keypoints. The heuristic consists of analyzing the distribution of geodesic distances of a keypoint. Let D_p^v be the geodesic distances from a keypoint v to its p nearest keypoints. We define the *density* of a keypoint v as

$$\mathrm{de}nsity(v) = \frac{mean(D_n^v)}{mean(D_3^v)}, \tag{7.1}$$

where $n = |K_X|$.

Grouped keypoints are expected to have a small $mean(D_3^v)$ compared to $mean(D_n^v)$, giving a high density value. In contrast, isolated keypoints will have more similar values for these two quantities, with the density approaching to one. Therefore, keypoints with a low density value should be discarded. We remove keypoints with $density(v) < 30$ from K_X. The cut-off value was found empirically.

In addition, for each remaining keypoint in K_X, our method computes two descriptors which are based on diffusion geometry: HKS (heat kernel signatures [136]) and WKS (wave kernel signatures [4]). Descriptions for these descriptors were given in Chapter 3.

The use of these two descriptors is related to their ability for feature localization [17]. The heat kernel signature is a collection of low-pass filters which inhibit high frequencies. For this reason, its nature is more global and it may affect the exact localization of correspondences. In contrast, the wave kernel signature is a collection of band-pass filters which reduces the impact of low frequencies, allowing a better feature localization. Therefore, we take advantage of these facts and use the HKS and WKS in different levels of our representation. The former is used for describing regions in the internal nodes of our decomposition tree and the latter is used for describing keypoints in the leaf nodes.

In summary, after the pre-processing step, we have the following information: a set of keypoints K_X, the complete set of geodesic distances between keypoints, and descriptors (HKS and WKS) for each keypoint.

7.1.2 Decomposition

Our decomposition algorithm relies on the distribution of keypoints on the shape's surface. Our algorithm performs a hierarchical clustering in the geodesic space of keypoints. More specifically, the algorithm first looks for large groups of keypoints, and recursively decomposes the groups into smaller groups. Each group of keypoints determines a mesh region. The decomposition ends when we cannot divide a group into smaller groups or when the area of the region that covers a group is very small relative to the area of the original shape. The outcome is a tree which represents the decomposition process. Fig. 7.1 shows a decomposition tree obtained with our algorithm. Furthermore, Algorithm 7.1 presents the pseudo-code of the decomposition algorithm in detail.

The input of the decomposition algorithm is a node T, which contains the original shape and all information computed in the pre-processing step (see Sec. 7.1.1). Technically, the output of this method is a tree with T as root node. The first part of the algorithm checks whether it is possible to continue decomposing a node (lines 3-13). The first check is through the comparison of areas between the shape at the input node and the original shape. If the mesh at node T is too small, then T is marked as leaf node. Subsequently, we propose to apply a medoid-based adaptive clustering to find groups of keypoints. These groups will generate regions with their associated new nodes. If it is not possible to obtain more than two clusters, the input node is marked as leaf.

There are two points of our algorithm that deserve carefully attention: the clustering

Algorithm 7.1 GenerateTree(T)

```
Require: Node $T$
Ensure: Node $T$ and its associated tree

 1: areaOriginal ← mesh.getArea()
 2: areaNode ← $T$.mesh.getArea()
 3: if areaNode $< 0.1\times$ areaOriginal then
 4:     isLeaf ← true
 5: end if
 6: if not isLeaf then
 7:     Let R be the intra-cluster threshold.
 8:     Let S be the inter-cluster threshold.
 9:     $C$ ← clustering($T$.distancesKeypoints(), R, S)
10:     if $|C| < 2$ then
11:         isLeaf ← true
12:     end if
13: end if
14: if isLeaf then
15:     $T$.leaf ← true
16:     return $T$
17: end if
18: for each cluster $c$ in $C$ do
19:     Create a new node $T_c$
20:     $(o_c, r_c)$ ← GetMin3DSphere(c.getKeypoints())
21:     $k_c$ ← c.getMedoid()
22:     $T_c$.mesh ← $T$.mesh.getPatch($o_c$, $r_c$, $k_c$)
23:     $T_c$.descriptor ← $T$.getPatchDescriptor(c.getKeypoints())
24:     Propagate keypoints from $T$ to $T_c$
25:     Propagate keypoint distances from $T$ to $T_c$
26:     Propagate keypoint descriptors from $T$ to $T_c$
27:     $T$.children[c] ← $T_c$
28: end for
29: for each cluster $c$ in $C$ do
30:     for each cluster $g$ in $C$ do
31:         $T$.childrenDistance[$c$,$g$] ← $d_g(k_c, k_g)$
32:     end for
33: end for
34: for each cluster $c$ in $C$ do
35:     $T$.children[$c$] ← GenerateTree($T$.children[$c$],$\delta$)
36: end for
37: return $T$
```

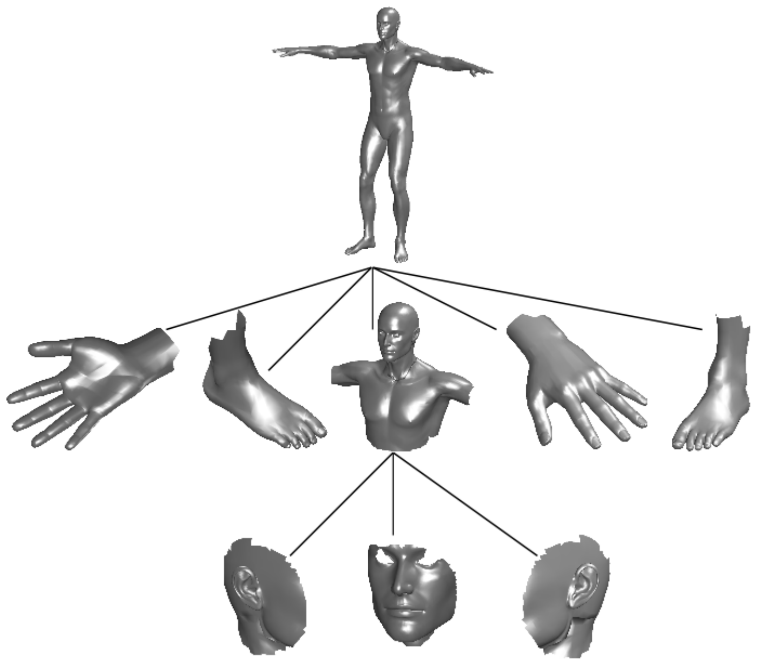


Figure 7.1: A decomposition tree of a human shape. The root node contains the original mesh and its related data. The first decomposition generates five regions. The last level contains smaller regions from the head.

algorithm (line 9) and the creation of new nodes (lines 18-28). We dedicate the following sections to describe these two points.

Adaptive Clustering with Medoids

Our algorithm is a variant of the adaptive clustering used to compute key-components. The main differences lie in two aspects. First, we use a medoid-based approach which prevents the computation of the multi-dimensional scaling. Second, we take an adaptive approach for determining the clustering thresholds depending on the hierarchical nature of our process.

Our method takes advantage of the distance matrix already computed in the pre-processing step and therefore, it is not necessary performing a multi-dimensional scaling to the keypoints. The clustering algorithm iterates over the set of keypoints assigning them to near clusters or creating new clusters otherwise. The decision of assigning a keypoint to a cluster or creating a new one depends on two parameters: R (intra-cluster threshold) and S (inter-cluster threshold). These parameters depends on the area of the mesh in the input node, and are obtained using an empiric Gaussian function as follows:

$$R = 0.4 \times \mathrm{e}^{-\frac{(areaRatio-1)^2}{0.5}}, \tag{7.2}$$

where $areaRat$io is the quotient between $areaN$ode and $areaOriginal$. This formulation was designed to control the clustering thresholds according to the size of the region. Eq. 7.2

distributes the values for R in the interval $[0.1, 0.4]$[1] depending on the area of the node region. This is consistent with the restriction in line 3. Additionally, the inter-cluster threshold S is always set to $2 \times R$. Therefore, the threshold values vary between $R = 0.4, S = 0.8$ and $R = 0.1, S = 0.2$.

Once the keypoints have been clustered, we need to compute the medoids of each resulting cluster. Let $C = \{c_1, \dots, c_n\}$ be a cluster where each c_i is a keypoint, the medoid of C is defined as the keypoint that is approximately in the center of the distribution of the cluster. More formally,

$$medoid(C) = \arg\min_{c \in C} \sum_{k=1}^{n} \mathrm{d}_g(c, c_k). \tag{7.3}$$

In addition, if a cluster has a few elements (less than ten in our experiments), the cluster is removed. The algorithm repeats the assignment and update steps until reaching a number of iterations (for all our experiments, we use ten iterations). After the clustering, each cluster will generate a new node in the decomposition tree.

Node Creation

Several steps are performed to create a new node in the tree from a cluster c. First, we compute the key-component for the cluster c. Second, we compute a descriptor for the new region. Let $C = \{c_1, \dots, c_n\}$ be the set of keypoints of a cluster, each associated to a HKS descriptor. The descriptor of the region determined by the cluster C is

$$f_{region}(C) = \frac{\sum_{i=1}^{n} HKS(c_i)}{n}, \tag{7.4}$$

i.e. the average descriptor of the keypoint collection in the cluster. Third, all information about keypoints and geodesic distances are propagated from the parent node T to the new created node. Finally, the new node is stored as a child of the parent node T.

Subsequently, our decomposition algorithm computes the geodesic distances between regions (lines 29-33). We use the medoid keypoint as a reference to accomplish this goal. Let T_{c_1} and T_{c_2} two nodes resulting from the decomposition of the mesh at node T. The distance between the regions associated to T_{c_1} and T_{c_2} is

$$\mathrm{d}_{reg}(T_{c_1}, T_{c_2}) = \mathrm{d}_g(k_{c_1}, k_{c_2}), \tag{7.5}$$

where k_{c_1} and k_{c_2} stand for the medoids of clusters in nodes T_{c_1} and T_{c_2}, respectively. As a last step, our algorithm proceeds recursively for each child node (lines 34-36).

[1] The threshold values represent a fraction of the original area, so $R = 0.1$ really means $R = 0.1 \times$ *areaOriginal*. We omitted this to facilitate the explanation.

7.2 Hierarchical Matching

This section describes the algorithm to find the correspondences between two shapes. The algorithm 7.2 presents the pseudo-code of our method. This algorithm requires two trees T and P that represent the decomposition of two shapes as described in Sec. 7.1.2. The overall algorithm is based on the ability of matching regions in the internal nodes and keypoints in the leaf nodes.

Algorithm 7.2 Matching(T,P)

```
Require: Node $T$
Require: Node $P$
Ensure: A set of correspondences $S$
 1: if not $T$.leaf and not $P$.leaf then
 2:     $Corr \leftarrow$ MatchingInternalNodes($T$, $P$)
 3:     $S \leftarrow \{\}$
 4:     for each match $(t,p) \in Corr$ do
 5:         $L \leftarrow$ Matching($T$.children[$t$], $P$.children[$p$])
 6:         $S \leftarrow S \bigcup L$
 7:     end for
 8: else
 9:     $S \leftarrow$ MatchingLeafNodes($T$, $P$)
10: end if
11: return $S$
```

The method is performed by depth levels (see Fig. 7.2). First, the matching of root nodes (level 0) implies to find correspondences between their children (level 1). Each correspondence generates a recursive call to the matching algorithm. The method proceeds until reaching the leaf nodes. In that case, we look for the best correspondence set between the keypoints in the leaf nodes. If at any time during the process, it is required to match an internal node of a tree with a leaf node of the other, the internal node is treated as a leaf node. This is possible because every internal node contains the information to behave as a leaf node. Next, we describe how to perform the matching in the aforementioned cases.

7.2.1 Matching of Internal Nodes

Let T and S two internal nodes from different trees. Each node has children represented as $children(T) = \{t_1, \ldots, t_n\}$ and $children(S) = \{s_1, \ldots, s_m\}$, where T has n children and S has m children. We define a boolean indicator variable as follows

$$x(\mathrm{i}, j) = \begin{cases} 1, & \text{if } t_\mathrm{i} \text{ matches } s_j \\ 0 & \text{otherwise.} \end{cases} \tag{7.6}$$

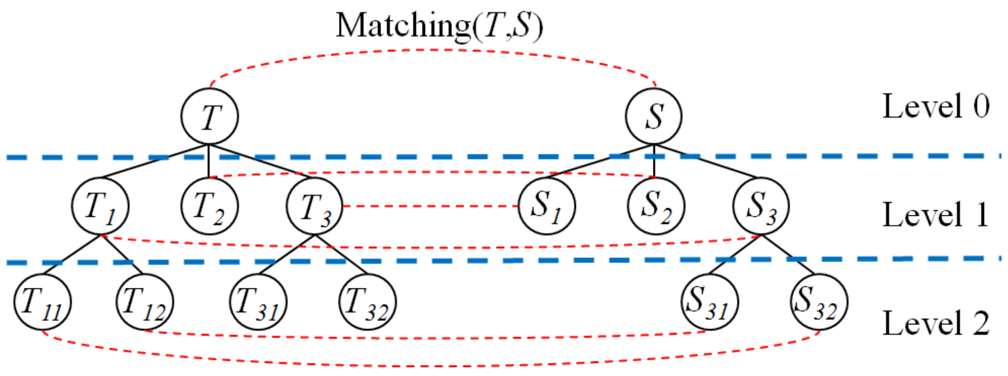


Figure 7.2: Representation of the matching process. The initial call to $Matching(T, S)$ (level 0) tries to find correspondences between nodes the internal in level 1. In the figure, the correspondences $\{(T_1, S_3), (T_2, S_2), (T_3, S_1)\}$ were found. Each pair generates a recursive call to the matching algorithm. The correspondence (T_1, S_3) causes a matching between internal nodes. In contrast, correspondences (T_2, S_2) and (T_3, S_1) drive to a matching of internal nodes. Note that T_3 is not a leaf node. However it contains the enough information to be considered as a leaf node, and therefore it can be matched to S_1.

Then, we formulate a quadratic optimization function as follows:

$$\begin{aligned} F(x) =& \alpha \sum_{\mathrm{i},j,\mathrm{i}',j'} |\mathrm{d}_{reg}(t_\mathrm{i}, t_{\mathrm{i}'}) - \mathrm{d}_{reg}(s_j, s_{j'})| x(\mathrm{i}, j) x(\mathrm{i}', j') + \\ & \beta \sum_{\mathrm{i},j} \| f_{region}(t_\mathrm{i}) - f_{region}(s_j) \|_2 x(\mathrm{i}, j) + \\ & \gamma \sum_{\mathrm{i},j} |area(t_\mathrm{i}) - area(s_j)| x(\mathrm{i}, j) \end{aligned} \tag{7.7}$$

where α, β and γ weight the contribution of each term in the function. The optimization function has two linear terms and a quadratic term. On the one hand, the linear terms evaluate the similarity between region descriptors and the consistency of areas. It is expected that two matched regions have similar descriptors and similar areas, indeed minimizing the linear terms. On the other hand, the quadratic term imposes a geometric consistency constraint. If there are two correspondences $x(\mathrm{i}, j)$ and $x(\mathrm{i}', j')$, it is expected that the geodesic distance between regions i and i$'$ in one shape is quite similar to the geodesic distance between regions j and j' in the other shape. Finally, the goal is to obtain the minimizer of F

$$x* = \arg\min_x F(x), \tag{7.8}$$

subject to

$$\sum_\mathrm{i} x(\mathrm{i}, j) = 1 \ \forall j \text{ and } \sum_j x(\mathrm{i}, j) = 1 \ \forall \mathrm{i}. \tag{7.9}$$

The constraint in Eq. 7.9 controls the multiplicity of an element in the correspondence set. That is to say, every t_i only can correspond to a unique s_j, and vice versa.

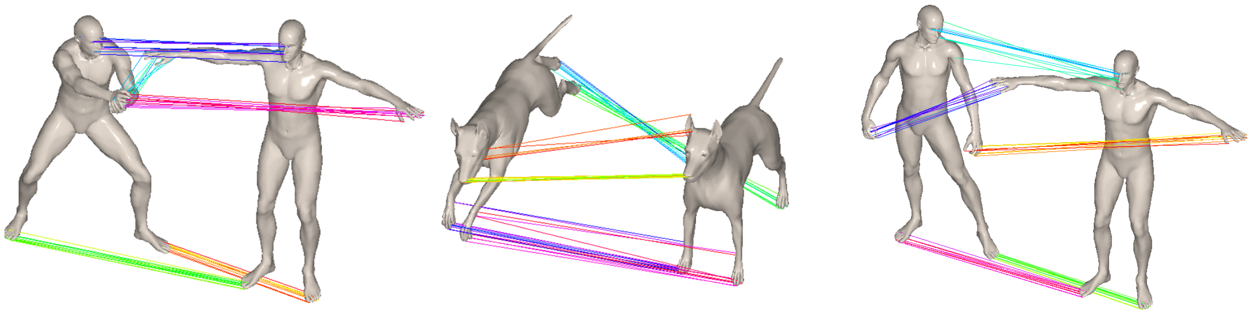


Figure 7.3: Correspondences found with our approach. Left: # correspondences = 66, geodesic error = 2.83, matching time = 0.12 sec. Middle: # correspondences = 58, geodesic error = 2.62, matching time = 0.08 sec. Right: # correspondences = 75, geodesic error = 3.51, matching time = 0.12 sec.

7.2.2 Matching of Leaf Nodes

Unlike the matching of internal nodes, in the leaf nodes we need to find correspondences between keypoints. Let T and S two leaf nodes from different trees. Each node has a set of keypoints represented as $keypoints(T) = \{t_1, \dots, t_n\}$ and $keypoints(S) = \{s_1, \dots, s_m\}$, where T has n keypoints and S has m keypoints. Using Eq. 7.6 to define a boolean indicator variable, we formulate the optimization function for a leaf node as

$$\begin{aligned} L(x) = & \alpha \sum_{\mathrm{i},j,\mathrm{i}',j'} |\mathrm{d}_g(t_\mathrm{i}, t_{\mathrm{i}'}) - \mathrm{d}_g(s_j, s_{j'})| x(\mathrm{i}, j) x(\mathrm{i}', j') + \\ & \beta \sum_{\mathrm{i},j} \|WKS(t_\mathrm{i}) - WKS(s_j)\|_2 x(\mathrm{i}, j). \end{aligned} \tag{7.10}$$

Note that the distances between regions in Eq. 7.7 have been replaced by geodesic distances between keypoints in Eq. 7.10. In addition, wave kernel signatures are used as descriptors for matching keypoints. Similarly to the matching of internal nodes, our goal is to find a minimizer of L in the same way as Eq. 7.8 and the same constraints as Eq. 7.9.

7.3 Experimental Evaluation and Discussion

We used two datasets to evaluate our method: the SHREC'2010 correspondence dataset [14] and the Shape Matching Benchmark proposed by Kim et al. [70]. In this section, we first detail the experimental setup for the experiments and then we show and discuss the results obtained with each dataset.

7.3.1 Experimental Setting

We detail our configuration as follows. First, we simplified the models to 10,000 vertices. The final correspondences were mapped back to the original shapes for evaluation. Second, we computed 100 Harris keypoints for each shape. Remember that this number can change after the filtering. Third, for Eq. 7.7 we use the weights: $\alpha = 5 \times 10^{-4}$, $\beta = 1$, and $\gamma = 5 \times 10^{-2}$. Fourth, for Eq. 7.10 we use the weights: $\alpha = 5 \times 10^{-4}$, and $\beta = 1$. Finally, we used a branch-and-bound algorithm with LP-relaxation [8] to solve the integer quadratic programs.

7.3.2 SHREC'2010 Dataset

Evaluation Criterion

For the quantitative results, we follow the methodology and notation proposed in [14]. We define our correspondence set as $C = \{(y_k, x_k)\}_{k=1}^{M}$, where M is the number of correspondences, and y_k and x_k are keypoints in the transformed and null shape, respectively. The ground-truth is composed by two correspondence set C_0 and $\hat{C}_0$, containing the point-to-point correspondences for each vertex in a transformed shape and their symmetric counterpart, respectively.

The measure to quantify the quality of the correspondence set C is

$$D(C) = \frac{1}{M} \min \left\{ \sum_{k=1}^{M} \mathrm{d}_g(x_k, x'_k), \sum_{k=1}^{M} \mathrm{d}_g(x_k, x''_k) \right\} \tag{7.11}$$

where $(y_k, x_k) \in C$, $(y_k, x'_k) \in C_0$ and $(y_k, x''_k) \in \hat{C}_0$.

Note that we also use the symmetric ground-truth to measure the localization error. Thus the final measure is the minimum between the exact correspondences and the symmetric counterpart.

Qualitative Results

We present results of our method in Fig. 7.3. The figures show examples with near-to-perfect localization of correspondences. An advantage of our method is the generation of very similar decomposition trees for near-isometric shapes. This enables the matching to be effective, yet fully exploiting the proposed hierarchical approach in favor of efficiency.

On the other hand, the addition of perturbations in meshes may affect the overall performance of finding correspondences. Figure 7.4 illustrates the correspondences found with our algorithm in presence of strong transformations. Even in the presence of shotnoise (on the left) and Gaussian noise (on the right), our algorithm performs acceptably. Note that there are parts where correspondences were not found (for instance, the hands in the right

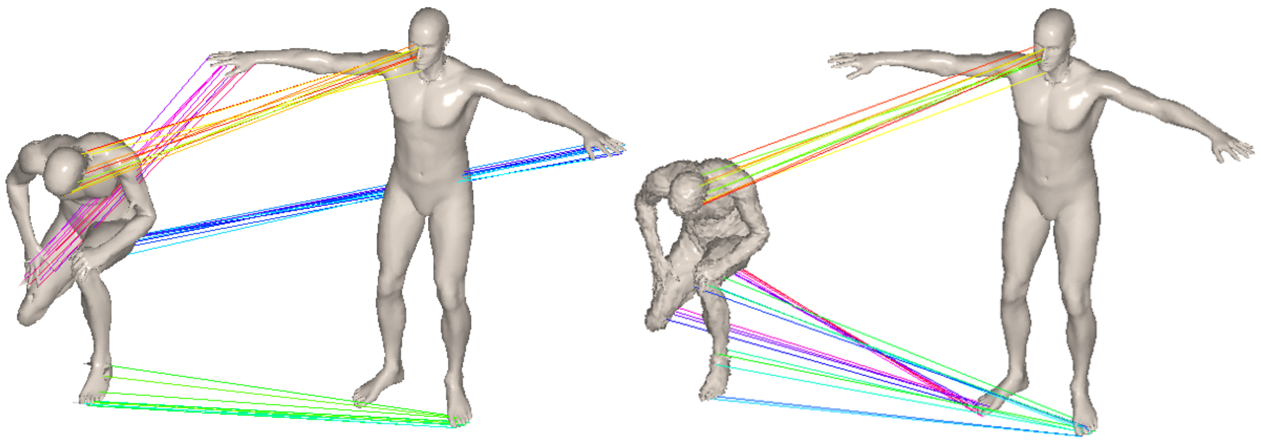


Figure 7.4: Correspondences found in presence of perturbations. Left: shotnoise, strength level 4 (# correspondences = 59, geodesic error = 5.23, matching time = 0.08 sec.). Right: noise, strength level 5 (# correspondences = 49, geodesic error = 5.42, matching time = 0.03 sec.).

	Strength				
Method.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
GMDS [20]	39.92	36.77	35.24	37.40	39.10
Game-theoretic [114]	10.28	12.51	11.73	14.35	18.26
Elastic Net [115]	**7.36**	8.62	10.49	21.72	**6.51**
Our method	7.99	**8.07**	**8.23**	**8.77**	9.38

Table 7.1: Average geodesic error of correspondences with respect to the strength level.

figure). This is because perturbations affect the decomposition process, and particularly in this case, noise prevented the detection of robust regions of interest in hands. Nevertheless, other regions were well detected and interestingly those regions were well matched to the null shape. Therefore, the decomposition method delivers robust regions (and their associated keypoints) which arrive to reliable correspondences.

Quantitative Results

In this section, we present the performance of our method using the geodesic error described in Eq. 7.11 and compare it with state-of-the-art methods. We compare our method with the elastic net approach [115], the game-theoretic approach (the variant which merges correspondences gathered from 25 games, as reported in [114]) and the GMDS method as reported in the SHREC 2010 contest [14]. These three methods deliver in average 50 correspondences. Similarly, our method computes 45 correspondences in average.

Table 7.1 presents the average geodesic error of our method and the compared approaches with respect to the strength level of transformation. It is worth noting that our method significantly obtains low localization errors for correspondences, specially in the stronger levels.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	9.37	7.28	6.47	**6.11**	6.34
Topology	9.02	8.23	8.97	9.88	10.17
Holes	**7.13**	**6.46**	**6.45**	**7.55**	**8.50**
Micro holes	5.86	5.60	5.92	**6.05**	6.18
Scale	6.45	6.57	7.30	7.60	7.79
Local scale	9.45	**10.20**	9.64	**9.84**	9.74
Sampling	10.18	11.62	12.96	15.78	19.20
Noise	**5.65**	8.15	**7.79**	**8.02**	8.58
Shot noise	8.80	**8.50**	8.56	8.04	7.88
Average	7.99	**8.07**	**8.23**	**8.77**	9.38

Table 7.2: Average geodesic error per transformation and per level. Average number of correspondences: 45.

Table 7.2 reports the average error per transformation and per strength level. Numbers in bold represent an improvement with respect to the state of the art. Our algorithm presents low error values in almost all transformations in strength level 4. In our opinion, the effectiveness of the correspondences is associated to the decomposition tree. That is, we believe that the decompositions are highly repeatable and consistent which agree with the results obtained for the key-components in Sec. 4.2. For this reason, the matching in high levels of the tree is very robust and therefore the region correspondences are well found. This leads to the propagation of good hypotheses for the correspondences, and hence the localization error is decreased.

Asa a result, the decomposition tree represents a mesh in an effective way, even in presence of severe perturbations. Hence this fact encourages us to think that our method is suitable for realistic applications. However, note that our method did not improve with respect to the topology transformation. The reason relies on the use of geodesic distances for the geometric consistency, which are sensitive to topological changes. A solution would be the use of a more robust manner for measuring intrinsic distances (for instance diffusion distances). Also keep in mind that we have used the original versions of the heat and wave kernel signatures. Applying scale-invariant versions of these descriptors could further improve our results.

7.3.3 Shape Matching Benchmark

In this section, we evaluate our algorithm using the benchmark described in Sec. 3.2.4.

Evaluation Criterion

Given two meshes M_1 and M_2, a dense correspondence set $C = \{(x, y) | x \in M_1 \text{ and } y \in M_2\}$, and the ground-truth $\hat{C} = \{(\hat{x}, \hat{y}) | \hat{x} \in M_1 \text{ and } \hat{y} \in M_2\}$, the error measure is defined as

$$Err(C, \hat{C}) = \sum_{\mathrm{i=1}}^{|C|} \mathrm{d}_g(y, \hat{y}) \tag{7.12}$$

where d_g is the geodesic distance in M_2 normalized by $\sqrt{Area(M_2)}$. This error measures how far a predicted correspondence is with respect to the real correspondence.

A convenient way to evaluate a correspondence set is by plotting the distribution of localization error in different distance thresholds. In addition, the benchmark provides information about symmetries, so it is possible to evaluate the possible symmetric flips in the predicted correspondences. Therefore, we present we show two plots depending on the use or not of the symmetry correspondences.

It is also worth noting that the benchmark aims to evaluate dense correspondences. In our case, our algorithm provides sparse correspondences. When this case, the evaluation scripts (provided with the benchmark) produces a full correspondence set by interpolating the sparse set using a method based on GMDS.

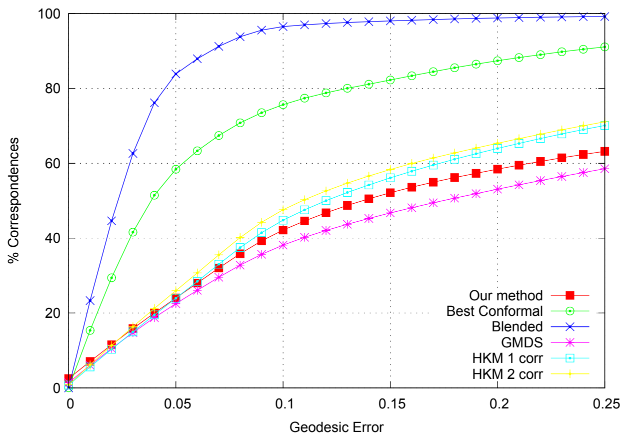
Results

In this section, we show the obtained results and compare them with state-of-the-art methods. We compare our approach with the following methods:

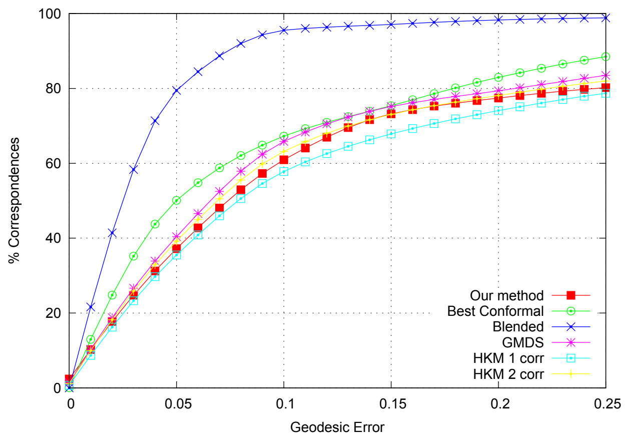
- Blended Intrinsic Maps [70].
- Best Conformal Map [70].
- Heat Kernel Matching with 1 correspondence [103].
- Heat Kernel Matching with 2 correspondences [103].
- Generalized Multi-Dimensional Scaling (GMDS) [20].

First, we show the distribution of localization errors in Figure 7.5. In the case of the unique ground-truth (Figure 7.5 (A)), our method is slightly better than GMDS and worst than the other methods. This result is expected since our method does not take into account any information about orientation. Therefore, the low performance is due to mistakes with symmetric correspondences. On the other hand, when the symmetric flips are used (Figure 7.5 (B)), our method gets a higher percentage of correspondences (almost a 20% consistently for each geodesic error). Most importantly, our method is comparable in performance to GMDS, and the Heat Kernel Matching methods. All these observations suggest us that our method finds a good set of correspondences which could be useful to find a further dense correspondence map.

In addition, we show and compare some interesting statistics in Table 7.3. The first column shows the averaged maximal per map geodesic error for each method. Our method obtains a lower maximal than the Heat Kernel Matching algorithms. Note that this statistic is computed on the unique ground-truth, so it allows us to compare the methods considering the exact matches. The second column shows the percentage of perfect matches on the entire



(a)



(b)

Figure 7.5: Performance of our method and state-of-the-art methods. This plot shows the percentage of correspondences within the prescribed distance to the ground-truth correspondence. (A) Using the unique ground-truth. (B) Using symmetric flips.

Method	**Averaged Maximal**	**% Perfect Matches**
Blended	0.31	0.002
Best Conformal	0.54	0.042
GMDS	0.97	1.392
HKM 1 corr	1.27	0.769
HKM 2 corrs	1.21	0.852
Our method	1.16	2.505

Table 7.3: Averaged maximal per map geodesic errors and percentage of perfect matches for each method in our comparison.

collection of correspondences. It is important to remark the improvement in the number of perfect matches in our method. By perfect match we mean that a given algorithm found the exact correspondence in the unique ground-truth. From the two previous statistics, we however note that our method finds a very good set of correspondences but also it delivers a set of bad correspondences with high localization error. We believe that the perfect matches are due to the matching between robust keypoints. As we showed in Chapter 4, the Harris keypoints are very robust and repeatable in presence of isometric transformations. Therefore, it is expected that our method match keypoints which represent exact matches in the shapes. On the other hand, we believe that the high localization error is due to a bad propagation of matches in our hierarchical approach. That is, the matching in lower levels of the hierarchy rely on the matching in higher levels. If our algorithm gets a bad matching of regions, this error is propagated to the next level. As consequence, some keypoints are matched to keypoints which are far away from the exact correspondences.

A note about execution time

Our matching algorithm takes in average 0.1 seconds to find the correspondences between two shapes. All our experiments were run on a 64-bits Linux system with Intel Core-i7 (3.40GHz) processors and 32GB of RAM. Our algorithms were implemented in C/C++ with interfaces MEX/MATLAB.

The low matching time can be explained by the fast matching in each level of the hierarchy. First, the matching of region is very fast since there are few regions to be matched. After that, the algorithm performs in the same manner for each corresponding region. Again, we only need to find correspondences between a small set of nodes. When a leaf node is reached, the number of keypoints to be matched is small with respect to the entire set of keypoints. In our opinion, the efficiency of our method is obtained thanks to the ability of solving small correspondences problems.

7.4 Concluding Remarks

We proposed a novel hierarchical approach to address the problem of finding reliable correspondences in non-rigid shapes. In our experiments, we showed that our method is robust to severe perturbations, making it suitable for realistic applications. Also, our approach outperformed the state of the art with respect to the localization error of correspondences. In addition, our matching algorithm is efficient thanks to the use of the hierarchical structure of decomposition, which allows to reduce the search space. In the future, we plan to use more robust descriptors and intrinsic distances to improve our results.

Chapter 8

Conclusions

In this thesis, we proposed algorithms to take advantage of local features for shape matching and retrieval. Our main conclusion lies in the effective use of local features to represent shapes in several contexts. To achieve this, the detection of robust local features was the most important aspect to achieve. To this respect, the Harris 3D algorithm has proven to be a robust and efficient algorithm to detect interest points on meshes. Most importantly, the high repeatability of the keypoints and their distinctive distribution on the shape's surface led us to develop robust representations to face the shape matching and retrieval.

To the light of the previous observation, we proposed two new forms of representation for 3D shapes based on robust local features. The first strategy was to group keypoints. We noted that by grouping keypoints in a geodesic sense, the regions denoted by the groups were distinctive in objects in the same class. For this reason, we proposed an efficient method to find repeatable salient regions on meshes, which we called key-components. Interestingly, the repeatability of key-components was high in presence of transformations. Likewise, the spatial grouping of keypoints allowed us to define a effective representation for generic shapes. In this case, we proposed a data-aware partitioning algorithm which was important to improve the performance of generic shape retrieval. Finally, the grouping strategy was useful to construct robust signatures for non-rigid shapes. In addition, the use of the Signature Quadratic Form Distance allowed us to apply these signatures in a large-scale non-rigid shape retrieval scenario. This combination resulted in a very efficient approach to retrieve non-rigid shapes. In our opinion, the grouping strategy is successful to enhance the representational power of local features for shapes. In addition, this approach reduces the amount of information we need to represent a shape. That is, the number of key-components, partitions, or signatures is considerably less than the number of keypoints. This fact led us to define efficient algorithms for generic and non-rigid shape retrieval.

The second strategy was to represent a shape as a hierarchy. This representation arose from observing the robustness of the key-components. We proposed to find key-components in a recursive way and capture the decomposition process in a tree structure. Internal nodes contain regions and leaf nodes contain keypoints. We also proposed an algorithm to perform the search of correspondences between two shapes under these hierarchical representations. In our experiments, we showed that our approach reduced the localization error and improved considerably the search time. We believe that our hierarchical representation and matching algorithm have a great potential to improve both the localization of correspondences and the searching time.

Based on our results, we believe that all our ideas presented in this thesis have done a valuable contribution to the state of the art in local features, shape matching and shape retrieval.

8.1 Future Work

Many research questions and new ideas have emerged from the results and the lessons learned in this thesis work. Next, we present the future research directions related to our work.

8.1.1 Representations

In this thesis, we assumed that 3D shapes contain detectable features. Nevertheless, in certain shapes, it could not be possible to find distinctive features. For example, in CAD domain, shapes are commonly represented by planes and regular surfaces which lack of features. Another example is an sphere, whose main feature is its curvature regularity. In this sense, we plan to investigate new representations that are able to determine the best characterization for a shape. For this purpose, we need high-level representations with the sufficient awareness to represent a mesh with its salient characteristics, whether keypoints, regular regions, or a combination of both. In addition, it would be interesting to provide high-level algorithms to evaluate the similarity of shapes with these representations.

On the other hand, the use of meshes to represent 3D models is extensive in the retrieval literature. However, we believe that with the massive use of cheap 3D scanning devices, new tools for processing and analyzing point clouds will be necessary. In this sense, we plan to investigate the use of local features in point clouds. As point clouds commonly have a high resolution, it will be necessary efficient algorithm for detecting local structures. Also, local descriptors for point cloud need to be evaluated, and subsequently new approaches for matching will also deserve attention.

8.1.2 Scalability and Efficiency

In the 3D shape retrieval community, it is common to have datasets with a few thousands of models for evaluation. However, this situation could radically change in the short term. Therefore, we need to start thinking the problem in a large-scale sense. Our results with the applications of the SQFD and metric indexing encourage us to believe that better improvements can be done in order to achieve better performances. In the future, we plan to research more approaches for indexing which, to our knowledge, is a promising way to go even more faster in content-based searches. New approaches such as the Ptolemaic tables [51] could be useful for shape retrieval.

Regarding the searching of correspondences, we believe that our matching approach can be improved if we use new approaches to solve the quadratic assignment problem. Specifically we plan to investigate graph-based methods to improve the efficiency of our method.

8.1.3 Robustness

In this work, state of the art descriptors such as DESIRE, PANORAMA, HKS and WKS have been useful to show the utility of our proposed representations in shape retrieval and matching. However, there is much to be done regarding the robustness. For instance, in the context of missing data, it would not be possible to guarantee a robust behavior of the aforementioned descriptors. The solution to the problem of missing data is crucial to facilitate effective tools for matching data coming from real acquisition devices. Moreover, we recently proposed a SHREC track [133] intended to evaluate algorithms for large-scale shape retrieval. The results were important to realize that the problem is very challenging and that so much work need to be done in that direction. In the future, we plan to investigate robust descriptors with the ability of dealing with missing data. We also plan to look for representations derived from local features which can tackle the lack of information. We want to provide algorithms for the efficient matching through these new representations.

Appendix A

Auxiliary Integrals for Harris 3D

This appendix contains the evaluation of integrals which are useful to obtain the final evaluations for our Harris 3D operator.

A.0.4 Evaluation of integral $\int_0^\infty \mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x$

Let us consider the following integral of a Gaussian in $\mathbb{R}^2$

$$\begin{aligned}\int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2}\mathrm{d}A &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-(x^2+y^2)/2\sigma^2}\mathrm{d}x\mathrm{d}y \\ &= \left(\int_{-\infty}^{\infty} \mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x\right)\left(\int_{-\infty}^{\infty} \mathrm{e}^{-y^2/2\sigma^2}\mathrm{d}y\right) \\ &= \left(\int_{-\infty}^{\infty} \mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x\right)^2.\end{aligned} \tag{A.1}$$

In addition, since the Gaussian is symmetric in $x = 0$, the following holds

$$\int_0^\infty \mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x = \frac{1}{2}\int_{-\infty}^{\infty} \mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x \tag{A.2}$$

Therefore, if we assume that the integral in A.1 is positive (as we will show later), then

$$\int_0^\infty \mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x = \frac{1}{2}\sqrt{\int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2}\mathrm{d}A} \tag{A.3}$$

To evaluate the integral in A.1, we transform the integral to polar coordinates where

$r^2 = x^2 + y^2$ and $\mathrm{d}A = r\mathrm{d}r\mathrm{d}\theta$. Thus, we have

$$\begin{aligned}\int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2}\mathrm{d}A &= \int_0^{2\pi}\int_0^{\infty} \mathrm{e}^{-r^2/2\sigma^2}\cdot r\cdot \mathrm{d}r\mathrm{d}\theta \\ &= 2\pi\int_0^{\infty} r\cdot \mathrm{e}^{-r^2/2\sigma^2}\mathrm{d}r\end{aligned} \tag{A.4}$$

Now, we make a change of variable $s = r^2$ and therefore $\mathrm{d}s = 2r\mathrm{d}r$. Hence, we obtain

$$\begin{aligned}\int_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2}\mathrm{d}A &= \pi\int_0^{\infty} \mathrm{e}^{-s/2\sigma^2}\mathrm{d}s \\ &= -2\sigma^2\pi\int_0^{\infty}\frac{-1}{2\sigma^2}\mathrm{e}^{-s/2\sigma^2}\mathrm{d}s \\ &= -2\sigma^2\pi\left(\mathrm{e}^{-s/2\sigma^2}\right)\Big|_0^{\infty} \\ &= -2\sigma^2\pi(-1) \\ &= 2\sigma^2\pi\end{aligned} \tag{A.5}$$

Hence, according to A.3, the final evaluation of our integral is

$$\int_0^{\infty} \mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x = \frac{\sigma\sqrt{2\pi}}{2} \tag{A.6}$$

We make use of integration by parts for evaluating this integral. The theorem of integration by parts states that

$$\int_a^b u\mathrm{d}v = uv|_a^b - \int_a^b v\mathrm{d}u \tag{A.7}$$

For the evaluation of the integral in this section, we will make the following replacements: $u = x$ $\mathrm{d}v = \mathrm{e}^{-x/2\sigma^2}\mathrm{d}x$, $\mathrm{d}v = x\mathrm{d}x$. In addition, $\mathrm{d}u = \mathrm{d}x$ and $v = -2\sigma^2\mathrm{e}^{-x/2\sigma^2}$. Applying the theorem of integration by parts, we have

$$\begin{aligned}\int_0^{\infty} \mathrm{e}^{-x/2\sigma^2}\cdot x\cdot \mathrm{d}x &= -2\sigma^2\left(\mathrm{e}^{-x/2\sigma^2}\cdot x\right)\Big|_0^{\infty} - \int_0^{\infty} -2\sigma^2\mathrm{e}^{-x/2\sigma^2}\mathrm{d}x \\ &= 0 + 2\sigma^2\int_0^{\infty}\mathrm{e}^{-x/2\sigma^2}\mathrm{d}x \\ &= 2\sigma^2\left(-2\sigma^2\mathrm{e}^{-x/2\sigma^2}\right)\Big|_0^{\infty}\end{aligned} \tag{A.8}$$

Finally, the integral evaluates to

$$\int_0^{\infty} \mathrm{e}^{-x/2\sigma^2}\cdot x\cdot \mathrm{d}x = 4\sigma^4 \tag{A.9}$$

A.0.5 Evaluation of integral $\int_0^\infty \mathrm{e}^{-x^2/2\sigma^2} \cdot x^2 \cdot \mathrm{d}x$

To evaluate the integral, we use the integration by parts. Here, we make the following replacements: $u = x$, $v = \mathrm{e}^{-x^2/2\sigma^2}$, $\mathrm{d}u = \mathrm{d}x$, and $\mathrm{d}v = \frac{-x}{\sigma^2}\mathrm{e}^{-x^2/\sigma^2}$. Thus, we proceed as follows

$$\begin{aligned}\int_0^\infty \mathrm{e}^{-s^2/2\sigma^2} \cdot x^2 \cdot \mathrm{d}x &= -\sigma^2 \int_0^\infty x \cdot \left(\frac{-x}{\sigma^2}\mathrm{e}^{-x^2/2\sigma^2}\right)\mathrm{d}x \\ &= -\sigma^2(\mathrm{e}^{-x^2/2\sigma^2} \cdot x)|_0^\infty + \sigma^2 \int_0^\infty \mathrm{e}^{-x^2\sigma^2}\mathrm{d}x \qquad \text{(A.10)} \\ &= 0 + \sigma^2 \int_0^\infty \mathrm{e}^{-x^2/\sigma^2}\mathrm{d}x\end{aligned}$$

Using the result obtained in A.0.4, the final result is

$$\int_0^\infty \mathrm{e}^{-s^2/2\sigma^2} \cdot x^2 \cdot \mathrm{d}x = \frac{\sigma^3}{2}\sqrt{2\pi} \qquad \text{(A.11)}$$

A.0.6 Evaluation of integral $\int_0^\infty \mathrm{e}^{-x^2/2\sigma^2} \cdot x^3 \cdot \mathrm{d}x$

We use a change of variables to easily evaluate this integral. We make $s = x^2$ and $\mathrm{d}s = 2x\mathrm{d}x$. Hence, the integral evaluates to

$$\int_0^\infty \mathrm{e}^{-x^2/2\sigma^2} \cdot x^3 \cdot \mathrm{d}x = \frac{1}{2}\int_0^\infty \mathrm{e}^{-s/2\sigma^2} \cdot s\mathrm{d}s \qquad \text{(A.12)}$$

Finally, using the result obtained in A.0.4, we obtain

$$\int_0^\infty \mathrm{e}^{-x^2/2\sigma^2} \cdot x^3 \cdot \mathrm{d}x = 2\sigma^4 \qquad \text{(A.13)}$$

Bibliography

[1] Alexander Agathos, Ioannis Pratikakis, Stavros Perantonis, and Nickolas S. Sapidis. Protrusion-oriented 3D mesh segmentation. *The Visual Computer*, 26(1):63–81, August 2009.

[2] Dragomir Anguelov, Praveen Srinivasan, Hoi-Cheung Pang, Daphne Koller, Sebastian Thrun, and James Davis. The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces. In *NIPS*, 2004.

[3] Indriyati Atmosukarto, Katarzyna Wilamowska, Carrie Heike, and Linda G. Shapiro. 3D object classification using salient point patterns with application to craniofacial research. *Pattern Recognit.*, 43(4):1502–1517, 2010.

[4] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *IEEE Int. Conf. in Computer Vision Workshops*, pages 1626–1633, 2011.

[5] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. Pose-Consistent 3D Shape Segmentation Based on a Quantum Mechanical Feature Descriptor. In *DAGM-Symposium*, pages 122–131, 2011.

[6] Christian Beecks, Merih Seran Uysal, and Thomas Seidl. Signature quadratic form distances for content-based similarity. In *Proc. of the ACM Int. Conf. on Multimedia*, MM '09, pages 697–700, New York, NY, USA, 2009. ACM.

[7] Christian Beecks, Merih Seran Uysal, and Thomas Seidl. Signature Quadratic Form Distance. In *Proc. of the ACM Int. Conf. on Image and Video Retr.*, CIVR '10, pages 438–445, New York, NY, USA, 2010. ACM.

[8] A. Bemporad, D. Mignone, and M. Morari. An Efficient Branch and Bound Algorithm for State Estimation and Control of Hybrid Systems. In *European Control Conference*, Karlsruhe, Germany, August 1999.

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

[10] S. Biasotti, X. Bai, B. Bustos, A. Cerri, D. Giorgi, L. Li, M. Mortara, I. Sipiran, S. Zhang, and M. Spagnuolo. SHREC'12 Track: Stability on Abstract Shapes. In *Proc.*

Eurographics Workshop on 3D Object Retrieval (3DOR), pages 101–107, 2012.

[11] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[12] Mario Botsch, Renato Pajarola, Baoquan Chen, and Matthias Zwicker, editors. *Symposium on Point Based Graphics, Prague, Czech Republic, 2007. Proceedings*. Eurographics Association, 2007.

[13] Edmond Boyer, Alexander Bronstein, Michael Bronstein, Benjamin Bustos, Tal Darom, Radu Horaud, Ingrid Hotz, Yosi Keller, Johannes Keustermans, Artiom Kovnatsky, Roee Litman, Jan Reininghaus, Ivan Sipiran, Dirk Smeets, Paul Suetens, Dirk Vandermeulen, Andrei Zaharescu, and Valentin Zobel. SHREC 2011: robust feature detection and description benchmark. In *Proc. Eurographics 2011 Workshop on 3D Object Retrieval (3DOR'11)*, pages 71–78. Eurographics Association, 2011.

[14] Alexander Bronstein, Michael Bronstein, Benjamin Bustos, Umberto Castellani, Marco Crisani, Bianca Falcidieno, Leonidas J. Guibas, Ioannis Kokkinos, Vittorio Murino, Ivan Sipiran, Maks Ovsjanikov, Giuseppe Patane, Michela Spagnuolo, and Jian Sun. SHREC 2010: Robust feature detection and description benchmark. In *Proc. Eurographics Workshop on 3D Object Retrieval*, pages 79–86. Eurographics Association, 2010.

[15] Alexander Bronstein, Michael Bronstein, Leonidas Guibas, and Maks Ovsjanikov. Shape Google: Geometric Words and Expressions for Invariant Shape Retrieval. *ACM Trans. Comput. Graph.*, 30(1), 2011.

[16] Alexander Bronstein, Michael Bronstein, and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[17] Alexander M. Bronstein. Spectral descriptors for deformable shapes. *CoRR*, abs/1110.5015, 2011.

[18] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Three-Dimensional Face Recognition. *Int. J. Comput. Vision*, 64(1):5–30, 2005.

[19] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Efficient Computation of Isometry-Invariant Distances Between Surfaces. *SIAM J. Sci. Comput.*, 28(5):1812–1836, September 2006.

[20] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. *Proc. of the Natl. Acad. of Sci.*, pages 1168–1172, 2006.

[21] M.M. Bronstein and I. Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1704–1711, 2010.

[22] Matthew Brown and David G. Lowe. Automatic Panoramic Image Stitching using

Invariant Features. *Int. J. Comput. Vis.*, 74(1):59–73, 2007.

[23] Benjamin Bustos, Daniel Keim, Dietmar Saupe, Tobias Schreck, and Dejan Vranić. An experimental effectiveness comparison of methods for 3D similarity search. *Int. Journal on Digital Libraries, Special issue on Multimedia Contents and Management in Digital Libraries*, 6(1):39–54, 2006.

[24] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan Vranic. Automatic selection and combination of descriptors for effective 3D similarity search. In *Proc. Int. Symposium on Multimedia Software Engineering*, 2004.

[25] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan V. Vranic. Feature-based similarity search in 3D object databases. *ACM Comput. Surv.*, 37(4):345–387, 2005.

[26] Benjamin Bustos, Tobias Schreck, Michael Walter, Juan M. Barrios, Matthias Schaefer, and Daniel A. Keim. Improving 3D similarity search by enhancing and combining 3D descriptors. *Multimedia Tools Appl.*, 58(1):81–108, 2011.

[27] Benjamin Bustos and Ivan Sipiran. 3D Shape Matching for Retrieval and Recognition. In Nick Pears, Yonghuai Liu, and Peter Bunting, editors, *3D Imaging, Analysis and Applications*, pages 265–308. Springer London, 2012.

[28] Umberto Castellani, Marco Cristani, S. Fantoni, and Vittorio Murino. Sparse points matching by combining 3D mesh saliency with statistical descriptors. *Comput. Graph. Forum*, 27(2):643–652, 2008.

[29] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, September 2001.

[30] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On Visual Similarity Based 3D Model Retrieval. *Comput. Graph. Forum*, 22(3):223–232, 2003.

[31] Hui Chen and Bir Bhanu. Human Ear Recognition in 3D. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(4):718–737, 2007.

[32] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729 vol.3, 1991.

[33] Clement Creusot, Nick Pears, and Jim Austin. A Machine-Learning Approach to Keypoint Detection and Landmarking on 3D Meshes. *International Journal of Computer Vision*, 102(1-3):146–179, 2013.

[34] Mohamed Daoudi, Lahoucine Ballihi, Chafik Samir, and Anuj Srivastava. Three-dimensional face recognition using elastic deformations of facial surfaces. In *Proc. Int. Conf. Multimedia and Expo*, pages 97–100. IEEE, 2008.

[35] J. Digne, J.-M. Morel, N. Audfray, and C. Mehdi-Souzani. The Level Set Tree on

Meshes. In *Proc. of the Fifth Int. Symposium on 3D Data Processing, Visualization and Transmission*, Paris, France, May 2010.

[36] A. Dubrovina and R. Kimmel. Matching shapes by eigendecomposition of the Laplace-Beltrami operator. In *3DPVT*, 2010.

[37] Helin Dutagaci, Chun Cheung, and Afzal Godil. Evaluation of 3D interest point detection techniques via human-generated ground truth. *The Visual Computer*, 28:901–917, 2012.

[38] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1285–1295, 2003.

[39] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[40] Yi Fang, Mengtian Sun, Minhyong Kim, and Karthik Ramani. Heat-Mapping: A Robust Approach Toward Perceptually Consistent Mesh Segmentation. *IEEE Computer Vision and Pattern Recognition*, 2011.

[41] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25(1):130–150, 2006.

[42] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. Int. Conf. and Exhib. on Comput. Graph. and Interact. Tech. SIGGRAPH '97*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[43] K. Gebal, J. A. Bærentzen, H. Aanæs, and R. Larsen. Shape Analysis Using the Auto Diffusion Function. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1405–1413, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[44] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Proc. Eurographics Symposium on Geometry Processing*, page 197, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[45] Daniela Giorgi, Silvia Biasotti, and L. Paraboschi. SHREC: Watertight Models Track, 2007. http://watertight.ge.imati.cnr.it/.

[46] Przemyslaw Glomb. Detection of Interest Points on 3D Data: Extending the Harris Operator. In *Computer Recognition Systems 3*, volume 57 of *Advances in Soft Computing*, pages 103–111. Springer Berlin / Heidelberg, May 2009.

[47] Afzal Godil, Helin Dutagaci, Ceyhun Burak Akgül, Apostolos Axenopoulos, Benjamin Bustos, Mohamed Chaouch, Petros Daras, Takahiko Furuya, Sebastian Kreft, Zhouhui Lian, Thibault Napoleon, Athanasios Mademlis, Ryutarou Ohbuchi, Paul L. Rosin, Bülent Sankur, Tobias Schreck, Xianfang Sun, Masaki Tezuka, Anne Verroust-Blondet,

M. Walter, and Yücel Yemez. SHREC'09 Track: Generic Shape Retrieval. In Michela Spagnuolo, Ioannis Pratikakis, Remco C. Veltkamp, and Theoharis Theoharis, editors, *Proc. Workshop on 3D Object Retr. (3DOR)*, pages 61–68. Eurographics Association, 2009.

[48] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. Context-Aware Saliency Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(10):1915–1926, 2012.

[49] Simon Goodall, Paul H. Lewis, Kirk Martinez, Patrick A. S. Sinclair, Fabrizio Giorgini, Matthew Addis, Mike J. Boniface, Christian Lahanier, and James Stevenson. SCULPTEUR: Multimedia Retrieval for Museums. In *Proc. ACM Int. Conf. on Image and Video Retrieval(CIVR)*, volume 3115 of *Lecture Notes in Computer Science*, pages 638–646. Springer, 2004.

[50] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Proc. of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[51] Magnus Lie Hetland, Tomáš Skopal, Jakub Lokoč, and Christian Beecks. Ptolemaic access methods: Challenging the reign of the metric space model. *Information Systems*, 38(7):989–1006, 2013.

[52] Masaki Hilaga, Yoshihisa Shinagawa, Taku Komura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc. Int. Conf. and Exhib. on Comput. Graph. and Interact. Tech.(SIGGRAPH)*, pages 203–212, 2001.

[53] H.T. Ho and D. Gibbins. Curvature-based approach for multi-scale feature extraction from 3D meshes and unstructured point clouds. *IET Comput. Vis.*, 3(4):201, 2009.

[54] Donald D Hoffman and Manish Singh. Salience of visual parts. *Cognition*, 63(1):29–78, 1997.

[55] Jiaxi Hu and Jing Hua. Salient spectral geometric features for shape matching and retrieval. *The Visual Computer*, 25(5-7):667–675, 2009.

[56] Jing Hua, Zhaoqiang Lai, Ming Dong, Xianfeng Gu, and Hong Qin. Geodesic distance-weighted shape vector image diffusion. *IEEE Trans. on Vis. and Comput. Graph*, 14(6):1643–50, 2008.

[57] Peng Huang, Adrian Hilton, and Jonathan Starck. Shape Similarity for 3D Video Sequences of People. *Int. J. of Comput. Vis.*, 89(2-3):362–381, February 2010.

[58] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. In *Proc. of the Symposium on Geometry Processing*, pages 1449–1457. Eurographics Association, 2008.

[59] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. on Graph.*, 25(3):569, July 2006.

[60] Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. Shape Decomposition using Modal Analysis. *Computer Graphics Forum*, 28(2):407–416, 2009.

[61] Varun Jain and Hao Zhang. A spectral approach to shape-based retrieval of articulated 3D models. *CAD*, 39:398–407, 2007.

[62] Andrew Johnson. *Spin-Images: A Representation for 3D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[63] Andrew E. Johnson and Martial Hebert. Control of Polygonal Mesh Resolution for 3-D Computer Vision. *Graphical Models and Image Processing*, 60(4):261–285, 1998.

[64] Ioannis A. Kakadiaris, Georgios Passalis, George Toderici, Takis Perakis, and Theoharis Theoharis. Face Recognition, 3D-Based. In Stan Z. Li and Anil K. Jain, editors, *Encycl. of Biometrics*, pages 329–338. Springer US, 2009.

[65] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *Visual Computer*, 21(8):649–658, 2005.

[66] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors, June 2003.

[67] Daniel A. Keim. Efficient Geometry-based Similarity Search of 3D Spatial Databases. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *Proc. ACM Int. Conf. on Management of Data(SIGMOD)*, pages 419–430. ACM Press, 1999.

[68] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas A. Funkhouser. Learning part-based templates from large collections of 3D shapes. *ACM Trans. Graph.*, 32(4):70, 2013.

[69] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring Collections of 3D Models Using Fuzzy Correspondences. *ACM Trans. Graph.*, 31(4):54:1–54:11, July 2012.

[70] Vladimir G. Kim, Yaron Lipman, and Thomas Funkhouser. Blended Intrinsic Maps. *ACM Trans. Graph.*, 30(4):79:1–79:12, July 2011.

[71] R. Kimmel and J. A. Sethian. Computing Geodesic Paths on Manifolds. In *Proc. Natl. Acad. Sci. USA*, pages 8431–8435, 1998.

[72] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel. Coupled quasi-harmonic bases. *Computer Graphics Forum*, 32(2pt4):439–448, 2013.

[73] Ivan Laptev. On Space-Time Interest Points. *Int. J. of Comput. Vis.*, 64(2-3):107–123, 2005.

[74] Ivan Laptev and Patrick Pérez. Retrieving actions in movies. In *Int. Conf. in Comput. Vis.*, pages 1–8, 2007.

[75] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. In *Proc. Int. Conf. and Exhib. on Comput. Graph. and Interact. Tech. SIGGRAPH '05*, pages 659–666, New York, NY, USA, 2005. ACM.

[76] Wee Kheng Leow and Rui Li. The analysis and applications of adaptive-binning color histograms. *Comput. Vis. Image Underst.*, 94:67–91, April 2004.

[77] Bo Li and Henry Johan. 3D model retrieval using global and local radial distances. In *Proc. Int. Workshop on Advanced Image Tech.*, 2010.

[78] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen. SHREC '11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes. In *Proc. Eurographics Workshop on 3D Object Retrieval (3DOR)*, pages 79–88, 2011.

[79] Zhouhui Lian, A. Godil, Xianfang Sun, and Hai Zhang. Non-rigid 3D shape retrieval using Multidimensional Scaling and Bag-of-Features. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 3181–3184, 2010.

[80] Zhouhui Lian, Afzal Godil, Benjamin Bustos, Mohamed Daoudi, Jeroen Hermans, Shun Kawamura, Yukinori Kurita, Guillaume Lavoué, Hien Van Nguyen, Ryutarou Ohbuchi, Yuki Ohkita, Yuya Ohishi, Fatih Porikli, Martin Reuter, Ivan Sipiran, Dirk Smeets, Paul Suetens, Hedi Tabia, and Dirk Vandermeulen. A comparison of methods for non-rigid 3D shape retrieval. *Pattern Recognition*, 46(1):449–461, 2013.

[81] Zhouhui Lian, Afzal Godil, and Jianguo Xiao. Feature-Preserved 3D Canonical Form. *International Journal of Computer Vision*, 102(1-3):221–238, 2013.

[82] Yaron Lipman and Thomas Funkhouser. Möbius voting for surface correspondence. *ACM Trans. Graph.*, 28(3):72:1–72:12, July 2009.

[83] Roee Litman, Alexander M. Bronstein, and Michael M. Bronstein. Diffusion-geometric maximally stable component detection in deformable shapes. *Computers & Graphics*, 35(3):549–560, 2011. Shape Modeling International (SMI) Conference 2011.

[84] Yi Liu, Hongbin Zha, and Hong Qin. Shape Topics: A Compact Representation and New Algorithms for 3D Partial Shape Retrieval. In *Proc. IEEE Conf. on Comput. Vis. and Pattern Recognit. CVPR '06*, pages 2025–2032, Washington, DC, USA, 2006. IEEE Computer Society.

[85] Marco Loog and François Lauze. The improbability of harris interest points. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 32(6):1141–7, June 2010.

[86] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. of Comput. Vis.*, 60(2):91–110, 2004.

[87] Mona Mahmoudi and Guillermo Sapiro. Three-dimensional point cloud recognition via distributions of geometric distances. *Graphical Models*, 71(1):22–31, 2009.

[88] Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. In *Proceedings of the eighth annual symposium on Computational geometry*, SCG '92, pages 1–8, New York, NY, USA, 1992. ACM.

[89] Facundo Mémoli. On the use of Gromov-Hausdorff Distances for Shape Comparison. In Botsch et al. [12], pages 81–90.

[90] Facundo Mémoli and Guillermo Sapiro. A Theoretical and Computational Framework for Isometry Invariant Recognition of Point Cloud Data. *Found. Comput. Math.*, 5(3):313–347, July 2005.

[91] Mark Meyer, Mathieu Desbrun, Peter Schroder, and Alan H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. *Vis. and Math. III*, pages 35–57, 2003.

[92] A. Mian, M. Bennamoun, and R. Owens. On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes. *Int. J. of Comput. Vis., Special Issue on 3D Object Retrieval*, 2009.

[93] María Luisa Micó, José Oncina, and Enrique Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.*, 15(1):9–17, January 1994.

[94] Krystian Mikolajczyk. Scale & Affine Invariant Interest Point Detectors. *Int. J. of Comput. Vis.*, 60(1):63–86, October 2004.

[95] Michela Mortara, Giuseppe Patanè, Michela Spagnuolo, Bianca Falcidieno, and Jarek Rossignac. Blowing Bubbles for Multi-Scale Analysis and Decomposition of Triangle Meshes. *Algorithmica*, 38:227–248, October 2003.

[96] John Novatnack and Ko Nishino. Scale-Dependent 3D Geometric Features. In *Proc. Int. Conf. on Comput. Vis.*, pages 1–8. Ieee, October 2007.

[97] Marcin Novotni and Reinhard Klein. A Geometric Approach to 3D Object Comparison. In *Proc. Shape Modeling Int.*, pages 167–175. IEEE Computer Society, 2001.

[98] Ryutarou Ohbuchi, Kunio Osada, Takahiko Furuya, and Tomohisa Banno. Salient local visual features for shape-based 3D model retrieval. In *Proc. Shape Modeling Int.*, pages 93–102. IEEE, 2008.

[99] Robert Osada, Thomas A. Funkhouser, Bernard Chazelle, and David P. Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, 2002.

[100] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional Maps: A Flexible Representation of Maps Between Shapes. *ACM Trans. Graph.*, 31(4):30:1–30:11, July 2012.

[101] Maks Ovsjanikov, Alexander M. Bronstein, Leonidas J. Guibas, and Michael M. Bron-

stein. Shape Google: a computer vision approach to invariant shape retrieval. In *Proc. Workshop on Non-Rigid Shape Anal. and Deform. Image Alignment(NORDIA)*, 2009.

[102] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of Continuous Variability in Collections of 3D Shapes. *ACM Trans. Graph.*, 30(4):33:1–33:10, July 2011.

[103] Maks Ovsjanikov, Quentin Mérigot, Facundo Mémoli, and Leonidas J. Guibas. One Point Isometric Matching with the Heat Kernel. *Comput. Graph. Forum*, 29(5):1555–1564, 2010.

[104] Panagiotis Papadakis, Ioannis Pratikakis, Stavros Perantonis, and Theoharis Theoharis. Efficient 3D shape matching and retrieval using a concrete radialized spherical projection representation. *Pattern Recognition*, 40(9):2437–2452, 2007.

[105] Panagiotis Papadakis, Ioannis Pratikakis, Theoharis Theoharis, Georgios Passalis, and Stavros J. Perantonis. 3D Object Retrieval using an Efficient and Compact Hybrid Shape Descriptor . In Stavros J. Perantonis, Nikolaos Sapidis, Michela Spagnuolo, and Daniel Thalmann, editors, *Proc. Workshop on 3D Object Retr. (3DOR)*, pages 9–16. Eurographics Association, 2008.

[106] Panagiotis Papadakis, Ioannis Pratikakis, Theoharis Theoharis, and Stavros Perantonis. PANORAMA: A 3D shape descriptor based on panoramic views for unsupervised 3D object retrieval. *Int. Journal of Computer Vision*, 89(2-3):177–192, 2009.

[107] E. Paquet and H.L. Viktor. Exploring Protein Architecture using 3D Shape-based Signatures. In *Proc. Int. Conf. Eng. in Med. and Biol.*, pages 1204 –1208, 2007.

[108] Eric Paquet and Herna L. Viktor. Capri/MR: exploring protein databases from a structural and physicochemical point of view. *Proc. VLDB*, 1(2):1504–1507, 2008.

[109] Georgios Passalis, Ioannis A. Kakadiaris, and Theoharis Theoharis. Intraclass Retrieval of Nonrigid 3D Objects: Application to Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:218–229, 2007.

[110] Oscar Pedreira and Nieves R. Brisaboa. Spatial Selection of Sparse Pivots for Similarity Search in Metric Spaces. In *Proceedings of the 33rd conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM '07, pages 434–445, Berlin, Heidelberg, 2007. Springer-Verlag.

[111] J. Pokrass, A. M. Bronstein, M. M. Bronstein, P. Sprechmann, and G. Sapiro. Sparse Modeling of Intrinsic Correspondences. *Computer Graphics Forum*, 32(2pt4):459–468, 2013.

[112] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-spectra as fingerprints for shape matching. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, SPM '05, pages 101–106, New York, NY, USA, 2005. ACM.

[113] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-Beltrami spectra as

"Shape-DNA" of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

[114] E. Rodolà, A.M. Bronstein, A. Albarelli, F. Bergamasco, and A. Torsello. A game-theoretic approach to deformable shape matching. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 182–189, 2012.

[115] E. Rodola, A. Torsello, T. Harada, Y. Kuniyoshi, and D. Cremers. Elastic Net Constraints for Shape Matching. In *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[116] Olaf Ronneberger, Hans Burkhardt, and Eckart Schultz. General-Purpose Object Recognition in 3D Volume Data Sets Using Gray-Scale Invariants - Classification of Airborne Pollen-Grains Recorded with a Confocal Laser Scanning Microscope. In *Int. Conf. Pattern Recognit.*, volume 2, 2002.

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

[118] Y. Sahillioğlu and Y. Yemez. 3D Shape correspondence by isometry-driven greedy optimization. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 453–458, 2010.

[119] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of Interest Point Detectors. *Int. J. of Comput. Vis.*, 37(2):151–172, 2000.

[120] Tobias Schreck, Maximilian Scherer, Michael Walter, Benjamin Bustos, Sang Ming Yoon, and Arjan Kuijper. Graph-based combinations of fragment descriptors for improved 3D object retrieval. In *Proc. ACM Multimedia Systems*, 2012.

[121] Ariel Shamir. A survey on Mesh Segmentation Techniques. *Comput. Graph. Forum*, 27(6):1539–1556, 2008.

[122] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual Part Analogies in 3D Objects. *International Journal of Computer Vision*, 89(2-3):309–326, 2010.

[123] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, 24(4):249–259, March 2008.

[124] Philip Shilane and Thomas Funkhouser. Selecting Distinctive 3D Shape Descriptors for Similarity Retrieval. In *Proc. IEEE Int. Conf. on Shape Model. and Appl. SMI '06*, page 18, Washington, DC, USA, 2006.

[125] Ivan Sipiran. Local features for partial shape matching and retrieval. In *Proc. ACM Multimedia*, pages 853–856, 2011.

[126] Ivan Sipiran and Benjamin Bustos. Scalable 3D Shape Retrieval using Local Features

and the Signature Quadratic Form Distance. Manuscript in preparation.

[127] Ivan Sipiran and Benjamin Bustos. A Robust 3D Interest Points Detector Based on Harris Operator. In *Proc. Eurographics Workshop on 3D Object Retrieval (3DOR)*, pages 7–14, 2010.

[128] Ivan Sipiran and Benjamin Bustos. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, 27:963–976, 2011.

[129] Ivan Sipiran and Benjamin Bustos. Key-component Detection on 3D Meshes using Local Features. In *Proc. Eurographics Workshop on 3D Object Retrieval (3DOR)*, pages 25–32, 2012.

[130] Ivan Sipiran and Benjamin Bustos. A Fully Hierarchical Approach for Finding Correspondences in Non-Rigid Shapes. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 817–824, 2013.

[131] Ivan Sipiran and Benjamin Bustos. Key-components: Detection of Salient Regions on 3D Meshes. *The Visual Computer*, 2013. Accepted with minor revision.

[132] Ivan Sipiran, Benjamin Bustos, and Tobias Schreck. Data-aware 3D partitioning for generic shape retrieval. *Computer & Graphics*, 2013. To appear.

[133] Ivan Sipiran, Rafael Meruane, Benjamin Bustos, Tobias Schreck, Henry Johan, Bo Li, and Yijuan Lu. SHREC'13 Track: Large-scale partial shape retrieval using simulated range images. In *Proc. 6th Eurographics Workshop on 3D Object Retrieval (3DOR'13)*, 2013. To appear.

[134] P. Skraba, M. Ovsjanikov, F. Chazal, and L. Guibas. Persistence-based Segmentation of Deformable Shapes. In *CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment*, June 2010.

[135] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23:399–405, August 2004.

[136] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Comput. Graph. Forum*, 28(5), 2009.

[137] Mengtian Sun, Yi Fang, and Karthik Ramani. Center-Shift: An approach towards automatic robust mesh segmentation (ARMS). In *CVPR*, pages 630–637, 2012.

[138] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton Based Shape Matching and Retrieval. In *Proc. Shape Modeling Int.*, page 130, Washington, DC, USA, 2003. IEEE Computer Society.

[139] Gary K. L. Tam and Rynson W. H. Lau. Deformable Model Retrieval Based on Topological and Geometric Signatures. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):470–482, May 2007.

[140] Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools Appl.*, 39(3):441–471, 2008.

[141] A. Tevs, A. Berner, M. Wand, I. Ihrke, and H.-P. Seidel. Intrinsic Shape Matching by Planned Landmark Sampling. *Computer Graphics Forum*, 30(2):543–552, 2011.

[142] A. Tevs, M. Bokeloh, M. Wand, A. Schilling, and H. P Seidel. Isometric registration of ambiguous and partial data. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1185–1192, 2009.

[143] Roberto Toldo, Umberto Castellani, and Andrea Fusiello. Visual Vocabulary Signature for 3D Object Retrieval and Partial Matching. In *3DOR*, pages 21–28, 2009.

[144] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance Evaluation of 3D Keypoint Detectors. *International Journal of Computer Vision*, 102(1-3):198–220, 2013.

[145] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A Survey on Shape Correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[146] Dejan Vranic. *3D Model Retrieval.* PhD thesis, University of Leipzig, 2004.

[147] Dejan V. Vranic. DESIRE: a composite 3D-shape descriptor. In *Proc. IEEE Int. Conf. on Multimedia and Expo*, 2005.

[148] Chaohui Wang, Michael M. Bronstein, Alexander M. Bronstein, and Nikos Paragios. Discrete Minimum Distortion Correspondence Problems for Non-rigid Shape Matching. In *Proceedings of the Third International Conference on Scale Space and Variational Methods in Computer Vision*, SSVM'11, pages 580–591, Berlin, Heidelberg, 2012. Springer-Verlag.

[149] Xu-lei Wang, Yi Liu, and Hongbin Zha. Intrinsic Spin Images: A subspace decomposition approach to understanding 3D deformable shapes. In *Proc. 3DPVT*, 2010.

[150] L.A. Wolsey. *Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, 1998.

[151] Jeng-Sheng Yeh, Ding-Yun Chen, Bing-Yu Chen, and Ming Ouhyoung. A web-based three-dimensional protein retrieval system by matching visual similarity. *Bioinformatics*, 21(13):3056–3057, 2005.

[152] Chun-Fong You and Yi-Lung Tsai. 3D solid model retrieval for engineering reuse based on local feature correspondence. *The Int. J. of Adv. Manuf. Technol.*, 46(5-8):649–661, May 2009.

[153] Tsz-Ho Yu, OliverJ. Woodford, and Roberto Cipolla. A Performance Evaluation of Volumetric 3D Interest Point Detectors. *International Journal of Computer Vision*, 102(1-3):180–197, 2013.

[154] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. Surface feature detection and description with applications to mesh matching. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 373–380, 2009.

[155] Yun Zeng, Chaohui Wang, Yang Wang, Xianfeng Gu, D. Samaras, and N. Paragios. Dense non-rigid surface registration using high-order graph matching. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 382–389, 2010.

[156] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. In *Proc. of the Symposium on Geometry Processing*, SGP '08, pages 1431–1439. Eurographics Association, 2008.

[157] Juyong Zhang, Jianmin Zheng, Chunlin Wu, and Jianfei Cai. Variational mesh decomposition. *ACM Trans. Graph.*, 31(3):21:1–21:14, June 2012.

[158] Youyi Zheng, Daniel Cohen-Or, and Niloy J. Mitra. Smart Variations: Functional Substructures for Part Compatibility. *Computer Graphics Forum (Eurographics)*, 32(2pt2):195–204, 2013.

[159] Guangyu Zou, Jing Hua, Ming Dong, and Hong Qin. Surface matching with salient keypoints in geodesic scale space. *Comput. Animat. and Virtual Worlds*, 19(3-4):399–410, 2008.

[160] Guangyu Zou, Jing Hua, Zhaoqiang Lai, Xianfeng Gu, and Ming Dong. Intrinsic geometric scale space by shape diffusion. *IEEE Trans. on Vis. and Comput. Graph.*, 15(6):1193–200, 2009.