Shape Distinction for 3D Object Retrieval

Philip Nathan Shilane

A Dissertation

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

By the Department of

Computer Science

April, 2008

Abstract

In recent years, there has been enormous growth in the number of 3D models and their availability to a wide segment of the population. Examples include the National Design Repository which stores 3D computer-aided design (CAD) models for tens of thousands of mechanical parts, the Protein Data Bank (PDB) that has atomic positions for tens of thousands of protein molecules, and the Princeton Shape Benchmark with thousands of everyday objects represented as polygonal surface models. With the availability of free interactive tools for creating 3D models and graphics cards for home computers, we can expect 3D data to become ever more widely available.

Given the availability of 3D data, searching for a 3D object in a large database is a core problem for numerous applications including object recognition and the reuse of expertly created data. This raises two key research problems: 1) How can we improve search techniques? and 2) How do we evaluate 3D search techniques?

The first contribution of this dissertation is an analysis technique to select the most important or distinctive regions of an object. Our approach identifies regions of a surface that have shape consistent with objects of the same type and different from objects of other types. By focusing a retrieval method on the most important regions of an object, we can improve retrieval performance in comparison to alternative feature point selection techniques. We investigate properties of shape distinction including techniques for calculating distinction, a method for visualizing differences in a database, and a prediction algorithm based on likelihoods of local shapes. We also demonstrate that shape distinction can be used in graphics applications such as mesh simplification and icon generation.

The second contribution is a new methodology to analyze shape retrieval methods with a common data set of classified 3D models and software tools called the Princeton Shape Benchmark (PSB). Based on experiments with several different retrieval methods, we find that no single method is best for all classifications of objects, and thus the main contribution of the PSB is a framework to evaluate retrieval methods.

Acknowledgments

Completing a dissertation is a multi-stage process involving taking classes, studying previous research, and exploring new areas of science. Numerous people provided guidance and encouragement along the way, from my earliest days through my time at Princeton University.

I would like to thank my advisor, Dr. Thomas Funkhouser, for his mentorship. Besides prodding my research onto productive paths, he contributed in numerous ways (small and large) to the completion of this project. He set an ongoing example of the persistent scientist. The other graphics professors, Drs. Adam Finkelstein and Szymon Rusinkiewicz, provided feedback on my research throughout graduate school and reviewed this dissertation. I would also like to thank Drs. David Dobkin, Andrea LaPaugh, and Kai Li for joining my dissertation committee and advising me at key moments during graduate school.

Numerous collaborators at Princeton University played an important role either assisting with this project or helping form my opinions about quality research. I would like to thank: Dr. Daniel Aliaga, Benedict Brown, Michael Burns, Paul Calamia, Forrestor Cole, Phillip Davidson, Dr. Doug DeCarlo, Christopher DeCoro, Nathaniel Dirkson, Aleksey Golovinskiy, Dr. Matthew Hibbs, William Kiefer, Dr. Jason Lawrence, Melissa Lawson, Dr. Diego Nehab, Joshua Podolak, Dr. Ayellat Tal, Corey Toler, and Dr. Tim Weyrich. My research is built upon the foundation laid by Drs. Patrick Min and Michael Kazhdan, previous members of the shape-matching research group.

Research is not completed with sweat and drive alone; I would like to thank several groups that provided partial funding for this work: the Princeton University Departmental Award; National Science Foundation Grants IIS-0612231, CCR-0093343, CNS-0406415, and 11S-0121446; Air Force Research Laboratory Grant FA8650-04-1-1718; and the Google Research Grant.

Outstanding educators and researchers encouraged my interests long before graduate school. From Stanford University, I would like to thank Dr. Mark Musen for giving me my first research opportunity in computer science and Julie Zelenski for explaining the finer details of the teaching process. From my hometown of Carl Junction, Missouri, educators Pamela Babbitt, Paul Foster, Linda Scruggs, and Dr. Paul Teverow motivated my research interests, first in history and then in the sciences.

I would like to thank several other friends for their support. They include roommates, officemates, fellow travelers, classmates, and cooking collaborators: Bryson Bennett, Dr. Christopher Callison-Burch, Melissa Carroll, Catherine Crump, Dr. Daniel Dantas, James Eric DeGruson, Dr. Eujin Goh, Adam Goldman, David Huang, Dr. Brian Milch, Ryan Pierce, Haakon Ringberg, and Eric Silverberg.

The support of my family has been invaluable. I am lucky that my parents, Dr. Lewis and Roberta Shilane, have advanced training in mathematics, library science, education, medicine, and Spanish, and they shared their love of learning and their work ethic with me. Watching my two siblings, David and Amy, achieve their own goals has been a joy as well. My grandparents, aunts, uncles, cousins, and in-laws have patiently supported me. Finally, I would like to thank my wife, Dr. Julie Shilane, for being a part of my life.

Dedicated to my parents

Dr. Lewis and Roberta Shilane

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, there has been enormous growth in the number of 3-dimensional (3D) models and their availability to a wide segment of the population. Examples include the National Design Repository, which stores 3D computer-aided design (CAD) models for tens of thousands of mechanical parts; the Protein Data Bank (PDB) with atomic positions for tens of thousands of protein molecules; medical collections such as the Visible Human; and the Princeton Shape Benchmark (PSB) with 36,000 everyday objects represented as polygonal surface models. The number of constructed or measured 3D models is experiencing enormous growth, and with the availability of free interactive tools for creating 3D models, we can expect the number of 3D models to continue to grow. As models have become more widespread, graphics cards for home computers have become faster and inexpensive, making 3D data available to nearly everyone. A key problem for everyday users who interact with 3D objects is how to search for 3D data within large collections.

Those who are not computer experts interact with 3D objects in numerous ways, and 3D shape similarity queries are useful in several applications. In the mechanical CAD community, computer models of individual pieces such as bolts and brackets are often combined into complex mechanical parts, and shape similarity search can aid their work. Manufacturing companies could create databases of these parts [57], so when designing a new device, the question is how to reuse or modify an existing part instead of designing one from scratch. Similarly, in the computer graphics community, databases of common, household objects have been created [45]. When designing virtual worlds, novice users

can quickly create complex scenes using a database of objects instead of creating each object individually [35]. For both the CAD and computer graphics communities, a user may start with a rough model that approximates his or her goal and then improve the model with detailed subparts that already exist.

In the fields of molecular biology and chemistry, the 3D structure of proteins can be determined using X-ray crystallography [44] or nuclear magnetic resonance spectroscopy, and approximately 45,000 structures are currently stored in the PDB [10]. Studying the properties of a protein experimentally in a laboratory is time intensive, so any information about similar proteins that can guide the experiments is of benefit. While a protein sequence can be used as a search term into the PDB, the 3D structure of a protein can also suggest distantly related proteins [38], which can guide laboratory studies.

For a computer vision system, 3D data scans of the environment can be the input to a recognition system that matches the scene data to objects in a labeled database. Systems that scan objects in the environment typically involve some combination of mounted cameras or camera and laser system, which triangulate positions in the environment to create a range image indicating the distance from the camera system to the surface of the object. Scans of real-world objects often have noise in the measurements or missing regions due to obscured points of view. Even when the camera system has an unobstructed view of the object, portions of the object may self-occlude regions. A retrieval system using scanned input needs to handle imperfect and incomplete data. Because of the incomplete data problem, research in this area has moved toward local shape matching. The idea is that since the entire query object is not available, small features of the query object can be compared to corresponding features in a database.

Searching for textual information on the World Wide Web has become commonplace using search engines such as Google or Yahoo, but we are still at the early stages of performing 3D search. To make 3D data become truly useful, we need a search engine for 3D objects that produces accurate retrieval results efficiently. An example is shown in Figure 1.1, where search results for the key word “plane” are shown. The image thumbnails on the right show pictures of 3D objects retrieved from the database. While text search is typically able to retrieve at least one object of the desired type, Min et al. [88] have shown that most 3D data lacks consistent textual labels, causing a mixture of airplanes and mathematical planes to match the search term “plane” in this example. These results are typical for search engines that use text as the query into a database of

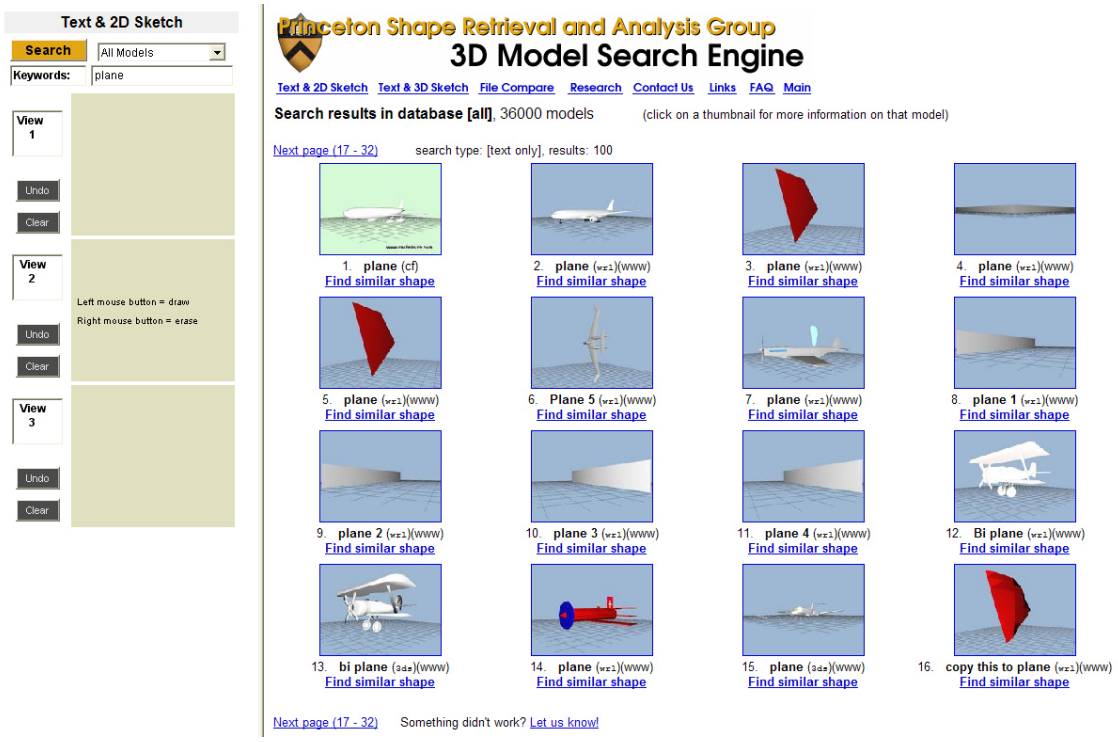


Figure 1.1: An example of text search for a 3D database. Searching for the term “plane” resulted in a variety of airplanes, as well as less-desired results.

non-textual objects. Designers who create original 3D objects often neglect to label each object, or they use labels that are only relevant within a narrow context. For example, an airplane could be labeled “airplane” indicating its function, or “Spirit of St. Louis,” indicating it is the airplane that Charles Lindbergh flew across the Atlantic. Text-based searches are often insufficient for these reasons and neglect important shape-based information. Similar issues apply to image search, where a combination of text and image similarity search is effective [100].

Given at least one good result from a database using a textual search or other method, though, we can use the shape of the object itself as a query into the database. The example search engine from the Princeton Shape Retrieval and Analysis Group [107] (Figure 1.1) supports that functionality with the “Find similar shape” links under the images. Clicking on the “Find similar shape” link under the first airplane image causes a shape similarity search to be performed, using the airplane itself as the 3D query. The goal is to then find objects in the database that have a similar 3D shape as the query object. As shown in Min et al. [88] and Figure 1.2, shape search of a 3D query object, versus text search,

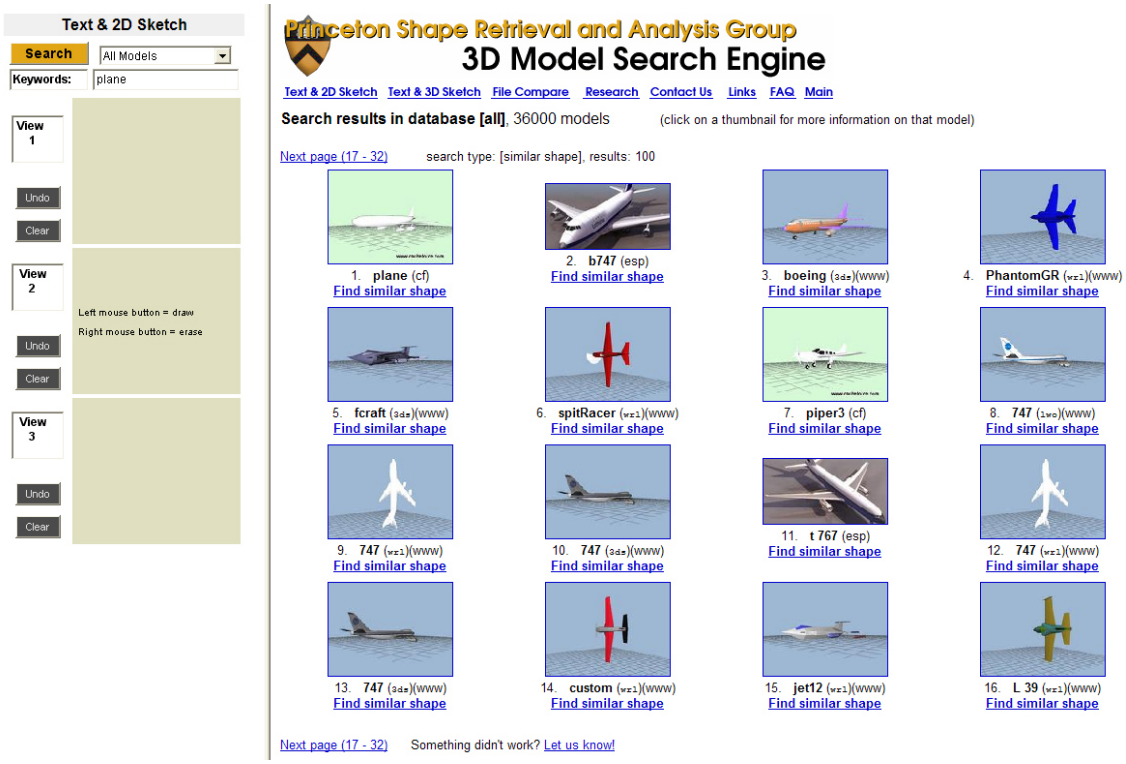


Figure 1.2: An example of shape search for a 3D database using the first airplane as a shape-based query. Airplanes with similar shapes were found.

can greatly improve retrieval results. A shape similarity method considers the geometric form of a shape and may either match a shape as a whole (global matching) or subparts (local matching). Local matching is well motivated when subparts of an object are either optional or can be located in multiple configurations such as for articulated machinery. This dissertation builds upon the idea of local feature matching, but extends the previous work by identifying important regions of shapes that distinguish objects of the correct class from other types of objects.

Representing the 3D structure of a shape with a feature vector, called a shape descriptor, is among the most effective search techniques. Using shape descriptors, shapes in a database and the query shape are transformed into a representation that can be compared directly such that a distance score can be calculated. Figure 1.3 shows a typical shape search engine system, in which there is a shape descriptor for each object represented by the "?" symbol. The distance between the query shape descriptor and every descriptor in the database is calculated and the distances are sorted from least to greatest. The closest results are then presented to the user.

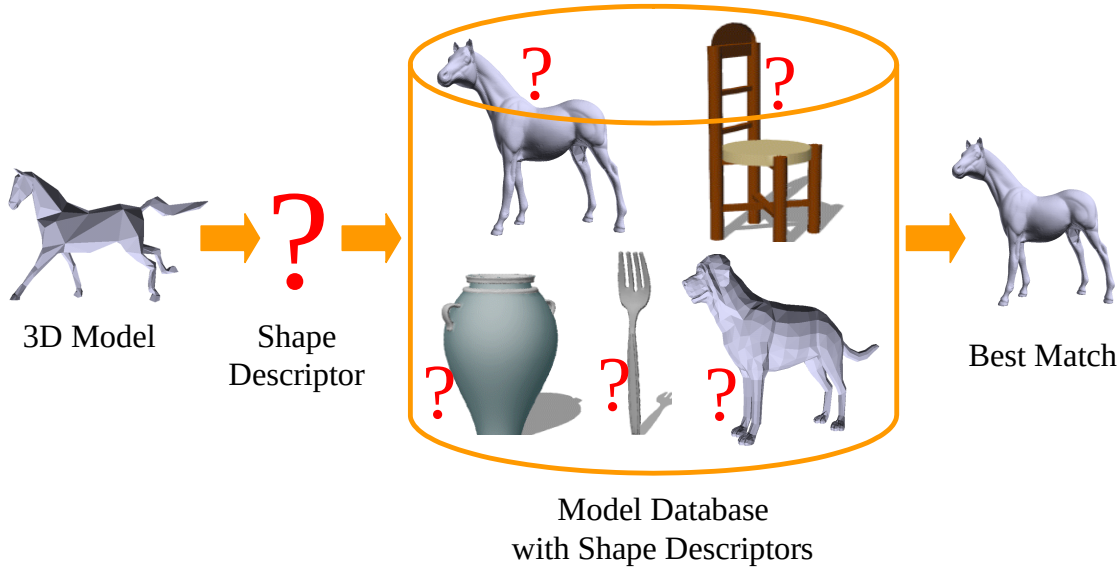


Figure 1.3: When a user presents a query model to a shape search engine, the query is converted into a shape descriptor (the "?" symbol) and compared to the descriptors representing all of the shapes in the database. The best matches are returned as results.

The multiple representations of 3D objects have led to some terminology ambiguity. *Shape* is the most generic term and refers to the abstract notion of the form of an object and corresponds to its semantic meaning. *Model* generally refers to the representation of a 3D object without specifying details of the representational format, meaning the *shape* of an object is represented by a model. Objects created by artists are also generally referred to as models. The term *mesh* specifically refers to a representation of the surface of a model as opposed to point samples or volumetric densities. A horse model has the shape of creatures in the semantic group "horse" whether it is represented as a surface mesh, points, or density values. In this dissertation, we focus mostly on surface models because they are common within the graphics community, though many of the retrieval techniques presented can be adapted to other representations (or other representations can be converted to surfaces). We use the terms shape, model, and mesh interchangeably, unless otherwise noted.

Contributions

This dissertation creates a technique to determine the distinctive regions of 3D meshes and focuses shape matching on those regions. To measure the importance of mesh surfaces, we create a methodology for evaluating shape retrieval methods. Specifically, we make the following research contributions:

Distinctive Regions Selecting the most "important" regions of a surface is useful for shape matching and a variety of applications in computer graphics and geometric modeling. While previous research has analyzed geometric properties of meshes in isolation, we select regions that distinguish a shape from objects of a different type. Our approach to analyzing distinctive regions is based on performing a shape-based search using each region as a query into a database. Distinctive regions of a surface have shapes consistent with objects of the same type and different from objects of other types. An important property of distinctive regions is that they correspond to features that distinguish between classes as opposed to previous techniques that determined important regions for each mesh in isolation.

Shape Retrieval with Distinction A shape matching method that focuses on the distinctive regions of a shape can improve retrieval success relative to considering all regions of a shape equally. To achieve this goal, the system maintains a priority queue of potential sets of feature correspondences (partial matches) sorted by a cost function accounting for both feature dissimilarity and the geometric deformation. Only partial matches that can possibly lead to the best full match are popped off the queue, and thus the system is able to find a provably optimal match while investigating only a small subset of potential matches in a few seconds per query. By filtering the set of shape descriptors representing each shape in the database to include only the most distinctive, the matching algorithm produces better retrieval accuracy than previously tested methods.

Updating Distinction Distinction values for each mesh region are a measure of how well each region corresponds to objects of the same class and distinguishes from objects of different classes. As a database undergoes the addition and removal of objects, distinction values must be updated to reflect changes in the database, which can be computationally infeasible. By calculating an approximation to distinction that

only requires a small set of similar regions that can be found with a spatial index structure, distinction scores can be updated efficiently.

Predicting Distinction The definition of distinction is based on analyzing a classified database, which facilitates processing target models, but cannot be directly applied to query shapes. Using a prediction model, distinction scores can also be found for new query models, even though their classification is unknown. During a preprocessing phase, a training set of models is analyzed with the following steps: descriptor likelihood is measured with a multi-variate Gaussian distribution of real-valued shape descriptors, the distinction score of each descriptor is calculated from a training set, and these performance values are averaged for every likelihood value. For a new shape presented to the system, distinction values are predicted across the surface of the shape based on a function that maps from the likelihood for local shape descriptors to predicted distinction values. Using predicted distinction values provides favorable retrieval performance while reducing the query time.

PSB The Princeton Shape Benchmark is a publicly available database of polygonal models collected from the World Wide Web and a suite of tools for comparing shape matching and classification algorithms. A key feature of the benchmark is that it provides multiple semantic labels for each 3D model. For instance, it includes one classification of the 3D models based on function, another that considers function and form, and others based on how the object was constructed (e.g., man-made versus natural objects), and these classifications can expose different properties of shape-based retrieval algorithms. Experiments with a large number of shape descriptors show that no single descriptor is best for all types of objects, and thus, the main contribution of the PSB is to provide a framework to determine the conditions under which each descriptor performs best.

Dissertation Structure

The remainder of this dissertation describes the background, design decisions, and results of this project. Chapter 2 discusses previous work on shape retrieval techniques. Chapter 3 gives an overview of defining distinction, and Chapter 4 explains our method in further detail as well as the effect of various parameter options. Chapter 5 demonstrates how to incorporate distinctive regions into a priority-driven search engine to improve efficiency and accuracy of retrieval. As new shapes are added to a database, distinction scores are updated, and we present an efficient method in Chapter 6. Chapter 7 presents a method for predicting distinction scores for new query models using a classified training database. In Chapter 8, we demonstrate how distinction can be used in various graphics applications. The Princeton Shape Benchmark is described in Chapter 9, and state-of-the art shape descriptors are compared. Finally, we conclude in Chapter 10 with a summary and a discussion of possible future work.

Chapter 2

Background and Related Work

To organize and analyze the large amount of 3D data that has become available, numerous search techniques have been developed. The general approach is to measure the *similarity* between a query object and every object in a database, which leads to the question, "How is the similarity of 3D shapes measured?" Alternatively, how can we measure the dissimilarity of 3D shapes such that objects of the same semantic class have a small dissimilarity score and objects of different semantic classes have a large dissimilarity score.

A common technique is to convert a model into a feature vector representation, called a shape descriptor, and then the dissimilarity of 3D objects is measured by the distance between their feature vector representations. A database of shapes can be compactly represented by their respective shape descriptors. Then, when a user presents a query shape to the system, the query shape is converted into a shape descriptor and compared against the database. One of the main advantages of using shape descriptors for matching is the ability to support high-throughput search results because the compact representation can be compared quickly and index structures can be used for further efficiency. There are several properties to consider when discussing shape descriptors:

- represents important shape properties
- compact to store
- quick to compute
- quick to compare

- insensitive to noise
- independent to 3D representation
- invariant to similarity transformations

To explain these properties in more detail, consider answering a query into a shape database with millions of models. Besides representing important features for matching, the shape descriptors must be small enough to fit within main memory for efficient comparison, calculating the descriptor for the query should be quick, and comparing the query descriptor against the entire database should rapidly return results. Also, the shape descriptor should be robust to small errors common in meshes such as missing or disconnected polygons and different polygon representations of similar surfaces.

Invariance to similarity transforms refers to a group of transformations that preserve what humans generally think of as the appearance of a shape such as position, rotation, and scale. A 3D model is typically defined with surface positions relative to a local coordinate system without any guarantee that the coordinate system is consistent across models. Ideally, a shape descriptor representation for an object would be the same regardless of its position, rotation, or scale so that if a model undergoes a similarity transformation, its descriptor will be unaffected. An alternative to designing a shape descriptor that is invariant to certain transformation is to normalize a model before calculating a descriptor. The main normalization technique is to transform all models such that their descriptors can be compared directly. Position differences can be handled by translating the center of mass of the object to $(0,0,0)$ of the coordinate system. Rotation can be handled either by determining the important axis of the shape and rotating those to align with the x,y,z planes, searching over all rotations when matching, or by designing a descriptor that is invariant to rotation about one or more axes by construction [68, 69, 130]. Scale information should be preserved for certain shape matching problems involving scans of real world objects such as proteins [1] or archaeological artifacts [54]. For artistically generated models, though, scale is often arbitrary and a common normalization technique is to scale a model to fit within a unit sphere or according to other properties. Shape normalization and invariance techniques are more thoroughly described by Kazhdan [65].

2.1 Global Shape Descriptors

Creating shape descriptors for content-based retrieval has been an active area of research for over a decade. In this section, we review the main types of shape descriptors. See the survey papers by Tangelder et al. [121], Bustos et al. [16], and Iyer et al. [56] for a thorough discussion and comparison of shape descriptors.

Simple Properties: Many simple properties of shapes have been considered as shape descriptors. For manifold models of fixed scale, volume has been considered [141, 142] as well as ratios of surface area to volume [24, 56]. The aspect ratio of bounding boxes can be used for aligned models [101, 142]. These properties create a low dimensional feature vector of limited descriptive power.

Surface Normals and Curvature: Representing surface normals (vectors perpendicular to small surface patches) and curvature properties is useful for distinguishing between boxy, man-made objects and natural, smooth surfaces. Horn developed a descriptor called the Extended Gaussian Image (EGI) [52] that records the distribution of surface normals binned to regularly positioned samples on a sphere, though rotation normalization must be performed. A similar descriptor was developed by Shum et al. [134], and when comparing two descriptors, the minimum distance was calculated across all rotations. The EGI descriptor was extended by Kang et al. [64] to incorporate with the distribution of normals the distance from the surface to the centroid in a complex-valued spherical function. The distribution of local curvature was used as a descriptor by Zaharia et al. [139]. First, local curvature was measured as a function of principal curvatures, and a histogram of these values formed the descriptor.

Surface Distribution: The position of a shape's surface is perhaps its most important property and has been investigated with many descriptors. The distribution of the distances between random points on the surface was defined by Osada et al. [99] as the D2 descriptor. For CAD models, the D2 descriptor was modified by Ip et al. [55] to incorporate whether the line segments fall completely within the volume of the model, outside the volume, or extends through both regions. Ohbuchi et al. [96] also modified the D2 descriptor to include the angle between the line segment and the normal of the surface. The higher moments of surface area have also been used for retrieval after removing the lower moments that incorporate translation and rotation information[28, 103, 131]. Saupe et al. [109] measured the distance from the centroid to the surface, sampled at regular

positions on a sphere. A more detailed version was developed by Vranic et al. [131], which considered distances from the centroid to the most distant surface region within a given shell. Ankerst et al. [1] established Shape Histograms as the distribution of the model partitioned by bins defined by evenly spaced sectors and shells. A discrete version of the Shape Histogram that measures the occupancy of voxel cells has also been used as a descriptor [70, 73, 95].

Morphing Distance: Several shape descriptors have been developed that approximate the amount of deformation needed to morph one mesh to another. Kazhdan et al. [65, 69] developed several shape descriptors that improve upon a voxel descriptor along with techniques for creating rotation-invariant descriptors. First, the Euclidean distance transform is calculated on a binary voxel grid representing the surface of a shape. Then, the values are composed with a Gaussian function and represented with spherical samples for concentric shells, which is called the Gaussian Euclidean Distance Transform. The distance between two descriptors of this form is an upper-bound on the deformation between shapes. Using a spherical representation, a rotation-invariant descriptor called the Harmonic Shape Descriptor was created by decomposing the function into spherical harmonics and storing the norm of each harmonic frequency. This general technique could be used to transform 3D functions that are rotation-dependent into a rotation-invariant version, though there is a loss of information in the conversion. Funkhouser et al. [37] presented the Fourier Shape Descriptor (FSD) based on a modification of this technique that stores the amplitude of every spherical harmonic coefficient, making the descriptor invariant to rotation about one axis.

Symmetry: Symmetry is an important feature of 3D shapes, and objects of the same semantic class often have similar planes reflecting portions of the model onto itself. Early research explored how to measure imperfect symmetries for 2D shapes by defining a continuous symmetry measure [136, 137, 138]. This was extended to 3D shapes by reflecting the surface about a plane and measuring the distance of the reflected surface to the original surface [66, 67]. For all planes through the center of mass, these distance values define a shape descriptor parameterized by the normal to the reflection plane. Podolak et al. [106] considered all reflection planes, instead of just those through the center of mass. Both symmetry descriptors were shown to improve retrieval when used in combination with non-symmetry based descriptors.

Image-based: Image-based methods are based on the property that similar shapes have similar appearances from one or more camera positions. Multiple images of shapes in a database are captured from several camera positions around each shape. The images are often processed to either produce a binary representation [80] or to extract boundary contours [36]. Then images of the query shape or sketch produced by a user are processed in a similar manner and compared to the database. Min [86] formalized the match as an optimization process for parameterized ovals approximating each image. Chen et al. [19] represented binary images by their Zernike moments and Fourier coefficients and found the minimum distance between corresponding images to produce high quality retrieval results. An improvement to this technique is to capture depth buffer images (distances from the camera) represented with their Fourier coefficients as described by Heczko et al. [48] and implemented by Vranic [129]. Recording the distance to the surface of a model is roughly similar to early stages of computer vision techniques for constructing the surface of objects with a camera system.

Matching with Global Descriptors

When calculating the distance between two instances of a shape descriptor, several approaches have been considered. Often, there is a strict ordering for the dimensions of the descriptor, meaning that the ith feature of descriptor X can be directly compared against the ith feature of descriptor Y. Then, a Minkowski distance L can be calculated with the following definition.

$$L_p(X,Y) = \left(\sum_{i=0}^{d-1} |X_i - Y_i|^p \right)^{\frac{1}{p}} \qquad X,Y \in \mathbb{R}^d, p \geq 1$$

For L_2, this is the well known Euclidean distance function. Various other distance metrics were investigated by Osada et. al [99]. Weights can also be applied to place importance on particular dimensions in a straightforward manner. For most of the types of descriptors considered in this dissertation, the L_2 distance is used.

The Minkowski distance cannot be used directly when the order of dimensions changes between descriptors. In the extreme case, there is no correspondence between the order of X and Y. A more common case is when X and Y are related by a rotation, typi-

cally because there was ambiguity about the coordinate system when calculating a shape descriptor. The optimal rotation is found by considering all constant offsets between dimensions of the descriptors.

$$\min_{\Delta} \left(\sum_{i=0}^{d-1} |X_i - Y_{(i+\Delta) \bmod d}|^p \right)^{\frac{1}{p}} \qquad X, Y \in \mathbb{R}^d, p \geq 1$$

Normalizing shapes for rotation or constructing rotation invariant descriptors are two ways to address this issue so that searching for the correct Δ is unnecessary, which makes indexing descriptors possible.

2.2 Local Shape Descriptors

While the discussion of shape descriptors has mostly focused on global representations of a shape by a single descriptor, an alternative approach is to represent a shape by many local descriptors. Then, each shape has a collection of descriptors, each describing a small region along with the position of the local descriptor either relative to a reference frame (e.g. center of mass) or other local descriptors. Using local descriptors for feature matching has numerous advantages when dealing with the following issues:

Missing Data: Many 3D models created from scanned objects are missing surfaces because of occlusion. In some cases, a model only consists of a single range image from one camera position, so the entire backside of the object is missing. Local descriptors handle these data sets naturally, because local matches can be found.

Articulation: A model may represent an underlying object that has articulated limbs, and a matching algorithm should be able to handle articulation. Examples of such models include animals that can have limbs in different positions and vehicles that may have their hoods or doors open. Local descriptors can identify local matches and allow an algorithm to consider topological similarities in the face of global differences.

Feature Importance: Because a shape descriptor converts from a model to a different representation, locality information often is lost, so it is difficult to assign importance values on specific regions. A classic example from identifying cars is that the Mercedes hood ornament is highly correlated with Mercedes vehicles, while the

rest of a sedan is fairly consistent across brands of cars. Using local descriptors, importance values can be determined for each local region.

Several researchers have investigated shape descriptors to determine feature correspondences (e.g., [7, 20, 40, 63, 94]). The general strategy is to compute multiple local shape descriptors for every object, each representing the shape for a region centered at a point on the surface of the object. Virtually any global descriptor could be used as a local descriptor by limiting the scale of the descriptor to a local region, but in this section, we present several of the more effective local descriptors.

Perhaps the most thoroughly investigated local shape descriptor is the Spin Image by Johnson et al. [63]. A spin image is calculated relative to a basis point on the surface of a model, and a cylindrical coordinate system is created relative to the surface normal. The distribution of nearby points is measured based on two parameters: distance from the basis point perpendicular to the normal and signed distance along the normal. Because of the cylindrical coordinate system, all points along a circle centered on a line aligned with the surface normal project to the same bin in the spin image, which provides invariance to rotation about the normal.

Numerous other local descriptors have been investigated as well. Chua et al. [20] created the Point Signatures by measuring the Euclidean distance from points along the circumference of a circle to a model's surface. Since the starting position of the circumference is not specified, during matching, all possible rotations of the descriptor must be considered. Shape Contexts [34, 72] is similar to the FSD but positions the shells progressively farther apart so there is more information recorded near the center of the descriptor before calculating the spherical harmonic coefficients.

Matching with Local Descriptors

When matching with local shape descriptors versus global descriptors, the matching algorithm is more complex and involves optimizing the difference between numerous local shape descriptors or uses descriptors to create an alignment between shapes before a more expensive difference calculation.

Recently, several researchers have investigated approaches to partial shape matching based on feature correspondences (e.g., [7, 20, 40, 63, 94]). The general strategy is to compute multiple local shape descriptors (shape features) for every object. Then,

the similarity of any pair of objects is determined by a cost function determined by the optimal set of feature correspondences at the optimal relative transformation, where the optimal match minimizes the differences between corresponding shape features and the geometric distortion implied by the feature correspondences. This approach has been used for recognizing objects in 2D images [7, 9], recognizing range scans [63], registering medical images [5], aligning point sets [21], aligning 3D range scans [42, 79], and matching 3D surfaces [94, 111].

The challenge is to find an optimal set of feature correspondences efficiently. One approach is to consider an association graph [6] containing a node for every possible feature correspondence and an edge for every compatible pair of correspondences. If each node is weighted by the dissimilarity of its associated features and each edge is weighted by the cost of the geometric deformation implied by its associated pair of correspondences, then finding the optimal set of k feature correspondences reduces to finding a minimum weight k-clique in the association graph. Researchers have approached this problem with algorithms based on branch-and-bound [42], integer quadratic programming [9], etc. However, previous work in this area has been aimed at pairwise alignment of objects, and current solution methods are generally too slow for a search of large databases.

"Bag of words" approaches can be used to discretize descriptor space, where descriptors are binned into a discrete set of possible values. For example, Mori et al. [89] clusters descriptors into "shapemes," builds a histogram of shapemes for every object, and then approximates the similarity of two objects by the similarity of their histograms. Grauman et al. [46] extended this approach to consider pyramids of clusters. However, these methods make little or no use of the geometric arrangements of features, which is an important property when distinguishing among models.

Another approach using local shape descriptors is based on the RANSAC algorithm [30, 111]. Sets of k feature correspondences are generated, where k is large enough to determine an aligning transformation, and the remaining features are used to score how well the objects match after the implied alignment. For example, Johnson et al. [63] finds small sets of compatible feature correspondences, computes the alignment providing a least-squared best fit of corresponding features, and then verifies the alignment with an iterative closest point algorithm [11]. Shan et al.[111] proposed a "Batch RANSAC" version of this algorithm that considers matches to all target objects in a database all

at once, generating candidate matches preferentially for the target objects with features compatible with ones in the query.

Graph matching algorithms represent a shape by a set of regions with connections between adjacent regions. Then, a match score between two models is related to the similarity of their graph representations, where each graph node may also have properties such as a local shape descriptor. A model may be divided into parts using the Medial Axis Transform [14], Reeb graph [49], thinning operator [39, 58, 119], or other technique. Constructing a graph for a model in the context of noise leads to matching methods that attempt to compensate for changes in topology [104, 110, 115, 116]. Besides the difficulty of constructing consistent graphs, computing the matching subgraph (subgraph isomorphism) is known to be computationally inefficient.

2.3 Selection of Local Shape Descriptors

Considering all possible feature correspondences for matching may be infeasible, because the number of possible feature correspondence sets grows exponentially with the set size. Naively checking all possible sets of k feature correspondences among n features on two objects takes $O(n^k)$ operations. In practice, searching the space of potential feature correspondences for a single pair of surfaces can take several seconds or minutes, and using these methods to find the best matches in a large database is impractical. A common technique is to subsample important shape descriptors to create a subset representative of the original shape. There has been a long history of related work in cognitive psychology, computer vision, computer graphics, geometric modeling, statistics, and pattern recognition.

Perceptual Criteria: There have been several attempts to select regions of 3D shapes that humans find visually important in object recognition, perceptual psychology, and computer vision. For example, Howlett et al. [53] used an eye-tracker to record which surfaces of a 3D model people tend to focus on and then used that information to assign importance to vertices in a mesh simplification algorithm. While this method captures a useful notion of surface importance, it is viewpoint-dependent and requires human analysis of every 3D mesh, which is impractical for the large databases of 3D meshes targeted by our system.

Several psychophysical experiments have found that the human visual system quickly processes regions of high curvature (e.g., [50]), and these findings have been applied extensively for object recognition in computer vision [82, 83]. For example, combinations of filters measuring edges and local maxima of curvature in 2D images have been used to focus scene recognition algorithms [33]. More recently, curvature filters have also been applied to define measures of saliency for mesh processing applications. For example, Lee et al. [77] use a center-surround filter of curvature across multiple scales to select salient regions for mesh simplification and viewpoint selection. Similarly, Gal et al. [40] compute the saliency of a region based on its size relative to the whole object, its curvature, the variance of the curvature, and the number of curvature changes within the region. They use this measure to guide partial shape matching, self-similarity detection, and shape alignment. Li et al. [79] compute surface signatures describing the curvature and other properties for local regions and only keep the ones with significantly non-zero magnitude. Novotni et al. [94] select points found as local extrema of the differences of Gaussian filters applied to the characteristic function of the surface.

While these approaches are able to select regions that may be visually noticable, they focus on curvature and other measures appropriate for manifold surfaces. Thus, they cannot be used effectively for the majority of 3D computer graphics models which often contain disjoint and intersecting polygons. More importantly, they measure how much a region sticks out from the rest of the object rather than how important the region is for defining the object type.

Statistical Criteria: Numerous techniques have been developed for selecting important features in the realm of statistical analysis and pattern recognition, which are covered in several classical books [27, 47, 84]. The problem is that given a set of feature vectors and labeled training data to select a subset of features that is most useful for classification. Many techniques from this field can also produce real-valued importance scores for each feature. Discriminant analysis [74] or analysis of variance (ANOVA) [85] selects feature vectors that are consistent within a class and have a large separation from other classes of objects. Using regression analysis [23, 32], a subset of features can be selected with stepwise selection that either grows or shrinks a subset of features to optimize a function. Stepwise methods [59] add or remove features using a variety of error metrics including a measure of group differences or the Mahalanobis distance between groups and stop when altering the subset of features would create an insignificant change to the accuracy. Linear

regression was used to remove outlier points for 2D matching by Dryden et al. [26] The problem we address differs from classical statistical analysis because these approaches assume a correspondence between features, and they select dimensions of a feature vector rather than positions on a surface.

In the shape matching literature, the relative rarity of local surface patches has been used as a measure of importance to guide several shape matching systems without requiring correspondences between patches. Typically, representations of local shape (shape descriptors) are computed for many regions of a surface, and then they are weighted according to a measure of uniqueness when point sets are aligned and/or matched. For example, Chua at al. [20] found "selective points" on a surface by comparing their descriptors to others in a local neighborhood and used the descriptor that was most unique for shape matching. Johnson [62] computed the likelihood of each shape descriptor based on a Gaussian distribution of the descriptors for each mesh and then selected only the least likely, i.e. rarest, descriptors to speed up surface matching. However, these methods only find descriptors that are rare – they do not specifically find ones that are distinctive of an object class.

Shan et al. [111] used shape descriptor matching to define the importance of points on a mesh after calculating correspondences in a database. This method selects points based on how well they match multiple points on one object (e.g., a spherical region will be selected if there is an object with many spherical regions in the database), and thus it provides a measure of stability for shape matching, rather than a measure of importance for object classification, as is provided by our method.

2.4 Research Challenges

There are several research challenges to improve all phases of a retrieval system. These challenges include increasing the accuracy of search results, improving the speed of finding results, and answering the underlying question of how to even evaluate success.

First, certain features of a shape are often more important for defining its class than others. For a chair, the back, seat, and legs define the class regardless of the level of ornamentation, color, or the aspect ratio of the object. The exact position of certain features is also irrelevant for some objects - e.g. whether a chimney is on the side or rear

of a roof. A shape matching algorithm should be able to place more weight on important regions.

Second, while local matching can handle certain issues more effectively than global descriptors, it requires a large increase in the number of descriptors and slows matching time. Handling articulation and occluded data naturally is one of the advantages of using local descriptors. There is also an opportunity to focus matching on local regions that are judged to be more important. Using local descriptors may involve dozens to hundreds of descriptors, so the question remains how to realize the advantages of local matching in an efficient search algorithm.

Third, indexing structures are needed to speed up the search for neighbors in the high dimensional space of shape descriptors. Typically, a user is satisfied with a few pages of search results, so it is unnecessary to calculate the distance between the query and every other shape if the nearest neighbors can be found directly. A nearest neighbor search is usually accelerated with indexing data structures, but indexing generally degrades to linear search time when the dimensionality is beyond fifty.

Fourth, given the large number of shape retrieval methods investigated in the literature, users are often left wondering, "Which technique works best?" It is not a straightforward question because most papers in the field evaluate a shape retrieval technique on a customized data set without comparing to previous techniques, and they have used their own measurement of retrieval performance.

We address several of these research challenges in this dissertation. Based on analyzing shapes in a database, we improve upon standard shape retrieval techniques by identifying the most distinctive regions of each shape (Chapters 3 and 4). We focus a local shape matching method on those regions using a priority-driven search algorithm that efficiently searches for the best matches before calculating the distance to more distant matches based on monotonically increasing partial distances (Chapter 5). As a database changes, distinction scores for hundreds of thousands of descriptors must be updated, and we explore several techniques involving a standard index structure (Chapter 6). Besides analyzing distinctive regions of models in a database, we predict important regions for query models to improve matching (Chapter 7). Finally, with the Princeton Shape Benchmark, we create a methodology to evaluate shape retrieval performance (Chapter 9). This allows us to directly compare shape retrieval algorithms on the same data set and report standardized performance metrics.

Chapter 3

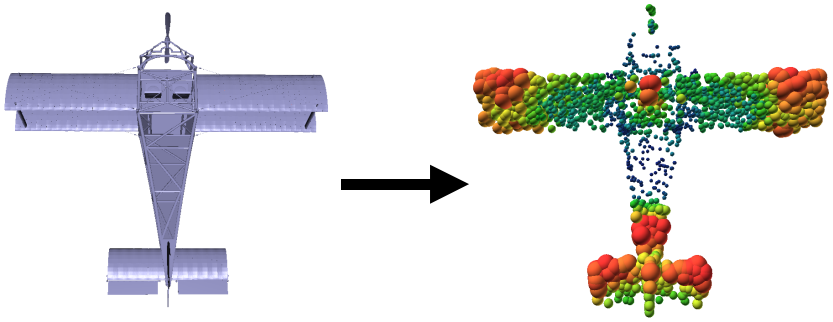
Introducing Distinction

Introduction

Many problems in computer graphics, computer vision, and geometric modeling require reasoning about which regions of a surface are most "important." For example, in an object classification system, a query might be compared only against the most important regions of a target object to provide more discriminating matches. For mesh simplification algorithms, the importance of vertices may guide the order in which they are decimated, and in an icon generation system, the most important regions of an object should be visible. Our approach is to compute local shape descriptors, analyze which descriptors best represent each class of objects, and integrate that analysis information into a shape matching algorithm (Shilane and Funkhouser [113]).

Although there has been significant progress in algorithms for determining important regions of polygonal meshes, most prior work has focused on geometric properties of every mesh in isolation. For example, Lee et al. [77] defined a measure of mesh saliency using a center-surround operator on Gaussian-weighted mean curvatures. Related measures of mesh importance have been defined by Gal et al. [40], Li et al. [79], Novotni et al [94], Gelfand et al. [42], and others. However, almost all of these methods simply select regions where the curvature of a surface patch is different than in its immediate neighborhood.

Intuitively, the important regions of an object for recognition are not the ones with specific curvature profiles, but rather the ones that distinguish it from objects of other types, i.e., the *distinctive* regions. For example, consider the biplane shown in Figure 3.1. Most people will tell you that the important features of the biplane are the wings and tail. Those features are unique to biplanes and thus distinguish the biplane from other types of vehicles. Generalizing this idea, we define the *distinction* of a surface region as how useful it is for distinguishing the object from others of different types.



Mesh Mesh Distinction

Figure 3.1: Distinctive regions of a plane correspond to the important regions that define the object type and distinguish the plane from other types of objects. Regions shown in red are the most distinctive, blue are least distinctive, and green are in the middle. This result corresponds to our intuition that the wings and tail are important features of a plane.

Intuitively, regions that are common among many object classes are not distinctive (e.g., planar regions, spherical regions, etc.), while others that are found in only one object class are very distinctive (e.g., the head of a wrench). In our system, each region of a shape is considered as a query into a classified database, the best matching region on each shape is determined with a distance metric, and the matches are sorted from closest match to furthest match. This list is the retrieval result for each region. The retrieval list is evaluated, and we assign a continuous value of distinction to every surface region, with “0” indicating that the region is not distinctive at all (i.e., that region could be found equally well in any object class), “1” indicating that the region is perfectly distinctive (i.e., that region is found in only one object class), and values in between representing the degree to which the region distinguishes the object class.

This definition of distinction has an important implication: in order to determine how distinctive a surface region is, we must not only consider its properties, but we must also consider how consistent those properties are within other instances of the same

object type and how unique those properties are with respect to other object types under consideration. For example, if we consider the biplane among other types of airplanes, we find that the wings are the most distinctive features. However, if we consider it as a biplane among other types of objects (tables, animals, cars, etc.) many of which have large flat regions, we find that the tail is most distinctive (Figure 3.1). In general, the distinctive regions of a surface will be different depending on the granularity and range of object types under consideration.

3.1 Examples

To help the reader understand which regions are found distinctive by the proposed method, we show a sampling of images depicting which regions are found to be distinctive for a variety of object classes in a variety of databases. Implementation details are provided in Chapter 4. In all images, regions shown in red are the most distinctive, blue regions are least distinctive, and green regions are in the middle. When computing distinction, local shape descriptors were generated to include a small region of each object (0.25 times the mesh radius). For example, in Figure 3.2, the ears of the Stanford Bunny are unique to rabbits (red) and thus distinguish the bunny from other classes of animals, while the shape of the body is not very distinctive (blue). Similarly, the head of the wrench, wheels of the vehicles, pot of the plant, and struts of the guitar are important parts for distinguishing each class of objects within the Princeton Shape Benchmark [114].

Our next example shows distinctive regions found for three helicopters (all except the right-most image of Figure 3.3). In this case, distinction was measured with respect to a database of flying vehicles dominated by airplanes. The propellers are red (the most distinctive region) in every case, which matches our intuition that the part that distinguishes a helicopter from other flying vehicles is its propeller. For comparison sake, we show the mesh saliency values computed by Lee et al. [77] for the third helicopter. The areas predicted to be salient by their approach highlight portions of the cockpit and tail due to variations of curvature there, and do not detect the importance of the propellers.

A second example shows regions that distinguish cars from other classes of vehicles (Figure 3.4). In this case, distinction was measured with respect to a database containing cars, planes, and jeeps selected from the PSB. The wheels are most distinctive (red) in

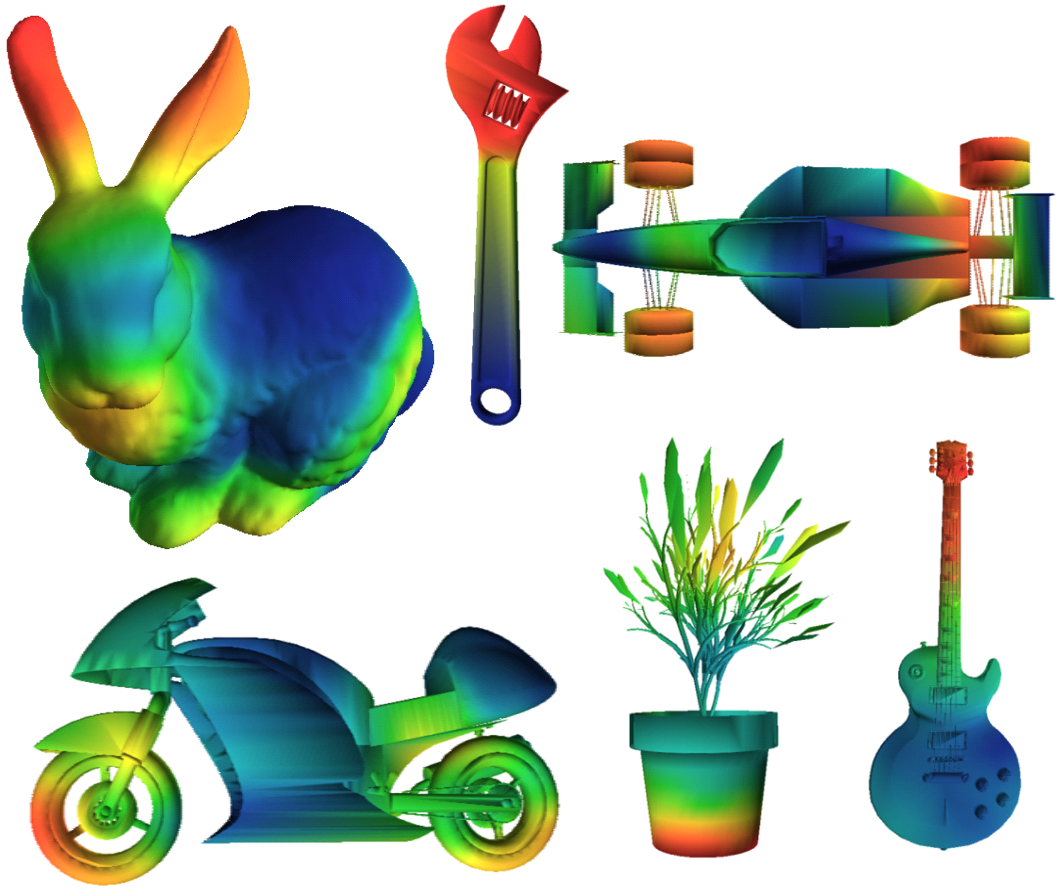


Figure 3.2: Distinctive regions of meshes correspond to the important regions that define their class of objects. In all images, regions shown in red are the most distinctive, blue are least distinctive, and green are in the middle. Models are from the Princeton Shape Benchmark, and regions were sized to be 0.25 times the mesh radius.

all cases, although for the top-left car, the front and rear of the car are equally distinctive probably because of the different aspect ratio of this car relative to other cars. Again, for comparison sake, we show the mesh saliency values computed by Lee et al. for the top-right car – the regions predicted to be salient by their approach do not correspond as well to distinguishing parts.

A third example shows the regions found to be distinctive for a set of humans standing in a spread-eagle pose in Figure 3.5. In this case, distinction was measured with respect to all classes in the test database of the PSB. For thirteen of the fifteen examples, the arms are found to be the most distinctive region. For the other two, the top of the head is most distinctive (those two people have wider arms and a slightly different pose than

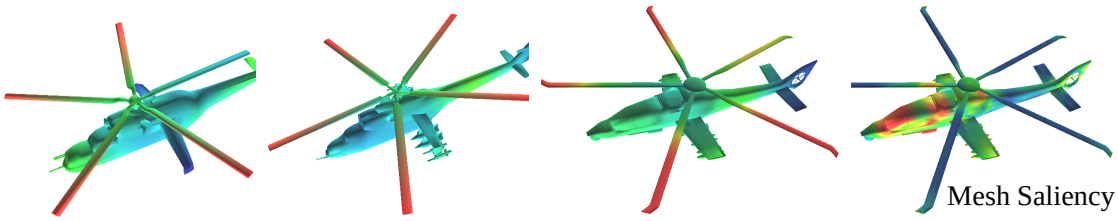


Figure 3.3: Visualizations of distinctive regions (red) on three helicopters with respect to a database of flying vehicles selected from the PSB. The two right-most images show a comparison of distinction to mesh saliency as computed by Lee et al. for the same helicopter model. Regions were calculated at 0.25 times the mesh radius.

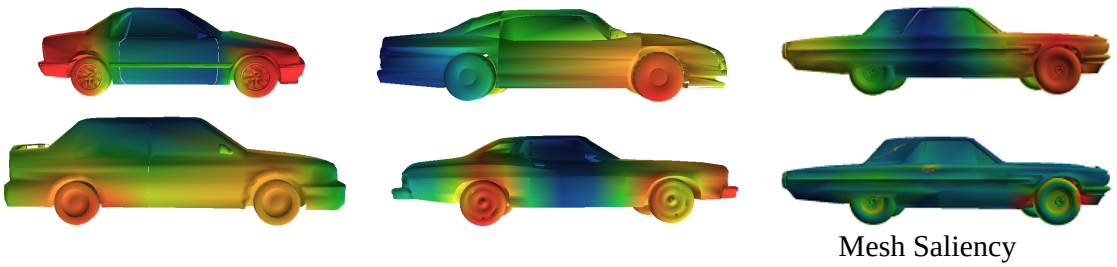


Figure 3.4: Visualizations of mesh distinction for five cars as computed with our method. The right-most column shows a comparison between distinction and saliency for the same car model. Note that the tires are consistently distinctive, but not particularly salient. Regions were calculated at 0.25 times the mesh radius.

the others). This result is interesting because the region found to be most distinctive is not obviously what a human might choose as the distinguishing feature of this class at first glance. However, the PSB test database has 92 different classes, including “humans in a standing pose,” “humans in a walking pose,” “faces,” “heads,” “skulls,” “hands,” etc., and thus it is indeed the pose of the arms that best differentiates this class from the others. This example points out both an advantage and disadvantage of our approach: our method can automatically determine the differences between classes in a database, but the distinctive regions may not correspond to semantic parts.

Figure 3.5: Visualizations of mesh distinction for fifteen humans. Note that the distinctive regions for humans in this pose are typically around the elbow area. This region best differentiates this class of objects from other classes of human models in the Princeton Shape Benchmark with regions calculated at 0.25 times the mesh radius.

Chapter 4

Computing Distinction

Introduction

Our definition of shape distinction requires evaluating every surface region to determine how well it matches shapes of the correct class. In this chapter, we describe a method for computing distinction that is motivated by shape retrieval applications (Shilane and Funkhouser [113]). A shape retrieval method may match subregions of shapes using local shape descriptors to represent each region, where the similarity of two shapes is related to the similarity of their subparts. This may involve computing numerous shape descriptors, possibly at multiple scales. Our technique for computing shape distinction fits into this pipeline by analyzing local shape descriptors in the database during a preprocessing phase. We assign a distinction score that relates the value of matching each region relative to the current database, which can be used to focus a shape matching algorithm. In the remainder of this chapter, we explain our specific technique and explore properties of distinction scores.

The organization of this chapter is as follows: In Section 4.1, we explain our technique for processing meshes to compute distinction scores. The subsections that follow explain each step of our technique in more detail. Then, in Section 4.2, we demonstrate how various parameters affect distinction scores. Finally, Section 4.3 provides a discussion of conclusions and limitations of our approach.

4.1 System Overview

Computation of surface distinction proceeds in our system as shown in Figure 4.1. Given a database of meshes partitioned into object classes, we first generate for each mesh a set of random points that are the centers of spherical regions covering its surface at multiple scales. Then, for every region, we compute a shape descriptor representing the distribution of surface area within that region. Next, we compare the difference between all pairs of shape descriptors to produce a ranked list of matches for each descriptor ordered from best to worst. The ranked lists are then analyzed to produce measures of how distinctive different local regions are – i.e., how many descriptors from the same class of objects appear near the front of their ranked lists. These measures can be directly used to improve shape matching applications, and can also be mapped from the regions back onto the vertices of the mesh and used to guide mesh visualization, processing, and analysis. The following subsections describe each of these steps in detail.

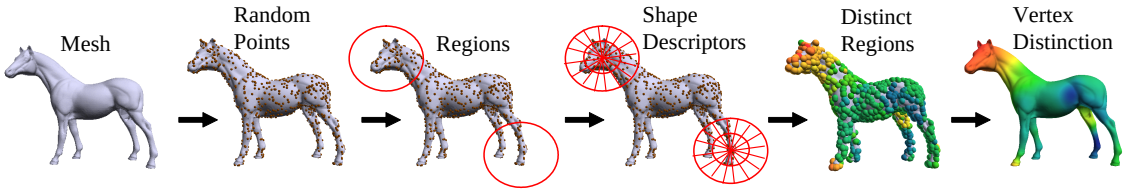


Figure 4.1: Overview of our technique for computing distinction on the surface of a mesh.

4.1.1 Constructing Regions

The first step of our process is to define a set of local regions covering the surface of the object. In theory, the regions could be volumetric or surface patches; they could be disjoint or overlap; and they could be defined at any scale.

In our system, we construct overlapping regions defined by spherical volumes centered on points sampled from the surface of an object (Figure 4.2). This choice supports robust processing of arbitrary surface meshes with degenerate topology, and it naturally supports overlapping regions at multiple scales. Formally, the database consists of a set of meshes $\{M_1,...,M_m\}$, each mesh M_j has a set of points $\{p_{1,j},...,p_{n,j}\}$ where $p \in R^3$, and each point has a set of scales $\{s_1,...,s_h\}$, where a spherical region $r_{i,j,k}$ has center $p_{i,j}$ and radius s_k. We have experimented with two different point sampling methods, one that

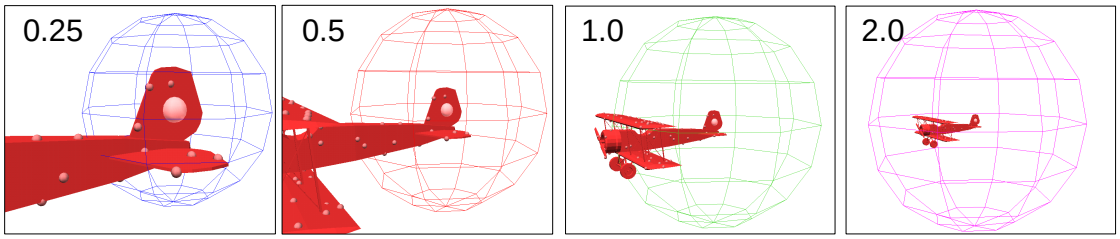


Figure 4.2: Four regions are shown with the same center point but different scales. At the 0.25 scale, the tail is included, and at larger scales, progressively more of the plane is included. At the 2.0 scale, the entire plane is included, even when the region is centered on the end of the plane.

selects points randomly with uniform distribution with respect to surface area and another that selects points at vertices of the mesh with probability equal to the surface area of the vertices' adjacent faces. Of course, other sampling methods that sample according to curvature, saliency, other surface properties, or based on volumetric properties [91] are possible as well. However, we found that they do not give significantly different performance, and so we consider only random sampling with respect to surface area in the remainder of this dissertation. In most of our experiments, we consider four different scales for every point, where the smallest scale has radius 0.25 times the radius of the entire object and the other scales are 0.5, 1.0, and 2.0 times, respectively. Note that the biggest scale is just large enough to cover the entire object for the most extreme position on the surface, and the smallest scale is large enough to contain easily recognizable shape features.

Our implementation for selecting random points on a surface as centers for these spherical regions follows the approach of Osada et al. [99]. We have modified their algorithm slightly to make sampling more efficient and to stratify samples in meshes with large triangles. Specifically, in the first stage, we allocate a number of points to every triangle in proportion to its surface area. Then, in the second stage, we sample the allocated number of points from every triangle uniformly with respect to its surface area. This method is faster than Osada's method, taking $O(n)$ rather than $O(nlogn)$ for a mesh with n triangles.

4.1.2 Describing Shapes

In the second step of our process, for each spherical region $r_{i,j,k}$, we generate and store a shape descriptor $x_{i,j,k}$ which has dimension d. There will be many such regions for every surface, so the shape descriptors must be quick to compute and concise to store. Since we will be matching all pairs of the shape representations, they must be indexable and quick to compare. Also, our methods should work for any input object representation; they must be independent of shape description, insensitive to topology, and robust to common input file degeneracies. Finally, since we aim to model how distinctive the shape of each surface region is, they must be discriminating of similar versus dissimilar shapes.

There are many shape descriptors that meet some or all of these goals (see surveys in [16, 56, 121]). For example, Belongie et al. [8] have used shape contexts for describing local regions of 2D images, and Kortgen et al. [72] have extended their method to 3D. However, shape contexts are dependent on a particular orientation and thus require alignment within a global coordinate system or searching possible rotations as they are matched [34]. Johnson et al. [63] have used spin images to represent the shapes of local regions with orientation dependence on just the normal to the surface at a sample point, and Vranic et al. [130] have described Fourier descriptors that could be used to provide invariance to all rotations except those around the surface normal. However, those methods are sensitive to normal orientation, which is highly variable in sparsely sampled point sets considered in this dissertation. Kazhdan et al. [69] described a Harmonic Shape Descriptor (HSD) that is invariant to all rotations. The main idea is to decompose a spherical region into concentric spherical shells of different radii, compute the spherical harmonic decomposition for a function describing the shape in each of those shells, and then store the amplitudes of the harmonic coefficients within every frequency (order) to form a feature vector for indexing and matching (see [36] for details).

In our system, we have experimented with three different shape descriptors based on spherical harmonics. All three decompose a sphere into concentric shells of different radii and then describe the distributions of shape within those shells using properties of spherical harmonics. The first (SD) simply stores the amplitude of all shape within each shell (the zero-th order component of spherical harmonics) – it is a one-dimensional descriptor equivalent to the Shells shape histogram of [2]. The second (HSD) stores the amplitude of spherical harmonic coefficients within each frequency – it is equivalent to the Harmonic Shape Descriptor of [36, 69]. The last (FSD) descriptor stores the ampli-

tude of every spherical harmonic coefficient separately – it is similar to the Harmonic Shape Contexts of [34]. In all of our experiments, we utilize 32 spherical shells and 16 harmonic frequencies for each descriptor.

We chose these shape representations for several reasons. First, they are well-known descriptors that have been shown to provide good performance in previous studies [34, 114]. Second, they are reasonably robust, concise, and fast to search. Finally, they provide a nested continuum with which to investigate the trade-offs between verbosity and discrimination – SD is very concise (32 values), but not that discriminating; HSD is more verbose (512 values) and more discriminating; and FSD is the most verbose (4352 values) and the most discriminating. The three descriptors are related in that each of the more concise descriptors is simply a subset or aggregation of terms in the more verbose ones (e.g., the SD descriptor stores the amplitude of only the zero-th order spherical harmonic frequencies). Thus, the L_2 difference of each descriptor provides a lower bound on the L_2 difference between the more verbose ones, which enables progressive refinement of descriptor differences, which could be exploited in future work.

Our method for computing the HSD for all regions of a single surface starts by computing a 3D grid containing a Gaussian function of the surface's Euclidean Distance Transform (GEDT) [65]. This function, which is one at the surface and falls off gradually with Euclidean distance, provides a soft penalty function for matching surfaces by comparison of volumes. The GEDT grid resolution is chosen to match the finest sampling rate required by the HSD for regions at the smallest scale; the triangles of the surface are rasterized into the grid; and the squared distance transform is computed and stored. Then, for every spherical region centered on a point sampled from the surface, a spherical grid is constructed by sampling the GEDT at regular intervals of radius and polar angles; the Spharmonickit software [118] is used to compute the spherical harmonic decomposition for each radius; and the amplitudes of the harmonic coefficients within each frequency are stored as a shape descriptor. The amplitudes of the harmonic coefficients (or frequencies, depending on the type of shape descriptor) are computed; the shape descriptors are compressed using principal component analysis (PCA); and the dimensions associated with the top C eigenvalues (C $\sim$ 10%) are stored as a shape descriptor.

For each 3D object, computing the three types of shape descriptors centered at 128 points for 4 scales (0.25, 0.5, 1.0, and 2.0) takes approximately four minutes overall and generates around 1 MB of data per object. One minute is spent rasterizing the

triangles and computing the squared distance transform at resolution sufficient for the smallest scale descriptors, almost two minutes are spent computing the spherical grids, and a few seconds are spent decomposing the grids into spherical harmonics for each object. Compression amortizes to approximately one minute per object for FSDs and approximately 1 second per object for HSDs.

4.1.3 Measuring Distinction

In the third step of our process, we compute how distinctive every shape descriptor is with respect to a database containing multiple classes of objects. Our goal is to compute a continuous measure that reflects how well the shape descriptor for a local region of a surface matches others within the same class of objects relative to how well it matches descriptors in other classes. Descriptors whose best matches are all from its own class are distinctive, while ones that match descriptors in a wide variety of classes equally well are not. While we would ideally like to calculate the distinction value for all combinations of local descriptors and at all scales, this is computationally infeasible. Instead, we make an independence assumption and calculate distinction for each descriptor independently, modeling distinction with an information retrieval metric.

Given the distance from the ith descriptor of mesh M_j (i.e. descriptor $x_{i,j,k}$) to the closest descriptor of every other mesh in the database, where the distance to mesh M_t is:

$$dist(x_{i,j,k}, M_t) = \min_b ||x_{i,j,k} - x_{b,t,k}||$$

We sort the distances from smallest to largest to create the retrieval list for $x_{i,j,k}$. This retrieval list represents the order of matching results in the database if we had used a single descriptor from M_j as a query. We then compute the distinction of the descriptor by evaluating a retrieval performance metric that measures how well meshes in the query's class appear near the front of the list. Retrieval metrics typically evaluate a retrieval list into a score between 0 and 1, where scores closer to 1 indicate a better retrieval list. There are numerous evaluation metrics that could be used to convert a retrieval list into a numeric score. While none of these statistics are new, we include detailed descriptions for completeness.

Nearest Neighbor is the percentage of the closest matches that belong to the same class as the query. This statistic provides an indication of how well a nearest neighbor classifier would perform. Obviously, an ideal score is 100%, and higher scores represent better results.

First-Tier and Second-Tier refer to the percentage of models in the query's class that appear within the top K matches, where K depends on the size of the query's class. Specifically, for a class with $|C|$ members, $K = |C| - 1$ for the first tier, and $K = 2 * (|C| - 1)$ for the second tier. The first tier statistic indicates the recall for the smallest K that could possibly include 100% of the models in the query class, while the second tier is a little less stringent (i.e., K is twice as big). These statistics are similar to the "Bulls Eye Percentage Score" ($K = 2 * |C|$), which has been adopted by the MPEG-7 visual SDs [139]. In all cases, an ideal matching result gives a score of 100%, and higher values indicate better matches.

E-Measure is a composite measure of the precision and recall for a fixed number of retrieved results [125]. The intuition is that a user of a search engine is more interested in the first page of query results than in later pages. So, this measure considers only the first 32 retrieved models for every query and calculates the precision and recall over those results. The E-Measure is defined as [125, 78]:

$$E = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

The E-measure is equivalent to subtracting van Rijsbergen's definition of the E-measure from 1. The maximum score is 1.0, and higher values indicate better results.

Discounted Cumulative Gain (DCG) is a statistic that weights correct results near the front of the list more than correct results later in the ranked list under the assumption that a user is less likely to consider elements near the end of the list. Specifically, the ranked list R is converted to a list G, where element G_i has value 1 if element R_i is in the correct class and value 0 otherwise. Discounted cumulative gain is then defined as follows [60]:

$$DCG_i = \left\{ \begin{array}{ll} G_1, & i = 1 \\ DCG_{i-1} + \frac{G_i}{lg_2(i)}, & otherwise \end{array} \right\}$$

This result is then divided by the maximum possible DCG (i.e., that would be achieved if the first C elements were in the correct class, where C is the size of the class) to give the final score:

$$DCG = \frac{DCG_k}{1 + \sum_{j=2}^{|C|} \frac{1}{lg_2(j)}}$$

where k is the number of models in the database.

Each of these measures has trade-offs in terms of how much of the retrieval list is included in the calculation (nearest neighbor uses the first retrieval result, while DCG requires the full list) versus the time necessary to calculate the results (nearest neighbor could be quickest using an indexing structure to find the closest result and DCG the slowest). We have selected the DCG [60] retrieval measure for most of our experiments because it has been shown to provide the most stable retrieval measure in previous studies [78, 114]. The choice of retrieval metric is considered in greater detail in Chapter 6. The distinction score for a descriptor $x_{i,j,k}$ associated with position $p_{i,j}$ and scale s_k is the DCG score calculated for the retrieval list when using $x_{i,j,k}$ as a query.

$$D(x_{i,j,k}) = D(p_{i,j}, s_k) \equiv DCG$$

In our system, we compute and store a measure of retrieval performance for every shape descriptor of every object during an off-line processing phase. Comparing two HSD descriptors takes 2.5E-6 seconds on a 2.2 GHz computer running Linux, and in general takes $O(sn^2m^2)$ time to make all pairs of comparisons, where s is the number of scales, n is the number of descriptors per mesh, and m is the number of meshes in the database. This process takes 37 hours for 128 points at four scales for 907 meshes in the Princeton Shape Benchmark. However, it must be done only once per database during a batch processing phase – we computed the distinction for each descriptor once and then stored it in a file for use multiple times in various applications.

4.1.4 Mapping to Vertices

The final step of the process is to map the computed measure of class distinction back onto the vertices of the mesh. While this step is not required of all applications (e.g.,

shape matching), it is useful for several mesh processing tasks (e.g., mesh simplification) that need to have a measure of importance associated directly with every vertex. An alternative to mapping distinction scores from samples on the surface is to calculate shape descriptors at each vertex and calculate distinction directly, but we have typically been working with a vastly smaller number of descriptors than the number of vertices per mesh.

Our approach to this problem is quite straightforward. We simply model distinction as a mixture of Gaussians. For every vertex, we estimate how distinctive it is by computing a weighted average of the DCG values that have been computed for nearby shape descriptors for a given scale where the weights are determined by the value of a Gaussian function of the distance between the vertex and the middle of the surface region, $p_{i,j}$.

Consider mesh M consisting of a set of shape descriptors each with a center position $p \in R^3$ and distinction score D defined for each scale s, where $D(p,s)$ is calculated as described in Section 4.1.3. For every vertex v on the mesh of M, distinction is defined as follows:

$$D(v,s) = \frac{\sum_{p \in M} D(p,s) e^{\frac{-\|p-v\|^2}{2\sigma^2}}}{\sum_{p \in M} e^{\frac{-\|p-v\|^2}{2\sigma^2}}}$$

While using the Euclidean distance instead of geodesic distance ignores connectivity information, it is robust to disconnected meshes. Also, since the regions selected on each shape are generally overlapping and nearby descriptors tend to be similar, it is reasonable to assume, and we observe in practice, that distinction scores change smoothly across a mesh. In all of the following results, we set σ=0.1 times the mesh radius.

4.2 Results

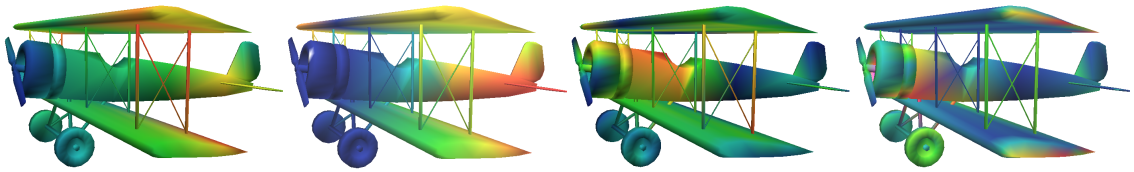
The methods described in the previous section have been tested on several databases of 3D meshes. In this section, we present images depicting mesh distinction for several examples and investigate: 1) how sensitive our results are to different parameter settings, and 2) how mesh distinction compares to previous measures of importance (i.e., saliency).

4.2.1 Effect of Database

A key feature of our formulation for mesh distinction is that the results depend on the database under consideration and how it is partitioned into object classes. In this section, we investigate how the distinctive regions of a surface might be affected by changes in the database.

Figure 4.3 shows four images of the same biplane, the first three of which are colored by mesh distinction as computed with our method at 1024 positions at the 0.25 scale, while the fourth image from the left shows the mesh colored by mesh saliency. The difference between the first three images is only that different databases were used to evaluate the DCG measure during the computation of mesh distinction. The left-most image shows that the wings and tail are most distinctive with respect to the other 91 classes in the Princeton Shape Benchmark. The second image shows that the tail is most distinctive with respect to a smaller database containing other classes of vehicles (cars, jeeps). The third image shows that the struts between the wings and cockpit are most distinctive with respect to a database containing different classes of planes (commercial jets, fighter planes, etc.). In contrast, the fourth image shows that mesh saliency is unaffected by database changes.

In short, the distinctive area of the biplane changes depending on the database under consideration. This is a very useful feature of our method, as it allows the measure to adapt to finer differences in databases with more similar object classes.



Princeton Shape Benchmark | Vehicle DB | Plane DB | Mesh Saliency

Figure 4.3: The distinctive surfaces of the biplane depends on the database under consideration.

4.2.2 Effect of Scale

Another factor affecting the distinction of surfaces is the scale (size of the region) covered by each spherical shape descriptor. Figure 4.4 compares mesh distinction computed for a model of a dog with respect to other quadrupeds with shape descriptors covering 0.25, 0.5, 1.0, and 2.0 of the mesh radius. As the scale of the shape descriptors vary, the distinctive regions vary. At the smallest scale, the head is the most distinctive region, while at the largest scale, the most distinctive region is centered on the front feet. Compared to other quadrupeds in this database, at a small scale, the head is the most distinguishing local feature. At larger scales, the aspect ratio of dogs versus taller animals such as horses causes an extremity to be the center of the most distinguishing region. This result is typical, smaller scales usually choose a region with a small part having a distinctive shape, while larger scales usually choose an extremity that provides a distinctive center point for describing the global shape of the mesh.

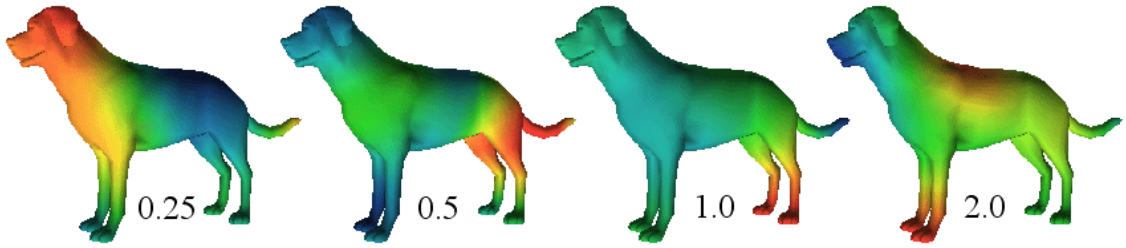


Figure 4.4: As the scale of the shape descriptor increases, different surfaces become distinctive.

These images highlight that distinction is dependent on the scale selected, and we have specifically preserved these differences as compared to combining distinction in a multiscale method as was calculated for saliency [77]. For shape matching purposes, descriptors can be calculated at multiple scales, so it is natural to focus a matching technique on distinctive regions at the appropriate scale, which we explore in Chapter 5.

4.2.3 Alternatives to Distinction

We next investigate how well distinction scores correspond to alternative importance measures across an entire database of shapes. Instead of using shape distinction, there are possibly other techniques for selecting important regions on a shape by focusing on properties that are intrinsic to the shape.

Distance: Surfaces of a shape near the center of mass or near an extremity may represent important regions. We have noticed examples such as the biplane in Figure 3.1 where positions on the extremity have high distinction, which motivates this investigation.

Surface Area: The amount of surface area enclosed within each region varies across the shape depending on the curvature of the shape and scale of the descriptor. We might expect that regions that include a large amount of surface area are more distinctive, while regions that mostly enclose empty space are less distinctive.

Likelihood: Previous projects [20, 62] have treated shape descriptors as high dimensional feature vectors and selected the least likely descriptors for matching, so shape descriptor likelihood is possibly a good indicator of distinction. We assumed a multivariate Gaussian distribution with mean and covariance measured from the descriptors to calculate likelihood.

Saliency: Shape saliency finds the regions of shapes that stick out and are important for visual representation [77], so we considered saliency as a property similar to distinction. Saliency scores were calculated by the saliency.exe program (provided by Chang Lee [77]) on the vertices of a mesh, and saliency scores were interpolated to the centers of the regions.

We compared each of these techniques to distinction for the 907 test models of the Princeton Shape Benchmark across all descriptor scales. We created 256 regions on each shape, created shape descriptors at multiple scales, and calculated distinction for each descriptor. We also calculated for each position the distance from the center of mass of the shape, the amount of surface area (for regions of each scale), the likelihood based on a Gaussian distribution, and the saliency score[1]. We then calculated the correlation score r [105] comparing distinction values to each alternative technique at each scale independently:

$$r = \frac{1}{(n-1)\sigma_x \sigma_{D(M_j,s_k)}} \sum_{i=1}^{n} (x_i - \overline{x})(D(p_{i,j}s_k) - \overline{D(M_j,s_k)})$$

[1]The saliency.exe program was not able to process one model, likely because it was a highly broken mesh with many disconnected polygons.

The term x_i is one of the alternative techniques calculated at position $p_{i,j}$ of mesh M_j, and $\overline{D(M_j, s_k)}$ is the average distinction score over the n regions of M_j at scale s_k. We found that, in all cases, the correlation score was between -0.04 and 0.07, where scores closer to either -1 or 1 indicate a linear relationship (negative or positive respectively), and values close to zero indicate little or no association. These correlation scores for the 1.0 scale indicate that neither Distance ($r = -0.04$), Surface Area ($r = 0.07$), Likelihood ($r = 0.04$), nor Saliency ($r = 0.03$) correlates well to distinction.

While this study only considers a linear relationship between distinction and other properties, it is clear that each property is unable to consistently predict which shape surfaces match within a class and to distinguish shapes from different classes.

4.3 Conclusion

In summary, we have defined distinctive regions of a 3D surface to be those whose shapes provide the best retrieval performance when matched against other shapes in a database of objects partitioned into classes. This definition produces measures of distinction that adjust to the types of classes in the database and provides shape information at multiple scales. For a number of examples, we have shown that the most distinctive parts are consistent within a class and sometimes correspond to identifiable parts of a surface.

There are several limitations of our approach. First, the retrieval measure (DCG) used in our implementation is slow to compute. Even with approximation techniques and index structures, analyzing a database with thousands of meshes may take hours, while calculating saliency takes 4.3 seconds per model and less than a second is required to calculate likelihood or select descriptors randomly. Since distinction can be calculated during the preprocessing of the database, we believe it is a worthwhile step to dramatically improve retrieval performance.

Second, our implementation has focused on distinction in 3D surface shape. While this is well-motivated for some applications such as shape matching, perhaps other approaches based on distinction in 2D images of 3D shapes would be better for others (e.g. visualization applications). We believe that the principles outlined in this dissertation can help guide further work in determining and utilizing the important regions of objects.

Chapter 5

Matching with Distinction

Introduction

One of the main advantages of shape distinction is to guide the process of matching shapes represented by local descriptors. Before a query shape is even presented to a retrieval system, descriptors for target shapes in a database can be filtered by selecting the most distinctive. Even with an importance function based on shape distinction, there will be numerous possible feature correspondences that must be investigated. Checking all possible sets of correspondences will take time exponential in the size of the set, which is too slow for many retrieval applications.

In this chapter, we introduce a priority-driven algorithm for searching all objects in a database at once by focusing on distinctive regions (Funkhouser and Shilane [37]). The algorithm is given a query object and a database of target objects, all represented by sets of local shape features, and its goal is to produce a ranked list of the best target objects sorted by how well any subset of k features on the query match features on the target object. To achieve this goal, the system maintains a priority queue of potential sets of feature correspondences (partial matches) sorted by a cost function accounting for both feature dissimilarity and geometric deformation. Initially, all pairwise correspondences between the features of the query and features of target objects are loaded onto the priority queue. Then, at every step, the best partial match m is popped off the priority queue, new partial matches are created by extending m to include compatible feature correspondences, and those new partial matches are added to the priority queue. This process is iterated until

the desired number of full matches with *k* feature correspondences have been popped off the priority queue.

The advantage of this approach is that the algorithm provably finds the optimal set of matches over the entire database while investigating only a small subset of the potential matches. Like any priority-driven backtracking search (e.g., Dijkstra's shortest path algorithm), the algorithm considers only the partial matches that can possibly lead to the lowest cost match (Figure 5.1). Although some poor partial matches are generated, they never rise to the top of the priority queue, and thus they incur little computational overhead. By using a single priority queue to store partial matches for all objects in the database at once, we achieve great speedups when retrieving only the top matches – if a small set of target objects match the query well, their feature correspondences will be discovered quickly, and the details of other potential matches will be left unexplored. This approach largely avoids the combinatorial explosion of searching for multi-feature matches in dissimilar objects.

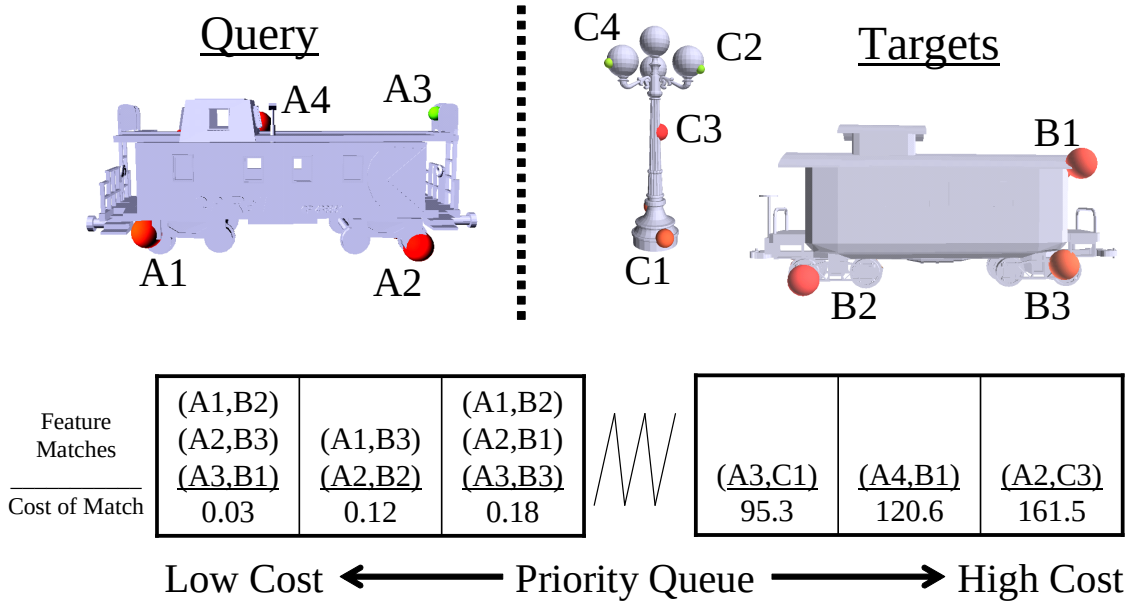


Figure 5.1: Priority driven search: a priority queue (bottom) stores potential matches of features (labeled dots) on a query to features of all target objects at once. Matches are extended only when they reach the top of the priority queue (the leftmost entry), and thus high cost feature correspondences sit deep in the priority queue and incur little computational expense.

This chapter makes several research contributions. In addition to the idea of priority-driven search, we explore ways of improving computational efficiency and retrieval performance of multi-feature matching algorithms: 1) we use ranks rather than L_2 differences to measure feature similarity; 2) we use surface distinction to select features; and, 3) we match features at multiple scales. Finally, we provide a working shape-based retrieval system and analyze its performance over a wide range of options and parameter settings. We find that our system provides significantly better retrieval performance than previous shape matching approaches on the Princeton Shape Benchmark while using increased, but reasonable, processing and storage costs.

The organization of this chapter is as follows. Section 5.1 contains an overview of the priority-driven search algorithm followed by a detailed description for every algorithmic step. Section 5.2 compares the performance of the priority-driven search approach to other state-of-the-art shape matching methods and investigates how modifying several aspects of the algorithm impacts its performance. Finally, Section 5.3 provides a brief discussion of conclusions and limitations.

5.1 System Execution

Execution of our system proceeds in two phases: a preprocessing phase and a query phase. We provide an overview of both phases before explaining each in greater detail.

During the preprocessing phase, we build a multi-feature representation of every object in the database. First, we generate for each object a set of spherical regions covering its surface at different scales and compute a descriptor of the shape within that region. Second, we compute differences between all pairs of descriptors at the same scale and associate with every descriptor a mapping from rank to difference. Finally, we select a subset of features to represent each object based on how distinctive they are of their object class. The result of this preprocessing is a set of "shape features" (or "features," for short) for every object, each with an associated position (p), normal ($\vec{n}$), radius (r), and shape descriptor (a feature vector of numbers representing a local region of shape), and a description of how discriminating its shape descriptor is with respect to others in the database.

For every query, our matching procedure proceeds as shown in Figure 5.2. The inputs are: 1) a query object, *query*, 2) a database of target objects, *db*, each represented by a set of shape features, 3) a cost function, *cost*, measuring the quality of a proposed set of feature correspondences, 4) a constant, *k*, indicating the number of feature correspondences that should be found for a complete match, and 5) a constant, *c*, indicating the number of objects for which to retrieve optimal matches. The output is a list of the best matching target objects, *M*, along with a description of the feature correspondences and cost for each one.

```
PriorityDrivenSearch(Object query, Database db,
    Function cost, int k, int c)
  # Create correspondences
  foreach Object target in db
    foreach Feature q in query
      foreach Feature t in target
        p = CreatePairwiseCorrespondence(q, t, cost)
        if (IsPlausible(p))
          AddToPriorityQueue(Q, p)
          AddToList(C[target], p)
          if (cost(p) < cost(M[target]))
            M[target] = p

  # Expand matches until find complete ones
  complete_match_count = 0
  while (complete_match_count < c)
    # Pop match off priority queue
    m = PopBestMatch(Q)
    target = GetTargetObject(m)

    # Check for complete match
    if (IsMatchComplete(m, k))
      RemoveMatchesFromPriorityQueue(Q, target)
      complete_match_count++
      continue;

    # Extend match
    foreach PairwiseCorrespondence p in C[target]
      m' = ExtendMatch(m, p, cost)
      if (IsPlausible(m'))
        AddToPriorityQueue(Q, m')
        if (cost(m') < cost(M[target]))
          M[target] = m'

  # Return result
  return M
```

Figure 5.2: Pseudo-code for priority-driven search.

Initially, a priority queue, Q, is created to store partial matches, and an array, M, is created to store the best match to every target object. Then, all pairwise correspondences between the features of the query and features of the target objects are created, stored in lists associated with the target objects, and loaded onto the priority queue. The priority queue then holds all possible matches of size 1. Then, until c complete matches have been found, the best partial match, m, is popped off the priority queue. If it is a complete match (i.e., the number of feature correspondences satisfies k), then the search of that target object is complete, and the priority queue is cleared of partial matches to that object. Otherwise, for every feature correspondence between the query and the target of m, the match is extended by one feature correspondence to form a new match, m'. The best match for every target object is retained in an array, M, when it is added to the priority queue. This process is iterated until at least c full matches with k feature correspondences have been popped off the priority queue for c distinct target objects, and the array of the best matches to every target object, M, is returned as the result.

The computational savings of this procedure come from two sources. First, matches are considered from best to worst, and thus, poor pairwise correspondences are never considered for extension and add little to the execution time of the algorithm. Second, after complete matches for at least c target objects have been added to the priority queue, it is possible to determine an upper-bound on the cost of matches that can plausibly lead to one of the best matches. If the score computed for an extended match, m', is higher than that upper bound, then there is no reason to add it to the queue, and it can be ignored. Similarly, if a match, m, is popped off the queue, then it is provably the best remaining match – i.e., no future match can be considered with a lower cost. Thus, the algorithm can terminate early (immediately after c best matches have been popped off the priority queue) while still guaranteeing an optimal solution.

Of course, there are many design decisions that impact the efficacy of this search procedure, including how shape descriptors are computed (Section 5.1.1), selecting distinctive descriptors from target shapes (Section 5.1.2), what cost function is used (Section 5.1.3), how implausible matches are culled (Section 5.1.4), and so on. The following subsections describe our design decisions in detail, and Section 5.2 provides the results of experiments aimed at evaluating the impact of each one on search speed and retrieval performance.

5.1.1 Computing Shape Descriptors

The first step of our system is to generate local regions for a shape and construct shape descriptors, which is identical to the procedure described in Section 4.1.2. We randomly select point samples on the surface of each mesh and consider spherical regions centered at those positions across multiple scales. In our matching experiments, we consider four scales ($0.25, 0.5, 1.0, and 2.0$ times the shape's radius). Each region is represented by a shape descriptor that provides a compact feature vector representation that can be efficiently calculated and compared. Most of our experiments compare three related descriptors: the Shells Descriptor (SD), Harmonic Shape Descriptor (HSD), and Fourier Shape Descriptor (FSD).

5.1.2 Selecting Distinctive Features

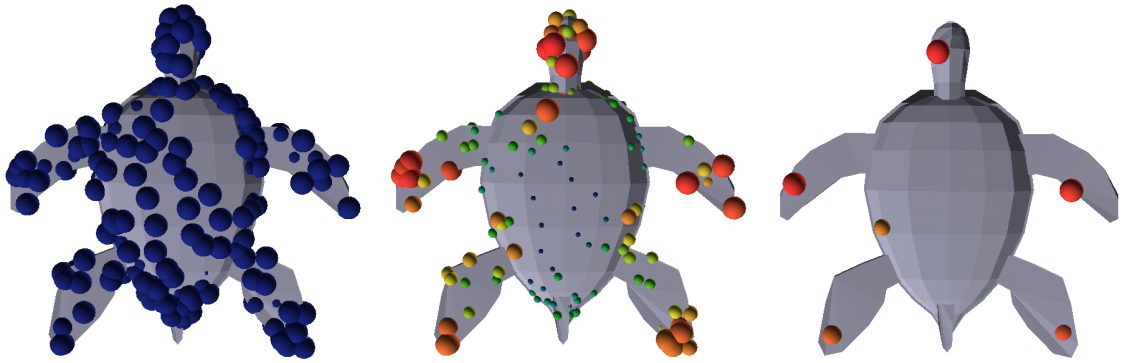
The next step of our system for preprocessing target models has two phases. In the first phase, we calculate for each descriptor a measure of how well it matches shapes of the same class versus shapes of other classes, which we call a rank-to-difference mapping. In the second phase, we filter the set of descriptors to a small set that are most distinctive.

Selecting a subset of local shape descriptors is a well known technique for speeding up retrieval, and several researchers have proposed different methods for this task. The simplest technique is to select features randomly [34, 63, 89]. Other methods have considered selecting features based on surface curvature [132], saliency [40], likelihood within the same shape [42, 63], persistence across scales [42], and number of matches to another shape [111].

In the first phase, for every feature, we compute the L_2 difference of its shape descriptor to the best match of every other object in the database, sort the differences from best to worst, and save them in a *rank-to-difference mapping* (RTD). To save space, we store an approximation to the RTD containing log(N) values by sampling distances at exponentially larger ranks. We then use the RTD to estimate the distinction (DCG) of every shape feature.

In the second phase, we employ a greedy algorithm to select a small set of features to represent every target object (Figure 5.3). The selection algorithm iteratively chooses the feature with highest DCG whose position is not closer than a Euclidean distance

threshold, *minlength*, to the position of any previously selected feature. This process avoids selecting features nearby each other on the mesh and provides an easy way to vary the subset size by adjusting the distance threshold.



(a) All Features (b) Feature Distinction (c) Selected Features

Figure 5.3: Feature selection: (a) positions sampled randomly on surface, (b) computed DCG values used to represent feature distinction (red is highest, blue is lowest), and (c) features selected to represent the object during matching.

The net result of this process is a small set of features for every target object, each with an associated position (p), normal ($\vec{n}$), radius (r), a set of shape descriptors (SD, HSD, and FSD), a rank-to-difference-mapping (RTD), and a retrieval performance score (DCG). The storage for the resulting data required at query time is approximately 100 KB per object.

5.1.3 Creating Pairwise Feature Correspondences

When given a query object to match to a database of target objects, the first step is to compute the cost of pairwise correspondences between features of the query to features of the target. The key to this step is to develop a cost function that provides low values only when two features are compatible and gradually penalizes pairs that are less similar. The simplest and most common approach is to use the L_2 difference between their associated shape descriptors. This approach forms the basis for our implementation, but we augment it in three ways.

First, given features F_1 and F_2, we compute the L_2 difference, D, between their shape descriptors. Then, we use the rank-to-difference mappings (RTD) of each feature to convert D into a rank (i.e., where that distance falls in the ranked list associated with each feature). The new difference measure (C_{rank}) is the sum of the ranks computed for F_1 with respect to the RTD of F_2, and vice versa:

$$C_{rank} = Rank(RTD_1, D) + Rank(RTD_2, D)$$

This feature rank cost (which we believe is novel) avoids the problem that very common features (e.g., flat planar regions) can provide false positive matches when L_2 differences are small. Our approach considers not the absolute difference between two features, but rather their difference relative to the best matching features of other objects in the database. Thus, a pair of features will only be considered similar if both rank highly in the retrieval list of the other.

Second, we augment the cost function with geometric terms. For part-in-whole object matching, we can take advantage of the fact that features are more likely to be in correspondence if they appear at the same relative position and orientation with respect to the rest of their objects. Thus, for each feature, we compute the distance between its position and the center of mass of its object (R), scaled by the average of R for all features in the object ($RAVG$), and we add a distance term C_{radius} to the cost function accounting for the difference between these distances:

$$C_{radius} = |\frac{R_1}{RAVG_1} - \frac{R_2}{RAVG_2}|$$

We also compute a normalized vector $\vec{r}$ from the object's center of mass to the position of each feature and store the dot product of that vector with the surface normal ($\vec{n}$) associated with the feature. The absolute value of the dot product is taken to account for the possibility of backfacing surface normals. Then, the difference between dot products for any pair of features is used to form a normal consistency term to the cost function:

$$C_{normal} = ||\vec{r_1} \cdot \vec{n_1}| - |\vec{r_2} \cdot \vec{n_2}||$$

Overall, the cost of a feature correspondence is a simple function of these three terms:

$$C_{correspondence} = \alpha_{rank} C_{rank}^{\gamma_{rank}} + \alpha_{radius} C_{radius}^{\gamma_{radius}} + \alpha_{normal} C_{normal}^{\gamma_{normal}}$$

The α coefficients and γ exponents are used to normalize and weight the terms with respect to each other.

Of course, computing all potential pairwise feature correspondences between a query object and a database of targets is very costly. If the query has M_Q features and each of N targets has M_T selected features, then the total number of potential feature correspondences is $N \times M_Q \times M_T$. To accelerate this process, we utilize conservative thresholds on each of the three terms (*maxrank*, *maxradius*, *maxnormal*) to throw away obviously poor feature correspondences. The terms are computed and the thresholds are checked progressively in order of how expensive they are to compute (e.g., C_{rank} is last), and thus there is great opportunity for trivial rejection of poor matches with little computation. Indexing and progressive refinement could further reduce the compute time as described in Chapter 10.

5.1.4 Searching for the Optimal Multi-Feature Match

The third step of the query process is to search for the best multi-feature matches between the query object and the target objects. This is the main step of priority-driven search.

A priority queue is used to store incomplete sets of feature matches during a backtracking search. Initially, all pairwise correspondences (computed as described in the previous subsection) are loaded onto the priority queue. Then, the best partial match, *m*, is repeatedly popped off the priority queue, and then extended matches are created for every compatible feature correspondence and loaded onto the priority queue. This process is iterated until at least *c* full matches with *k* feature correspondences have been popped off the priority queue for distinct target objects.

As a partial match is extended to include one more feature correspondence, two extra terms are added to the cost function to account for geometric deformations implied by multiple pairwise feature correspondences (Figure 5.4). First, a chord length term C_{length} is added to penalize matches with inconsistent inter-feature lengths. Specifically, for every pair of feature correspondences in *m*, we compute the length of the chord between

feature positions in the same object (L), scaled by the average of L over all features in the object ($\overline{L}$). Then, we compute the difference between these distances and normalize by the greater of the two to produce the length term of the cost function:

$$C_{length} = \frac{|\frac{L_1}{\overline{L}_1} - \frac{L_2}{\overline{L}_2}|}{max(\frac{L_1}{\overline{L}_1}, \frac{L_2}{\overline{L}_2})}$$

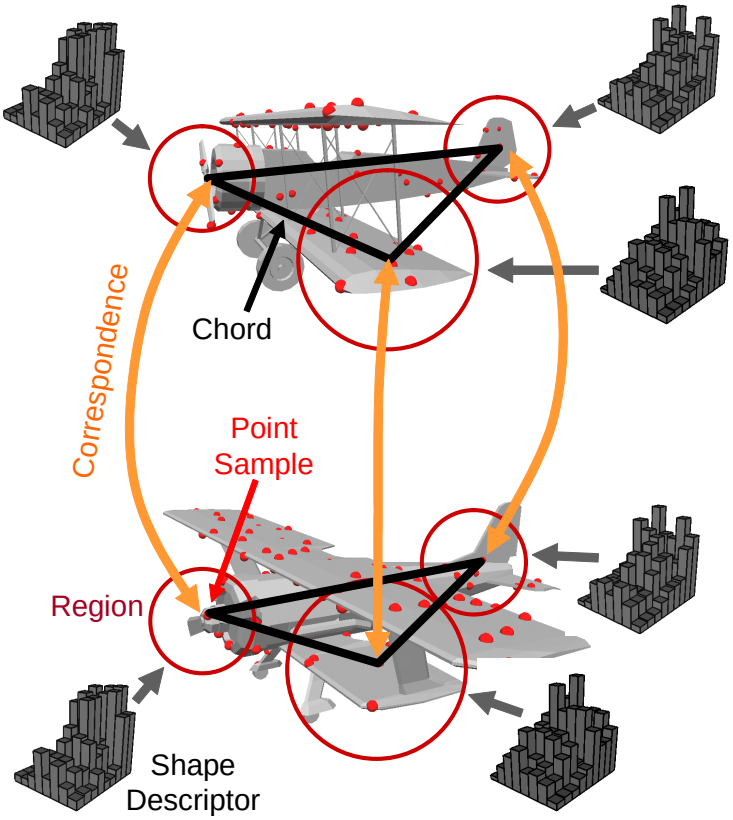


Figure 5.4: A 3-feature match for two airplanes. Red points represent feature positions on the surface. For three features, red circles represent regions, gray histograms represent shape descriptors, orange lines represent feature correspondences, and black lines represent lengths between features of same object. Consistency of all shape descriptors, lengths, and angles is required for a good match.

Second, a surface orientation term is added to penalize matches with pairs of feature correspondences whose surface normals are inconsistent. This term penalizes both mismatches in the relative orientations of the two pairs of normals with respect to one another and mismatches in the orientations of the normals with respect to the chord between the features. If $\vec{v}_1$ is the normalized vector between features 1a and 1b with normals $\vec{n_{1a}}$ and

$\vec{n_{1b}}$ in object 1, and similar variables describe the relative orientations of features in object 2, then the orientation term of the cost function can be computed as follows:

$$\begin{aligned} C_{orient} = \quad & ||\vec{n_{1a}} \cdot \vec{n_{1b}}| - |\vec{n_{2a}} \cdot \vec{n_{2b}}|| + \\ & ||\vec{v_1} \cdot \vec{n_{1a}}| - |\vec{v_2} \cdot \vec{n_{2a}}|| + \\ & ||\vec{v_1} \cdot \vec{n_{1b}}| - |\vec{v_2} \cdot \vec{n_{2b}}|| \end{aligned}$$

These terms are also weighted and raised to exponents to provide normalization when added to the overall scoring function computed for a match with k feature correspondences:

$$C_{chord} = \alpha_{length} C_{length}^{\gamma_{length}} + \alpha_{orient} C_{orient}^{\gamma_{orient}}$$

As in the previous section, we utilize conservative thresholds on C_{length} and C_{orient} (*maxlength* and *maxorientation*) to throw away obviously poor feature correspondences. We also utilize a threshold on the minimum distance between features within the same object (*minlength*) in order to avoid matches comprised of features in close proximity to one another.

The overall cost of a match is the sum of the terms representing differences in the k feature correspondences and the geometric differences between the k(k-1)/2 chords spanning pairs of features:

$$C_{match} = \sum_{i<k} C_{correspondence}(i) + \sum_{i<j<k} C_{chord}(i,j)$$

5.2 Results

In this section, we present results of experiments with priority-driven search. We investigate the performance of the method in relation to the state of the art in shape-based retrieval and investigate the impact of several design choices on the speed and quality of retrieval results. Using our priority-driven search algorithm, we compare focusing a search algorithm on distinctive regions of shapes versus regions selected with other techniques.

In a representative preprocessing phase, we generate features at 128 surface points with 4 different scales for every object. For every feature, we compute its shape descrip-

tors, RTDs, and DCGs, and then we select the most distinctive set of descriptors. The total preprocessing time for all 907 objects is 70 hours and the total size of all data generated is 1GB, of which 64MB represents the selected features that are stored in memory for target objects during the query phase.

During the query phase, we perform a series of "leave-one-out" classification tests with the Princeton Shape Benchmark. In each test, every object of the database is used as a *query object* to search databases containing the remaining $N-1$ target objects. Standard information retrieval metrics, such as precision, recall, nearest neighbor classification rate (1-NN), first-tier percentage (1-tier), second-tier percentage (2-tier), and discounted cumulative gain (DCG), are computed to measure how many objects in the query's class appear near the top of its ranked retrieval list, and those metrics are averaged for all queries.

Unless otherwise stated, experiments were run on Linux server with a x86_64 processor running at 2.2 GHz and with 12 GB of memory. Parameters for the "base configuration" of the system were set as follows: c = 1, k = 3, number of features per object = 128, number of feature scales = 4 (0.25, 0.5, 1.0, and 2.0), shape descriptor type = HSD, compression ratio = 10X, $maxradius = maxnormal = maxlength = maxorientation = 0.25$, $minlength = 0.3 \cdot RAVG$, α_{rank} = 0.01, $\alpha_{radius} = \alpha_{normal} = \alpha_{length} = \alpha_{orient}$ = 1, and γ_{rank} = 4, $\gamma_{radius} = \gamma_{normal} = \gamma_{length} = \gamma_{orient}$ = 2. These parameters were determined empirically and used for all experiments without adjustment, except in Section 5.2.1 where the FSD shape descriptor was used, and in Section 5.2.3 where the impact of specific parameter settings was studied.

5.2.1 Comparison to Previous Methods

The goal of the first experiment is to evaluate the retrieval performance of the proposed priority-driven search (PDS) approach with respect to previous state-of-the-art shape-based retrieval methods:

Depth Buffer Descriptor (DSR740B) This shape descriptor achieved the highest retrieval performance in the study of Bustos et al. [17] and our own study with the PSB. It describes an object by six depth buffer images captured from orthogonal parallel projections [48]. Images are stored as Fourier coefficients of the lowest frequencies, and differences between Fourier coefficients provide a measure of

object dissimilarity. We use Dejan Vranic's implementation of this method [129] without modification and ran it on a 2 GHz Pentium 4 running Windows XP.

Light Field Descriptor (LFD) It represents an object as a collection of images rendered from uniformly sampled positions on a view sphere [19]. The dissimilarity of two objects is defined as the minimum L_1-difference between aligned images of the light field, taken over all rotations and all pairings of vertices on two dodecahedra. We use the original implementation provided by Chen et al. without modification and ran it on a 2 GHz Pentium 4 running Windows XP. This shape descriptor achieved among the highest retrieval performance on the Princeton Shape Benchmark.

Global Harmonic Shape Descriptor (GHSD) The GHSD is the shape descriptor currently used in the Princeton 3D Search Engine [36]. It describes an object by a single HSD feature positioned at the center of mass and scaled to include the entire mesh. We include it in this study to provide an apples-to-apples comparison to a method that matches a single global shape descriptor of the same type used in our study.

Random A random retrieval list is created for every query as a baseline for retrieval performance.

Figure 5.5 shows a precision-recall plot comparing the average retrieval performance for all queries for each of these shape matching methods. Briefly, precision and recall are metrics used to evaluate ranked retrieval lists. If one considers the top M matches for any query, *recall* measures the fraction of the query's class found, and *precision* measures the fraction of objects found from the query's class – higher curves represent better retrieval performance. The Random line provides a baseline for comparison, and the increase around Recall = 0.1 relates to averaging the results of different class sizes together.

Timing statistics and standard retrieval performance measures are shown in Table 5.1. The leftmost column indicates the shape matching method (PDS is the one described in this chapter). The remaining columns list the average time required for one query into the database (in seconds) and various retrieval measures.

From these statistics, we see that the priority-driven search algorithm provides the best retrieval performance of the tested methods on this data set. The improvement in nearest

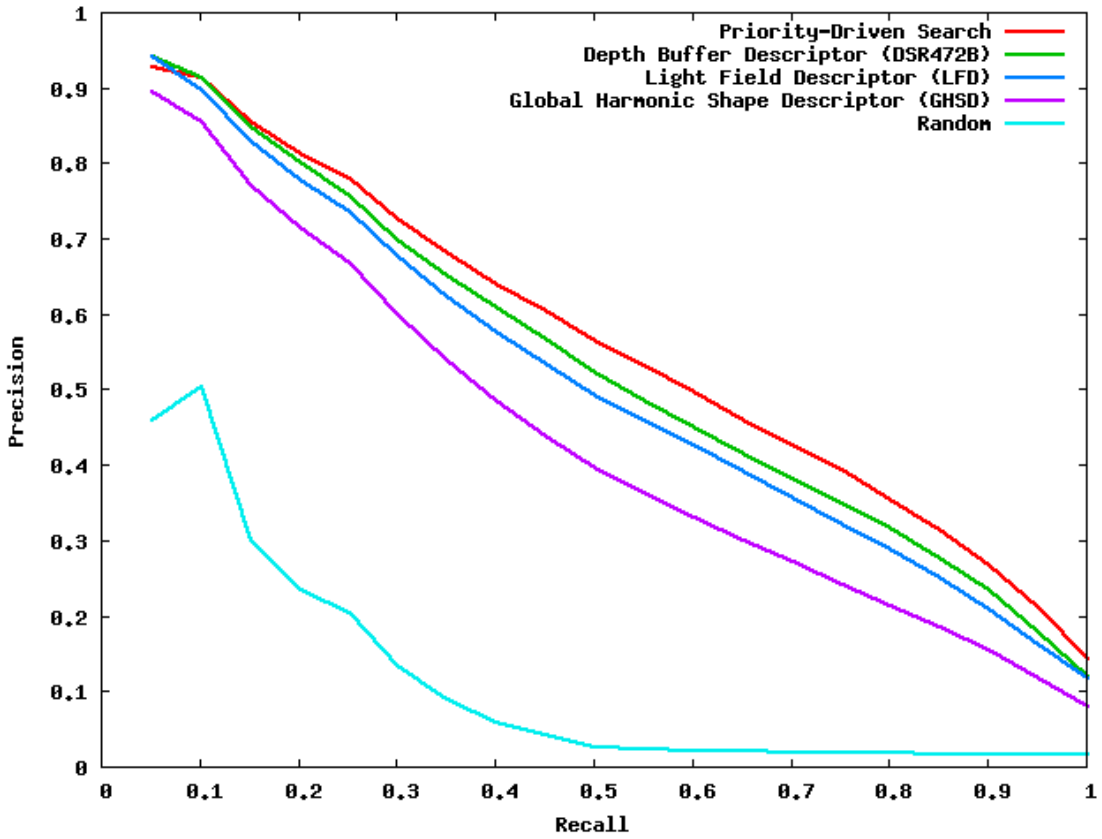


Figure 5.5: Precision-recall plot comparing priority-driven search (PDS) to other state-of-the-art shape matching methods using the Princeton Shape Benchmark.

Method	Time	1-NN	1-Tier	2-Tier	E-Meas.	DCG
PDS	2.4	69.2	43.5	55.7	31.3	68.7
DSR740B	0.005	66.5	40.3	51.2	29.5	66.3
LFD	-	65.0	37.2	47.4	27.1	63.6
GHSD	0.003	55.6	30.9	41.1	24.0	58.4
Random	0	1.7	1.6	3.4	2.2	26.1

Table 5.1: Comparison of retrieval statistics between priority-driven search (PDS) and other methods on the Princeton Shape Benchmark. (Timing results are in seconds.)

neighbor classification rate over the Depth Buffer Descriptor is 4% (69.2% vs. 66.5%) and the improvement over the Light Field Descriptor is 6.4% (69.2% vs. 65.0%).

However, the PDS algorithm takes considerably more compute time to preprocess the database (4-5 minutes per object), more memory per object (100 KB per target object), and more time to find matches (2.4 seconds per query) than the other tested shape descriptors. Almost all of the query processing time is spent establishing the

cost of feature correspondences, and less than a tenth of a second is spent finding the optimal multi-feature match with priority driven search. Thus, we believe that simple improvements to the basic algorithm (e.g., compression, indexing, etc.) will significantly improve the processing speed and that query processing times less than a second are possible in this framework (Section 5.3).

In any case, it seems that priority-driven search is well-suited for batch applications where retrieval accuracy is premium. Often, query results can be computed off-line and cached for later interactive analysis – e.g., for discovery of relationships in mechanical CAD, molecular biology, etc. Even interactive search engines can benefit from off-line preprocessing with high-accuracy matching methods, for example, to preprocess queries that find a shape similar to another in the database (over 90% of the 3D queries to the Princeton 3D Search Engine are of this type [87]).

5.2.2 Evaluation of Algorithmic Contributions

The goal of the second experiment is to understand which algorithmic features of the priority-driven search algorithm contribute most to its timing and retrieval performance. To study this question, we started with the "base configuration" and ran the system multiple times on the Princeton Shape Benchmark with different aspects of the system enabled and disabled.

Distinction (D) If enabled, a small subset of features (~ 7) was selected for matching within every target object, as described in Section 5.1.2. Otherwise, all features were included within the target objects.

Rank (R) If enabled, the cost of two corresponding shape descriptors (C_{rank}) was the sum of the two ranks in their respective retrieval lists, as described in Section 5.1.3. Otherwise, it was the direct L_2 distance between shape descriptors (the most common measure of descriptor difference in other systems).

Multi-Scale (S) If enabled, the costs of the best matches found at all four scales were summed. Otherwise, the cost of the best match found among features at scale 1.0 was used (the scale that gave the best retrieval performance on its own).

Results of this experiment are shown in Table 5.2 and Figure 5.6. The first three columns of Table 5.2 indicate whether each of the three algorithmic features (R, M,

and D) are enabled (Y) or disabled (N), and the remaining columns provide retrieval performance statistics (note that the top row shows the performance statistics for PDS with all its algorithmic features enabled: Y Y Y).

D	R	S	1-NN	1-Tier	2-Tier	E-Meas.	DCG
Y	Y	Y	62.8	40.2	51.7	29.7	65.6
N	Y	Y	66.6	37.2	48.7	28.0	64.4
Y	N	Y	60.3	33.2	43.5	25.8	60.5
Y	Y	N	57.0	33.3	44.3	25.8	59.1
N	N	Y	63.4	32.7	42.6	25.0	60.6
N	Y	N	60.7	31.6	42.6	24.8	59.0
Y	N	N	51.0	28.7	39.7	23.0	55.7
N	N	N	57.1	28.5	38.7	22.8	56.4

Table 5.2: Results of experiments to investigate the individual and combined value of three algorithmic features of priority-driven search (PDS). The top row represents the base PDS algorithm (Y Y Y). Other rows represent variants of the algorithms with three algorithmic features (D = distinctive, R = rank, and S = multi-scale feature selection) enabled (Y) or disabled (N). Differences in the results achieved with these variants provide insights into which aspects of the PDS algorithm contribute most to its results.

From these results, we see that the retrieval performance of our system comes from several sources. That is, all three algorithmic features tested contribute a modest but significant improvement to the overall result. Specifically, if we consider the incremental improvements in nearest neighbor classification rates (1-NN) of the combinations shown in Figure 5.6, we find that using descriptor ranks rather than L_2 differences provides a 6% improvement (60.7% vs. 57.1%) and using multi-scale features further boosts performance by another 16% (66.6% vs 57.0%). Selecting distinctive features of target objects degrades performance slightly when used alone, especially for the NN metric, but has an overall positive benefit when used in combination with the other algorithmic features. This is most noticeable with the DCG score increasing 16% (65.6% vs. 56.4%), though the NN score when using distinction features decreases in several examples. This is likely because the NN metric is sensitive to the first retrieval result, and our measure of distinction (DCG) is based on evaluating the entire retrieval list.

With respect to timing, the main expense of the priority driven search implementation is establishing the initial set of pairwise feature correspondences ($\sim$0.3 seconds per query per scale). By comparison, the time required to search for the best multi-feature match is negligible ($<$0.1 seconds). So, the timing results are currently dominated by the number

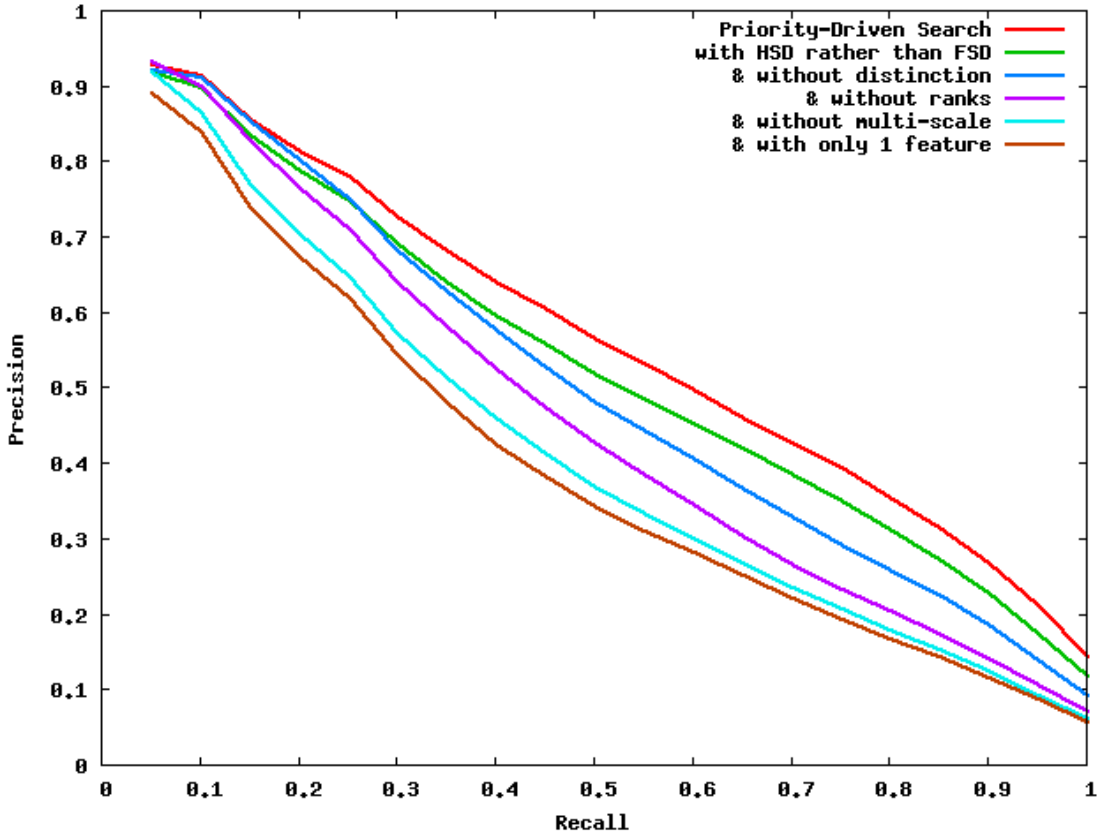


Figure 5.6: Precision-recall plot showing the relative contributions of different algorithmic features of priority-driven search. The top curve (red) shows the retrieval performance of the best performing set of options for the PDS algorithm (it is the same as the red curve in Figure 5.5). The second curve (green) shows the result of using HSDs rather than FSDs as shape descriptors (it represents the "base configuration" for the study in Section 5.2.3). The third curve (blue) shows the results without selecting a subset of distinctive features on target objects; the fourth curve (magenta) shows the same, but using L_2 differences instead of ranks to measure feature correspondence costs; the next-to-bottom curve (cyan) also disables multi-scale feature matching (all features are matched only at scale 1.0); and the bottom curve (brown) shows the results when finding only one point per match rather than 3. Note how the retrieval performance degrades significantly when each of these algorithmic features is disabled.

of features considered for each target object and the number of scales considered for each feature.

Overall, we find that choosing distinctive features (D) improves both precision and speed significantly; using ranks rather than L_2 differences (R) improves precision with

negligible extra compute time; and using features at four scales (S) improves precision, but incurs four times the computational expense.

5.2.3 Investigation of Parameter Settings

The goal of the third experiment is to investigate in detail how various options of the priority-driven search system affect the timing and retrieval performance. Of course, there is a large space of possible options, and thus we are forced to focus our discussion on small "slices" through this space. Our approach is to center our investigation on the "base configuration" set of options described in the beginning of this section and to study how timing and retrieval statistics are affected independently as one option is varied at a time.

The results of this study are shown in Table 5.3(a-d) – each table studies the impact of a different option, and different rows represent a different setting for that option. Please note that rows marked with an '*' represent the same data – they provide results for the base configuration through which slices of option space are being studied.

Impact of shape descriptor type (Table 5.3(a)) More verbose descriptors generally provide better retrieval performance, albeit at higher storage and compute costs. For example, the Fourier Shape Descriptor (FSD) provides better nearest neighbor classification rates (69.2%) than the Harmonic shape descriptor (HSD) (62.8%). However, it is also eight times bigger, and thus eight times more expensive to compare. There is a further decrease in retrieval performance and improvement in comparison time when using the SD shape descriptor. Further study is required to determine which descriptors provide the best "bang for the buck" for specific applications and how multiple descriptors can be combined to provide the accuracy of the most verbose ones while incurring query times of the smaller ones (Section 5.3).

Impact of feature scale (Table 5.3(b)) Medium scale features (radius = 0.5-1.0) provide better retrieval performance than small and large scales in this test, and multi-scale features perform the best of all (nearest neighbor classification rates are 62.8% with multi-scale versus 57.2% with the best single scale (0.5)). Interestingly, summing the cost functions computed for matches at all four scales separately ("Multi-scale") provides better retrieval performance than matching features at all scales simultaneously ("All"). The difference is that the same set of features must match at all 4 scales in "All," while

Descriptor	Time	1-NN	1-Tier	2-Tier	E-Meas.	DCG
SD	1.1	54.2	30.5	40.5	23.8	57.9
HSD *	1.2	62.8	40.2	51.7	29.7	65.6
FSD	2.4	69.2	43.5	55.7	31.3	68.7

(a) Shape descriptor type

Radius	Time	1-NN	1-Tier	2-Tier	E-Meas.	DCG
0.25	0.3	48.1	24.2	33.8	20.0	51.5
0.5	0.3	57.2	30.8	41.5	23.8	57.4
1.0	0.3	57.0	33.3	44.3	25.8	59.1
2.0	0.3	51.3	29.2	39.6	23.0	55.5
Multi-scale *	1.2	62.8	40.2	51.7	29.7	65.6
All	0.6	60.9	33.0	46.2	26.1	61.2

(b) Scales used for matching shape features

# Points	Time	1-NN	1-Tier	2-Tier	E-Meas.	DCG
64	0.6	64.5	38.0	50.0	28.8	64.3
128 *	1.2	62.8	40.2	51.7	29.7	65.6
256	4.0	64.2	40.9	53.2	30.4	66.2
512	17.6	65.5	42.1	54.1	31.0	66.9

(c) Number of sample points per object

k	Time	1-NN	1-Tier	2-Tier	E-Meas.	DCG
1	1.2	61.7	38.7	50.5	29.5	64.3
2	1.2	63.4	39.3	50.9	29.6	64.7
3 *	1.2	62.8	40.2	51.7	29.7	65.6
4	1.2	63.0	40.0	51.8	29.8	65.2
5	1.2	61.4	40.1	51.6	30.0	65.2

(d) Number of feature correspondences per match (k)

Table 5.3: Results of experiments to investigate the impact of several options on the query time (in seconds) and retrieval performance of priority-driven search.

different features can be selected independently for each scale in "Multi-scale." This result seems to suggest that features persistent across multiple scales are not necessarily as useful for classification as ones that are very distinctive at a particular scale.

Impact of the number of sample points per object (Table 5.3(c)) Including more sample points for each object improves retrieval performance in this test, at least up to 512 points. The DCG classification rate is 66.9% for 512 points per object, while it is 66.2% for 256 points, 65.6% for 128 points, and 64.3% for 64 points. Although a small set of distinctive features are ultimately selected for every target object during a preprocess, features centered at all sample points of the query object are candidates for a match, and thus the compute time for each query should be proportional to the number of points (the quadratic growth observed in this experiment is an artifact of our implementation).

Impact of number of feature correspondences (Table 5.3(d)) Matching large numbers of features does not improve retrieval performance in this study. In fact, matching more than 3 features seems to degrade performance. This result may be because features are quite large scale and spread apart, and thus 3 features may describe the shape as well as is possible with the HSD feature representation. Interestingly, matching larger numbers of features also does not increase query times – this is because the priority-driven search algorithm is able to find good matches in time that is largely independent of the number of possible matches – it investigates only the good matches and ignores the rest.

5.2.4 Alternative Selection Techniques

Using the framework of priority-driven search, we can compare filtering local descriptors using distinction relative to other techniques for selecting a small subset of descriptors, such as methods that select regions of a shape in isolation, i.e., without considering the context of a database. We consider three common techniques for filtering descriptors from individual models. Several previous projects [34, 89] randomly selected descriptors on the mesh surface, which is a simple technique. The least likely shape descriptors have been used for matching by [20, 62] under the assumption that rare descriptors correspond to important shape features. "Salient" regions [40, 77, 94], surfaces where the shape sticks out or has variable curvature, have been studied as well. Likelihood was calculated by considering shape descriptors as feature vectors and assuming a multivariate Gaussian distribution calculated from the database, and saliency was calculated using an executable provided by Lee at al. [77].

We compare our technique of using the most distinctive descriptors against the alternatives of selecting descriptors randomly or based on likelihood or saliency. Figure 5.7 shows regions selected on the same helicopter model using the four techniques. The color of the sphere centered on each region indicates the distinction score associated with the region, where red spheres indicate higher distinction scores. Selecting regions based on the likelihood, saliency, or random selection leads to representing meshes with regions that perform poorly in retrieval tasks. A similar number of regions were selected for the helicopter in all cases, but some selected regions are not visible in Figure 5.7 because they appear on the backside of the mesh.

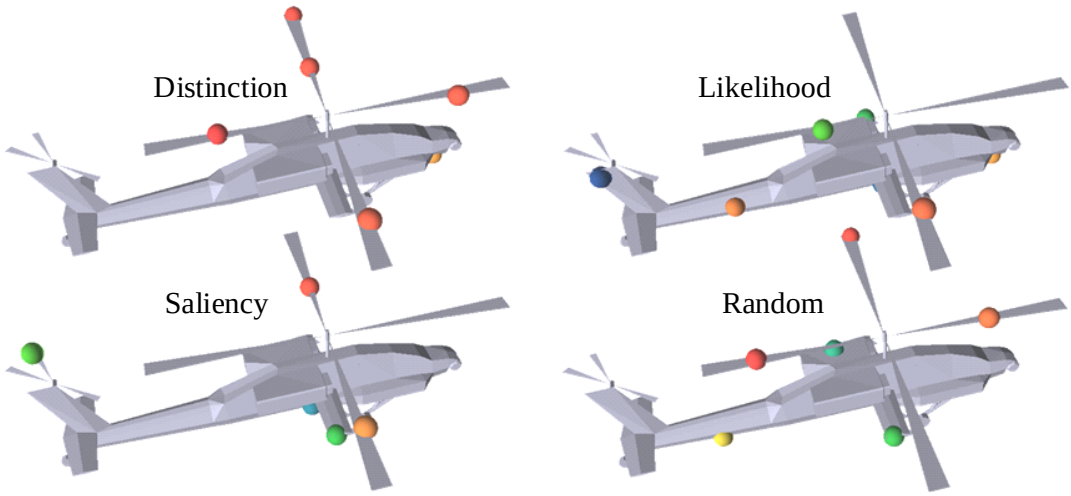


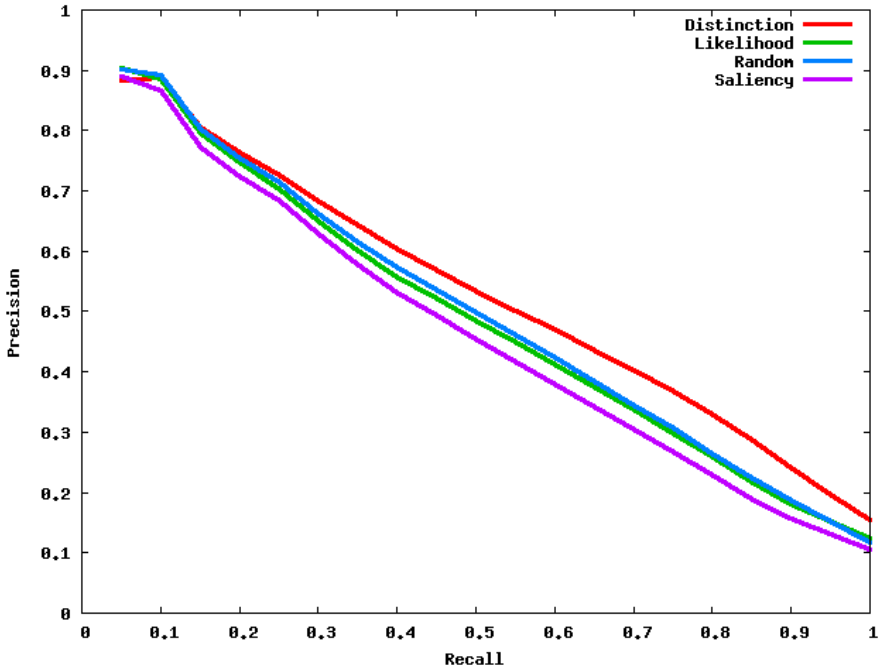
Figure 5.7: Descriptors are selected based on distinction, likelihood, saliency, or are selected randomly. The coloring of the sphere is based on distinction scores, indicating that descriptors with poor distinction scores are selected with the other techniques. A similar number of descriptors are selected for all four techniques, although some appear on the backside of the mesh.

In this study, 128 shape descriptors were used for the query mesh and matched against a subset of the descriptors from every other mesh based on a selection technique. In our experiments, we explored a range of parameter settings for the priority-driven search algorithm and found that the relative performance of the three techniques was consistent across all settings. In the following discussion, we selected parameters that optimized the retrieval performance independently for descriptors selected based on likelihood, saliency, or distinction scores as well as descriptors selected randomly.

Figure 5.8 shows precision-recall plots of retrieval results achieved with the proposed method during a leave-one-out study with the training and test sets of the PSB. Retrieval statistics are also shown in Table 5.4. Column 1 lists the method used for selecting descriptors for retrieval and Column 2 lists the number of descriptors K selected during matching. Columns 3-7 and 8-12 show several measures of retrieval performance, and in all cases, higher scores indicate better retrieval performance.

Looking at both the plots and tables, the first result to notice is that selecting features based on distinction provides better retrieval performance than selecting them based on saliency, likelihood, or at random. When considering multipoint matching (K=3) with all four scales on the PSB Test set, the DCG score for Distinction is 65.6% as compared to

a.) Training set of the PSB



b.) Test set of the PSB

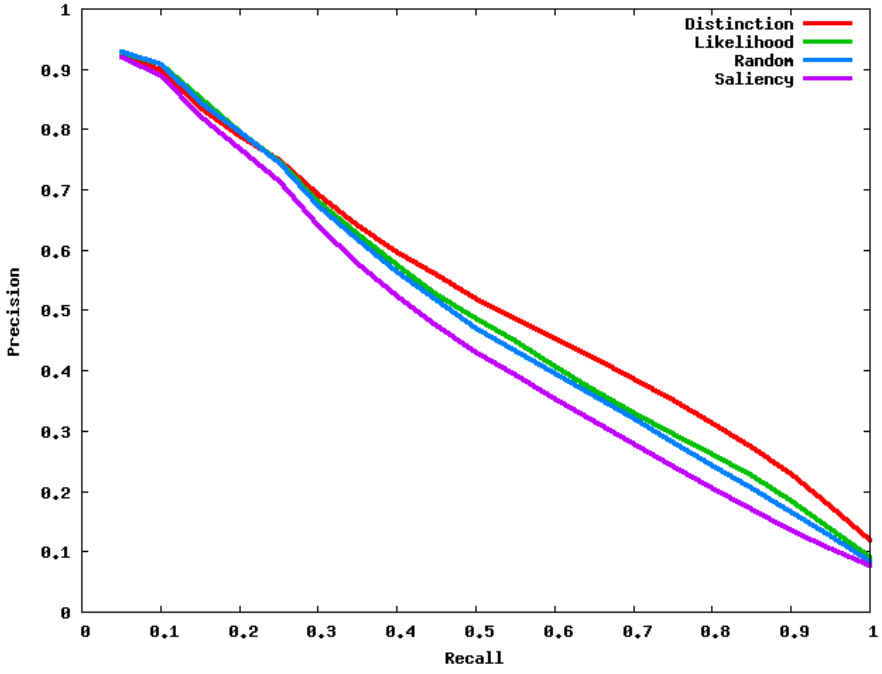


Figure 5.8: Selecting distinctive features to focus matching leads to better retrieval performance than selecting features that are least likely, most salient, or randomly selected.

		PSB Training Set					PSB Test Set				
Descriptor Selection	K	NN (%)	1-Tier (%)	2-Tier (%)	E (%)	DCG (%)	NN (%)	1-Tier (%)	2-Tier (%)	E (%)	DCG (%)
Distinction	3	68.3	42.1	54.2	29.3	67.6	62.8	40.2	51.7	29.7	65.6
Likelihood	3	64.9	37.4	49.2	27.3	64.4	66.5	37.0	49.0	27.9	64.2
Random	3	68.2	39.0	50.0	27.3	65.8	66.5	36.0	47.7	27.5	63.4
Saliency	3	61.7	35.1	46.6	25.6	62.7	61.6	33.5	44.5	25.9	60.9
Distinction	1	54.7	34.9	48.5	26.0	61.5	51.9	33.3	46.7	26.5	59.7
Likelihood	1	55.8	31.3	43.6	24.0	59.5	55.5	29.7	40.9	23.8	57.8
Random	1	56.3	31.8	43.6	24.1	59.9	55.6	30.0	41.4	24.0	57.8
Saliency	1	54.6	30.6	42.8	23.5	59.1	53.4	29.2	40.6	23.3	57.4
Centroid	1	54.1	28.6	38.1	21.7	57.0	53.3	26.3	35.1	21.1	54.4
Oracle	1	92.6	54.6	63.4	33.1	81.1	89.5	53.5	63.3	34.2	79.7

Table 5.4: Selecting the most distinctive descriptors from the target set improves retrieval relative to selecting based on likelihood, saliency, or at random. Retrieval improves when using several local descriptors on the query (K=3 in this result) as compared to using a single descriptor. Using the most distinctive descriptors improves over using a single global descriptor (Centroid), while there is still room to improve upon these results since a single descriptor selected by an oracle outperforms any other technique. All experiments are with meshes from the Training and Test sets of the Princeton Shape Benchmark.

64.2% for Likelihood, 63.4% for Random, and 60.9% for Saliency, and across most metrics, Distinction outperforms Likelihood, Random, and Saliency. For single descriptor matching, K=1 at the 1.0 scale, Distinction (DCG=59.7%) also beats Likelihood, Random, and Saliency with DCG scores of 57.8%, 57.8%, and 57.4%, respectively. While the numbers change somewhat for the PSB Training set, the qualitative results are the same. Nearest Neighbor scores are possibly lower for Distinction in some cases because of the instability of only considering the first retrieval result. Of course, the improved retrieval performance using distinction comes at the cost of increased computation time and the requirement that the database is classified.

Besides investigating feature selection methods, we also compared retrieval with distinct descriptors to two other retrieval methods that provide an informative comparison for retrieval performance. The most common shape matching technique [16, 121] is to use a single shape descriptor centered at the centroid of each shape with a region size large enough to include the entire shape (Centroid). Matching a single descriptor at the 1.0 scale on the surface of a shape has better retrieval performance than using the Centroid with DCG scores of 54.4% for Centroid as compared to 59.7%, 57.8%, 57.8%, and 57.4% for Distinction, Likelihood, Random, and Saliency, respectively, on the Test set. The DCG score for Distinction increases to 65.6% when matching with three descriptors

combined at each of four scales. Of course, this improved retrieval performance comes at some cost (retrieval time of 1.2 seconds versus 3 milliseconds), but we believe that surface descriptors are preferable when retrieval performance is critical.

We next compared our technique of selecting distinctive descriptors versus a best-case method where an oracle selects a single descriptor from the surface of the query shape across all scales. For each query shape, the single descriptor with the highest distinction score (calculated during preprocessing) was selected to use as the query, and the closest match to each target in the database was found. Although this process is not usually possible in a real application (since the class of the query is generally unknown), it provides an upper bound on the retrieval performance possible with surface descriptors. The Oracle technique has a DCG score of 79.7% on the Test set, which dramatically outperforms all other selection techniques we have considered. This result suggests that future work should focus on improving the selection of descriptors from the query shape and that using surface descriptors for shape matching has the potential to achieve accurate retrieval results.

We also investigated how often the most distinctive region exists at a particular scale. Table 5.5 shows the percentage of time a descriptor from each scale was selected by the Oracle technique. On both the PSB Training and Test sets, descriptors from every scale were selected as the most distinctive, though the 1.0 scale was selected most often.

	Scale			
Database	0.25	0.5	1.0	2.0
PSB Training	18.3%	23.4%	38.5%	19.8%
PSB Test	20.9%	27.0%	34.4%	17.6%

Table 5.5: With the Oracle selection method, the most distinctive feature was selected to represent each query shape across all scales. Every scale was used for matching, though the 1.0 scale was selected most often for both the PSB Training and Test databases.

5.3 Conclusion

This chapter describes an algorithm for multi-feature matching of 3D shapes with priority-driven search that focuses on distinctive regions of target meshes. The two main contributions are an algorithm for searching a database for the best multi-feature matches and an exploration of the benefit of focusing the algorithm on distinctive regions. Perhaps just as valuable is the investigation of factors that contribute to speed and retrieval performance improvements in a multi-feature matching system. We find that: 1) using ranks to measure the cost of a feature correspondence is more effective that using L_2 differences directly; 2) matching features at different scales independently and then adding the resulting costs is an effective way to combine shape information from multiple scales; and 3) selecting target features based on how distinctive they are of their object's class can improve both search speed and retrieval performance significantly beyond other selection techniques.

Perhaps the most interesting question for further study is to investigate how best to recognize 3D objects from their parts. Of course, this is an active topic in computer vision, but the issues for 3D shapes are different than they are for 2D images. Our study seems to suggest that just a few shape features are sufficient to recognize most 3D objects. It will be interesting to see whether other object types follow this pattern and whether effective algorithms can be developed using even fewer features.

Chapter 6

Updating Distinction

Introduction

Calculating shape distinction can be a time consuming process, since our main method of using the Discounted Cumulative Gain metric requires comparing every shape descriptor against every other descriptor in a database (an $O(n^2)$ operation for n descriptors in a database). Also, as new models and classes of models are added to a database, distinctive features may change because of the relationship between distinction scores and feature similarity/dissimilarity. With a naive implementation, distinction scores for the entire database would have to be entirely recalculated as new models are inserted. While this may be a reasonable preprocessing step for moderate sized databases, as databases continue to grow, calculating distinction could become impractical.

There are numerous fields where databases of 3D meshes are updated on a regular basis. The Google Earth tool has been downloaded by over 200 million users, and there is a community of designers submitting new models regularly. The Protein Data Bank has grown at an increasing rate since the 1970's, and during 2007 alone, there were over 7,200 new structures submitted. With the increasing number of 3D models available on the Internet, 3D search engines [107, 90] need to be periodically updated so users can find new content, and shape distinction scores need to be calculated for new models. Importantly, current models need to be be updated as well, since distinctive regions are those that match meshes of the correct class, and newly added meshes may cause a ripple-effect, changing distinction scores throughout the database.

Our goal is to design an efficient distinction algorithm that handles both static and growing databases. We pursue two methods for achieving this goal. First, besides the DCG evaluation metric, which requires a full retrieval list, we investigate alternative metrics that only require a fixed length retrieval list to approximate distinction. Second, we use an index structure to quickly find a fixed number of nearest neighbors for each descriptor without searching the entire database. For these approaches, there is a trade-off between calculation time and resulting retrieval performance.

The remainder of this chapter is organized as follows: The next section describes our methods for approximating shape distinction. Section 6.1.1 considers evaluation metrics besides DCG, and Section 6.1.2 describes a spatial indexing structure that is effective with shape descriptors. In Section 6.2, we present results comparing the retrieval performance of approximate distinction in relation to calculation time. Finally, we summarize the conclusions and limitations of our approach in Section 6.3.

6.1 Method

Our method to create an efficient shape distinction algorithm is to approximate distinction with an evaluation metric that only requires a fixed number of nearest-neighbor retrieval results for each descriptor. We consider several metrics common in the field, and also introduce a modification to Discounted Cumulative Gain. To take advantage of metrics that only require a short retrieval list, we present an index for shape descriptors that supports nearest neighbor search efficiently for databases undergoing dynamic changes.

6.1.1 Retrieval Measures for Defining Distinction

When calculating distinction values, the choice of retrieval metric can impact the quality of results. Metrics vary by whether they use a fixed portion of a retrieval list or the entire retrieval list and how the position of correct results are weighted when calculating a score. While retrieval metrics may have different ranges of possible scores, generally, all metrics can be converted to fit within the range of zero to one and are compatible with our definition of distinction.

Besides the standard retrieval metrics described in Section 4.1.3 (NN, First Tier, Second Tier, E Measure, and DCG), we considered several other metrics. NN 3 is similar to NN but considers the nearest three neighbors weighted by proximity. Top Ten is similar to First and Second Tier but uses only the first ten retrieval results. A limitation of many of these approaches (except NN 3 and DCG) is that any retrieval result within the portion of the list considered has equal weight in calculating a distinction score. As an example, for Top Ten, the distinction score is the same if the retrieval result consists of either one correct match followed by nine incorrect matches or nine incorrect matches followed by one correct match.

Intuitively, correct results near the front of the retrieval list should have a larger value when calculating a distinction score. The DCG metric matches this intuition with a correct retrieval result at position x having weight $\frac{1}{\lg x}$ in the final score. Logarithmic functions decrease rather slowly, so to increase the weight of correct results near the front of the list, we adjust the DCG metric to be weighted by $\frac{1}{x}$, $\frac{1}{x^2}$, $\frac{1}{x^4}$, $\frac{1}{x^8}$, or $\frac{1}{x^{16}}$, which places increasing emphasis towards the front of the retrieval list. We refer to these function as DCG $lg(x)$, DCG x, DCG x^2, ... , and DCG x^{16}. The normalization terms for the augmented DCG functions are updated accordingly to divide by the maximum possible scores.

A key observation about the various versions of the DCG metric is that DCG is a summation over a fixed range R divided by a normalization term,

$$\frac{\sum_{x=1}^{R} \frac{Q(x)}{f(x)}}{\sum_{x=1}^{C} \frac{1}{f(x)}}$$

where $Q(x)$ has value one if the mesh at retrieval position x is of the same class as the query and zero otherwise. The bottom summation from 1 to class size C normalizes the result to be within the range zero to one. Ignoring the normalization term, when $f(x)$ is of the form x^k, then the summation is of the same form as the Riemann zeta function or p-series, though over a limited range. The Riemann zeta function is known to converge when $k > 1$, so we can approximate the true value with arbitrary accuracy, setting R based on the value of k. The modified DCG metrics can be approximated by the same property. We show that using an index structure that finds R nearest neighbors efficiently will allow us to approximate distinction and maintain high quality retrieval results.

Figure 6.1 shows a visualization of distinction scores using twelve different evaluation metrics on a dolphin model from the test set of the PSB. The distinction scores were calculated using 128 descriptors per mesh at the 0.5 scale, and the scores were normalized in the visualization to fall evenly between zero and one for each mesh. Distinction scores close to one are visualized with larger, reddish spheres, and distinction scores close to zero are visualized with smaller, bluish spheres. The first two rows of results (NN through E Measure) tend to show a lack of continuity in the distinction function with high distinction regions (red spheres) near low distinction regions (blue spheres), such as in the bottom-front flipper region. Since our shape descriptors tend to change in a gradual manner, distinction scores might be expected to change gradually as well, which is generally true in the bottom two rows of examples (DCG $lg(x)$ through DCG x^{16}). The distinction scores also appear to change between examples in the first two rows (tip of the dorsal fin varies between red and yellow) and have fewer changes between examples in the bottom two rows, likely because of the similarity of the augmented DCG functions.

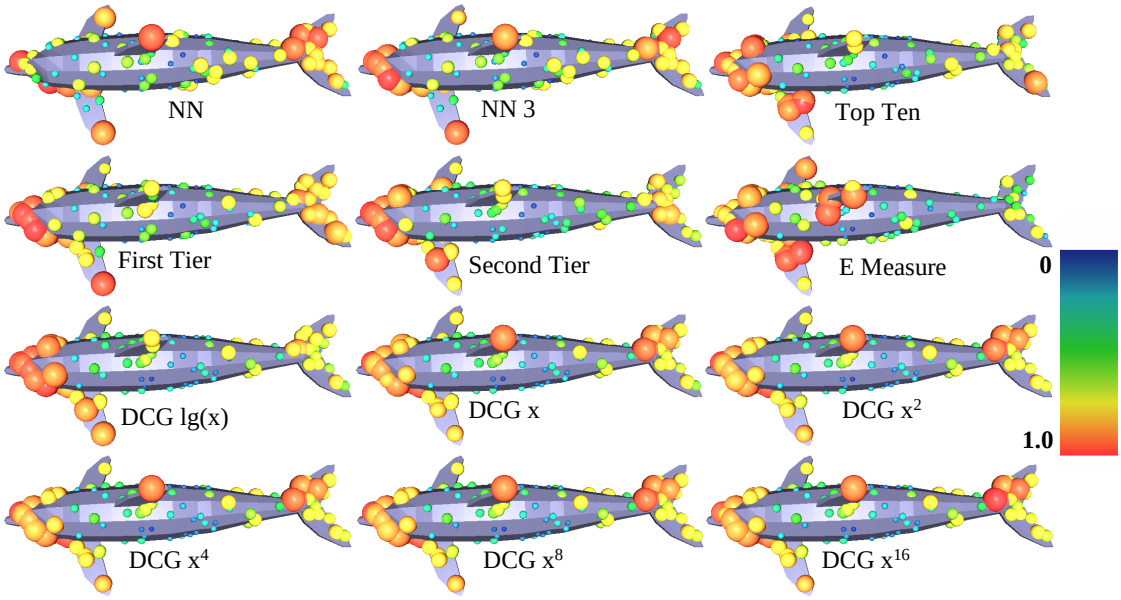


Figure 6.1: Twelve different evaluation metrics are shown for a single model with high distinction values colored red. Techniques that consider longer retrieval results (unlike NN) and weight results by position (such as DCG), tend to have smoothly changing scores that are more consistent. We also consider modified weighting functions for DCG besides the default $lg(x)$ function.

While this is only an anecdotal example of different retrieval metrics, it provides intuition about important properties needed when calculating distinction. A retrieval

metric should place more weight on correct results near the front of the retrieval list than on correct results near the end of the retrieval list, and the resulting scores should change smoothly across the surface of a mesh in the same way that region-based descriptors change smoothly. We provide a more quantitative analysis of various retrieval metrics in Section 6.2.

6.1.2 Nearest Neighbors with a Cover Tree Index

Calculating approximate distinction can be performed quickly using an index structure that finds a small set of nearest neighbors without resorting to considering all descriptors in the database. The definition of DCG and many alternative distinction metrics can be augmented to only consider a partial list (e.g., DCG_R) consisting of calculating a retrieval metric for the best R matches to the query descriptors. The definition of NN, NN 3, Top Ten, First Tier, Second Tier, and E Measure all use a fixed retrieval list by definition. Then, when calculating approximate distinction, instead of searching the entire database for matches, only the subset of the database that closely matches the query needs to be considered. Depending on the time to find R nearest neighbors, this approximation can be an efficient technique for calculating distinction. A spatial index that supports a quick nearest neighbor search can also be effective when updating distinction in a dynamic database.

There are numerous techniques for finding nearest neighbors using spatial index structures that allow neighbor search to focus on the best matches. Designing indexing techniques is an active area of research that is improving our ability to perform similarity queries in high dimensional spaces. For an overview of the field, see [4, 18, 135]. Selecting the best technique for shape descriptors is beyond the scope of this dissertation, but we investigat the cover tree index and demonstrate a favorable trade-off between distinction accuracy and processing time.

We provide a brief overview of the cover tree index structure–for more details, see Beygelzimer et al. [12]. The cover tree builds an index based on the intrinsic dimensionality c of the data, where high dimensional data sets often effectively reside in a lower dimensional subspace. Consider a bounding sphere S_0 surrounding a high dimensional point and a small set of its nearby neighbors in the data set. Enclosing bounding spheres $S_1, ...S_m$ each have a radius a constant size larger than that of the smaller sphere such

that the radius of sphere S_j is larger than the radius of sphere S_{j-1}. If the intrinsic dimensionality of the data set is low, then the number of points within a sphere will be less than twice that of the next smaller sphere. A cover tree is constructed with a hierarchy of spheres of increasing radius, and similarity queries can be handled by moving through the hierarchy to the data closest to the query. Many data sets in practice expand at a slow rate (where the expansion constant of the spheres grows faster than the number of neighbors), so a cover tree is effective at spatial indexing. When a data set with n points has expansion constant c, a cover tree can be constructed in $O(c^6 n \log n)$ time, support insertions and removals in $O(c^6 \log n)$ time, queried in $O(c^{12} \log n)$ time, and stored in $O(n)$ space. These properties allow approximate distinction to be updated efficiently for a rapidly changing database.

6.2 Results

In this section, we present results of our technique for updating distinction. We investigate several retrieval metrics to calculate distinction and show the effect on overall retrieval performance. Using a cover tree index, we compare retrieval performance with metrics that require R neighbors in relation to the time to find R neighbors.

6.2.1 Alternative Retrieval Metrics

In Section 6.1.1, we provided a qualitative analysis of shape retrieval metrics suggesting that more weight should be placed on correct results near the front of the retrieval list, and using more retrieval results leads to greater accuracy (NN only uses one match, which is probably too sensitive). Using the priority-driven search algorithm, we can perform a quantitative analysis of each retrieval metric.

Our experimental method is largely similar to previous examples for the preprocessing phase. We calculate descriptors at multiple scales and create a retrieval list for each descriptor. We then evaluate the distinction of each descriptor using a variety of retrieval metrics that convert from a retrieval list into a score between zero and one and select a subset of descriptors with high distinction scores that are well spread-out across a mesh. Then, the PDS algorithm is performed with each type of distinction score.

		PSB Training Set					PSB Test Set				
Descriptor Selection	K	NN (%)	1-Tier (%)	2-Tier (%)	E (%)	DCG (%)	NN (%)	1-Tier (%)	2-Tier (%)	E (%)	DCG (%)
Random	3	68.2	39.0	50.0	27.3	65.8	66.5	36.0	47.7	27.5	63.4
NN-1	3	66.0	38.2	49.8	27.0	65.1	66.0	36.1	47.1	27.1	63.3
NN-3	3	65.6	38.0	49.8	27.0	65.1	66.3	36.3	47.2	27.1	63.4
Top Ten	3	66.2	39.4	51.1	27.6	65.9	66.0	37.6	49.0	28.2	64.2
First Tier	3	66.6	41.2	52.8	28.5	67.0	64.9	37.2	48.7	27.7	63.8
Second Tier	3	67.0	39.5	51.8	27.4	66.0	65.8	37.9	49.7	28.2	64.3
E Measure	3	66.8	41.3	53.4	28.9	67.2	65.3	37.8	49.9	28.4	64.2
DCG lg(x)	3	68.3	42.1	54.2	29.3	67.6	62.8	40.2	51.7	29.7	65.6
DCG x	3	68.2	41.6	54.0	29.3	67.4	65.4	39.9	51.4	29.5	65.4
DCG x^2	3	67.0	41.7	53.5	29.0	67.4	64.9	39.5	51.1	29.3	64.9
DCG x^4	3	65.8	41.9	53.5	29.0	67.5	64.5	39.7	50.8	29.2	65.2
DCG x^8	3	67.5	41.8	53.5	29.0	67.5	65.7	39.4	50.9	29.4	65.2
DCG x^{16}	3	68.0	41.6	52.8	29.0	67.5	64.6	39.1	50.1	29.0	64.8

Table 6.1: Twelve retrieval metrics are investigated as well as a baseline Random metric. Generally, using a larger portion of the retrieval list improves accuracy as well as placing weight on the first results. Increasing the weight function to extreme values x^8 and x^{16} tends to cause little change.

Table 6.1 shows the retrieval results on the PSB Training and Test sets using twelve retrieval metrics as well as a Random metric that provides a baseline for comparison. For the Random metric, scores were assigned to each descriptor randomly, without considering a retrieval list. The first column shows the metric used for calculating distinction scores during preprocessing, the second column shows the optimal number of feature matches (k), and the retrieval results of running our PDS algorithm for a given distinction metric are shown in the remaining columns. Note that the last five versions of DCG increase the weight of results near the front of the retrieval list. There are a number of parameters to the PDS algorithm, so we optimized the algorithm for each evaluation metric independently, and these results represent the best retrieval scores for each metric.

We generally see that increasing the number of retrieval results considered by the metric during the preprocessing phase improves retrieval using PDS. DCG scores increased from 63.4 for NN-1 to 65.6 when using the full retrieval list with the DCG $lg(x)$ metric. The metrics NN-1, Top Ten, First Tier, Second Tier, and E Measure use only a fixed portion of the retrieval list and weight any result within that region equally, which likely explains their performance. In all examples, a version of DCG that weights results based on position leads to better results. Among the versions of DCG considered in this study, the results are fairly similar, though scores decrease with extremely high weighting

values. These results are generally higher than the base line Random metric, though they are slower to calculate.

6.2.2 Time for K-Nearest Neighbors

We slightly modified code provided by the authors of the cover tree [75] to support shape descriptors. In our experiments, we built a cover tree for each scale of the descriptors independently, so we could separate the timing analysis from the number of scales. We used the PSB with 128 descriptors per model, considered each model as a query into the cover tree, and averaged the query times.

Figure 6.2 shows the timing results for the 1.0 scale as the number of neighbors increases. First, we notice that to perform a full search of the 907 Training set models in the database takes 35 seconds without any index structure, and the cover tree can only find 512 models with closest descriptors in that time because of overhead associated with the structure. 32 neighbors can be found in less than five seconds, and 128 neighbors can be found in less than ten seconds. We also merged the Training and Test sets of the PSB to perform a larger experiment with 1,814 models. 32 neighbors can be found in under 20 seconds and 128 neighbors in approximately 60 seconds. Without an index structure, finding neighbors by scanning 1,814 models would take approximately 70 seconds, and when searching for more than 128 neighbors, scanning provides faster results. If calculating distinction with up to 128 neighbors gives a reasonably good approximation to the true distinction values, then approximate distinction can be calculated for the 907 meshes in the Training set in approximately 2.5 hours per scale and ten hours for four scales as compared to 37 hours without the index.

6.2.3 Approximate Distinction versus Calculation Time

An efficient approximate version of distinction is useful for improving the preprocessing time of calculating distinction. To evaluate our approximation technique, we found a small set of neighbors using a cover tree index, calculated the approximated version of distinction, selected a small set of descriptors for each target model based on distinction scores, and performed a retrieval experiment. Table 6.2 shows several retrieval statistics with a fixed size retrieval list for both the PSB Training and Test sets using 128 descriptors

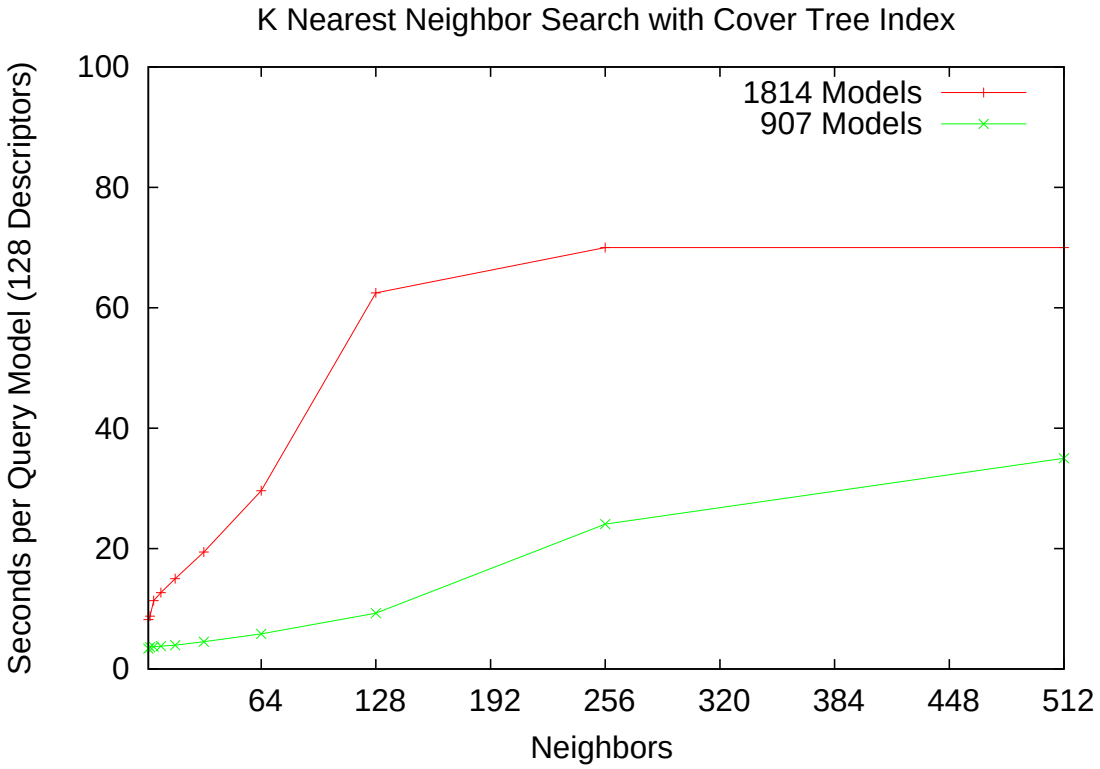


Figure 6.2: Calculating distinction for a new model can be performed efficiently using a cover tree indexing structure on the descriptors in the database. Times are shown for the 1.0 scale and 128 descriptors per model in the training set (907 models) and full PSB (1,1814 models), though timing results are consistent at all scales. Up to 128 neighbors can be found per model within 10 seconds when searching the PSB training set, and 64 neighbors can be found in approximately 32 seconds when searching the full PSB.

per model at four scales. Both the standard DCG $lg(x)$ and augmented DCG x^2 distinction metrics are shown with the number of neighbors R (second column) varying from two through the full retrieval list of 906. The third column shows the time to find R neighbors using the cover tree. For comparison, using a Random distinction function (randomly assigning distinction scores between zero and one) and using only the first retrieval result (NN) for distinction are also shown.

There are several important results shown in this experiment. First, retrieval performance only improves slightly as R increases from zero to 906. The DCG score increases slowly as R increases and is fairly flat, which suggests that there is little advantage to using more than 64 or 128 neighbors when calculating distinction. While calculating distinction with DCG x^2 has slightly worse performance than DCG, using x^2 can provably

			PSB Training Set					PSB Test Set				
Descriptor Selection	R	Time sec.	NN (%)	1-Tier (%)	2-Tier (%)	E (%)	DCG (%)	NN (%)	1-Tier (%)	2-Tier (%)	E (%)	DCG (%)
Random	0	0	68.2	39.0	50.0	27.3	65.8	66.5	36.0	47.7	27.5	63.4
NN-1	1	3.4	66.0	38.2	49.8	27.0	65.1	66.0	36.1	47.1	27.1	63.3
DCG lg(x)	2	3.6	68.5	40.3	51.8	28.0	66.8	65.8	37.4	48.9	28.4	64.3
DCG lg(x)	4	3.7	64.3	40.8	52.4	28.4	66.5	65.0	36.2	47.8	27.5	63.2
DCG lg(x)	8	3.8	65.3	40.9	52.8	28.5	66.5	63.7	36.9	48.1	27.4	63.4
DCG lg(x)	16	4.0	64.5	41.1	53.0	28.7	66.7	63.8	38.6	50.7	28.9	64.7
DCG lg(x)	32	4.5	64.8	41.4	53.6	28.9	67.1	63.7	37.9	49.5	28.6	64.2
DCG lg(x)	64	5.8	67.0	42.1	54.4	29.3	67.6	63.6	38.2	50.0	28.8	64.4
DCG lg(x)	128	9.3	65.7	42.2	54.3	29.3	67.4	64.1	39.8	51.5	29.7	65.2
DCG lg(x)	256	24.0	66.2	42.3	54.4	29.4	67.9	63.7	40.0	51.8	29.7	65.3
DCG lg(x)	906	35.0	68.3	42.1	54.2	29.3	67.6	62.8	40.2	51.7	29.7	65.6
DCG x^2	2	3.6	68.5	40.3	51.8	28.0	66.8	65.8	37.4	48.9	28.4	64.3
DCG x^2	4	3.7	65.0	40.6	52.3	28.4	66.4	64.5	37.7	49.6	28.6	64.2
DCG x^2	8	3.8	64.7	40.6	52.5	28.4	66.2	63.9	38.1	49.8	28.6	64.3
DCG x^2	16	4.0	64.8	40.7	52.9	28.4	66.4	64.8	38.1	50.3	28.7	64.5
DCG x^2	32	4.5	65.7	41.0	53.1	28.6	66.7	64.5	38.8	50.7	28.8	64.6
DCG x^2	64	5.8	66.4	41.4	53.3	28.8	67.1	64.6	38.9	50.5	29.0	64.6
DCG x^2	128	9.3	67.1	41.7	53.4	28.9	67.3	64.9	39.2	50.8	29.1	64.7
DCG x^2	256	24.0	66.8	41.7	53.5	29.0	67.3	64.1	39.3	50.9	29.2	64.8
DCG x^2	906	35.0	67.0	41.7	53.5	29.0	67.4	64.9	39.5	51.1	29.3	64.9

Table 6.2: Up to 128 nearest neighbors can be found in under 10 seconds using a cover tree index, which provides a good approximation to distinction and retrieval results similar to using the full retrieval list.

be approximated with a shorter retrieval list, which provides a theoretical justification for this approximation technique as the database size grows. Using a cover tree index, 128 neighbors can be found in under ten seconds per model with retrieval performance that is similar to the baseline technique of calculating distinction with a full search through the database, which requires 35 seconds. Using a descriptor index, an approximated version of distinction can be calculated for new meshes in a few seconds, while maintaining retrieval performance.

6.2.4 Updating Distinction when Inserting Models

Besides calculating distinction for newly inserted meshes, meshes in the database need to be updated as well. If a distinction score requires a full retrieval list such as the original version of DCG, then the entire database would need to be updated when inserting mesh M. Using an R approximation to distinction, the only meshes in the database that need to be updated are those with descriptors that would have mesh M within their first R neighbors. We present a simple technique for updating the necessary meshes and descriptors of the database after an insertion.

Our method is to record in an index structure all of the descriptors based on the distance to the Rth neighbor needed for approximate distinction, find the nearest N ($N \geq R$) neighbors for a new mesh M, and then update those meshes in the index that have R-distance greater than the distance to N. When finding the N neighbors using an index structure, distinction can be calculated for M, and distinction could be updated for those N neighbors as well.

The main observation of our technique is that models in the database that need to be updated are those that have M within their respective list of R neighbors, which we refer to as the reverse neighbor distance. Any descriptor with distance to their respective R neighbors greater than the distance from descriptors of mesh M to their Nth neighbors potentially need to be updated. Otherwise, if their distances were less, they would be within the the N neighbors of M's descriptors. This holds because the L_2 distance metric on descriptors is symmetric, $\|x_1 - x_2\| = \|x_2 - x_1\|$. As an example, consider a sorted list of R distances $(0.1, 0.2, 0.25, ..., 0.4, 0.41)$ and the N distance for a new descriptor added to the database is 0.39. Distinction needs to be updated for the descriptors that constitute the N distance as well as the descriptors in the R distance list that correspond to the values 0.4 and 0.41.

We performed an experiment to evaluate how many descriptors need to be updated when inserting meshes. We used the PSB training set with 128 descriptors, considered each model as a new query, and present timing results averaged over the four scales. For each new mesh inserted, N neighbors were found using a cover tree index. The distance to the Nth neighbor was then used to index into a Red-Black tree that has every descriptor entered by its distance to its R neighbor (creating an R approximation for distinction). Figure 6.3 shows the results of the experiment. The horizontal axis shows the results for various values of R when N is either 128 or 256, and the vertical axis shows the percent of the database that has to be considered for update. If there is sufficient time to find 256 neighbors for a new mesh, then an R approximation of up to 32 can be achieved while updating less than 10% of the database. If 32 neighbors provides sufficently accurate distinction, then the database can be updated reasonably quickly. When finding 128 neighbors for newly inserted meshes, an R accuracy of 16 can be achieved while updating less than 20% of the database. In the worst case, nearly every descriptor in the database may need to be updated, but we find that in practice it is only a reasonably small percentage of the database. Of course, this is only one technique for

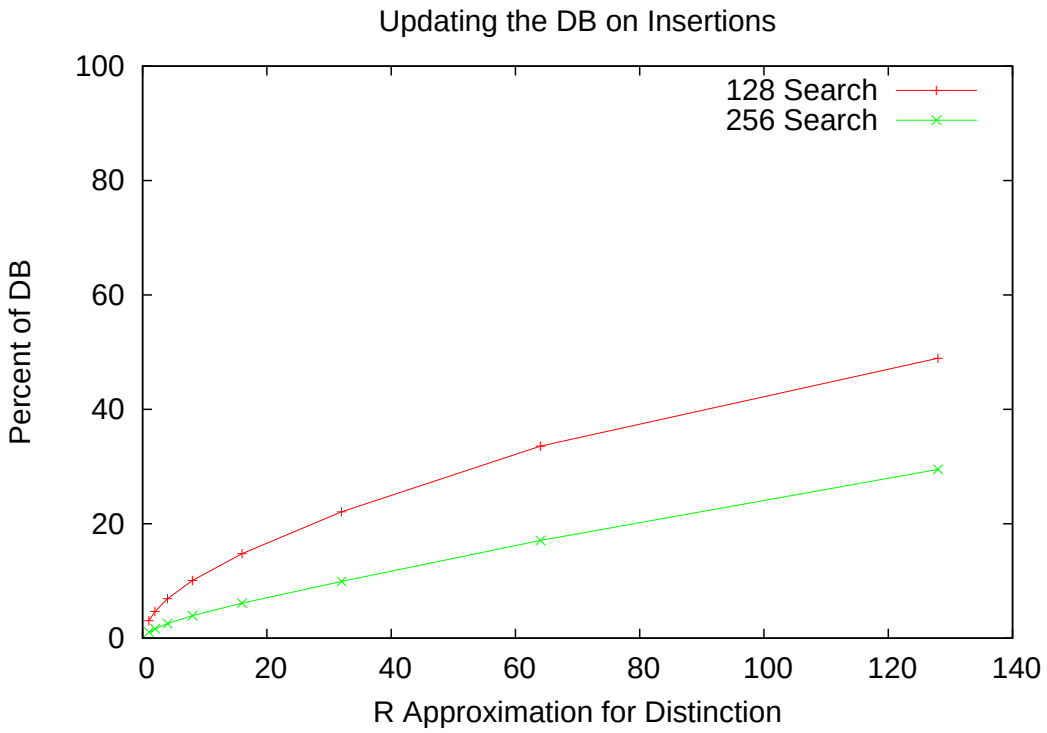


Figure 6.3: When inserting new meshes into a database, distinction scores for other models need to be updated. Using an index structure, 128 neighbors are found for each inserted mesh, and based on the distance to the last neighbor, the number of descriptors that need updating (to be *R* approximations) is plotted.

updating a dynamic database, and there are likely other efficient methods such as batch updating and improved indexing structures that may provide further speedup.

We have shown that there is a trade-off between the time to calculate distinction and the resulting retrieval accuracy. High retrieval accuracy can be achieved in a dynamic database while limiting the update overhead using a combination of techniques. For new meshes added to the database, distinction can be approximated with a small set of nearest neighbors found using a cover tree index. Distinction scores can then be updated for the rest of the database by considering the reverse nearest neighbor distance.

6.3 Conclusion

While we have not implemented an end-to-end system to update distinction scores while performing retrievals in a dynamic database, we have presented a plan for such a system and trade-offs between time to approximate distinction and the resulting retrieval accuracy. Good retrieval accuracy can be achieved in a dynamic database while limiting the update overhead using a combination of techniques. For new meshes added to the database, distinction can be approximated with a small set of nearest neighbors found using a cover tree index. The approximation has a provable error bound with a modified version of the DCG function. Distinction scores can also be updated for the rest of the database by considering the reverse nearest neighbor distance.

Our technique of approximating distinction and using a cover tree index to find nearest neighbors is a first approach to this problem. There are several limitations of our technique that deserve further research. While the cover tree has an update time that is logarithmic in the number of entries, shape descriptors for very large databases may not meet the intrinsic dimensionality requirement. Also, our approach for updating models in a database using the reverse nearest neighbor distance requires updates to 10-20% of the database.

Chapter 7

Predicting Distinction

Introduction

Performing shape-similarity retrieval with local descriptors can be quite slow when comparing every descriptor from the query against every descriptor of every shape in a database. Previously, we demonstrated a technique for reducing the number of comparisons by preprocessing a database to select distinctive descriptors for each target shape. An alternative technique is to filter the query shape to a small set of distinctive descriptors, but calculating distinction for query models is not possible, since the classification is generally unknown.

Our goal is to predict which query shape features are distinctive and focus similarity retrieval on those features. Our approach is to compute shape descriptors for several regions of each shape, map them into a space parameterized by their likelihood, predict their distinction based on a training set of labeled descriptors, and then select only the most distinctive descriptors to be used during retrieval (Shilane and Funkhouser [112]).

In this chapter, we address the research problem of predicting shape distinction. Specifically, we make the following contributions: 1) the definition of a mapping function for shape descriptor likelihood that separates descriptors with good retrieval performance and 2) an algorithm for learning the retrieval performance of descriptors from a training set.

The remainder of this chapter is organized as follows: The next section gives an overview of how shape distinction can be used to improve local matching for retrieval. In

Section 7.1.1, we define a mapping function based on the likelihood of shape descriptors. In Section 7.1.2, we show how to predict the retrieval performance of each descriptor from a training set. We review how to select a subset of descriptors from a query in Section 7.1.3. In Section 7.2, we provide empirical results demonstrating that our definition of predicted shape distinction is useful for retrieval, and we summarize our results in Section 7.3.

7.1 Overview of the Approach

The organization of our system is shown in Figure 7.1. During a training phase, a distinction function is learned. First, a shape descriptors are created for numerous regions of a shape. Then, the likelihood of each descriptor is evaluated along with its retrieval performance in the classified training database. A histogram of retrieval performance scores is built for different descriptor likelihood values.

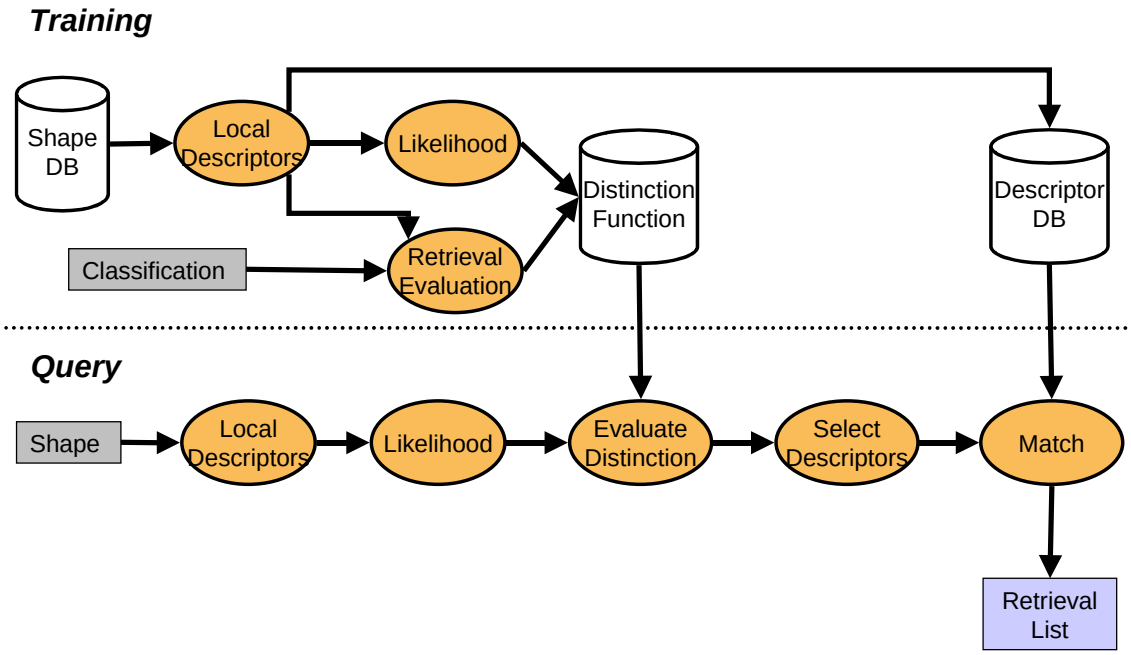


Figure 7.1: Diagram of training and query phases.

When a user presents a query shape to the system, distinction values are predicted for local descriptors on the query shape. First, local descriptors are generated across the surface in a manner similar to the training phase. The likelihood of each descriptor

relative to the training database is calculated. Then, based on the likelihood of each descriptor and the distinction function, a distinction value is predicted. A small set of the k most distinctive descriptors is then selected for the query. Each selected descriptor is matched against all descriptors in the database, and then the objects with the best sum of match scores for all k selected query descriptors are returned as the retrieval result.

The key step in this process is the method to predict distinction for every shape descriptor based on the average retrieval performance of descriptors with the same likelihood in a training set. More formally, predicted distinction function D maps descriptor d with likelihood function map into a bin representing descriptors from a training database with the same likelihood value as d. We represent these training descriptors with the same likelihood as the set F. The predicted distinction value for d is the average retrieval performance of the descriptors $f \in F$.

$$D(map(d)) = \frac{1}{|F|} \sum_{f \in F} RetrievalPerf(f)$$

There are several advantages to this approach. The main advantage is that our predicted distinction function D is based on the retrieval performance of descriptors from the training database. Another advantage is that D is independent of the type of descriptor, so it can be applied to many real-valued descriptors. Also, by defining a predicted distinction function in terms of descriptors mapped by likelihood, we have created a one-dimensional parameterization. This allows for a compact representation of predicted distinction as a table of average retrieval scores computed from a training set. The query descriptor with likelihood mapping to the highest predicted distinction can be used as the query into the database. If multiple descriptors for the query shape will be used for retrieval, D provides an ordering of the descriptors. Alternatively, while descriptors are being calculated for the query shape, predicted distinction can be determined for each descriptor, and the process can end when a descriptor with a sufficiently high distinction value is found. As such, we have a quick way to select the most distinctive descriptors for a query.

In the following sections, we investigate several research problems for creating the distinction function. We first define a likelihood model for shape descriptors and show how to use a training set to evaluate retrieval performance. We then select a subset of the most distinctive descriptors for a query shape and use the subset during retrieval. For comparison, we evaluate prediction function D against other common alternatives.

7.1.1 Mapping from Descriptors to Likelihood

The first issue in implementing our approach is to define a mapping function that clusters shape descriptors based on their retrieval performance. The challenge is to define a mapping such that descriptors near each other in the mapped space will have similar retrieval scores and be well separated from descriptors with different scores. There are many options for a mapping function. One approach is to use the full dimensionality of descriptors directly, though this could be a slow prediction function. Other mapping functions could use the local curvature or the descriptors' positions relative to a coordinate system such as the shape's center of mass.

We define a mapping function of shape descriptors using likelihood based on the work of [20, 62]. A rationale for this approach is that rare features (such as the wing-tips and tail of the plane in Figure 7.2) may be discriminating for retrieval, while common areas (such as the flat portions of the wings) may match numerous categories of shapes. Likelihood mapping has the advantage of being independent of the underlying real-valued feature vector used as a shape descriptor. Also, after descriptor statistics are estimated from the training set, the likelihood function can be evaluated quickly for queries.

A key question is then how to map descriptors to likelihoods. In previous work, Johnson et al. [62] used a mixture of Gaussian distributions to estimate descriptor likelihoods. However, if the distribution of our descriptors is normal, then perhaps we can use a single Gaussian distribution to achieve the same performance at less cost. Based on the assumption of a normal distribution of shape descriptors, the probability density of descriptor x can be modeled by a multivariate normal distribution [27]:

$$\text{density}(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)}$$

with mean μ and covariance Σ estimated from a training set and d equal to the dimensionality of the shape descriptor. Descriptor x is treated as a vector for this calculation.

Under floating point arithmetic, the exponential function rounds to zero for descriptors far from μ, so we work with the natural log of the density function. We also drop the normalization term since we are interested in the relative density of descriptors as opposed to their exact values. We refer to this function, p, as the likelihood of a descriptor:

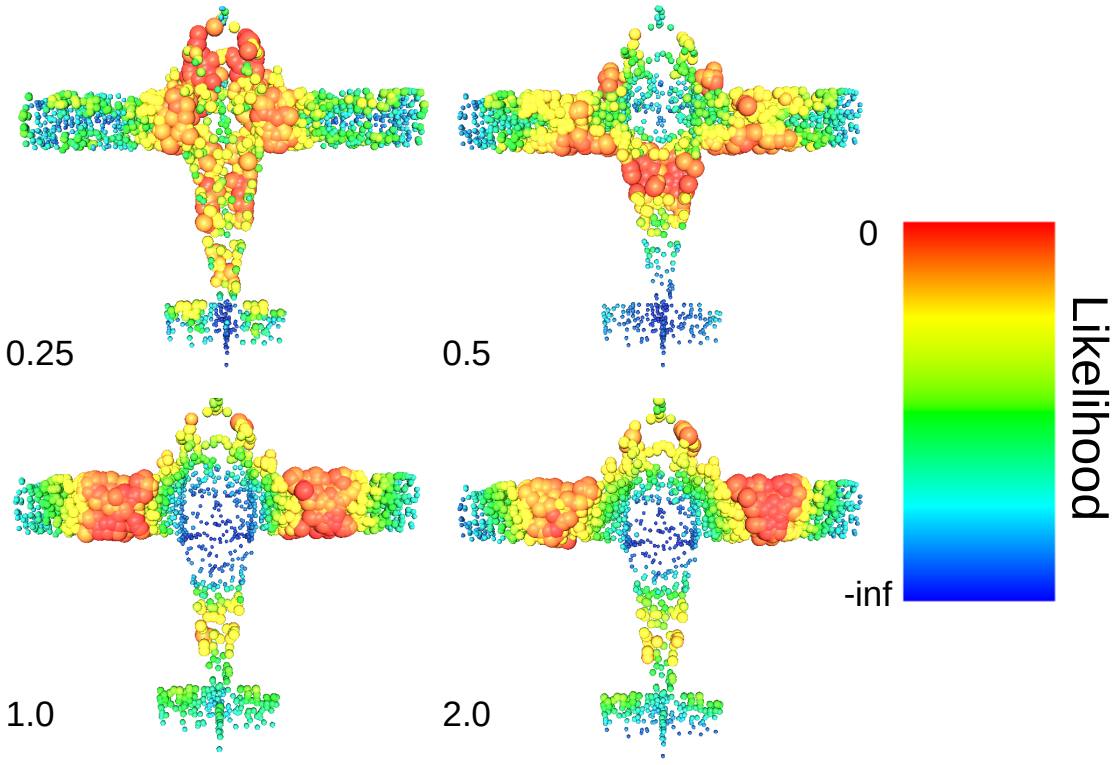


Figure 7.2: The likelihood of the descriptors is color coded with red indicating the most likely descriptors. Notice that the likelihood of the descriptors changes with the scale of the descriptor.

$$p(x) \propto \ln(\text{density}(x))$$

$$p(x) = -\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)$$

In practice, we calculate distinction function D from the training set with p as the mapping function, therefore $map \equiv p$. Bins partitioning the likelihood space hold the average retrieval performance of the training set descriptors. Since the distribution has a long tail of low likelihood, a threshold is selected and a bin represents all descriptors with likelihood below the threshold.

To evaluate this normality hypothesis, we generated 200,000 local descriptors on 100 shapes from the Princeton Shape Benchmark (PSB). For this experiment we used a version of the Harmonic Shape Descriptor representing a local region of each shape with 512 values. We compared the distribution of these descriptors against 200,000 feature vectors

randomly generated with distribution $N(0, 1)$ and 512 dimensions. Since our definition of likelihood incorporates a covariance matrix that accounts for correlated features, we evaluated the shape descriptors with a diagonal covariance matrix for this experiment. Figure 7.3 shows a quantile-quantile plot [51] comparing the shape descriptor distribution against the randomly generated feature vectors. A quantile-quantile plot is a visualization of the relationship between two distributions of data. For each $+$ marker, the horizontal position indicates the likelihood value for a quantile of the randomly generated data, and the vertical position for the maker indicates the likelihood for an equal quantile of the measured shape descriptor data. The straight line indicates the line of best fit between the distributions, which corresponds to a normal distribution with different mean and variance. While the shape descriptor distribution varies from the line of best fit, a normal distribution is a reasonable model for the majority of shape descriptors.

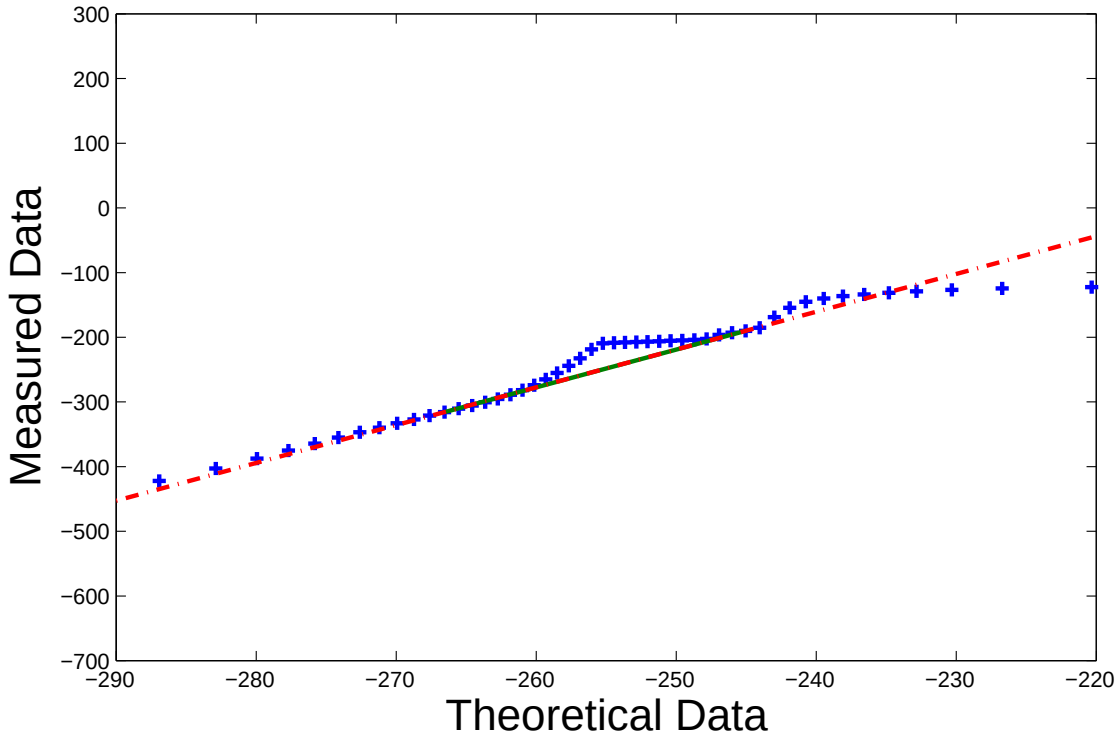


Figure 7.3: Quantile-Quantile plot of the likelihood of HSD descriptors against a randomly generated $N(0, 1)$ distribution. The $+$ markers indicate the relationship between the measured and randomly generated data. A normal distribution (dashed line) provides a good model of the shape descriptor distribution.

7.1.2 Mapping from Likelihood to Distinction

The second issue is to define a distinction function that maps descriptor likelihood to an expected retrieval score. For this step, we evaluate the retrieval performance of every local shape descriptor in a training set and build a histogram of average retrieval performance as a function of likelihood.

During a training phase, each query shape is presented to a retrieval system, and local descriptors are calculated over the shape. As in our previous work, a retrieval list is generated for each descriptor. Then, the quality of the retrieval list can be evaluated with any standard retrieval metric (Chapter 4.1.3). In our experiments, we use the Discounted Cumulative Gain (DCG).

For every query descriptor in the training set, we evaluate both its likelihood and its DCG retrieval performance. Then, we cluster descriptors into regular bins by likelihood and average the DCG scores for all descriptors in the same likelihood bin. The result is a histogram of average DCG scores indexed by likelihood that can be used as a map from likelihood to distinction.

7.1.3 Selecting Distinctive Descriptors

The next issue is to select the k most distinctive descriptors from each query shape to use during retrieval. Given a query shape, we compute the likelihood of every local descriptor and map likelihood scores to distinction, creating a predicted distinction score for each descriptor. We can then select a subset of descriptors using the same algorithm described in Section 5.1.2, that maintains a minimum distance constraint while selecting the k descriptors with the hightest predicted distincton. An example of our selection technique for a biplane query model is shown in Figure 7.4.

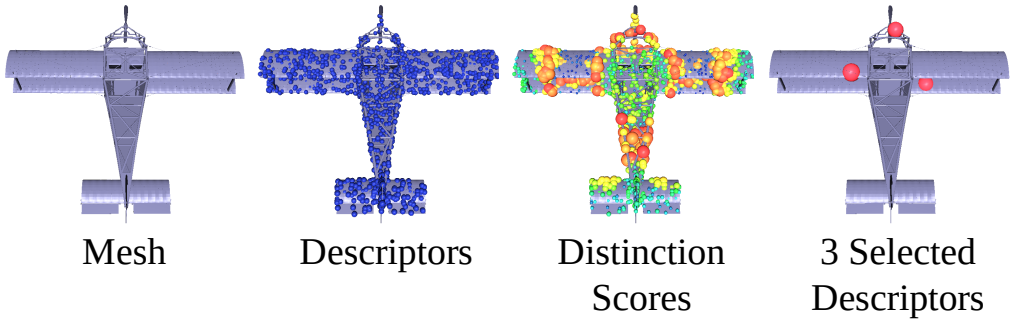


Figure 7.4: When a new query mesh is presented, shape descriptors are created at random positions, the predicted distinction scores are calculated based on the likelihood of each descriptor, and a subset of distinctive descriptors is selected to be used during retrieval.

7.2 Results

In this section, we evaluate the utility of selecting descriptors with predicted distinction based on likelihood and learned retrieval performance. We first describe the shape database and set of shape descriptors used for our experiments, and then we address the following research questions with empirical results.

- How well does a likelihood mapping predict distinction?
- Is predicted distinction useful for retrieval?
- Is our method of predicting distinction for a query shape better than other alternative approaches?

7.2.1 Shape Database

In this experiment, we evaluated 100 models[1] from the Princeton Shape Benchmark. We focused on this small subset of the PSB so that we could calculate a large number of local descriptors and thoroughly evaluate a likelihood function. The 100 shapes, evenly divided into ten classes, represent classes that are in different branches of the hierarchical classification, so a diverse set of classes was included.

During the preprocessing phase, 2,000 shape descriptors were computed over the surface of the mesh across four scales (0.25, 0.5, 1.0, and 2.0) as shown in Figure 7.5. Unless

[1]The shape classes include: biplane, spider, human with arms out, dome church, dining chair, rectangular table, ice cream, potted plant, sedan, and tank.

otherwise noted, all of the reported experiments are for a scale of 1.0, which generally includes about 30% of the surface area when positioned on an extremity. Though our technique is independent of the type of shape descriptor, we experimented with the Harmonic Shape Descriptor (HSD) and Shells Descriptor (SD) because these descriptors are simple to compute, invariant to rotations (which simplifies matching), quick to compare, and showed good performance in our previous studies.

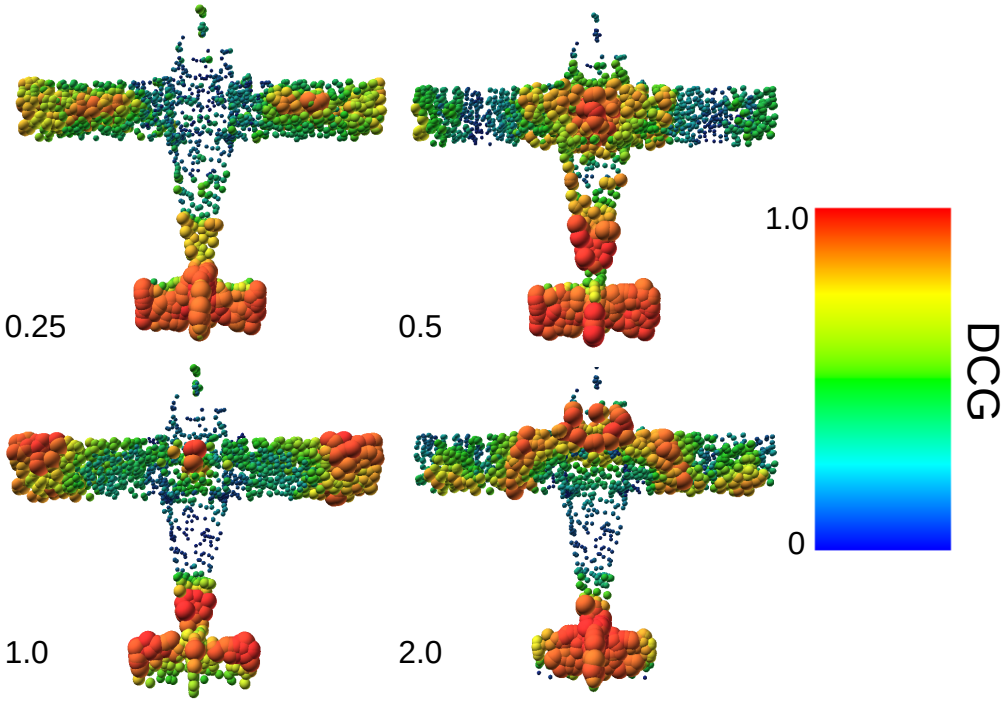


Figure 7.5: The distinction scores for local descriptors over the surface is shown with red indicating the best performance. Across multiple descriptor scales, the tail region of the plane has distinctive descriptors.

7.2.2 Mapping Functions

We first evaluated whether mapping descriptors based on their likelihood effectively groups descriptors with similar retrieval performance. For every shape descriptor, we performed a query into the database of descriptors for the 100 shapes and evaluated the likelihood of the descriptor and its retrieval performance. Figure 7.6 shows the resulting average retrieval performance as vertical bars for each likelihood value. The horizontal axis shows the likelihood. The left vertical axis is retrieval performance as measured by

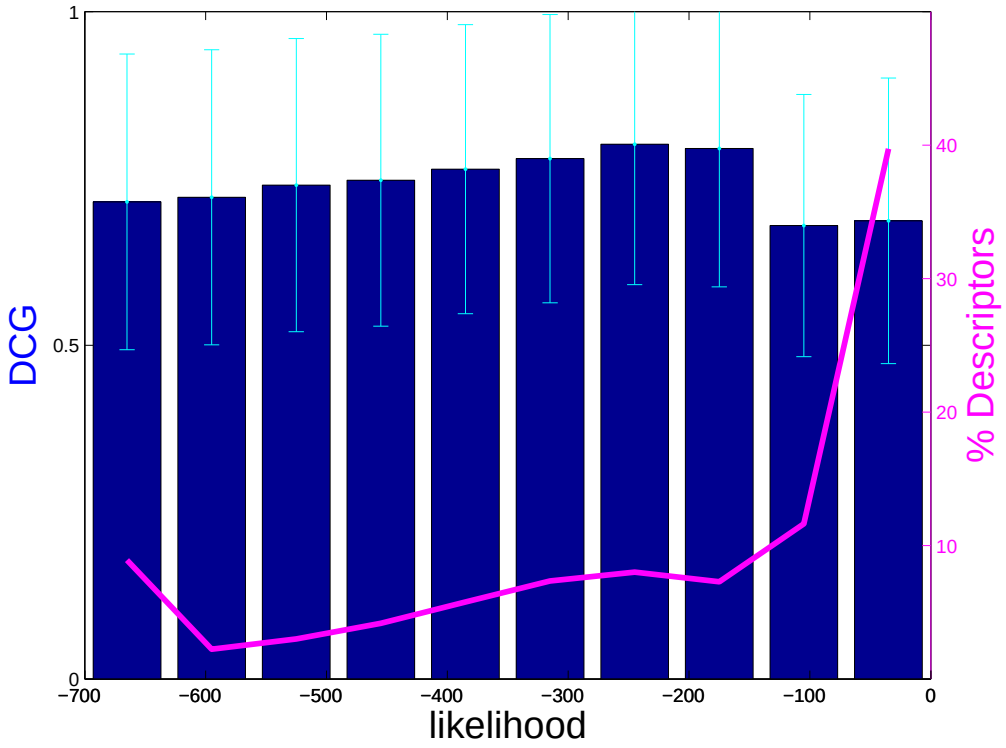


Figure 7.6: Using a likelihood mapping of HSD descriptors, the majority of descriptors fall within a poor retrieval group to the right. An area between the least likely and most likely descriptors tends to be better for retrieval.

DCG, with 1 standard deviation error bars shown in cyan. The magenta line indicates the percentage of descriptors that falls within each likelihood bin. Note that the axis for the magenta line is on the right side of the plot.

The most likely bin of the histogram (with 40% of the descriptors) contained descriptors with nearly the worst retrieval performance. We also found that grouping shape descriptors based on their likelihood effectively clusters descriptors with similar retrieval performances. Using a t-test, there is 99% confidence that the bin with the best performance varies significantly from the most common bin.

For comparison, we considered alternative mappings, such as the amount of surface area within the descriptor's radius, as well as the position of the descriptor relative to the shape's center of mass, in studies explained in Section 4.2.3. However, both alternatives failed to group descriptors with similar retrieval scores as well as likelihood.

7.2.3 Retrieval Results

We next evaluated whether using distinctive local descriptors can improve retrieval performance over competing methods. We performed a leave-one-out experiment where we held out one model as a query and trained the distinction function over the remaining models (this maximizes the size of our training and test sets, since each of the 100 models serves as a query once and the training set has the remaining 99 models). For each query, we matched its k most distinctive descriptors to all the descriptors of the other 99 models, and then we returned the models in a ranked retrieval list.

We adjust our shape matching algorithm to focus on shape descriptor similarity and neglect deformation since we are interested in investigating the value of predicting distinction. We take a simple approach in this study: we measure the sum of distances between all k descriptors from query X (represented as X^k) and the closest descriptors of model Y:

$$|X^k - Y| = \sum_{i}^{k} C(X_i^k, Y)$$

where $C(X_i^k, Y)$ is the minimal L_2 difference between X_i^k and all descriptors of Y.

Although this distance function does not consider the amount of deformation necessary to bring the corresponding regions of the shape into alignment, it is fast to compute, and it can be considered a lower-bound on more complex geometric distance functions such as the cost function used in the priority-driven search study.

Comparison to Global Shape Descriptors: Figure 7.7 shows a precision recall plot comparing retrieval with a single global descriptor versus using 10 descriptors with high distinction values. Higher lines indicate better retrieval performance. Also consider Table 7.1 that shows timing results and DCG scores for various configurations. For this experiment, ten descriptors were used from the query. Using these ten distinctive descriptors improves retrieval performance beyond a single global descriptor. To be fair, shape matching with a global descriptor is faster than with local descriptors, but the improved retrieval accuracy may be worth the extra time for certain applications.

Effect of selecting fewer descriptors: We also considered how retrieval performance varies with k, the number of descriptors selected for each query model. Figure 7.8 shows

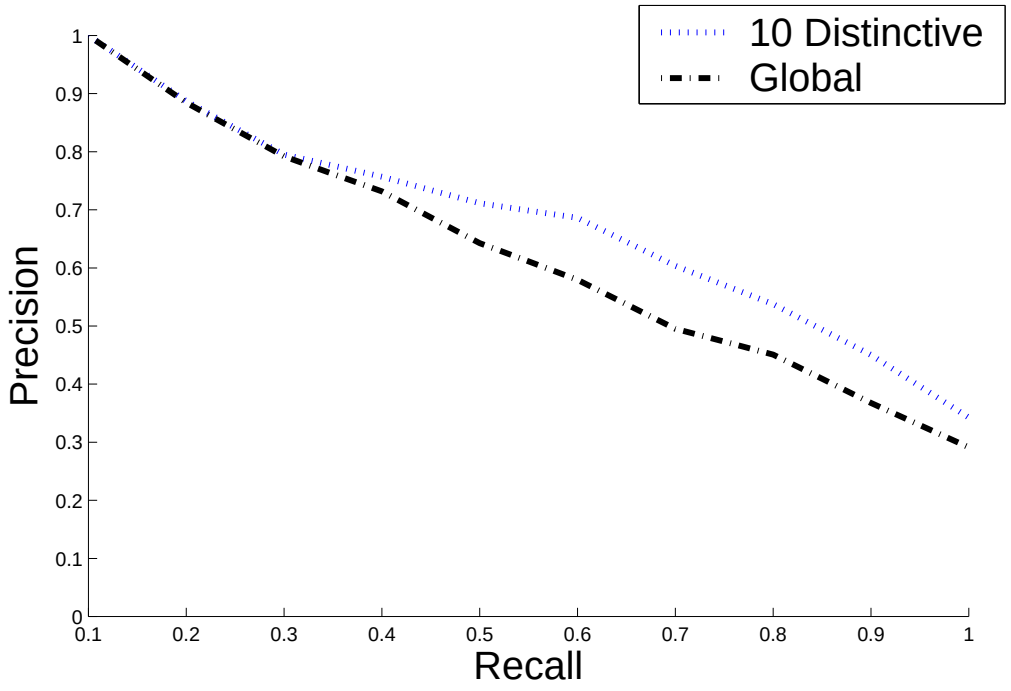


Figure 7.7: Using ten distinctive HSD descriptors improves retrieval compared to using a single global descriptor.

	Timing Results			
Descriptors	Generate Descriptors	Calculate Likelihood	Compare Descriptors	Retrieval DCG
Global HSD	0.35s	NA	0.000009s	0.762
3 HSD	81.5s	3.7s	0.0057s	0.785
10 HSD	81.5s	3.7s	0.018s	0.794
2,000 HSD	81.5s	NA	2.18s	0.796
Global SD	0.35s	NA	0.000001s	0.638
3 SD	68.7s	0.1s	0.0007s	0.679
10 SD	68.7s	0.1s	0.0016s	0.718
2,000 SD	68.7s	NA	0.56s	0.735

Table 7.1: Using a few distinctive features provides better matching results than a global descriptor and is faster than using the full set of local descriptors, with a modest decrease in retrieval accuracy. All timing results are for experiments on a computer running the Windows XP operating system on an Intel Pentium 4 processor running at 3 GHz with 1 GB of RAM.

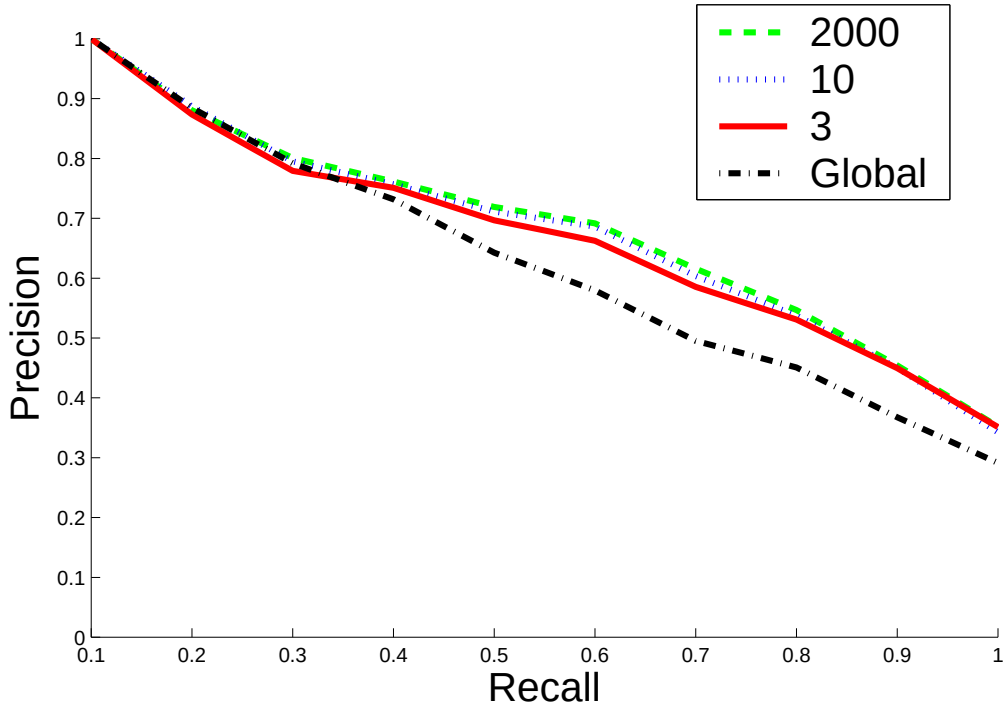


Figure 7.8: Performance decreases gradually as the number of distinctive HSD descriptors is reduced.

the retrieval performance when using different numbers of query descriptors. For most values of $k > 3$, retrieval performance remains almost as high as when using all 2,000 descriptors. This result shows that using a small number of distinctive descriptors can approximate the retrieval result of using the full set. Meanwhile, Table 7.1 shows that comparing a query shape against a shape in the database using the three most distinctive descriptors takes only $\frac{1}{350}$ of the time for using all 2,000. This combination provides a significant time savings with minimal loss of retrieval precision.

Comparison to other selection methods: We next evaluated how well our predicted distinction function compared to previous techniques for selecting local descriptors. We compared against three alternative approaches:

Least Likely DB For each model, the descriptors are sorted based on their likelihood as calculated from the distribution for the entire database.

Least Likely Model For each model, the descriptors are sorted based on their likelihood as calculated based on the distribution of descriptors for the model.

Random The descriptors are randomly sorted, providing a baseline for comparison.

Figure 7.9 shows the retrieval performance when combining descriptors with $k = 3$. In this plot, the vertical axis shows the percentage improvement over Random. Results for both SD and HSD descriptors are shown. Selecting the k descriptors with highest predicted distinction scores outperforms Global as well as Random, Least Likely DB, and Least Likely Model for most recall values. It should be noted that as k increases, the difference between all of the techniques decreases, since each shape becomes fully represented with the local descriptors.

This result demonstrates that distinctive features are generally better for retrieval than other approaches that focus on likelihood without consideration of how likelihood relates to retrieval performance. While this is the only retrieval result shown for the SD descriptor, our results on other experiments are consistent for both the SD and HSD descriptors.

7.3 Conclusion

The main contribution of our work is a method for selecting a subset of local shape descriptors for each query shape to use during matching. We map descriptors based on their likelihood and calculate the average distinction for each descriptor within a likelihood bin. From training data, we can efficiently predict distinction scores for descriptors from a query through a likelihood mapping.

During our experiments, we have demonstrated several important properties of distinctive descriptors. Descriptors with similar likelihoods have similar retrieval performance. However, the least likely descriptors do not have the best retrieval performance – although they are rarest, they are not the most distinctive. Rather, descriptors with intermediate likelihoods provide the best retrieval performance, and thus it is valuable to store a mapping from likelihood to retrieval performance and to use that mapping for selecting query descriptors during shape matching. We find that distinctive descriptors from the query can be combined to improve retrieval over using a single global descriptor, and a small subset of distinctive descriptors can approximate the retrieval performance of the full set while decreasing retrieval times. We also find that distinctive descriptors are better for retrieval than alternative approaches such as either selecting descriptors randomly or selecting those that are least likely.

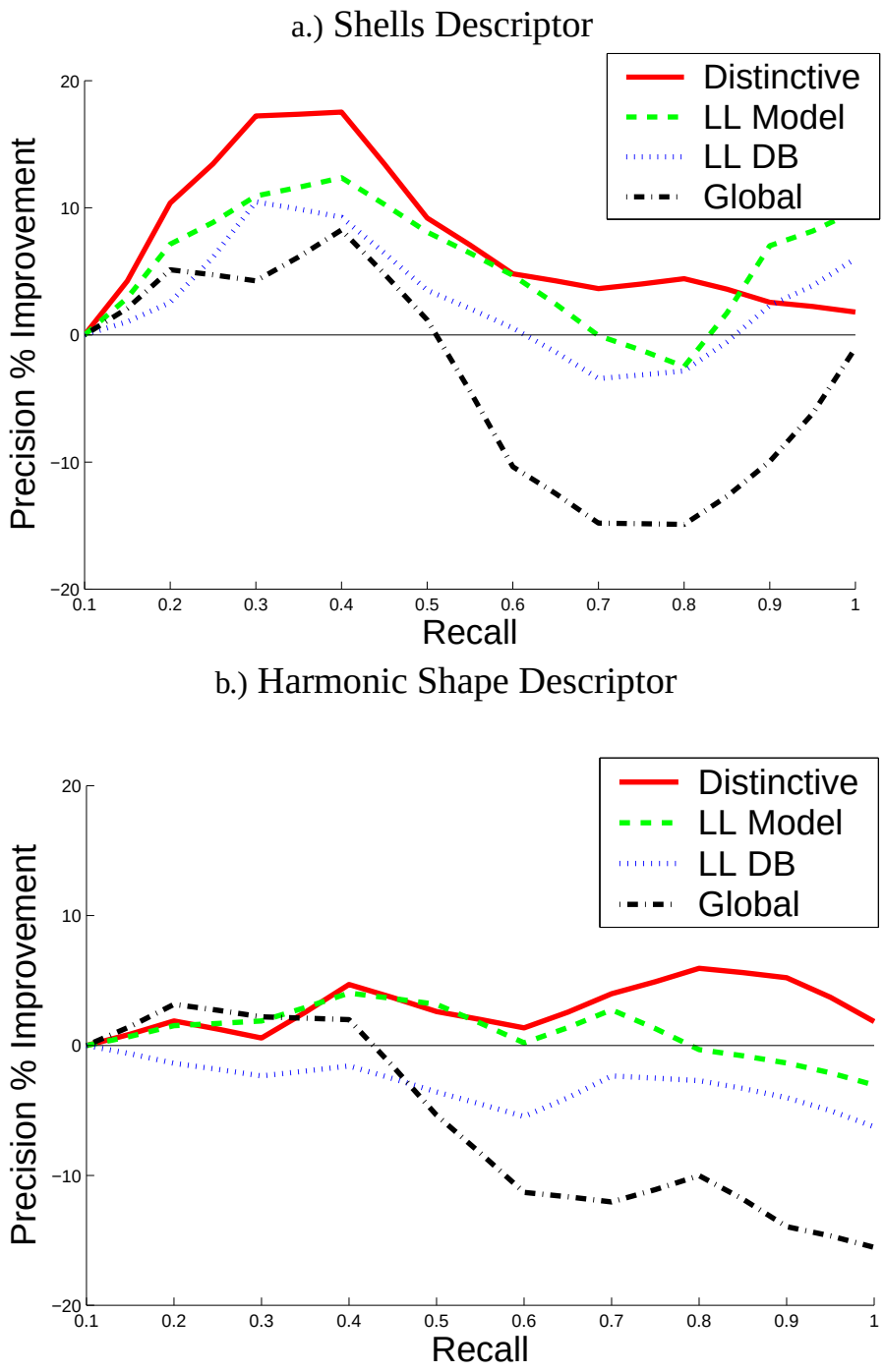


Figure 7.9: Distinctive descriptors have better retrieval performance than using randomly selected descriptors, least likely (LL) descriptors, or a global descriptor. Precision values are shown as improvement over randomly selected descriptors.

Chapter 8

Applications of Distinction

Finding distinctive regions on a mesh is potentially useful for numerous graphics applications beyond shape matching. With mesh processing algorithms, an importance score over the surface of a mesh can often provide useful information to guide which regions should be processed/retained most fully. For example, a remeshing process may allocate more polygons for important regions, and an alignment algorithm may use a cost function that places more weight on aligning distinctive regions of similar meshes. In this chapter, we consider two applications: mesh simplification and icon generation (Shilane and Funkhouser [113]).

8.1 Mesh Simplification

Creating a simplified version of a complex mesh is important for many applications. For example, consider an online parts catalog where images of many tools are shown on-screen at the same time. To improve rendering times, the mesh representing each tool might be simplified, since rendering time is related to the number of polygons representing a shape. However, to preserve object recognition and emphasize differences within a large collection of meshes, the distinctive features of each tool should be simplified less than the rest of the mesh.

Most techniques for mesh simplification have focused on minimizing the geometric error in a simplified mesh (e.g., [41]), while others have attempted to minimize errors

in the rendered images. In particular, Lee et al. [77] used their estimation of mesh saliency to weight vertices in a quadrics-based simplification scheme. We follow this work by weighting vertices instead with mesh distinction scores. Since surface distinction identifies parts that are consistent within a class and distinguish from other classes, we expect the simplification algorithm to preserve distinctive features better than other approaches. While features that are salient to the human visual system may not necessarily be preserved with our shape-matching approach, distinguishing features will be preserved while common features are simplified, which, under extreme simplification, will produce a mesh caricature.

To review, quadric error simplification works by contracting an edge of a mesh with the least quadric error. The quadric error for each vertex is a measure of how far that vertex has moved during simplification. Consider all planes incident to vertex v, where each plane p is represented by normal vector (a,b,c) and offset d as augmented vector (a,b,c,d). The quadric error for v is the squared distance to the set of all such incident planes $p \in P$, $E_v = v^t(\sum_{p \in P} pp^t)v$, and the error for an edge is the sum of the error for the two vertices on the edge. When an edge is selected for contraction, the optimal position for the new vertex v' is selected that minimizes the error. Then, the error for v' is the sum of the errors for the two removed vertices.

We augment this basic algorithm by adjusting the error for each edge based on how distinctive its two vertices are. If D_v is the distinction of mesh regions mapped onto vertex v as described in Section 4.1.4, then the new error for every edge e is $E_e = D_{v_1}Q_{v_1} + D_{v_2}Q_{v_2}$. To accentuate the difference between distinctive and non-distinctive regions, however, the distinction scores for the lower 65% of vertices was set to the minimal distinction value. After each edge is collapsed, the new vertex is assigned an error that is the maximum of the two vertices collapsed so that distinctive regions preserve their scores without being averaged with nearby areas.

Simplification results achieved with this method are shown in Figures 8.1 and 8.2. For the hammer example shown in Figure 8.1, descriptors were computed for 1024 regions at scale 0.25, and distinction was computed within the context of a database containing four hammers among nineteen meshes representing other classes of tools (screwdrivers, wrenches, and shovels). For a database of tools, the distinguishing features are generally at the functional end of the object away from the handle. The mesh for this hammer was then simplified from 1,700 triangles to 300 triangles (Figure 8.1) using the distinction-

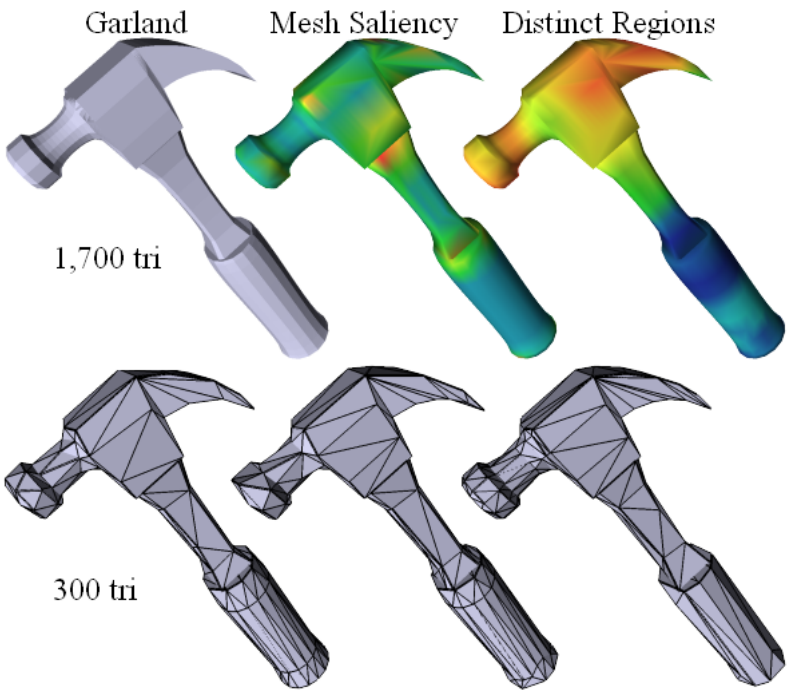


Figure 8.1: Simplification results using Garland's method, mesh saliency, and distinctive regions. Notice that more detail is preserved in the head of the hammer by focusing on distinctive regions.

weighted error metric. Note that the head of the hammer is the most distinctive region of the mesh and remains well-preserved. For comparison sake, we show simplifications to the same triangle count achieved using Garland's standard quadric error in the first column and using Lee's method of weighting the quadric error by mesh saliency in the third column. Note that our method preserves the head of the hammer, the most distinctive part, better than these other methods.

Figure 8.2 shows similar results achieved when simplifying the mesh of a horse. In this case, the head was found to be most distinctive in the context of a database containing four horses among five other classes of quadrupeds (rabbit, dog, feline, and pig). So, when the mesh was simplified from 97K triangles to 2K triangles, the fine details in and around the horse's head are well-preserved, while the body is greatly simplified. Since the competing methods do not identify these important regions of the horse as strongly, they provide more simplification to the head, while preserving detail in the creases of the body.

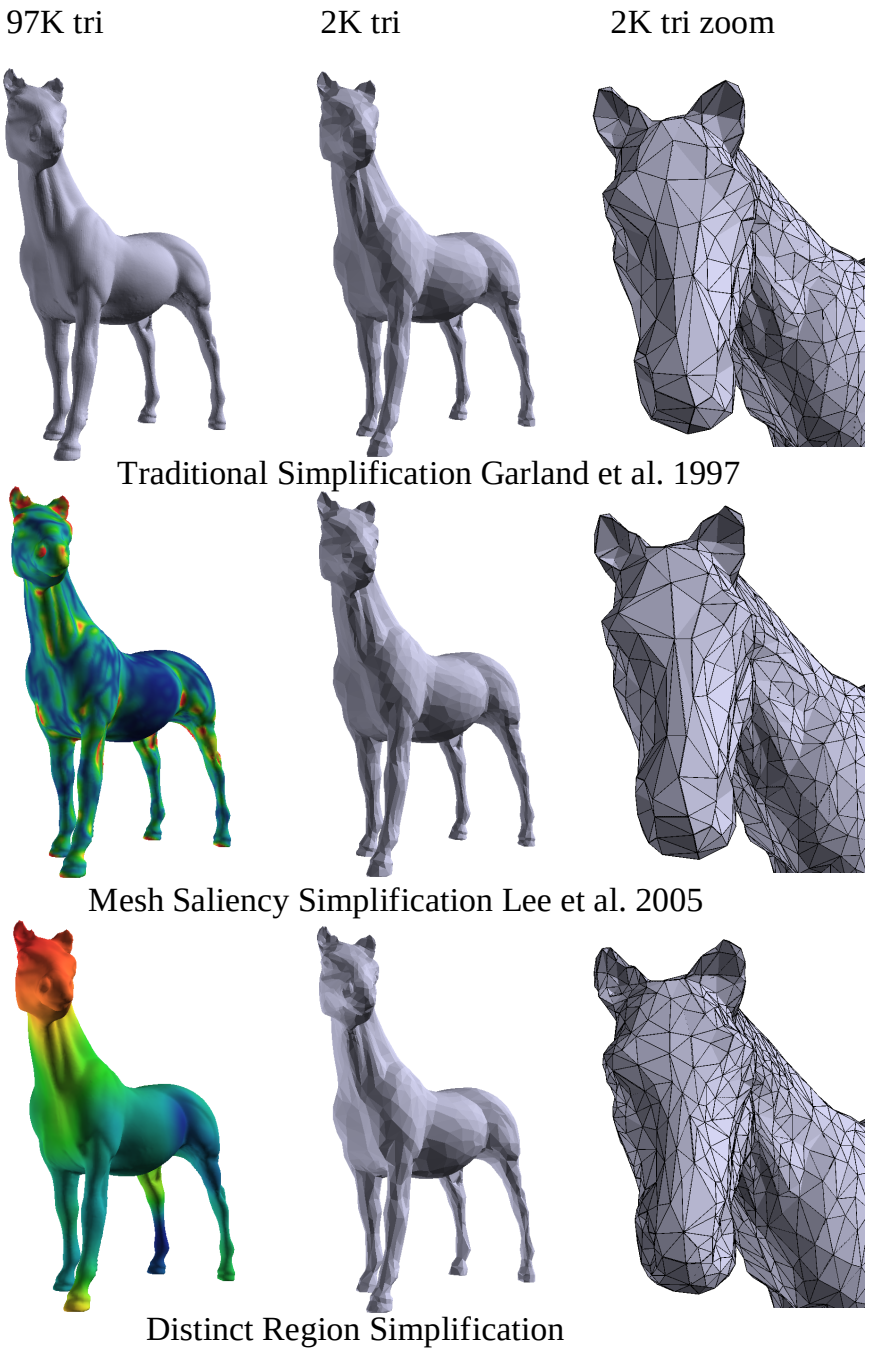


Figure 8.2: Simplification results using Garland's method, mesh saliency, and distinctive regions. Notice that details of the eyes and nose are better preserved using our method, while using mesh saliency, areas are preserved throughout the horse's body.

8.2 Icon Generation

With the increasing size of 3D model collections, quickly presenting models to a user is an ongoing problem. Whether the application is viewing a catalog of objects or presenting retrieval results in a search engine, the important features of shapes must be shown clearly to the user, perhaps with icons. Focusing an icon on the features that distinguish different classes of shapes could help increase comprehension.

Most previous work on icon generation has focused on the problem of viewpoint selection, that is, positioning a camera oriented towards the center of the object. Vázquez et al. [126] selected the position that maximized the entropy of the viewed mesh, where the optimal view would see all of the faces of the mesh with the same relative projected area. Blanz et al. [13] studied the preferences of users and found that views at an angle to the major axis were selected. Using their own definition of mesh saliency, Lee et al. [77] selected views that maximized the visibly salient regions.

We have developed a method for generating icons that displays only the most distinctive region of a mesh. Our algorithm is quite simple. After computing shape descriptors at the 0.25 scale for many points on the mesh, we select the single most distinctive position with respect to a chosen database. We then produce an image of the mesh zoomed in such that the view frustum exactly covers that most distinctive region. The rotation of the camera is chosen by the computer automatically with one of many possible heuristics or by interactive user control.

We find that this simple method produces useful icons for many classes of objects. For example, Figure 8.3 shows automatically generated icons for six shapes in the Princeton Shape Benchmark. For many meshes (such as the turtle, wrench, and car), large and recognizable features are visible in the icon. Showing a limitation of our approach, the biplane icon is focused on the tail region because that region distinguished planes from many other classes of shapes, but perhaps the tail is not the most semantically important feature to humans. However, it should be clearly stated that our measure of distinction is based on 3D shape matching not 2D image matching, and thus it is not guaranteed that the regions determined to be most distinctive by our method will match the ones most visually recognizable by a human. Nonetheless, we find that our simple method based on mesh distinction produces good icons in most cases.

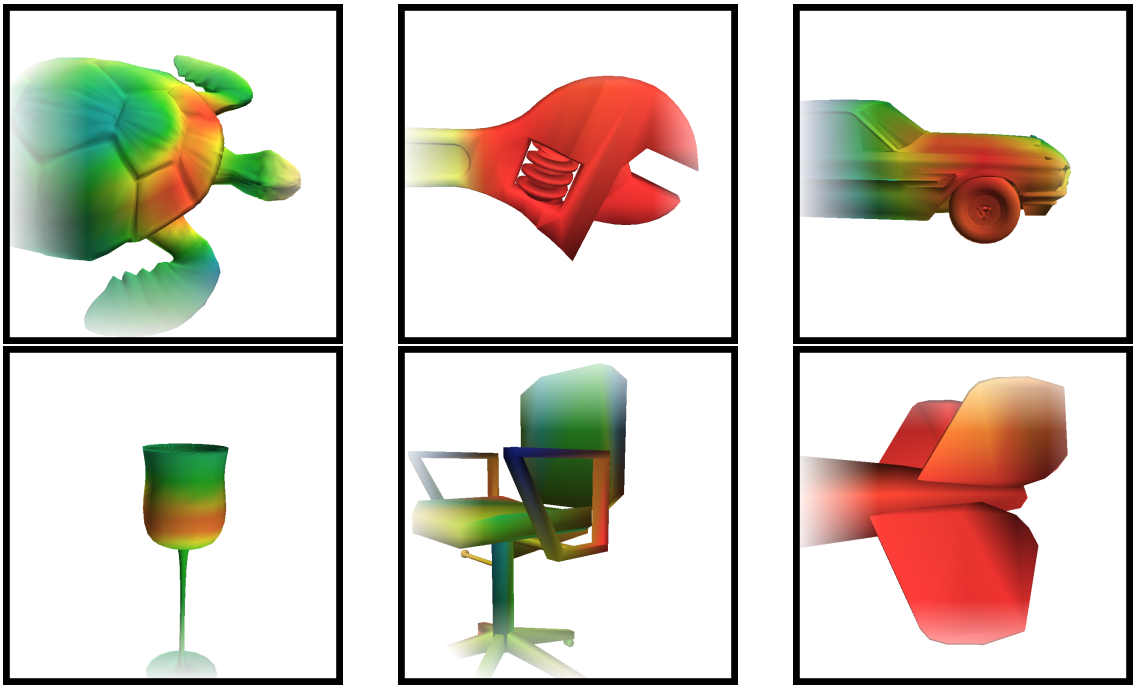


Figure 8.3: Icons showing the most distinctive surface region for each mesh.

8.3 Conclusion

We have demonstrated that shape distinction can be incorporated into computer graphics applications by focusing an algorithm on the distinctive regions of a mesh. During mesh simplification, important regions are preserved that distinguish an object from others in the database even at greatly reduced polygon counts. When creating icons of numerous models in a database, we have shown how to produce reasonable images by positioning the camera to focus on distinctive features.

Our definition of distinction has focused on shape similarity among 3D surface regions and directly improves shape matching. Other approaches based on distinction in 2D images of 3D shapes would likely be better for visualization applications. Even with these limitations, we have shown that shape distinction focuses mesh simplification and icon generation on important regions of shapes. Our applications of distinction are first steps – we believe that there is a wealth of new ways to utilize mesh distinction in these and other applications in computer graphics.

Chapter 9

Princeton Shape Benchmark

Introduction

To analyze shape distinction and perform the type of shape retrieval experiments shown throughout this dissertation, a standardized benchmark of classified shapes is needed. Despite decades of research on 3D shape representations and matching algorithms [81, 127], there still are no standard ways of comparing the results achieved with different methods. Usually, computed match results are evaluated according to how well they correlate with human-generated classifications. However, it seems that each research group has its own database of 3D models, own classifications, own suites of tests, and own metrics of success. Moreover, few publications contain results of tests comparing several approaches on the same data [19, 36, 78, 131].

In this chapter, we describe the Princeton Shape Benchmark (PSB), a publicly-available database of 3D models, software tools, and a standardized set of experiments for comparing 3D shape matching algorithms (Shilane et al. [114]). The database contains 1,814 polygonal models collected from the World Wide Web and classified by humans according to function and form. It includes a set of hierarchical classifications, separate training and test sets, annotations for each model, and a suite of software tools for generation, analysis, and visualization of shape matching results. The PSB classification and tools were used throughout this dissertation.

The main contribution of the Princeton Shape Benchmark is the proposed framework for comparison of shape matching algorithms. We demonstrate its use by exposing the differences between fourteen shape descriptors, including D2 shape distributions [98], Extended Gaussian Images [52, 64], Shape Histograms [3], Spherical Extent Functions [109, 130], Gaussian Euclidean Distance Transforms [69], Spherical Harmonic Descriptors [69], Light Field Descriptors [19], and Depth Buffer Descriptors [48].

In short, we find that no single shape descriptor performs best for all classifications, and no single classification provides the best evaluation of all shape descriptors. From this result, we conclude that it is only possible to understand the differences between shape descriptors by looking at the results of several experiments aimed at testing specific properties. The Princeton Shape Benchmark provides a standardized framework for this type of experimentation.

9.1 Related Work

The benefits of benchmarks have been well-demonstrated in many fields. For example, in computer architecture, the SPEC benchmarks [117] have been used successfully to compare processor performance. In text document retrieval, the Smart Collection [108] and TREC database [123] provide standard benchmarks. In computer vision, benchmarks are available for handwriting recognition (e.g., [76]), face recognition (e.g., [22]), and several other image analysis tasks [25]. There are even databases for specific types of 3D data – e.g., computer-aided design parts [31, 61] and protein structures [10].

Unfortunately, no standard benchmarks are available for matching of 3D polygonal models representing a wide variety of objects. Instead, several research groups have independently gathered databases of 3D models, generated different classifications, run different sets of tests, employed different metrics to quantify performance, and compared different shape descriptors.

Table 9.1 shows statistics for several 3D model databases currently in use for shape matching experiments. For each database, the table shows the total number of 3D models in the database, the number of classes, the number of models that have been classified, and the percentage of classified models in the largest class. Also, estimates of what percentage of classified models belong to different object types (vehicle, household,

animal, plant, architecture) appear in Table 9.2. The bottom row of each table shows statistics for the Princeton Shape Benchmark for comparison. From these statistics, we make several observations.

Database	Num Models	Num Classes	Num Classified	Largest Class
Osada [98]	133	25	133	20%
MPEG-7 [139]	1,300	15	227	15%
Hilaga [49]	230	32	230	15%
Technion [78]	1,068	17	258	10%
Zaharia [140]	1,300	23	362	14%
CCCC [130]	1,841	54	416	13%
Utrecht [122]	684	6	512	47%
Taiwan [19]	1,833	47	549	12%
Viewpoint [36]	1,890	85	1,280	12%
PSB	6,670	161	1,814	6%

Table 9.1: Summary of previous 3D model databases.

	Vehicles	Furniture	Animals	Plants	Household	Buildings
Osada [98]	47%	12%	12%	0%	24%	0%
MPEG-7 [139]	12%	0%	14%	13%	0%	7%
Hilaga [49]	12%	0%	23%	2%	12%	0%
Zaharia [140]	35%	0%	7%	7%	11%	0%
CCCC [130]	33%	13%	21%	5%	25%	0%
Utrecht [122]	73%	0%	0%	0%	0%	0%
Taiwan [19]	44%	13%	0%	0%	36%	0%
Viewpoint [36]	0%	42%	1%	0%	50%	0%
PSB	26%	11%	16%	8%	22%	6%

Table 9.2: Types of objects found in previous 3D model databases (shown as percentages of classified models).

First, most previous databases contain a small number of classified models. For example, the Osada database [98], which has been used in experiments by several research groups (e.g., [122]), contains only 133 models. Some of them appear in classes with only 2 other models, which makes it difficult to acquire statistically significant results in classification experiments. In other cases, the total number of 3D models in the database is quite large (> 1800), but only a small fraction of them are included in the classification.

For instance, the MPEG-7 database [139] contains 1,300 VRML models in all. But, only 227 (18%) of them are included in labeled classes, while the vast majority of models are lumped into a "miscellaneous" class that provides only "background noise" during shape matching experiments. To our knowledge, the only set of more than 1000 classified 3D polygonal models used for shape matching experiments is the Viewpoint database [128], as described in [36]. However, it is not available to the general public, and it is expensive to purchase, which makes its use as a standard benchmark problematic.

Second, most 3D model databases contain a limited range of object types and/or are dominated by a small set of object classes (see Table 9.2). For example, the Viewpoint database [36] contains only household objects, and the Utrecht database [122] contains mainly vehicles among its classified models. Even databases that have a wide variety of objects often contain a few classes with a disproportionately large number of models. For example, the MPEG-7 database contains 50 (22%) models representing letters of the alphabet among its 227 classified objects, and the Osada database contains 27 (20%) airplanes out of 133 objects. Of course, these large classes significantly bias (micro-) averaged retrieval results.

Third, current 3D model classifications have significantly different granularities. Some databases have classes with large, diverse sets of objects (e.g., "Kitchenware" [49]), while others have very small and specific classes (e.g., "motorcycles with 3 wheels" [140]). For example, the National Taiwan University database [19] has a single class containing all types of seats (dining room chairs, desk chairs, patio chairs, sofas, recliners, benches, and stools), while the Viewpoint database [36] has a separate class for each specific type. This difference in classification granularity can have an impact on retrieval and classification results, as significant differences between retrieval methods may be masked by classifications that are too coarse or too fine.

Finally, many 3D databases have classifications that mix function and form. For example, the MPEG-7 database contains several classes that group objects with similar semantics (e.g., "buildings"), while others group objects based solely on their shapes (e.g., the "aerodynamic" class contains fish, helicopters, and airplanes). Similarly, the Hilaga database [49] contains some classes corresponding grossly to functions (e.g., "Machine") and others corresponding directly to shapes (e.g., "Stick", "Donut", "Sphere", and "Many Holes"). Results achieved over these disparate class types are averaged together,

making it difficult to draw specific conclusions about why and when a shape matching method works well.

9.2 Overview

The Princeton Shape Benchmark provides a repository of 3D models and software tools for comparing shape matching algorithms. The motivation is to promote the use of standardized data sets and evaluation methods for research in matching, classification, clustering, and recognition of 3D models.

Version 1 of the benchmark contains a database of 1,814 classified 3D models collected from 293 different Web domains. For each 3D model, there is an Object File Format (.off) file with the polygonal surface geometry of the model, a textual information file containing meta-data for the model (e.g., the URL from whence it came), and a JPEG image file containing a thumbnail view of the model. We expect larger databases to be available in future versions.

In addition to the database of 3D models, the benchmark provides guidelines regarding its use. For instance, the 3D models are partitioned equally into training and test sets. The benchmark requires that algorithms be trained only on the training set (without influence of the test set); and then, after all exploration has been completed and all algorithmic parameters have been frozen, results should be reported for experiments with the test set.

In order to enable evaluation of shape matching algorithms for retrieval and classification tasks, the benchmark includes a simple mechanism to specify partitions of the 3D models into classes. In Version 1, we provide a hierarchical classification for 1,814 models (907 from the training set and 907 from the test set). At its finest granularity, this classification provides a tight grouping of objects based on both function and form. For example, there is a class for “birds in a flying pose” in the test database. Yet, it also includes a hierarchy of classes that reflects primarily the function of each object and secondarily its form. Continuing with the example, there are classes for “birds”, “flying creatures,” and “animals” at coarser levels of the hierarchy. Note that every level of the hierarchy is useful for a different type of evaluation.

Since arbitrarily many semantic groupings are plausible for a given set of 3D models, the benchmark provides a flexible mechanism for specifying multiple classifications. It

also includes a method for averaging over queries for models with certain geometric properties (e.g., "roughly spherical"). The differences in matching results achieved with respect to these different classifications and queries yield interesting insights into the properties of the shape retrieval algorithms being tested (e.g., algorithm X works better on round objects, but worse on elongated ones), and the combined results of multiple classifications provide a much more complete view of the differences between competing algorithms.

To standardize analysis of shape matching experiment results, the benchmark includes free source code for evaluation and visualization of 3D model matching scores. For instance, there are programs for: 1) generating precision-recall plots, 2) computing several retrieval statistics (e.g., nearest neighbor, 1st and 2nd tier, discounted cumulative gain, etc.), 3) producing color-coded similarity matrices, and 4) constructing web pages with thumbnails of the best ranked matches for a given query model. These programs provide a standard toolbox with which researchers can compare results of independently run tests in a consistent manner.

In summary, the benchmark provides a flexible framework for comparing shape matching algorithms. The remainder of the chapter describes many of the design decisions and issues that were addressed during its construction. Specifically, detailed descriptions of how our database was acquired, classifications were constructed, and models were annotated appear in Sections 9.3-9.5. Section 9.6 describes our software tools for evaluating matching results, and Section 9.7 presents experimental results obtained during tests with several different shape descriptors, classifications, and databases. Finally, Section 9.8 summarizes our finding.

9.3 Acquisition

The 3D models in the PSB were acquired from the World Wide Web by three automated crawls over a two year period. This section describes how they were found, processed to remove duplicates, converted to a common file format, and organized to form a database.

The first crawl was performed in October 2001 and targeted VRML files only. It began with the results of search engine queries for web pages linking to files with extension ".wrl" and ".wrz" and then crawled outward from those pages in a breadth-first fashion.

The crawl ran for 48 hours and downloaded 22,243 VRML files from 2,185 different Web sites [87].

The second crawl was executed in August 2002 and targeted VRML, 3D Studio, Autocad, Wavefront, and Lightwave objects, both in plain links as well as in compressed archive files (".tar" and ".zip"). Unlike VRML, the other formats were not designed to be used on the web and often are contained within compressed archives, so they typically cannot be located simply by file name extension. Instead, the second crawler searched for them using a focusing method, where web sites were crawled in priority order according to the number of pages already downloaded from that site containing 3D models. The crawl ran for 2 days and 16 hours and resulted in 13,217 3D model files and 5,539 compressed archive files containing 3D models. After expansion of archive files, there were 20,084 model files retrieved from 455 different sites [87].

The third crawl was executed in August 2003 and targeted models from known 3D model repositories (e.g., 3dcafe.com and avalon.viewpoint.com). The crawl ran for approximately 5 hours and resulted in 1,908 3D models in a variety of formats, downloaded from 16 different web domains.

These three crawls provided 44,235 model files. We ignored 3,763 of the models found in the second crawl because they had URLs in common with ones found in the first crawl. Another 6,863 models were thrown out because they contained no geometry or could not be parsed by our conversion software [97]. We culled 15,035 more models because their shapes were exact-duplicates or near-duplicates of some other model in the database. For example, we found multiple copies of the same model at different URLs (e.g., 483 spheres), multiple levels of detail for the same object, and different colors/textures for models with the same geometry. Finally, 11,904 models were eliminated manually because they came from application domains outside the scope of our benchmark. Specifically, we kept only models of "every-day objects," and threw out molecular structures, CAD parts, data visualizations, and abstract geometric shapes. The remaining 3D models form the database for our shape benchmark. In all, there are 6,670 unique models acquired from 661 distinct Web domains.

All of the remaining models were converted to the Object File Format (.off), a simple-to-parse polygonal format designed by the University of Minnesota Geometry Center [124]. During the conversion process, all color, texture, and scene graph information was eliminated, leaving a single indexed face set comprising a list of vertices (x,y,z) and a list of

polygons (v1, v2,...). We chose to make only these simple files available in the first version of the benchmark to focus matching experiments on geometric surface information only.

9.4 Classification

The PSB benchmark splits the 3D model database into training and test sets and partitions both test sets into classes (e.g., telephones, dogs, etc.) that can be used as labels in shape matching, retrieval, and classification experiments. In this section, we first explain how the models are partitioned into classes. Then, we discuss how training and test sets were formed. Finally, we describe the mechanisms provided for creating alternative classifications.

9.4.1 Base Classification

We manually partitioned the models of the benchmark database into a fine-grained set of classes. During this process, our goal was to create clusters of objects primarily based on semantic and functional concepts (e.g., furniture and table) and secondarily based on shape attributes (e.g., round tables). We use the hierarchical nature of this grouping strategy to form classifications at multiple granularities.

The steps used to produce our *base classification* proceeded as follows. First, we rendered thumbnail images for all 6,670 3D models and stored them in a single directory of a file system. Then, two students used Windows Explorer to create directories representing object classes and to move the thumbnail image files into the directories to indicate membership in the class. This process was executed iteratively until each class represented an atomic concept (e.g., a noun in the dictionary) and could not be partitioned further. Then, where appropriate, a few classes were further partitioned based on geometric attributes (e.g., “humans with arms out” versus “humans with arms down”). No textual information besides an integer model ID was available to the students (e.g., file names were hidden). So, we believe the students were not biased by auxiliary information during the formation of classes. The result of this process was a set of 1,271 classes partitioning the 6,670 models.

Many of the classes contained too few models to be included in meaningful experiments. For example, there were only two drill presses and three fire hydrants. So, we manually selected 161 classes, each containing at least four models, to be included in the first version of the benchmark (the other classes will be included in later versions). We also eliminated models from the largest classes (e.g., fighter jets and humans) so that every class contains at most 100 members ($\sim$6% of the classified models). The net result is our base classification, a set of 161 classes containing a total of 1,814 models.

9.4.2 Training and Test Sets

We then partitioned the models of the base classification into training and test sets. Our goal was to split the models as evenly as possible, producing two sets with similar types of classes, yet without splitting small classes, and without biasing either set with a large number of models of the same type. To meet these goals, we applied the following steps. First, all classes with 20 or more models were split equally between the training and test sets (models downloaded from the same domain were evenly distributed). Then, smaller classes were alternately placed in the training and test sets in a manner that ensured that both had a balanced number of classes for every object type (plants, animals, vehicles, etc.). Finally, we manually swapped a few small classes so that the training and test sets have an equal number of models. The final result is two sets of classified 3D models, one with 907 models partitioned into 90 classes to be used for training the parameters of shape matching algorithms, and the other with a different 907 models partitioned into 92 classes to be used for comparison with other algorithms. Detailed lists of the classes in both sets appear in Table 9.3.

Training		Test	
aircraft/airplane/F117	4	aircraft/airplane/biplane	14
aircraft/airplane/biplane	14	aircraft/airplane/commercial	11
aircraft/airplane/commercial	10	aircraft/airplane/fighter_jet	50
aircraft/airplane/fighter_jet	50	aircraft/airplane/glider	19
aircraft/airplane/multi_fuselage	7	aircraft/airplane/stealth_bomber	5
aircraft/balloon_vehicle/dirigible	7	aircraft/balloon_vehicle/hot_air_balloon	9
aircraft/helicopter	17	aircraft/helicopter	18
aircraft/spaceship/enterprise_like	11	aircraft/spaceship/enterprise_like	11
aircraft/spaceship/space_shuttle	6	aircraft/spaceship/flying_saucer	13
aircraft/spaceship/x_wing	5	aircraft/spaceship/satellite	7
animal/arthropod/insect/bee	4	aircraft/spaceship/tie_fighter	5
animal/arthropod/spider	11	animal/arthropod/insect/ant	5
animal/biped/human	50	animal/arthropod/insect/butterfly	7
animal/biped/human/arms_out	21	animal/biped/human	50
animal/biped/trex	6	animal/biped/human/arms_out	20
animal/flying_creature/bird/duck	5	animal/biped/human/walking	8
animal/quadruped/apatosaurus	4	animal/flying_creature/bird/flying	14
animal/quadruped/feline	6	animal/flying_creature/bird/standing	7
animal/quadruped/pig	4	animal/quadruped/dog	7
animal/underwater_creature/dolphin	5	animal/quadruped/horse	6
animal/underwater_creature/shark	7	animal/quadruped/rabbit	4
blade/butcher_knife	4	animal/snake	4
blade/sword	15	animal/underwater_creature/fish	17
body_part/brain	7	animal/underwater_creature/sea_turtle	6
body_part/face	17	blade/axe	4
body_part/head	16	blade/knife	7
body_part/skeleton	5	blade/sword	16
body_part/torso	4	body_part/face	16
bridge	10	body_part/hand	17
building/castle	7	body_part/head	16
building/dome_church	13	body_part/skull	6
building/lighthouse	5	book	4
building/roman_building	12	building/barn	5
building/tent/multiple__peak_tent	5	building/church	4
building/two_story_home	11	building/gazebo	5
chess_piece	17	building/one_story_home	14
chest	7	building/skyscraper	5
city	10	building/tent/one_peak_tent	4
computer/laptop	4	building/two_story_home	10

Continued on next page

Table 9.3 – continued from previous page

Training		Test	
display_device/tv	12	chess_set	9
door/double_doors	10	city	10
fantasy_animal/dragon	6	computer/desktop	11
furniture/bed	8	display_device/computer_monitor	13
furniture/desk/desk_with_hutch	7	door	18
furniture/seat/chair/dining	11	eyeglasses	7
furniture/seat/chair/stool	7	fireplace	6
furniture/seat/couch	15	furniture/cabinet	9
furniture/shelves	13	furniture/desk/school	4
furniture/table/rectangular	26	furniture/seat/bench	11
furniture/table/round	12	furniture/seat/chair/dining	11
furniture/table_and_chairs	5	furniture/seat/chair/desk	15
gun/handgun	10	furniture/shelves	13
gun/rifle	19	furniture/table/rectangular	25
hat/helmet	10	furniture/table/round/single_leg	6
ice_cream	12	geographic_map	12
lamp/desk	14	gun/handgun	10
liquid_container/bottle	12	hat	6
liquid_container/mug	7	hourglass	6
liquid_container/tank	5	ladder	4
liquid_container/vase	11	lamp/streetlight	8
microchip	7	liquid_container/glass_with_stem	9
microscope	5	liquid_container/pail	4
musical_instrument/guitar/acoustic	4	liquid_container/vase	11
musical_instrument/piano	6	mailbox	7
phone_handle	4	musical_instrument/guitar/electric	13
plant/flower_with_stem	15	newtonian_toy	4
plant/potted_plant	25	plant/bush	9
plant/tree	17	plant/flowers	4
plant/tree/barren	11	plant/potted_plant	26
plant/tree/palm	10	plant/tree/barren	11
sea_vessel/sailboat	5	plant/tree/conical	10
sea_vessel/sailboat/sailboat_with_oars	4	satellite_dish	4
sea_vessel/ship	10	sea_vessel/sailboat/large_sail_boat	6
shoe	8	sea_vessel/ship	11
sign/street_sign	12	sea_vessel/submarine	9
skateboard	5	sign/billboard	4
snowman	6	sink	4

Continued on next page

Table 9.3 – continued from previous page

Training		Test	
swingset	4	slot_machine	4
tool/screwdriver	5	staircase	7
tool/wrench	4	tool/hammer	4
vehicle/car/antique	5	tool/shovel	6
vehicle/car/sedan	10	umbrella	6
vehicle/car/sports	19	vehicle/car/race	14
vehicle/cycle/bicycle	7	vehicle/car/sedan	10
vehicle/military_tank	16	vehicle/covered_wagon	5
vehicle/pickup_truck	8	vehicle/cycle/motorcycle	6
vehicle/suv	4	vehicle/monster_truck	5
vehicle/train	7	vehicle/semi	7
watch	5	vehicle/suv/jeep	5
wheel/tire	4	vehicle/train/train_car	5
		wheel	4
		wheel/gear	9
Total	907	Total	907
Overall Total = 1,814			

Table 9.3: The PSB base classification.

9.4.3 Alternative Classifications

There are many possible classifications for a given set of 3D models. For instance, one person might group models based primarily on function (e.g., like our base classification), while another might group them according to how the objects are constructed (e.g., man-made versus natural), where they are used (e.g., office versus home versus outdoors), or who uses them (e.g., adults versus children). We believe that the results of shape retrieval experiments for multiple classifications are interesting, as they provide information about the circumstances in which each shape matching algorithm performs well/poorly. The cumulative results of experiments with multiple classifications can provide a more complete picture of the differences between competing shape matching algorithms than does any single classification alone.

To support multiple classifications, the benchmark includes a simple language in which researchers can define new classifications. Briefly, an ASCII file is used to specify a hierarchy of class names and to indicate which models belong to each class. We have used this language to create three alternatives to the base classification, each representing a different granularity of grouping. For instance, a coarse classification merges all types of tables into a single class, a coarser classification merges all furniture into one class, and the coarsest partitions objects based only whether they are man-made or appear naturally in the real world. We use these alternative classifications to compare shape matching algorithms in Section 9.7.

In the future, we expect that other researchers will use the language to define new classifications that we did not anticipate, thereby adding to the suite of experiments that can be used to compare shape matching algorithms.

9.5 Annotation

The benchmark includes several types of auxiliary information for each model in the database. For instance, the following meta-data is provided to help identify the source and object type for each model:

- **Model URL:** the Web address where the model was found on the Web. The last part of the URL provides the model's file name, which may be useful for semantic labeling. More importantly, the URL can be used to determine the owner of the model for assigning credit and attribution.
- **Referring URL:** the address of the Web page containing a link to the model. The textual anchor and context on this page may be useful for extracting information about the model (if the Web page still exists).
- **Thumbnail image:** an image of the model rendered with colors and textures as seen from a plausible viewpoint. This view of the model with all its surface attributes is useful for seeing what the model looked like in its original form.

In addition, the benchmark lists several geometric attributes for each 3D model (e.g., number of polygons, average dihedral angle, averaged depth complexity over all views, etc.), which are useful for identifying interesting subsets of the database. While these

attributes could be derived from the models, and thus are somewhat redundant, they provide a standardized set of values that can be used to avoid the risk that differences in implementations can cause differences in matching results. For instance, the following attributes provide useful data for normalizing 3D models for differences in translation, scale, and orientation:

- **Center of mass:** the average (x,y,z) coordinates for all points on the surfaces of all polygons. These values can be used to normalize models for translations.
- **Scale:** the average distance from all points on the surfaces of all polygons to the center of mass. This value can be used to normalize models for isotropic scales.
- **Principal axes:** the eigenvectors (and associated eigenvalues) of the covariance matrix obtained by integrating the quadratic polynomials $x_i \cdot x_j$, with $x_i \in \{x,y,z\}$, over all points on the surfaces of all polygons. These axes can be used to normalize models for rotations.

9.6 Evaluation

The benchmark includes several tools for evaluating and comparing how well shape matching algorithms work. These tools assume that every algorithm being evaluated can compute the "distance" between any pair of 3D models, producing positive values that are small if the models are similar and larger for pairs with greater shape differences. So, for a given shape matching algorithm and database of 3D models, we can compute a *distance matrix*, where element (i, j) represents the computed distance between models i and j. Similarly, for any given model Q, we can rank the others from best to worst according to their computed distances from Q. This *ranked list* corresponds to the retrieval result that would be returned if Q were provided as a query to a shape-based search engine.

Given a classification and a distance matrix computed with any shape matching algorithm, a suite of PSB benchmark tools produces statistics and visualizations that facilitate evaluation of the match results (i.e., how many of the top ranked models are from the same class as the query). We include detailed descriptions so that the reader can get a feel for the tools available in the benchmark and can understand the results presented in Section 9.7.

- **Best matches:** a web page for each model displaying images of its best matches in rank order. The associated rank and distance value appears below each image, and images of models in the query model's class (hits) are highlighted with a thickened frame. This simple visualization provides a qualitative evaluation tool emulating the output of many 3D model search engines (e.g., [19, 29, 36, 71, 102, 120, 130, 133, 142]).

- **Precision-recall plot:** a plot describing the relationship between precision and recall in a ranked list of matches. For each query model in class C and any number K of top matches, "recall" (the horizontal axis) represents the ratio of models in class C returned within the top K matches, while "precision" (the vertical axis) indicates the ratio of the top K matches that are members of class C. A perfect retrieval result produces a horizontal line across the top of the plot (at precision = 1.0), indicating that all the models within the query object's class are returned as the top ranked matches. Otherwise, curves that appear shifted up represent superior retrieval results (see Figure 9.2).

- **Distance image:** an image of the distance matrix where the lightness of each pixel (i, j) is proportional to the magnitude of the distance between models i and j [98]. Models are grouped by class along each axis, and lines are added to separate classes, which makes it easy to evaluate patterns in the match results qualitatively – i.e., the optimal result is a set of darkest, class-sized blocks of pixels along the diagonal indicating that every model matches the models within its class better than ones in other classes. Otherwise, the reasons for poor match results can often be seen in the image – e.g., off-diagonal blocks of dark pixels indicate that two classes match each other well.

- **Tier image:** an image visualizing nearest neighbor, first tier, and second tier matches [98]. Specifically, for each row representing a query with model j in a class with $|C|$ members, pixel (i, j) is: (a) black if model i is model j or its nearest neighbor, (b) red if model i is among the $|C|-1$ top matches (the first tier), and blue if model i is among the $2*(|C|-1)$ top matches (the second tier). As with the distance image, models are grouped by class along each axis, and lines are added to separate classes. This image is often more useful than the distance image because the best matches are clearly shown for every model, regardless of the magnitude of their distance values. The optimal result is a set of black/red, class-sized blocks of pixels along

the diagonal indicating that every model matches the models within its class better than ones in other classes. Otherwise, more colored pixels in the class-sized blocks along the diagonal represents a better result (see Figure 9.1).

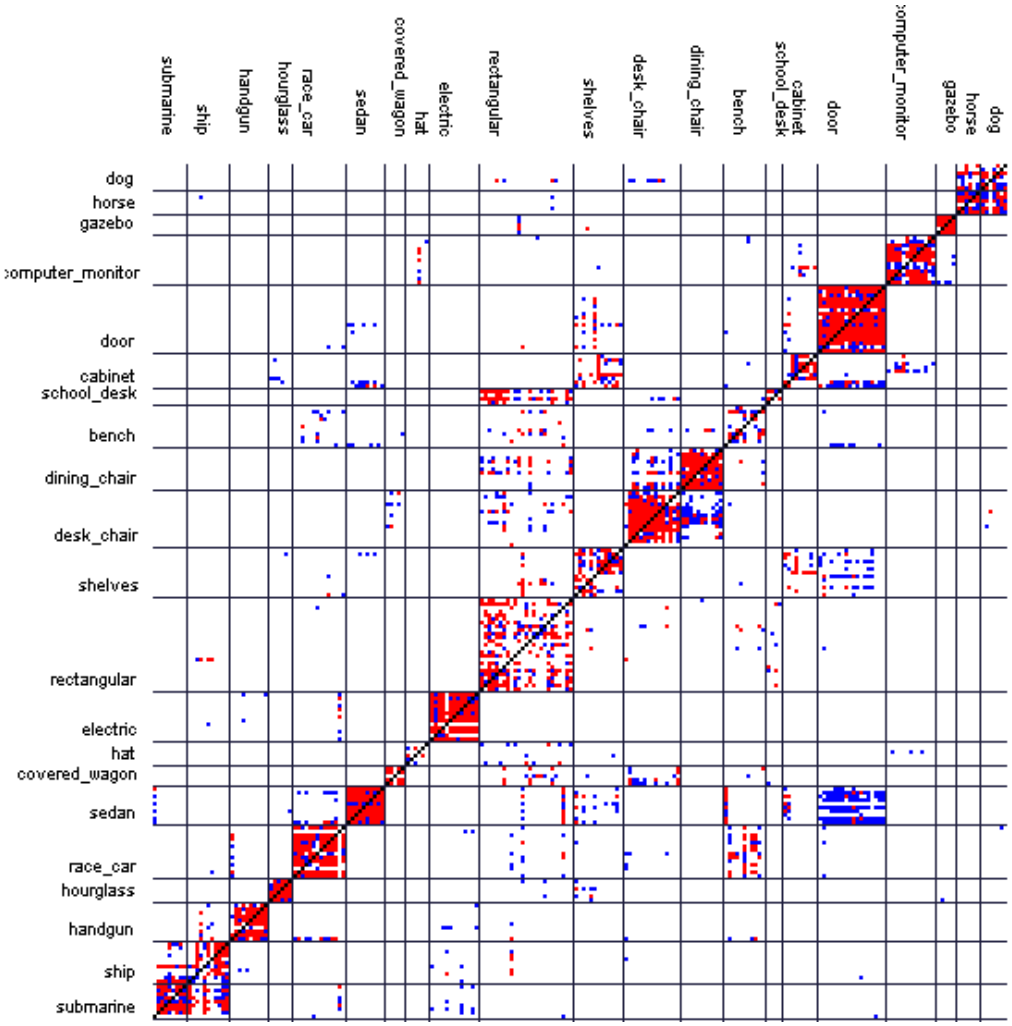


Figure 9.1: Tier image visualizing nearest neighbor (black), first tier (red), and second tier (blue) computed by matching every model (rows) with every other model (columns) in the base classification of the test set using the LFD algorithm – separating lines and labels indicate classes. Note that the full image is 907x907 "pixels," and only a small portion is shown.

In addition to these qualitative visualizations, the benchmark includes tools for computing quantitative statistics for evaluation of match results. Usually, the statistics are summarized by averaging over all query models (micro-average), with the query model removed from the matching results. However, our tools also support output of separate

statistics for each query model, averages for each class, an average of the averages for each class (macro-average), and averages over any user-supplied list of query models.[1] As will be shown in Section 9.7.4, this last feature is particularly useful for studying the quality of matches for models having specific properties. Specifically, our tools compute Nearest Neighbor, First Tier, Second Tier, E-Measure, and Discounted Cumulative Gain, which are described in Section 4.1.3.

The Discounted Cumulative Gain [78] metric incorporates the entire query result list in an intuitive manner, so we typically use it to summarize results when comparing algorithms. More specifically, we usually look at the "normalized DCG," which scales the DCG values down by the average over all algorithms tested and shifts the average to zero:

$$\text{Normalized}DCG_A = \frac{DCG_A}{\overline{DCG} - 1}$$

where DCG_A is the DCG value computed for algorithm A, and $\overline{DCG}$ is the average DCG value for all algorithms being compared in the same experiment. Positive/negative normalized DCG scores represent above/below average performance, and higher numbers are better (see the rightmost column of Table 9.4).

9.7 Results

In order to investigate the utility of the proposed benchmark, we used it to compare fourteen shape matching algorithms recently described in the literature. While the results of these experiments are interesting in their own right, the focus of our investigation is whether the database, classifications, annotations, and evaluation tools provided by the benchmark are useful for understanding the differences between the algorithms. Our hypothesis is that we might learn something about the algorithms that would have been difficult to discover without the benchmark tools.

[1] For precision-recall plots, the precision for each model (micro) or class (macro) is averaged using linear interpolation over the recall range $[1/|C|, 1]$, if there are C models in a class.

9.7.1 Shape Descriptors

The fourteen shape matching algorithms are all similar in that they proceed in three steps: the first step normalizes models for differences in scale and possibly translation and rotation; the second step generates a *shape descriptor* for each model; and the third step computes the distance between every pair of shape descriptors, using their L_2 difference unless otherwise noted. The differences between the algorithms lie mainly in the details of their shape descriptors:

- **D2 Shape Distribution (D2)**: a histogram of distances between pairs of points on the surface [98].
- **Extended Gaussian Image (EGI)**: a spherical function giving the distribution of surface normals [52].
- **Complex Extended Gaussian Image (CEGI)**: a complex-valued spherical function giving the distribution of normals and associated normal distances of points on the surface [64].
- **Shape Histogram (SHELLS)**: a histogram of distances from the center of mass to points on the surface [3]. This is similar to the Shells Descriptor used in other chapters.
- **Shape Histogram (SECTORS)**: a spherical function giving the distribution of model area as a function of spherical angle [3].
- **Shape Histogram (SECSHEL)**: a collection of spherical functions that give the distribution of model area as a function of radius and spherical angle [3].
- **Voxel**: a binary rasterization of the model boundary into a voxel grid.
- **Spherical Extent Function (EXT)**: a spherical function giving the maximal distance from center of mass as a function of spherical angle [109].
- **Radialized Spherical Extent Function (REXT)**: a collection of spherical functions giving the maximal distance from center of mass as a function of spherical angle and radius [130].
- **Gaussian Euclidean Distance Transform (GEDT)**: a 3D function whose value at each point is given by composition of a Gaussian with the Euclidean Distance Transform of the surface [69].

- **Harmonic Shape Descriptor (HSD)**: a rotation invariant representation of the GEDT obtained by computing the restriction of the function to concentric spheres and storing the norm of each (harmonic) frequency [69].
- **Fourier Shape Descriptor (FSD)** [69]: similar to the HSD, but the amplitude of every spherical harmonic coefficient is stored – it is similar to the Harmonic Shape Contexts of [34].
- **Light Field Descriptor (LFD)**: a representation of a model as a collection of images rendered from uniformly sampled positions on a view sphere. The distance between two descriptors is defined as the minimum L_1-difference, taken over all rotations and all pairings of vertices on two dodecahedra. [19]. We use the original implementation provided by Chen et al. without modification.
- **Depth Buffer Descriptor DSR740 (DBD)**: a collection of depth buffer images captured from orthogonal parallel projections. Images are stored as Fourier coefficients of the lowest frequencies, and differences between Fourier coefficients provide a measure of object dissimilarity [48]. We use Dejan Vranic's implementation of this method [129] without modification.

All computations were performed on a Windows PC with a Pentium 4 CPU running at 2.00 GHz and 512 MB of memory, except the LFD and FSD computations. The LFD was executed on a Windows PC with Pentium 4 CPU running at 2.4 GHz with 256 MB RAM and an NVIDIA GeForce 2 MX graphics card, and the FSD was computed on a x86_64 Linux server running at 2.2 GHz with 16 GB RAM.[2]

[2] Every model was normalized for size by isotropically rescaling it so that the average distance from points on its surface to the center of mass is 0.5. Then, for all descriptors except D2 and EGI, the model was normalized for translation by moving its center of mass to the origin. Next, for all descriptors except D2, SHELLS, HSD, and LFD, the model was normalized for rotation by aligning its principal axes to the *x*-, *y*-, and *z*-axes. The FSD only required alignment of the *x*-axis. The ambiguity between positive and negative axes was resolved by choosing the direction of the axes so that the area of the model on the positive side of the *x*-, *y*-, and *z*-axes was greater than the area on the negative side [28].

Every spherical descriptor (EGI, CEGI, Sectors, etc.) was computed on a 64×64 spherical grid and then represented by its harmonic coefficients up to order 16. Similarly, every 3D descriptor (e.g., Voxel and GEDT) was computed on a $64 \times 64 \times 64$ axial grid, translated so that the origin is at the point $(32, 32, 32)$, scaled by a factor of 32, and then represented by thirty-two spherical descriptors representing the intersection of the voxel grid with concentric spherical shells. Values within each shell were scaled by the square-root of the corresponding area and represented by their spherical harmonic coefficients up to order 16. Histograms of distances (D2 and Shells) were stored with 64 bins representing distances in the range $[0, 2]$. All descriptors, except LFD and DBD, were scaled to have L_2-norm equal to 1.

The LFD comprises 100 images encoded with 35, 8-bit, coefficients to describe Zernike moments and 10, 8-bit, coefficients to represent Fourier descriptors.

9.7.2 Base Classification Results

In our first experiment, we used each of the fourteen shape matching algorithms to compute the distances between all pairs of models in the test set and analyzed them with the benchmark evaluation tools to quantify the matching performance with respect to the base classification (the training set was not used for training any of the algorithms). Figure 9.2 shows a precision-recall plot showing the micro-averaged retrieval results achieved for this experiment, and Table 9.4 shows micro-averaged storage requirements, processing times, and retrieval statistics for each algorithm. We found that the micro and macro-average gave consistent results, and we decided to present micro-averaged statistics.

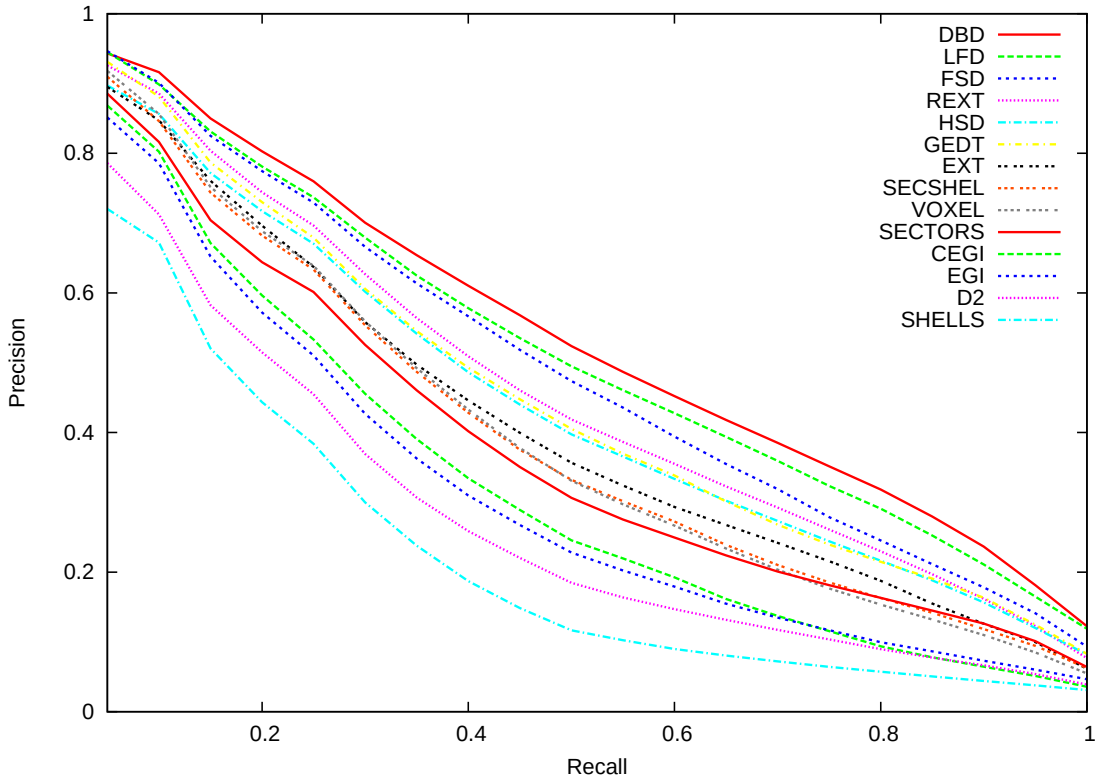


Figure 9.2: Precision-recall curves computed for fourteen shape descriptors for tests with the PSB base classification.

The Depth Buffer Descriptor encodes six images with 8-bit gray values encoding depth from the viewing plane. The Fourier transform is applied to the images and the k low-frequency values are recorded as a feature vector (k=73 by default).

Shape Descriptor	Storage Size (bytes)	Timing		Discrimination					
		Generate Time (s)	Compare Time (s)	Nearest Neighbor	First Tier	Second Tier	E-Measure	DCG	Normalized DCG
DBD	1,752	0.55	0.000018	66.5%	40.3%	51.2%	29.5%	66.3%	21.3%
LFD	4,700	3.25*	0.001300*	65.7%	38.0%	48.7%	28.0%	64.3%	17.7%
FSD	32,768	1.82*	0.000450*	63.1%	35.6%	45.5%	26.7%	62.6%	14.6%
REXT	17,416	2.22	0.000229	60.2%	32.7%	43.2%	25.4%	60.1%	10.0%
HSD	2,184	1.69	0.000027	55.6%	30.9%	41.1%	24.1%	58.4%	6.9%
GEDT	32,776	1.69	0.000450	60.3%	31.3%	40.7%	23.7%	58.4%	6.9%
EXT	552	1.17	0.000008	54.9%	28.6%	37.9%	21.9%	56.2%	2.8%
SECSHEL	32,776	1.38	0.000451	54.6%	26.7%	35.0%	20.9%	54.5%	-0.3%
VOXEL	32,776	1.34	0.000450	54.0%	26.7%	35.3%	20.7%	54.3%	-0.6%
SECTORS	552	0.90	0.000014	50.4%	24.9%	33.4%	19.8%	52.9%	-3.2%
CEGI	2,056	0.37	0.000027	42.0%	21.1%	28.7%	17.0%	47.9%	-12.4%
EGI	1,032	0.41	0.000014	37.7%	19.7%	27.7%	16.5%	47.2%	-13.6%
D2	136	1.12	0.000002	31.1%	15.8%	23.5%	13.9%	43.4%	-20.6%
SHELLS	136	0.66	0.000002	22.7%	11.1%	17.3%	10.2%	38.6%	-29.4%

Table 9.4: Comparing fourteen shape descriptors using the PSB base classification. (*Times were approximated by normalizing for processor speed.)

Surprisingly, we find that the top two shape descriptors (DBD and LFD) in this experiment were image-based. While the DBD uses depth information, the LFD uses only 2D projections to achieve high retrieval performance. The DBD not only has the best retrieval performance, it is among the fastest to compute and use for comparisons. Among the other descriptors, FSD, REXT, HSD, GEDT, and EXT provide the best matching performance. While FSD provides slightly better discrimination than the others, HSD and EXT are smaller and quicker to compare, suggesting they provide more "bang for the buck." The least discriminating descriptors are D2 and SHELLS. However, they are also the smallest and fastest to compare, which may be useful in certain applications.

Overall, we conclude that there is a quality-cost trade-off in the choice between shape descriptors, and no one descriptor beats the others in all respects.

9.7.3 Multi-Classification Results

In our second experiment, we investigated the impact of alternative classifications on the analysis of retrieval results. Specifically, we created three new classifications representing increasingly coarser groupings for the 907 models in the benchmark test set, and then we tested how these different classifications affect the evaluation of the fourteen shape matching algorithms.

The base classification provides the grouping with finest granularity in this experiment. It contains the 92 classes listed in Table 9.3. Most classes contain all the objects with a

particular function (e.g., microscopes). Yet, there are also cases where objects with the same function are partitioned into different classes based on their forms (e.g., round tables versus rectangular tables). In the alternative classifications, we recursively merge classes to form coarser granularity groups. Specifically, the "Coarse" classification merges objects with similar overall function to form 44 classes, the "Coarser" classification merges groups further to form the 6 classes listed in Table 9.1, plus a miscellaneous class not included in averaged retrieval results. Finally, the "Coarsest" classification merges those classes until just two classes remain: one with man-made objects and the other with naturally occurring objects.

Table 9.5 lists the normalized DCG scores achieved by the fourteen shape matching algorithms (rows) when evaluated with respect to the four different classifications (columns). From this table, we make two observations. First, as you might expect, the differences between shape matching algorithms are diminished when evaluated with coarser granularity classifications - i.e., the normalized DCG scores, which measure differences from the average, become less in columns further to the right. Second, we observe that the relative rankings of algorithms can vary significantly for different classifications. In particular, the EGI algorithm performs twelfth best with respect to the base classification (13.6% below the average). However, it performs best of all for the coarsest classification (3.0% above the average). Apparently, it is very good at determining the difference between man-made and natural objects, but not that good at telling apart the differences between specific classes. We conjecture that man-made objects have a narrower distribution of normals, making detection easy with EGIs. Similar behavior is shown by CEGI, which is a closely related algorithm.

These results provide a simple example of the value of using multiple classifications when evaluating shape matching algorithms. The information available in multiple classifications is more than in any one classification alone. We expect that many alternative semantic classifications will be made for these models in the future, exposing further differences between algorithms.

9.7.4 Query List Results

In our third experiment, we studied the properties of the fourteen shape matching algorithms further by looking at retrieval results with respect to the base classification

Shape Descriptor	Base (92)	Coarse (44)	Coarser (6)	Coarsest (2)
DBD	21.3%	11.4%	2.8%	0.3%
LFD	17.7%	8.8%	3.0%	0.3%
FSD	14.6%	7.0%	1.2%	0.0%
REXT	10.0%	5.3%	1.7%	0.2%
HSD	6.9%	4.4%	0.5%	-0.6%
GEDT	6.9%	3.4%	0.8%	-0.4%
EXT	2.8%	0.6%	0.3%	-0.6%
SECSHEL	-0.3%	-1.7%	-0.3%	-0.4%
VOXEL	-0.6%	-1.3%	-0.4%	-0.4%
SECTORS	-3.2%	-2.9%	-1.1%	-0.7%
CEGI	-12.4%	-2.6%	-0.2%	2.6%
EGI	-13.6%	-3.5%	-0.1%	3.0%
D2	-20.6%	-11.7%	-3.6%	-1.6%
SHELLS	-29.4%	-17.1%	-4.9%	-1.5%

Table 9.5: Evaluating fourteen shape descriptors using classifications of different granularity. The columns represent different classifications (with the number of classes listed in parenthesis), and the rows represent different shape descriptors. The numbers show normalized DCG scores averaged over all models.

averaged over sets of models with specific properties. Some of the properties were semantic (e.g., is a piece of furniture), others were procedural (e.g., aligned well with other members of its class), and the rest were geometric (e.g., roughly linear in shape). Our hope is that we can infer the conditions under which each shape matching algorithm performs best by comparing the retrieval results of this experiment.

Table 9.6 lists normalized DCG scores achieved by the fourteen shape matching algorithms (rows) with respect to the base classification when averaged over all models with specific properties (columns). The first column of numbers ("All Models") shows the average for all models, as a reference for comparison. The next six columns ("Animal"-"Vehicle") correspond to averages over the sets of models of the same object type. The next column ("Rotation Aligned") shows the average over all models for which our normalization steps were successfully able to align the model consistently with other members of is class. The following column ("Stick Shape") lists averages over the 200 models whose shape is most stick-like (as determined by the ratio of the largest and second largest eigenvalues of the covariance matrix of second order moments). Finally, the right-most column ("Complex Shape") shows averages over the 200 models with the most "complex shapes" (as estimated by the average pixel depth complexity when the model is rendered with parallel projection from viewpoints at the vertices of an

Shape Descriptor	All Models	Animal	Building	Furni-ture	House-hold	Tree & Plant	Vehicle	Rotation Aligned	Stick Shape	Complex Shape
DBD	21.3%	22.5%	21.3%	19.1%	21.2%	31.5%	20.5%	20.9%	19.3%	20.0%
LFD	17.7%	8.6%	38.2%	31.8%	22.0%	19.1%	11.7%	14.1%	5.8%	23.2%
FSD	14.5%	12.5%	4.4%	14.2%	12.7%	23.4%	17.3%	11.5%	12.0%	20.4%
REXT	10.0%	5.8%	2.1%	8.2%	10.5%	3.1%	12.5%	9.3%	7.7%	10.8%
HSD	6.9%	7.4%	-10.0%	7.2%	10.1%	14.2%	3.6%	4.8%	2.8%	5.4%
GEDT	6.9%	7.0%	1.2%	4.5%	7.9%	1.4%	7.1%	10.1%	6.2%	9.3%
EXT	2.8%	3.8%	9.3%	1.9%	5.2%	-11.1%	4.5%	2.4%	4.4%	2.3%
SECSHEL	-0.3%	-1.7%	-0.5%	-7.8%	3.8%	-15.2%	4.0%	2.6%	1.9%	-1.2%
VOXEL	-0.6%	1.1%	-1.2%	-4.5%	1.1%	-7.3%	2.2%	2.2%	3.3%	-3.2%
SECTORS	-3.2%	-4.8%	0.1%	-8.9%	0.9%	-25.2%	0.2%	-0.7%	2.3%	-5.1%
CEGI	-12.4	-6.6%	-24.8%	-1.9%	-21.8%	6.3%	-16.1%	-11.0%	-9.9%	-15.8%
EGI	-13.6%	-12.1%	-13.8%	-2.8%	-23.3%	6.8%	-15.4%	-13.4%	-11.7%	-12.3%
D2	-20.6%	-18.0%	-4.2%	-27.2%	-18.2%	-25.3%	-21.7%	-21.9%	-13.9%	-20.2%
SHELLS	-29.4%	-25.4%	-22.0%	-34.1%	-32.3%	-21.5%	-30.6%	-31.0%	-30.3%	-33.6%

Table 9.6: Evaluating retrieval performance for fourteen shape descriptors on query lists with specific object types and geometric properties using the PSB base classification. Numbers represent normalized DCG value averaged over models with the property listed in the column heading.

icosahedron). These latter properties are derived directly from the annotations provided with the benchmark.

With these results, we confirm that shape matching algorithms do not perform equally well on all object types. Although the ranking of algorithms is fairly consistent, there is sometimes a big difference in the relative performance of algorithms when focusing on models with specific properties. For instance, we note that SECTORS does better than EGI on household objects (0.9% above average versus 23.3% below average), while the opposite is true for trees and plants (25.2% below average versus 6.8% above average). Also, we see that the top ranked algorithms (DBD, LFD, FSD, REXT, and HSD) do worse on stick-shaped objects relative to other algorithms (the normalized DCG scores averaged for stick shaped objects are worse than the average over all models by 2.0%, 11.9%, 2.5%, and 2.3%, respectively), probably because the principal axes of sticks align well and/or the descriptors eliminate high-frequency information. Finally, we note that queries with "Rotation Aligned" models produce significantly different retrieval results, indicating that misalignment of models during normalization significantly affects the results achieved with some algorithms (GEDT, SECSHEL, VOXEL, and SECTORS).

9.7.5 Comparison with Other Databases

In our final experiment, we compared results of the Princeton Shape Benchmark database versus those achieved with other databases previously described in the literature [36, 98, 130, 139]. Our goal in this experiment was to validate whether our benchmark produces results consistent with those previously reported.

Table 9.7 shows the normalized DCG scores computed for twelve shape matching algorithms on six different databases. Note that the FSD and DBD descriptors were not available when analyzing these databases. We see that the results computed with the Osada [98] and MPEG-7 [139] databases are less consistent than the others. We conjecture that the reason is that they are relatively small (133 and 227 models, respectively) and have less variation of object types. The categorized models of the Utrecht [122] database are entirely airplanes, which probably explains why the descriptors clustered to a few values. Meanwhile, the relative performance of the algorithms on the other three databases appear fairly consistent. We expect that the minor differences between the databases can be explained by the differences in their object types.

Shape Descriptor	Osada [98]	MPEG-7 [139]	CCCC [130]	Utrecht [122]	VP [36]	PSB [ours]
LFD	14.9%	5.8%	20.3%	5.4%	17.7%	21.3%
REXT	8.6%	3.6%	11.3%	2.4%	8.5%	13.3%
HSD	12.1%	5.5%	12.5%	2.3%	10.6%	10.2%
GEDT	5.2%	2.5%	5.5%	4.3%	6.3%	10.1%
EXT	2.9%	0.4%	5.5%	2.4%	5.6%	6.0%
SECSHEL	-0.7%	-0.2%	-0.8%	2.2%	0.7%	2.8%
VOXEL	2.2%	1.3%	-0.5%	2.5%	0.4%	2.4%
SECTORS	-0.8%	-2.3%	-1.9%	2.3%	-1.6%	-0.3%
CEGI	-13.9%	-1.8%	-4.7%	-6.9%	-7.6%	-9.6%
EGI	-10.7%	-1.0%	-7.3%	-7.0%	-9.5%	-10.9%
D2	-1.1%	-4.3%	-16.6%	-3.1%	-12.8%	-18.2%
SHELLS	-18.7%	-9.6%	-23.2%	-6.8%	-18.2%	-27.3%

Table 9.7: Evaluating shape descriptors using different databases. Numbers represent normalized DCG averaged over all models in each database.

9.8 Conclusion

In summary, this chapter describes the Princeton Shape Benchmark, a publicly available framework for comparing shape matching algorithms. The benchmark includes a database of annotated 3D polygonal models, multiple classifications, and software tools for evaluating the results of shape matching experiments. All data and source code is freely available on the Web (http://shape.cs.princeton.edu/benchmark). As of December, 2007, the PSB has had 20,000 unique visitors and been downloaded over 8,500 times.

Since the original publication of the Princeton Shape Benchmark, there has been ongoing interest in comparing 3D shape retrieval techniques. The Network of Excellence AIM@SHAPE has organized the Shape Retrieval Contest (SHREC) [92, 93] in 2006 and 2007, which has included new shape matching techniques and several databases of models with specialized characteristics. Contestants from many countries compared their shape matching algorithms using standardized data sets and performance measures.

The main research contribution of this work is the methodology proposed for comparing shape matching algorithms. In particular, we advocate experimenting with several different classifications and query lists targeted at exposing specific differences between shape matching algorithms. Using this methodology, for example, we were able to discover that EGIs are good at discriminating between man-made and natural objects, but not that good at making detailed class distinctions. We also find that the Depth Buffer Descriptor [48], which is computed from multiple depth buffer images of a 3D model, is the most discriminating among the shape descriptors tested, with relatively low storage and computational requirements. We hope that results of this type encourage shape matching researchers to use the benchmark in future experiments, possibly creating new classifications and query lists of their own. In time, we expect that a common set of tests will emerge to form a de-facto standard for shape matching experiments.

Chapter 10

Conclusion and Future Work

Conclusion

In this dissertation, we explored techniques for retrieving 3D models from large databases. Our main focus has been on identifying the important, distinctive regions of 3D shapes and methods for focusing retrieval on those regions. We have been guided by the twin goals of improving the accuracy of shape retrieval and exploring the properties of shape distinction. There are six main contributions of this dissertation.

First, we have defined distinctive regions of 3D surfaces as those regions that provide the best retrieval performance relative to a classified database. Unlike previous techniques that have focused on inherent properties of a mesh (curvature or likelihood), shape distinction analyzes meshes in relation to an entire database. Shape distinction adjusts to the classes in the database, shape descriptors used for retrieval, scale of the descriptors, and even the retrieval metric used during analysis of the database. By visualizing distinction scores within a database, properties that define classes become readily apparent.

Second, we developed an algorithm for multi-feature matching of 3D shapes that can efficiently search a database for results within a few seconds. The main contribution is a priority-driven search algorithm that focuses on the best candidates without considering all possible combinations of matches. By focusing on the distinctive regions of shapes in the database, search speed and retrieval performance are significantly improved. Using the framework of priority-driven search, the effects of numerous parameters were

explored including: filtering the database based on shape distinction or other properties, the scale of shape descriptors, number of features used in matching, and a cost function for combining feature correspondences.

Third, we developed an efficient algorithm for calculating distinction for large databases undergoing additions. We compared several shape retrieval metrics and found that metrics that weight correct results near the front of the retrieval list more heavily than later results produce distinction scores that improve retrieval performance. We approximated distinction by modifying the DCG retrieval metric to use a small number of nearest neighbors that can be found with an index structure for shape descriptors and showed that retrieval performance improved with the accuracy of the approximation.

Fourth, we predicted distinction for the query model (as compared to target models) using a function that maps descriptor likelihood to distinction scores learned from a training set. Compared to several common selection techniques (selecting descriptors randomly or based directly on likelihood), using a likelihood mapping leads to better retrieval performance. The improvement is because descriptors grouped by their likelihood values have similar distinction scores, and we found that descriptors with likelihood values between the most rare and most common provide the best retrieval performance. Selecting a small set of descriptors with high predicted distinction not only improves retrieval performance, but filtering query descriptors also improves retrieval time.

Fifth, we demonstrated that shape distinction is useful for graphics applications that benefit from importance scores across the surface of a mesh. Differences within a database can be visualized since distinction directly relates to regions that are similar within a class and differ from shapes of other classes. Also, icons and mesh simplifications were generated by focusing on distinctive regions. Our methodology of learning importance scores directly from a training set can likely be applied to numerous other graphics applications.

Sixth, we introduced the Princeton Shape Benchmark: including a classified data set, tools for evaluating shape retrieval, and a methodology for comparing shape-matching techniques. Using the PSB, we were able to directly compare shape-matching techniques from the literature and show that visual-based descriptors such as the Depth Buffer Descriptor and Light Field Descriptor are among the best choices for many retrieval tasks. We also showed that for specific types of queries, the relative performance of descriptors can change dramatically, and a generally poor performing descriptor such as EGI has

the best performance on man-made versus natural objects. The PSB has become a de-facto standard in the computer graphics shape matching community and has hopefully improved the methodology of experiments in the field.

Future Work

There are several promising venues for future research related to shape distinction, retrieval from large databases, and the Princeton Shape Benchmark.

Defining Distinction

There are several strengths and weaknesses of our definition of distinction that should be considered and addressed in future work. First, the shape descriptor (HSD) used in our implementation is not the most descriptive possible. Experiments with the PSB have shown that the Depth Buffer Descriptor has better retrieval performance than the HSD, though using image-based descriptors for local matching has not been explored. However, a strength of our approach is that it is independent of a particular shape descriptor. So, in future work, we intend to investigate more descriptive shape representations to define mesh distinction.

Matching with Distinction

Results with the priority-driven search algorithm suggest several areas for improvement and future work. In particular, there are three main computational bottlenecks in the system: 1) constructing shape descriptors, 2) determining the distinction of shape descriptors, and 3) generating pairwise feature correspondences. There are many simple ways to speed up these steps, including random sampling, compression, and indexing. For example, the time required to establish the best pairwise correspondences between features could be improved with standard multi-dimensional indexing schemes. We have focused our efforts on utilizing distinction with the priority-driven search algorithm, and thus we have not yet investigated these options in detail.

Another interesting option is to compute the cost of feature correspondences in a weighted manner that more fully takes advantage of distinction scores. Currently, we

filter the set of descriptors to the most distinctive k that meet a separation threshold and use those during matching. The selected k descriptors have distinction scores that reflect their relative importance from a retrieval perspective and could be used as weights when calculating the cost of feature correspondences. The challenge of this approach is how to normalize the distinction weights since the weights are on the target models instead of on the query, so normalization would be across the entire database. Also, while more emphasis should be placed on correct matches in highly distinctive regions, our current priority-driven search algorithm focuses on low cost correspondences, which would be areas of low distinction.

Updating Distinction

The retrieval measure (DCG) used in most of our experiments can be slow to compute. While we have shown one method for improving the scalability by approximating DCG with a small set of neighbors and a cover tree index structure, this may only be practical for a few thousand models. Specifically, our method for handling a dynamic database is only a first solution to the problem. In future work, we would like to improve the scalability of our algorithm. Techniques to consider include hierarchical distinction on the surface of a mesh, updating distinction scores for clusters of similar regions, and further indexing techniques.

Predicting Distinction

Our prediction model for distinction can be extended in several ways. We assumed a normal distribution as an initial mapping, which also suggests a range of alternatives. Other distribution models may more accurately reflect the true likelihood, but all likelihood mappings condense the descriptors to one dimension parameterized by likelihood. Any likelihood model has the drawback of grouping all descriptors within a shell of equal likelihood, even if there is a large variation of retrieval performance within each shell. Other groupings of descriptors may better cluster those with similar distinction scores, though increasing the dimensionality of the mapped space can adversely affect the calculation time. The feature space of descriptors could be used directly by making a prediction based on similar descriptors (measured by the L_2 distance). Using nearby

descriptors is likely to provide a better clustering of retrieval scores, though a combination of indexing and compression techniques would be needed to minimize the lookup time.

Applications of Distinction

While we have shown that shape distinction provides a good measure of surface importance for icon generation and mesh simplification, analyzing distinction of 2D images of 3D shapes would likely provide higher quality results. Finally, we feel that extending our methodology of calculating an importance score from a well labeled training set can improve numerous graphics tasks beyond retrieval such as mesh alignment and 2D image retrieval.

Princeton Shape Benchmark

From experiences using the PSB, we suggest several areas for future research. First, the benchmark should be expanded to support other shape analysis tasks. Annotations for human-generated alignment transformations would facilitate evaluation of automatic registration algorithms, and consistent segmentations for classes of objects would provide training data for automatic techniques. Second, the results of Section 9.7.4 suggest that it is possible to build an adaptive multi-classifier that first checks the geometric properties of a given query model and then dynamically weights the distances computed by several shape matching algorithms to produce more discriminating results (e.g., [15, 43]). Third, as more and more data gets added to the benchmark, it will become possible to consider multi-classifiers that take into account both geometric and non-geometric attributes of 3D models (e.g., color, texture, scene graph structure, textual annotation, etc.).

Finally, while the PSB has helped standardize shape retrieval experiments, after five years it has begun to show its age. The shape-matching community has always had diverse interests because of the wide range of problems in the field, and a benchmark or multiple benchmarks should reflect the needs of the community. New benchmarks should be created with a larger number of classified meshes, expanded annotations, and subdivisions focusing on specific shape matching tasks such as: partial versus complete objects, manifold versus polygon-soup meshes, and domain specific meshes such as proteins, CAD models, and architectural objects. Hopefully, lessons learned from the PSB will provide guidance as the shape-matching field continues to mature.

Bibliography

[1] M. Ankerst, G. Kastenmüller, H. Kriegel, and T. Seidl. Nearest neighbor classification in 3D protein databases. In *ISMB*, 1999.

[2] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *Proc. SSD*, 1999.

[3] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. Nearest neighbor classification in 3D protein databases. In *Proc. ISMB*, 1999.

[4] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

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

[6] H. Barrow and R. Burstall. Subgraph isomorphism, matching relational structures and maximal cliques. *Inf. Process. Lett.*, 4:83–84, 1976.

[7] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *ICCV*, 2001.

[8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 24(24):509–522, 2002.

[9] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005.

[10] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.

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

[12] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *23rd International Conference on Machine Learning (ICML)*, pages 97–104, June 2006.

[13] V. Blanz, M. Tarr, H. Buelthoff, and T. Vetter. What object attributes determine canonical views? *Perception*, pages 575–599, 1999.

[14] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Proc. Models for the Perception of Speech and Visual Form*, pages 362–380, Cambridge, MA, November 1967. MIT Press.

[15] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranić. Using entropy impurity for improved 3D object similarity search. In *Proc. IEEE International Conference on Multimedia and Expo (ICME'04)*, pages 1303–1306. IEEE, 2004.

[16] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranić. Feature-based similarity search in 3D object databases. *ACM Computing Surveys*, 37(4):345–387, 2005.

[17] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranić. An experimental effectiveness comparison of methods for 3D similarity search. *International Journal on Digital Libraries, Special issue on Multimedia Contents and Management in Digital Libraries*, 6(1):39–54, 2006.

[18] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, 2001.

[19] D.-Y. Chen, M. Ouhyoung, X.-P. Tian, and Y.-T. Shen. On visual similarity based 3D model retrieval. *Computer Graphics Forum*, pages 223–232, 2003.

[20] C. Chua and R. Jarvis. Point signatures: A new representation for 3D object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1996.

[21] H. Chui and A. Rangarajan. A new algorithm for non-rigid point matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 44–51, 2000.

[22] CMU. Pose, Illumination, and Expression (PIE) database, 2003. http://www.ri.cmu.edu/projects/project_418.html.

[23] J. Copas. Regression, prediction, and shrinkage. *Journal of the Royal Statistical Society. Series B (Methodological)*, 45(3):311–354, 1983.

[24] J. Corney, H. Rea, D. Clark, J. Pritchard, M. Breaks, and R. MacLeod. Coarse filters for shape matching. *IEEE Computer Graphics & Applications*, 22(3):65–74, May/June 2002.

[25] P. Courtney and N. Thacker. Peformance characterization in computer vision, 2003. http://peipa.essex.ac.uk/benchmark.

[26] I. L. Dryden and G. Walker. Highly resistent regression and object matching. *Biometrics*, 55(3):820–825, September 1999.

[27] R. Duda, P. Hart, and D. Stork. *Pattern Classification, Second Edition*. John Wiley & Sons, New York, NY, 2001.

[28] M. Elad, A. Tal, and S. Ar. Content based retrieval of VRML objects - an iterative and interactive approach. In *6th Eurographics Workshop on Multimedia 2001*, 2001.

[29] T. T. Elvins and R. Jain. Web-based volumetric data retrieval. In *VRML '95*, pages 7–12, 1995.

[30] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[31] C. Foster, E. Hayes, C. Y. Ip, D. McWherter, M. Peabody, Y. Shapirsteyn, V. Zaychik, and W. C. Regli. National design repository project: A status report. In *Int'l Joint Confs. on Artificial Intelligence (IJCAI) AAAI/SIGMAN Workshop on AI in Manufacturing Systems*, August 2001.

[32] I. E. Frank and J. H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135, May 1993.

[33] S. Frintrop, A. Nüchter, H. Surmann, and J. Hertzberg. Saliency-based object recognition in 3D data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, sept 2004.

[34] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *European Conference on Computer Vision (ECCV)*, pages 224–237, May 2004.

[35] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004)*, 2004.

[36] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. *Transactions on Graphics*, 22(1):83–105, 2003.

[37] T. Funkhouser and P. Shilane. Partial matching of 3D shapes with priority-driven search. In *Symposium of Geometry Processing*, July 2006.

[38] T. A. Funkhouser, R. A. Laskowski, and J. M. Thornton. Protein function prediction by matching volumetric models of active sites. In *Automated Function Prediction Meeting (AFP)*, August 2006.

[39] N. Gagvani and D. Silver. Parameter controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999.

[40] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transaction on Graphics*, 25(1):130–150, January 2006.

[41] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 209–216, Aug. 1997.

[42] N. Gelfand, N. Mitra, L. Guibas, and H. Pottman. Robust global registration. In *Symposium on Geometry Processing*, 2005.

[43] G. Giacinto and F. Roli. Dynamic classifier selection. *Lecture Notes in Computer Science*, 1857, 2000.

[44] C. Giacovazzo, H. L. Monaco, D. Viterbo, F. Scordari, G. Gilli, G. Zanotti, and M. Catti. *Fundamentals of Crystallography*. International Union of Crystallography. Oxford University Press, 1992.

[45] Google. 3D Warehouse, 2007. `http://sketchup.google.com/3dwarehouse`.

[46] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[47] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer-Verlag, Berlin, Germany, 2001.

[48] M. Heczko, D. Keim, D. Saupe, and D. Vranić. *Methods for similarity search on 3D databases*, volume 2, pages 54–63. Springer-Verlag, 2002. (In German).

[49] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 203–212, August 2001.

[50] D. D. Hoffman and M. Singh. Salience of visual parts. *Cognition*, 63, 1997.

[51] R. Hogg and E. Tanis. *Probability and Statistical Inference*. Pearson Education, sixth edition, 2001.

[52] B. Horn. Extended Gaussian images. *Proc. of the IEEE*, 72(12):1671–1686, December 1984.

[53] S. Howlett, J. Hamill, and C. O'Sullivan. An experimental approach to predicting saliency for simplified polygonal models. In *Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization*, 2004.

[54] Q.-X. Huang, S. Flory, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. *Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006)*, 2006.

[55] C. Y. Ip, D. Lapadat, L. Sieger, and W. C. Regli. Using shape distributions to compare solid models. In *7th ACM/SIGGRAPH Symp. on Solid Modeling and Applications*, pages 273–280, June 2002.

[56] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. Three dimensional shape searching: State-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509–530, 2005.

[57] N. Iyer, S. Jayanti, and K. Ramani. An engineering shape benchmark for 3D models. In *Proc. ASME IDETC/CIE*, 2005.

[58] N. Iyer, Y. Kalyanaraman, K. Lou, S. Jayanti, and K. Ramani. A reconfigurable 3D engineering shape search system part I: shape representation. In *ASME DETC'03*, 2003.

[59] M. James. *Classification algorithms*. Wiley-Interscience, New York, 1985.

[60] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[61] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani. Developing an engineering shape benchmark for CAD models. *Computer-Aided Design*, 38:939–953, September 2006.

[62] A. Johnson. Surface landmark selection and matching in natural terrain. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 413–420, June 2000.

[63] A. Johnson and M. Hebert. Using spin-images for efficient multiple model recognition in cluttered 3-D scenes. *IEEE PAMI*, 21(5):433–449, 1999.

[64] S. Kang and K. Ikeuchi. Determining 3-D object pose using the complex extended Gaussian image. In *CVPR*, pages 580–585, June 1991.

[65] M. Kazhdan. *Shape Representations and Algorithms for 3D Model Retrieval*. PhD thesis, Department of Computer Science, Princeton University, 2004.

[66] M. Kazhdan, B. Chazelle, D. Dobkin, A. Finkelstein, and T. Funkhouser. A reflective symmetry descriptor. In *European Conference on Computer Vision (ECCV)*, pages 642–656, May 2002.

[67] M. Kazhdan, B. Chazelle, D. Dobkin, T. Funkhouser, and S. Rusinkiewicz. A reflective symmetry descriptor for 3D models. *Algorithmica*, May 2002.

[68] M. Kazhdan and T. Funkhouser. Harmonic 3D shape matching. *SIGGRAPH 2002 Visual Proceedings, Technical Sketch*, July 2002.

[69] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, June 2003.

[70] D. A. Kleim. Efficient geometry-based similarity search of 3D spatial databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 419–430. ACM Press, 1999.

[71] I. Kolonias, D. Tzovaras, S. Malasiotis, and M.G.Strintzis. Fast content-based search of VRML models based on shape descriptors. *IEEE Transactions on Multimedia*, 2003. accepted for publication.

[72] M. Kortgen, G.-J. Park, M. Novotni, and R. Klein. 3D shape matching with 3D shape contexts. In *7th Central European Seminar on Computer Graphics*, April 2003.

[73] H.-P. Kriegel, S. Brecheisen, P. Kroger, M. Pfeifle, and M. Schubert. Using sets of feature vectors for similarity search on voxelized CAD objects. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 587–598. ACM Press, 2003.

[74] P. Lachenbruch and M. Goldstein. Discriminant analysis. *Biometrics*, 35(1):69–85, March 1979.

[75] J. Langford. Cover tree for nearest neighbor calculations, 2007. `http://hunch.net/~jl/projects/cover_tree/cover_tree.html`.

[76] Y. Lecun. The MNIST database of handwritten digits, 2003. http://yann.lecun.com/exdb/mnist/.

[77] C. H. Lee, A. Varshney, and D. W. Jacobs. Mesh saliency. In *ACM SIGGRAPH*, pages 659–666, New York, NY, USA, 2005. ACM Press.

[78] G. Leifman, S. Katz, A. Tal, and R. Meir. Signatures of 3D models for retrieval. In *4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, pages 159–163, February 2003.

[79] X. Li and I. Guskov. Multi-scale features for approximate alignment of point-based surfaces. In *Symposium on Geometry Processing*, 2005.

[80] J. Löffler. Content-based retrieval of 3D models in distributed web databases by visual shape information. In *Proceedings of the International Conference on Information Visualizaiton*. IEEE Computer Society, 2000.

[81] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[82] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[83] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[84] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley-Interscience, New York, 1992.

[85] R. G. Miller. *Beyond ANOVA: Basics of Applied Statistics*. Chapman & Hall/CRC, New York, NY, 1997.

[86] P. Min. *A 3D Model Search Engine*. PhD thesis, Department of Computer Science, Princeton University, 2004.

[87] P. Min, J. Halderman, M. Kazhdan, and T. Funkhouser. Early experiences with a 3D model search engine. In *Proceeding of the Eighth International Conference on 3D web technology*, pages 7–18, 2003.

[88] P. Min, M. Kazhdan, and T. Funkhouser. A comparison of text and shape matching for retrieval of online 3D models. *European Conference on Digital Libraries*, Sept 2004.

[89] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, December 2001.

[90] National Taiwan University. 3D Model Retrieval System, 2003. `http://3d.csie.ntu.edu.tw/~dynamic/cgi-bin/DatabaseII_v1.8`.

[91] D. Nehab and P. Shilane. Stratified point sampling of 3D models. In *Proceedings of the 1st Eurographics Symposium on Point-Based Graphics*, pages 49–56, 2004.

[92] Network of Excellence AIM@SHAPE. SHREC, 2006. http://www.aimatshape.net/event/SHREC/shrec06.

[93] Network of Excellence AIM@SHAPE. SHREC, 2007. http://www.aimatshape.net/event/SHREC.

[94] M. Novotni, P. Degener, and R. Klein. Correspondence generation and matching of 3D shape subparts. Technical Report CG-2005-2, Universität Bonn, June 2005.

[95] M. Novotni and R. Klein. A geometric approach to 3D object comparison. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 167–175, 2001.

[96] R. Ohbuchi, T. Minamitani, and T. Takei. Shape-similarity search of 3D models by using enhanced shape functions. In *Theory and Practice of Computer Graphics 2003*, pages 97–104, 2003.

[97] Okino. Polytrans, 2003. http://www.okino.com/conv/conv.htm.

[98] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3D models with shape distributions. *Shape Modeling International*, pages 154–166, May 2001.

[99] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, 2002.

[100] S. Paek, C. Sable, V. Hatzivassiloglou, A. Jaimes, B. Schiffman, S. Chang, and K. McKeown. Integration of visual and text based approaches for the content labeling and classification of photographs. In *ACM SIGIR'99 Workshop on Multimedia Indexing and Retrieval*, 1999.

[101] E. Paquet, A. Murching, T. Naveen, A. Tabatabai, and M. Rioux. Description of shape information for 2D and 3D objects. In *Signal Processing: Image Communication*, volume 16, pages 103–122, 2000.

[102] E. Paquet and M. Rioux. Nefertiti: a query by content software for three-dimensional models databases management. *Image and Vision Computing*, 17(2):157–166, 1999. NRC 40243.

[103] E. Paquet and M. Rioux. Influence of pose on 3D shape classification. In *Digital Human Modeling for Design and Engineering*, pages 6–8, June 2000. NRC 43627.

[104] M. Pelillo, K. Siddiqi, and S. W. Zucker. Attributed tree matching and maximum weight cliques. In *Proc. Image Analysis and Processing*, pages 1154–1159. IEEE, 1999.

[105] J. Pitman. *Probability*. Springer-Verlag, Berlin, Germany, 1993.

[106] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser. A planar-reflective symmetry transform for 3D shapes. *Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006)*, 2006.

[107] Princeton Shape Retrieval and Analysis group. 3D model search engine, 2007. `http://shape.cs.princeton.edu/`.

[108] G. Salton. The smart document retrieval project. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 356–358, 1991.

[109] D. Saupe and D. V. Vranić. 3D model retrieval with spherical harmonics and moments. In B. Radig and S. Florczyk, editors, *DAGM 2001*, pages 392–397, September 2001.

[110] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing shock graphs. In *International Conference of Computer Vision (ICCV)*, pages 755–762, 2001.

[111] Y. Shan, B. Matei, H. S. Sawhney, R. Kumar, D. Huber, and M. Hebert. Linear model hashing and batch ransac for rapid and accurate object recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2004.

[112] P. Shilane and T. Funkhouser. Selecting distinctive 3D shape descriptors for similarity retrieval. In *Shape Modeling International*, pages 108–117, June 2006.

[113] P. Shilane and T. Funkhouser. Distinctive regions of 3D surfaces. *ACM Transactions on Graphics*, 26(2), June 2007.

[114] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton Shape Benchmark. In *Shape Modeling International*, pages 167–178, June 2004.

[115] A. Shokoufandeh and S. J. Dickinson. A unified framework for indexing and matching hierarchical shape structures. In *Proceedings of the 4th International Workshop on Visual Form*, volume 4, pages 67–84. Springer-Verlag, 2001.

[116] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *Int. Journal of Computer Vision*, 35(1):13–31, 1999.

[117] SPEC. Standard Performance Evaluation Corporation, 2003. `http://www.specbench.org/benchmarks.html`.

[118] SpharmonicKit 2.5. Fast spherical transforms: Spharmonickit, 1998. `http://www.cs.dartmouth.edu/~geelong/sphere/`.

[119] H. Sundar, D. Silver, N. Gagvani, and S. J. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International*, pages 130–139, 2003.

[120] M. T. Suzuki. A web-based retrieval system for 3D polygonal models. *Joint 9th IFSA World Congress and 20th NAFIPS International Conference (IFSA/NAFIPS2001)*, pages 2271–2276, July 2001.

[121] J. Tangelder and R. Veltkamp. A survey of content based 3D shape retrieval methods. In *Shape Modeling International*, pages 145–156, June 2004.

[122] J. Tangelder and R. C. Veltkamp. Polyhedral model retrieval using weighted point sets. In *Shape Modeling International*, May 2003.

[123] TREC. Text REtrieval Conference Data, 2003. http://trec.nist.gov/data.html.

[124] University of Minnesota. Object File Format, 1990. Geometry Center.

[125] C. K. van Rijsbergen. *Information Retrieval*. Butterworths, 1975.

[126] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 273–280. Aka GmbH, 2001.

[127] R. C. Veltkamp. Shape matching: Similarity measures and algorithms. In *Shape Modelling International*, pages 188–197, May 2001.

[128] Viewpoint Corporation, 2001. http://www.viewpoint.com.

[129] D. Vranić. Tools for 3D model retrieval, 2006. `http://merkur01.inf.uni-konstanz.de/3Dtools`.

[130] D. V. Vranić. An improvement of rotation invariant 3D shape descriptor based on functions on concentric spheres. In *IEEE International Conference on Image Processing (ICIP 2003)*, volume 3, pages 757–760, September 2003.

[131] D. V. Vranić and D. Saupe. 3D shape descriptor based on 3D Fourier transform. In K. Fazekas, editor, *EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001)*, pages 271–274, September 2001.

[132] S. Yamany and A. Farag. Surfacing signatures: An orientation independent free-form surface representation scheme for the purpose of objects registration and matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:1105–1120, 2002.

[133] Y. Yang, J. Yang, H. Yang, and O. Gwun. Indexing VRML objects with triples. In *SPIE Proceedings*, volume 4311, pages 236–243, 2001.

[134] H. yeung Shum, M. Hebert, and K. Ikeuchi. On 3D shape synthesis. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 526–531, 1996.

[135] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric space. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1993.

[136] H. Zabrodky, S. Peleg, and D. Avnir. Continuous symmetry for shapes. *2nd Intl. Workshop on Visual Form*, 1994.

[137] H. Zabrodsky, S. Peleg, and D. Avnir. A measure of symmetry based on shape similarity. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, pages 703–706, Los Alamitos, California, 15–18 1992. IEEE Press.

[138] H. Zabrodsky, S. Peleg, and D. Avnir. Symmetry as a continuous feature. *IEEE PAMI*, 17(12):1154–1166, December 1995.

[139] T. Zaharia and F. Preteux. 3D shape-based retrieval within the MPEG-7 framework. In *SPIE Conf. on Nonlinear Image Processing and Pattern Analysis XII*, volume 4304, pages 133–145, January 2001.

[140] T. Zaharia and F. Preteux. Shape-based retrieval of 3D mesh models. In *IEEE International Conference on Multimedia and Expo (ICME '2002)*, August 2002.

[141] C. Zhang and T. Chen. Efficient feature extraction for 2D/3D objects in mesh representation. In *ICIP*, 2001.

[142] C. Zhang and T. Chen. Indexing and retrieval of 3D models aided by active learning. In *ACM Multimedia*, 2001.