Winter 2004

Digital Library Services for Three-Dimensional Models

Hesham Anan
Old Dominion University

DIGITAL LIBRARY SERVICES FOR 3D MODELS

by

Hesham Anan
M.Sc. Computer Science, December 1997, Alexandria University, Egypt
B.Sc. Computer Science, June 1993, Alexandria University, Egypt

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2004

Approved by:

Kurt Maly (Co-Director)

Mohammed Zubair (Co-Director)

Frederic McKenzie (Member)

Chester Grosch (Member)

Christian Wild (Member)

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

DIGITAL LIBRARY SERVICES FOR 3D MODELS

Hesham Anan
Old Dominion University, 2004
Co-Directors of Advisory Committee: Dr. Kurt Maly
Dr. Mohammad Zubair

With the growth in computing, storage and networking infrastructure, it is becoming increasingly feasible for multimedia professionals – such as graphic designers in commercial, manufacturing, scientific and entertainment areas - to work with 3D digital models of the objects with which they deal in their domain. Unfortunately most of these models exist in individual repositories, and are not accessible to geographically distributed professionals who are in need of them.

Building an efficient digital library system presents a number of challenges. In particular, the following issues need to be addressed

- What is the best way of representing 3D models in a digital library, so that the searches can be done faster?
- How to compress and deliver the 3D models to reduce the storage and bandwidth requirements?
- How can we represent the user's view on similarity between two objects?
- What search types can be used to enhance the usability of the digital library and how can we implement these searches, what are the trade-offs?

In this research, we have developed a digital library architecture for 3D models that addresses the above issues as well as other technical issues. We have developed a prototype for our 3D digital library (3DLIB) that supports compressed storage, along with retrieval of 3D models. The prototype also supports search and discovery services that are targeted for 3-D models. The key to 3DLIB is a representation of a 3D model that is based on "surface signatures". This representation captures the shape information of any free-form surface and encodes it into a set of 2D images. We have developed a shape similarity search technique that uses the signature images to compare 3D models.

One advantage of the proposed technique is that it works in the compressed domain, thus it eliminates the need for uncompressing in content-based search. Moreover, we have developed an efficient discovery service consisting of a multi-level hierarchical browsing service that enables users to navigate large sets of 3D models. To implement this targeted browsing (find an object that is similar to a given object in a large collection through browsing) we abstract a large set of 3D models to a small set of representative models (key models). The abstraction is based on shape similarity and uses specially tailored clustering techniques. The browsing service applies clustering recursively to limit the number of key models that a user views at any time.

We have evaluated the performance of our digital library services using the Princeton Shape Benchmark (PSB) and it shows significantly better precision and recall, as compared to other approaches.

ACKNOWLEDGMENTS

First and foremost I wish to thank God without whose help I would have never been able to finish this work.

My sincere appreciation and gratitude goes to my co-advisors, Dr. Kurt Maly and Dr. Mohammed Zubair for their time, patience, support and guidance throughout the entire research. Not only did their insight and ideas help me to complete this dissertation, but also their kindness and understanding have helped me through the difficult times of my research.

Many thanks to my committee members, Dr Chris Wild, Dr Chester Grosch, and Dr. Frederic McKenzie for taking the time to read my dissertation and for their valuable feedback.

I would like to thank the faculty, staff, and colleagues at the Computer Science department of Old Dominion University for their help and encouragement. Special thanks to Phyllis Woods, the department office manager, for her kindness and her friendly assistance.

My thanks also go to Ahmed Al-Theneyan, Mohamed Kholief, Sameh Yamany, Xiaoming Liu, and Ashraf Amrou for their helpful discussions, and to Satish Kumar who helped in the database administration and backup. Special Thanks to Phil Shilane from Princeton University for providing me with the 3D models that we used in testing our digital library services.

Many thanks to my wife Rania Hussein and to my cousin, Ezzeldin Anan, for reviewing and editing my dissertation and to my sisters, Mona and Doaa, for their encouragement and support. Special thanks to my lovely daughters, Yomna and Aaya, whose love and smiles have helped me overcome the most difficult times.

Finally, I can not find the words that express my thanks, love, and gratitude to my beloved parents for their continuous encouragement and support throughout my entire life. To them, I owe my Ph.D.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

CHAPTER I
INTRODUCTION

1.1 Motivation

With the increase in the number and in the extent of using three-dimensional (3D) images and models, 3D models are spreading and their use is no longer limited to specialists. Recently, there have been rapid improvements in the technology of acquiring 3D models along with their visualization and modeling tools. With the easy acquisition of 3D models and because of the versatility of the acquiring tools, 3D models are currently used in many fields like medicine, engineering, entertainment, and electronic commerce.

In medicine, the number of 3D models repositories of the different parts of the human body is rapidly growing. The management and the access to those large repositories have become increasingly complex. The goal of any medical information system is to improve the quality and efficiency of healthcare processes by delivering the needed information at the right time. Such a goal will need more than a query by patient name, series ID or study ID for 3d models. Teaching and research in the medical field are expected to improve significantly through the use of search, retrieval, and navigation systems of 3D medical models. In teaching, it can help lecturers as well as students to browse educational 3D models repositories and visually inspect the results found, which can augment the educational quality. Content-based techniques can facilitate internet-based medicine teaching by browsing medical databases in a fast and efficient way. Research can also benefit from content-based retrieval methods. Researchers will have more options for the choice of cases to include into research and studies, where 3D data can be mined to find changes or interesting patterns which can lead to the discovery of new knowledge by combining the various knowledge sources. Another important issue with the 3D medical repositories is the need for efficient storage techniques for the models which need a large amount of storage spaces.

The journal model for this dissertation is the IEEE/ACM Transactions on Networking.

Engineers can use search engines that can mine catalogs of three-dimensional objects, like airplane parts or architectural features in designing a certain model. They sketch what they are thinking of, and the search engines should produce comparable objects. Search engines can also serve industrial companies by saving their engineers' time and energy in designing a specialized part, when someone else has already created something similar to it. Even though engineers use computer-aided design (CAD) to design different 3D parts, they are tediously examining each part separately because of the way the parts are normally catalogued. The visual channel of our brains has not been exploited effectively for information retrieval in engineering. Using shape information to retrieve similar 3D models will retrieve shape as well as other related knowledge. Engineers relate objects to 3D shapes more than to meta-data and hence a shape-based system could provide an answer where other methods fail. Effective deployment of a search/retrieval system requires a combination of text and shape-based searching.

In the area of entertainment, multimedia professionals like graphic designers can benefit from using a 3D based search/retrieval system when searching for 3D objects to use in making a film or in a 3D computer game, as text descriptions rarely convey the most valuable information. Text-based search usually retrieves also irrelevant information that consumes the programmer's time to filter them.

As shopping through the Internet is becoming a very acceptable way for many users, electronic commerce becomes one of the most important applications that needs significant improvements in the way of searching and displaying items a user is interested in buying. Unfortunately, most of the e-commerce applications use text-based search techniques and one has to manually search through many retrieved irrelevant objects, which causes many of the buyers to get bored and leave without buying. Using a 3D search engines will make life easier for Internet shoppers and will increase the percentage of online buyers.

After presenting only a few of the many applications that use 3D models, it becomes obvious that the volume of 3D models available in digital form is rapidly growing, which highlight the need for digital library services for 3D models. Traditional digital libraries have proved to work well with one-dimensional data such as reports. A

great number of digital libraries have been developed for handling two-dimensional data such as images. However, handling three-dimensional data in digital libraries is relatively new and faces many challenges such as: efficient storage, faster retrieval, and user-friendly search and discovery process.

Some of the previously listed applications require direct access to 3D models in archives as well as tools to find 3D models when necessary. Also, when those applications maintain or need to access large archives of 3D models, it will become convenient (if not necessary) to browse the large collection in a controlled manner that will present the user with as much information in as little time as possible.

1.2 Issues

The general objective of any information search/retrieval system is to minimize the time a user spends in finding the needed information. Handling three-dimensional data in digital libraries is relatively new and faces many challenges such as: efficient storage, faster retrieval, and user-friendly search and discovery process. We will address some important issues in designing a digital library to manage 3D objects, which are listed as follows.

Repositories issues

- What is an efficient representation of free-form 3D models that: (a) reduces the storage requirements, and (b) enables fast reconstruction of the 3D models to improve interactivity and response time?

User interfaces and human computer interaction issues

- How can we represent the user's perspective on similarity between two objects that helps in better discovery?
- How to minimize the response time in browsing and navigating through the search results?

In the following subsections, we will give an overview of the approach that we are adopting to address these issues, and then we will present the objectives of our research.

1.3 Approach

A digital library is a set of collections and services. To construct a digital library for 3D models, the collections will contain appropriate representation of 3D models and

the services will be the traditional library services such as storage, search, retrieval, publishing and management of the collections as well as advanced services such as hierarchical browsing to enhance the experience of browsing and retrieving 3D models.

In developing the digital library, we will address issues that were presented in the previous section. We shall now discuss specifically the approach we will use to address these issues.

1.3.1 Approach to Build the 3D Repository

The central idea here is to arrive at a representation of 3D models that will lead to efficient storage, progressive delivery, and a better similarity measure that will help in discovery. To our knowledge, there is no representation of 3D models currently that can be used both for compressed representation of the model and for computing a shape similarity measure at the same time.

To handle the issue of efficient representation of 3D models, we will develop a 3D model representation in which the model is represented as a sequence of 2D images, called surface signature images, which capture shape information of the model. This representation will

- reduce the storage requirements for 3D models
- and at the same time it can be used in shape similarity search of 3D models.
- Moreover, this representation can be used to progressively transmit and reconstruct the 3D model.

The multiple uses of the signature image representation means that we can work entirely in the signature image domain to perform operations, such as shape similarity computations or progressive reconstruction, which were not available before without using or extracting some features from the 3D model itself.

Using signature images as a representation of the 3D model should allow progressive retrieval of the model in which a rough view can be reconstructed even with only incomplete data. The model can be reconstructed with better accuracy as more data arrives.

The simple approach for retrieving models with different resolutions could have been also achieved by storing different versions of the same model at different resolution

levels. This can be done by using mesh simplification techniques to get a simpler representation of the same model. However, this approach will ultimately require more storage space than what is required by the original model (the original model needs to be stored as well as every simplified version). Moreover, if a low resolution model was transferred and there is a need to get one with a better resolution, the current model will be discarded and he transmission of the better resolution model will start from scratch. This is not the case in our representation where we will build upon previous transmissions.

Although there are many progressive representations of 3D models (as we will review in chapter 2), our representation is unique in that it does not require the transmission of data to be in a certain order and it can be also used as a feature to compute shape similarity.

Unlike text documents, one cannot search for a specific 3D model using only textual metadata. This leads to an inevitable need for content-based 3D shape retrieval methods. Hence, in this research we will use a shape similarity type of query in which a model will be represented as a query to retrieve models of similar shapes. Shape-based retrieval cannot be achieved by comparing the actual 3D models due to the complexity and the time needed for such an operation. Shape-based retrieval usually is achieved by extracting features from the 3D model and using these features to compare the similarity of models. Hence, the key for a good shape-based retrieval technique is extracting features from the 3D models that are discriminatory enough to provide accurate similarity data.

In order to accurately search for 3D models by shape, with a response time comparable to searching in document digital libraries, we shall use the signature images as a mean to compare the shape similarity of 3D models.

There are similarity based search services that exist in the literature, such as [9][6], [63], [83], [74], [28][84], [21], [62], [29] and [78]. In these services features are extracted from the 3D model for the sole purpose of shape similarity computation. However, in our approach, we use signature images which are utilized to reconstruct the 3D model. This does not only allow us to search directly in the domain used to represent the 3D model

but it also provides a proof that our shape-similarity measure is discriminatory enough to be used in accurate retrieval of 3D models. Such a proof does not exist in any of the shape features described in the literature. We will support this by comparing precision and recall results of shape-similarity searches to those reported in the literature and will prove that our shape-based search outperforms other existing methods.

1.3.2 Approach for Building User Friendly Interfaces

To address the issue of minimizing the time used in browsing and navigating the 3D models, we will provide convenient tools to browse large collections of 3D models. We shall use the signature image to cluster similar models together and provide a more abstract view based on clusters instead of displaying all the models when browsing. Moreover, we shall extend clustering to multiple levels to accommodate a large number of models.

Recently, some digital libraries that support 3D models started to emerge (as will be discussed in chapter 2) such as the Visible Human Project [8], Princeton 3D Model Search Engine [60] and 3DESS [40]. Though these libraries have collections of 3D models, none of them provide a complete set of services as we do. For example, the Visible Human Project and Princeton 3D Model Search Engine do not provide hierarchical browsing tools that can help in browsing large collections of 3D models. The Visible Human Project does not provide tools to search for 3D models by similar shape, unlike our digital library. Though Princeton 3D Model Search Engine and 3DESS provide shape-based search services, our approach should prove to provide better precision and recall. None of these libraries provide a progressive retrieval service as we propose.

1.3.3 Digital Library Services

To handle the above issues while providing a complete set of services, we have developed the architecture shown in Fig. 1. Our 3D Digital Library (3DLIB) consists of a set of modules; each module will provide an implementation for one of the digital library services. On the backend, we have a database server to store the 3D models along with their corresponding metadata and extracted features or signatures. 3DLIB will provide a

set of services that are extended for traditional digital libraries to support 3D models. These services are:

- **A storage service**: It will use a repository to store the 3D models. The repository will be implemented using an Oracle database and will be divided into three parts. One part will be for storing metadata of the 3D models such as the author of the model, description or annotation of the models, etc. Another part of the database will be used to store features about the 3D model "signatures". These signatures (as will be explained in chapter 3) will be used in comparing shapes of the 3D models and conducting similarity searches as well as retrieving and reconstructing the actual 3D model. Original models can be reconstructed from the signatures. And the third part will contain the actual full 3D models (although they could be reconstructed from the 2D signature images).
- **Progressive retrieval service**: a streaming service to progressively retrieve the 3D models. The compression format of the 3D models has the characteristic that not all data are required for an initial view of the model. This makes the compression technique amenable to progressive retrieval, and for fault-tolerance in network applications. In other words, the response time to view the initial model is smaller. The initial display is refined with time as more data arrives. In addition, if some data frames are lost, the end user will still be able to visualize the model and interact with it, though with lower resolution. This service achieves the objective of reducing the storage requirements for the 3D models and it helps in minimizing response time to the user, to help in enhancing the experience of retrieving 3D models.
- **Publishing service**: a service which will be used to upload 3D models and their metadata to the repository. This service will accept a 3D model from the client, will determine anchor points (points at which signatures are calculated), will compute the signatures, and will upload the signatures along with the metadata of the 3D models to the repository.
- **Search and discovery service**: This service should achieve the objective of being able to accurately search for 3D models. This service will support two basic types

of search: a text search and a shape-based search. In the text search mode, a user shall provide some textual metadata about the 3D model in a query interface designed as a web page. The metadata searcher shall try to find the results of the query in a local cache. If no results were found, a query shall be sent to the backend database. The results shall be then collected, sorted, grouped and presented back to the user through the web client. In the shape-based search, a user can select (or present) an existing 3D model and ask through the web interface to retrieve similar models (in shape). In this case, the signatures of the model shall be compared against the signatures of other models in the database and the results are sent back to the user. If the user decides to use an external model in the similarity query instead of using an existing model, an additional process to get anchor points and signatures at these anchor points has to be invoked first. A user also has the option of browsing the existing models, in which the user views hierarchies of the models grouped by shape (as will be illustrated in chapter 4).

1.4 Objectives

In summary, the **objectives of our research** are:

- To develop digital *library architecture* that supports storage, search, retrieval, publishing and management of 3D models. The design should also support advanced services such as hierarchical browsing and progressive retrieval, to enhance the experience of browsing and retrieving 3D models.
- To demonstrate the feasibility of developing a digital library for 3D models by deploying a *prototype with more than 1,500 models* to address issues of *scaling*.
- To *reduce the storage requirements* for 3D models by developing a compression algorithm to represent a 3D model as a sequence of 2D images (called signature images) and developing algorithms to reconstruct the 3D model from the signature images.
- To be able to accurately *search for 3D models using shape* with a response time comparable to searching in document digital libraries by using the signature images as a way to compare the shape similarity of 3D models.

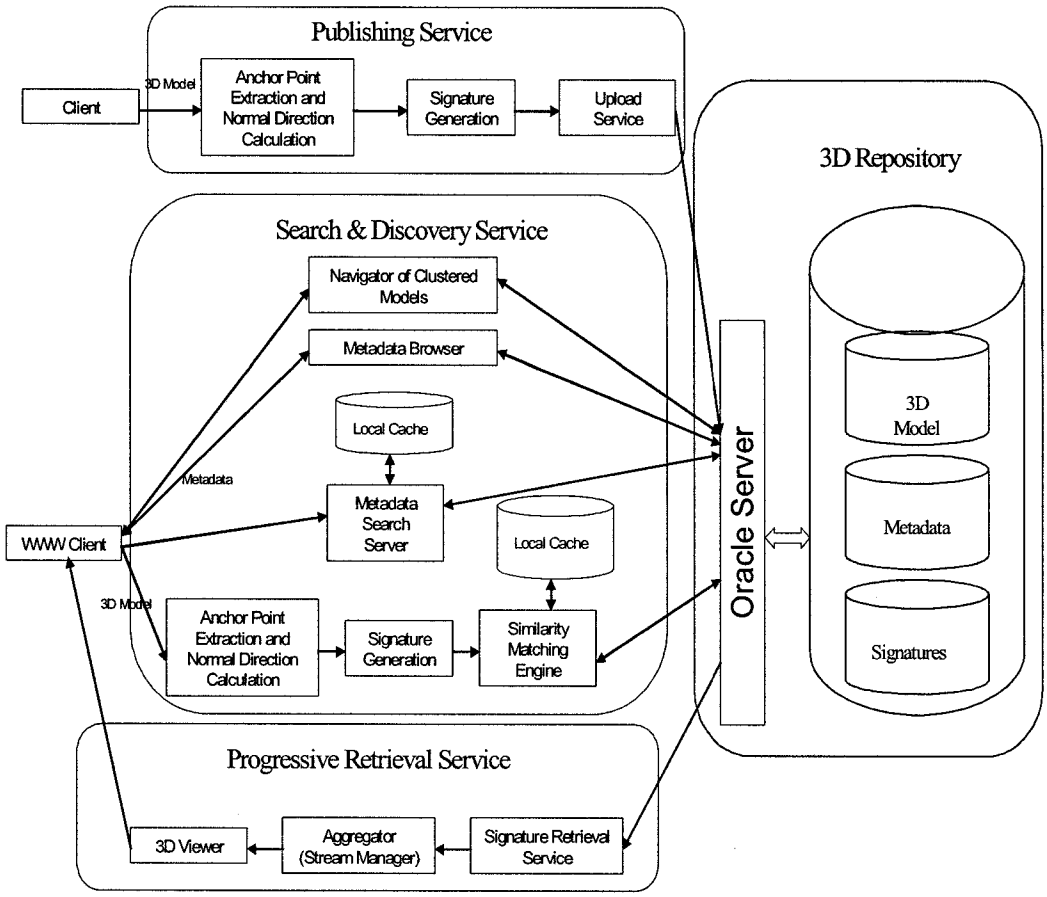


Fig. 1. 3DLIB digital library architecture.

- To provide convenient tools to *browse large collections* of 3D models by using the signature image to cluster similar models together and provide a more abstract view based on clusters, instead of displaying all the models to the user when browsing the database.
- To compare the performance of our 3D shape-based search against other known algorithms and to *evaluate the performance* of browsing.

1.5 Organization of the Dissertation

The rest of this thesis is organized as follows:

Chapter 2 - Background and Related Work: In chapter 2, we will present the background and related work in the areas of 3D representation, progressive compression, shape-based retrieval and clustering. Also, we will describe selected sample digital libraries.

Chapter 3 - Storage and Retrieval Services: In chapter 3, we will show details about the derivation of the surface signature representation of 3D models and how signatures can be used to compress and reconstruct the 3D model.

Chapter 4 - Search and Discovery Services: In chapter 4, we will present how surface signature was used in shape-based retrieval and will present details about our hierarchical browsing service.

Chapter 5 - Performance Evaluation: In chapter 5, we will describe the experiments we conducted to evaluate the performance of our digital library services and will compare the performance of the services to the performance of similar services in the literature.

Chapter 6 - Conclusions and Future Work: Finally, in chapter 6 we will summarize the contributions of our research and provide suggested directions for future work.

CHAPTER II
BACKGROUND AND RELATED WORK

2.1 Introduction

Our objective is to develop a digital library of 3D models with some services to facilitate storing, progressive compression, searching, browsing and retrieving 3D models.

This chapter highlights previous work related to our research. We begin by presenting the existing 3D Model Representation techniques. We follow this by a discussion of progressive compression of 3D Models. Then we talk about shape-based retrieval of 3D Models. As in our thesis we use clustering to group similar models, we present an overview of clustering algorithms. Finally we discuss digital libraries and talk about some samples of digital libraries that manage documents, images or 3D models.

2.2 3D Model Representation

The key to building an efficient digital library for 3D models is to use a 3D model representation that requires less storage, and results in user-friendly discovery services. There are different existing representations for 3D models that vary in their efficiency (in terms of computational complexity or storage), simplicity (in terms of the ease of acquisition or hardware acceleration), and usability. These representations can be divided into the following categories:

- Raw data
- Surfaces
- Solid Models

In the following subsections, we will give examples for each one of these representations.

2.2.1 Raw Data

Raw data is the basic unsophisticated form of 3D model representation. It is usually unstructured, easy to acquire, and may require significant computations and storage. **Point clouds** ([20], [36][), **range images**, and **polygon soup** [55] are examples of raw data.

2.2.1.1 Point Clouds

Point clouds are sets of unstructured points in the 3D space [36]. These points are usually acquired using special sensors or algorithms (such as stereo vision).

2.2.1.2 Range Images

Range images are acquired using laser sensors [68]. The images are rectangular grids of distances from the sensor to the objects being scanned. To capture a scene with a 3D laser scanning system it is often necessary to scan multiple views. Multiple views can be captured by moving the camera about the object and scanning previously occluded regions of the surface. Using this method each camera position has a unique coordinate system that results in accurate reconstruction of the object.

2.2.1.3 Polygons

Polygon soups [55] are collections of unstructured polygons. They are unstructured in the sense that they do not have special order nor do they define separate clusters for different objects.

The mathematical simplicity of polygonal meshes attracts 3D model designers to use in rendering datasets. Polygons act as a standard for designing 3D models; this is because most 3D model representations may be converted with arbitrary accuracy to a polygonal mesh.

Unfortunately, in order to accurately represent a three-dimensional model, large number of polygons is required. For example, a smooth looking sphere requires thousands of polygons to be accurately represented.

2.2.1.4 Discussion

Point clouds and range images are usually easy to acquire. However, polygons can be rendered easier than the later two representations. This is why the use of polygons, mostly triangles, is more common and it is even supported by hardware of most graphic cards.

2.2.2 Surfaces

2.2.2.1 Polygon Meshes

A polygon mesh is a collection of edges, vertices, and polygons that are connected in a way such that each edge is shared by at most two polygons [31]. An edge connects

two vertices, and a polygon is a closed sequence of edges. An edge can be shared by two adjacent polygons, and a vertex is shared by at least two edges.

2.2.2.2 Implicit Surface

A general surface in three dimensions can be represented as:

$S = \{(x,y,z): F(x,y,z)=0\}$.

This is referred to as an implicit representation of the surface.

where

$F(x; y; z) = 0$ is an implicit function that can determine if a point is inside (when $F(x; y; z) < 0$), outside (when $F(x; y; z) > 0$), or on (when $F(x; y; z)=0$) the surface. The equation of a sphere ($x^2 + y^2 + z^2 - r^2 = 0$, where r is the radius) is an example of the function F.

2.2.2.3 Parametric Surface

The parametric representation of surfaces maps a domain's portion to a patch of the surface in another domain. The surface is parameterized by two coordinates that can uniquely identify any position in the plane, and thus any position on the surface patch. A single surface patch may cover an entire surface or only a portion of it. In the latter case, a big part of the surface may be assembled using copies of one or many patches. Creating a rectangular grid over a part of the original domain and evaluating the mapping function at each point on the grid is a simple way of generating parametric surfaces.

2.2.2.4 Discussion

A parametric or implicit representation of surfaces is usually a complex task and in many situations can be applied only to specific types of models. On the other hand polygonal meshes are easier to acquire and process.

2.2.3 Solid Modeling

Solid modeling (sometimes called **volume modeling**) defines the techniques and data structures for constructing 3D objects as rigid solids [16].

Boundary representations (B-rep), spatial occupancy representations, and constructive solid geometry representations (CSG) are the most well known methods of representing 3D solid models.

2.2.3.1 Boundary Representations (B-rep)

A solid may be represented by a set of non-overlapping faces, whose union approximates the solid boundary. Such a scheme is called boundary representation or B-rep [9]. This scheme is based on the fact that an object can be defined by its boundary.

2.2.3.2 Spatial Occupancy Representations

Spatial occupancy representations use either volume elements (called voxels) or a hierarchical structure known as octree, to represent a solid model. In a voxel representation, a solid model is built up from a 3D binary array, where an occupied array element has the value 1 whereas an un-occupied element has a value of 0. The resolution is uniform throughout the voxel representation. An octree representation, however, has a varying resolution across the model, depending on the shape of the object. Octrees can be illustrated with quadtrees, which are a useful way of encoding a spatial occupancy array.

2.2.3.3 Constructive Solid Geometry Representation (CSG-rep)

In Constructive Solid Geometry representation, a binary tree is constructed for each object, where the tree's leaves are solid primitives such as cylinders, spheres, etc. A new solid is made via using Boolean operations such as union, intersection, and set differencing of these primitives.

2.2.3.4 Discussion

CSG-rep is very intuitive and it is excellent to use in situations where shape-based matching is needed. However, unless the 3D model is designed using special tools that keep track of primitive solids constructing the model, it will be a very complex task to get this set of primitive solids. Spatial occupancy is the simplest representation of solids and it can be manipulated to be space efficient as in the case of octree representation.

2.2.4 Discussion of 3D Models Representation

The use of the right representation depends on the application. For example, when models are acquired using scanning devices, raw model representation is used. For graphic design applications, there is more interest on the external shape of the model, which makes surface representation the best choice. For medical industrial design applications, there is interest in modeling the space the model occupies. Hence, solid

representation is the best choice in this case. In this research, we focus on surface representation of 3D models.

2.3 Progressive Compression

Researchers in the past had developed a set of techniques to compress 3D models using different resolutions depending on whether the viewer is far (low resolution) or near (high resolution). These techniques were initially used to render 3D models using multiple levels of details but they evolved to be progressive compression techniques. In these techniques a 3D model in low resolution is represented as a set of polygons. To support higher level of resolution, additional data is included in the representation, which helps in dividing the polygons and generates new polygons to achieve higher level of detail.

Most of the progressive compression techniques start with polygonal representation of the 3D model. The polygonal geometry of small, distant, or unimportant portions of the model is simplified by *polygonal simplification algorithms* to get a lower resolution representation to reduce the rendering cost.

This section presents several progressive compression techniques, each having its own input criteria and advantages. A majority of the algorithms assume that the input model contains only triangular polygons.

2.3.1 Classification of Algorithms

Erikson [29] classified the polygonal simplification algorithms based on the method used to produce a simplified model, into three categories: adaptive subdivision, geometry removal, and sampling. Erikson [29] also classified the algorithms into topology preserving algorithms or topology simplifying algorithms. In the following sections, we will briefly define each of the previously mentioned algorithms. The reader can refer to [29] and [55] for more details.

2.3.1.1 Adaptive Subdivision

This algorithm starts with a very simple base model (that captures the original model important features) and recursively subdivides it, to closely approximate the initial model. The algorithm stops when the approximation is close to the original model as determined by the user. Adaptive subdivision algorithms preserve the surface topology

and thereby appropriate for multi-resolution surface editing, because changes made at low levels of subdivision propagate naturally to higher levels.

2.3.1.2 Geometry Removal

This algorithm starts with the original mesh and iteratively removes faces or vertices from the mesh until the model can no longer be simplified and at the same time when the model is satisfactory to the user. Most of geometry removal algorithms preserve the topology by preventing vertex and face removal operations which may cause topology violations. Hence, they perform very well in removing redundant geometry such as coplanar polygons.

2.3.1.3 Sampling

This algorithm initially samples the geometry of the original model either with points on the model's surface or voxels superimposed on the model in a 3D grid. The user can control the amount of sampling either by specifying the number of points to be sampled or by choosing the dimensions of the grid. These algorithms work well on smooth models with no sharp corners, because high frequency features are difficult to sample accurately and this is where the sampling algorithms may not perform well.

2.3.2 Topology Preserving Algorithms

Topology refers to the connected polygonal mesh's structure. Local topology is the face or vertex's connectivity with the immediate neighborhood. When an algorithm preserves local topology, it does not change the local geometric structure and the number of holes of the object. Global topology is the geometric structure of the whole polygonal model and when an algorithm preserves it, it preserves local topology and does not create self-intersection in the object which occurs when two non-adjacent faces intersect each other.

Topology-preserving algorithms preserve either local or global topology and they work well in applications where surface topology can affect results. They also simplify some applications, such as multi-resolution surface editing, where a correspondence between an object's high detailed and low detailed representations is required.

2.3.3 Topology Simplifying Algorithms

Topology simplifying algorithms do not guarantee the preservation of local or global topology of a model. Thus, the algorithms can close up holes in the model and aggregate separate objects into assemblies as simplification progresses. These algorithms have fewer constraints to satisfy and they can simplify models to a greater degree than topology preserving algorithms. They can be used on scenes or objects when poor visual quality is acceptable.

2.3.4 Discussion

Most of the previous algorithms were designed to support multi-resolution rendering where models are rendered with few number of polygons when seen from a far view point and with higher number of polygons when seen from close view point. Compression of the 3D model was a second priority. None of these techniques was designed with shape-based retrieval in mind. The 3D model representation (signature images) we developed not only acts as a progressive representation of the 3D model but this representation acts as feature used in shape-based retrieval of 3D models. However our representation can not be used for multi-resolution due to the lengthy reconstruction time and due to the fact that using less number of signature images usually results in higher number of polygons in the reconstructed model.

2.4 Shape-Based Retrieval

The rapid growth of modeling and visualization techniques of 3D shapes has led to an explosion of the number of 3D models all over the world. Unlike text documents, 3D models can not be easily retrieved using textual conventions. This leads to an inevitable need for content based 3D shape retrieval methods. The problem of 3D object representation and recognition has attracted a lot of researchers, survey papers to this literature have been provided by Besl et al [9], Campbell et al [13], and Tangelder et al [78]. Content based 3D shape retrieval systems can be classified as shown in Fig. 2 [78]. We will discuss these shape-based retrieval algorithms in the following subsections.

2.4.1 Feature-based methods

In feature based shape matching methods, 3D models can be discriminated according to their geometric and topological properties. In the local based features

methods, for each surface point of the 3D shape a descriptor is used to represent the shape's feature. A shape descriptor is a point in a high dimensional space, two shapes are considered similar if they are close in this space. In global feature-based, view based, and in spatial map feature based methods, features from the 3D shape are represented using a single descriptor consisting of a d-dimensional vector of values, where the value of d is fixed for all shapes.

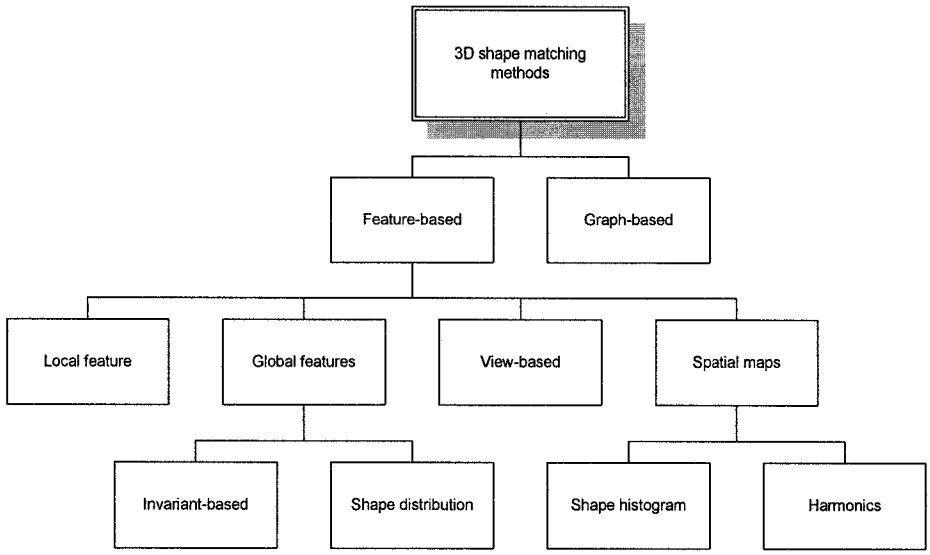


Fig. 2. Classification of shape-based retrieval techniques.

2.4.1.1 Local features methods

Local feature based similarity methods are based on accumulating information about the surface shape in the neighborhood of points on a 3D model's boundary. Such information is very complex and hence local feature based methods are difficult to be applied to 3D matching [78].

2.4.1.2 Global features methods

Global feature-based similarity methods describe the global shape of a 3D model. In invariant based methods, invariants or descriptors of the 3D shape are used. Examples of these descriptors include surface area, volume, and high order moments. Elad et al. [26] used moments based classifier and a weighted Euclidean distance measure to retrieve VRML objects.

In shape distribution methods, measures such as distance, angles, areas or volumes between surface points are used to represent a shape distribution of a 3D model. The shape matching complexity thus reduces to a comparison between probability distributions from being a feature correspondence problem. For example, Osada et al. [63] used random sampling to produce a continuous probability distribution to be used as a signature for 3D shape. The similarity between the objects was evaluated based on measuring distances between distributions. The D2 shape distribution developed by Osada [63] measures distances between pairs of random points on the surface to calculate the similarity between the objects. Horn et al [39] defined an Extended Gaussian Image (EGI) shape descriptor as a spherical function that gives the distribution of surface normals. Kang et al [45] defined a Complex Extended Gaussian Image (CEGI) shape descriptor as a complex-valued spherical function that gives the distribution of surface points' normals and their associated normal distances.

2.4.1.3 View-based methods

In view based similarity methods, if all viewing angles of two 3D models are similar then these two models are similar. Cyr and Kimia [21] recognize a 3D model by comparing its view with all the views of 3D models using shock graph matching. Chen et al [15] developed a Light Field shape Descriptor (LFD) to represent a model as a collection of images rendered from uniformly sampled positions on a view sphere. The minimum difference of all rotations and all pairings of vertices on two dodecahedrons (a dodecahedra is a polyhedron with 12 faces) defines the distance between two descriptors.

2.4.1.4 Spatial map-based methods

Spatial map methods use the spatial location of a 3D model in calculating similarity. The different sections of the model are arranged such that the locations of the

model's features are preserved. Shape histograms and harmonics are examples of spatial map based similarity methods.

In Shape Histograms, the bounding sphere of a 3D model is divided into a number of sectors. A histogram is constructed based on the volume of the 3D model lying in each sector. Ankerst et al. [6] use shape histograms that are built from uniformly distributed surface points taken from 3D molecular surfaces to analyze the similarity of the molecular surfaces. The shape descriptors (SHELLS, SECTORS, and SECSHEL) developed by [6] and used by [74] are based on shape histogram approach. SHELLS is a histogram of distances from the center of the object's body to points on the surface. SECTORS is a spherical function giving the distribution of the model's area as a function of spherical angle. SECSHEL is a collection of spherical functions that give the distribution of the model's area as a function of radius and spherical angle.

Harmonics based methods use spherical or Fourier functions to generate harmonic functions of a shape for shape comparison. For example, Vranic & Saupe [83] used the coefficients of 3D Discrete Fourier Transform as feature vector components of a 3D shape descriptor. Novotni and Klein [62] used 3D Zernike descriptors, an extension of spherical harmonics based descriptor, to capture object coherence in the direction along a sphere and in the radial direction. Kazhdan et al [46] presented a spherical harmonics based approach to transform rotation dependent shape descriptors into rotation independent ones: Gaussian Euclidean Distance Transform (GEDT), and Spherical Harmonic Descriptor (SHD). GEDT is a 3D function whose value at each point is given by the composition of a Gaussian with the Euclidean Distance Transform of the surface. SHD is a rotation invariant representation of the GEDT obtained by computing the restriction of the function to concentric spheres and storing the norm of each frequency. The authors' approach generalizes the method to compute voxel-based spherical harmonics shape descriptor, used by [74], which is defined as binary rasterization of the model boundary into a voxel grid. Vranic et al [83] developed the Radialized Spherical Extent Function (REXT) shape descriptor, which a collection of spherical functions giving the maximal distance from center of mass as a function of spherical angle and radius. Saupe et al [72] defined a Spherical Extent Function (EXT) shape descriptor as a

spherical function that defines the maximal distance from the object's center as a function of a spherical angle.

2.4.2 Graph-based methods

As we have seen in the previous section, feature based methods take into account the pure geometry of the 3D model. On the contrary, graph based methods use graph to extract a geometric meaning from a 3D model by showing how shape components are linked together.

Lou et al. [51] used global features and skeletal graphs to describe volume models, where they obtained a skeletal graph by using a thinning algorithm that erodes voxels of voxelized solid models to a one-voxel width. They matched two shapes by an algorithm that detects graph/sub-graph isomorphism based on a decision tree approach. Though their algorithm obtains a search time polynomial in the number of graph nodes, their approach is feasible for relatively small volume models.

2.4.3 Discussion

Shape-based retrieval methods that are graph-based are very slow and most of them can be applied to solid models only which makes it impractical to use in many situations [78].

Shape-based retrieval methods that use global features are usually fast. However, they usually have low discriminative power which leads to low precision in retrieving query results [78]. Methods that use local features use complex calculations and are less efficient than global-feature based methods. However, they have high discriminative power. Spatial maps provide good retrieval results within reasonable time. The shape-based similarity technique, we are using in this research combines local features with global features resulting in fast and precise retrieval of results.

2.5 Clustering

Clustering is an unsupervised process that structures a collection of unlabeled data. It is different from classification in that classification predicts the classes of unlabelled data based on a supervised training on pre-labeled data. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. Users have to provide a grouping criterion to constitute good clustering results that suits their needs. A cluster is

a collection of "similar" objects, where the similarity criterion is based on the distance between objects. The clustering process consists of the following steps.

- ***Preprocessing and feature selection***: This involves preprocessing and extracting appropriate features from data objects to construct n-dimensional feature vectors that are necessary for objects' representation. In order to reduce the dimensionality of the problem, a subset of all the available features should be chosen which in turn requires good domain knowledge and data analysis.
- ***Similarity measure***: This is a function which calculates the distance between objects to decide how close they are and hence whether they are similar or not.

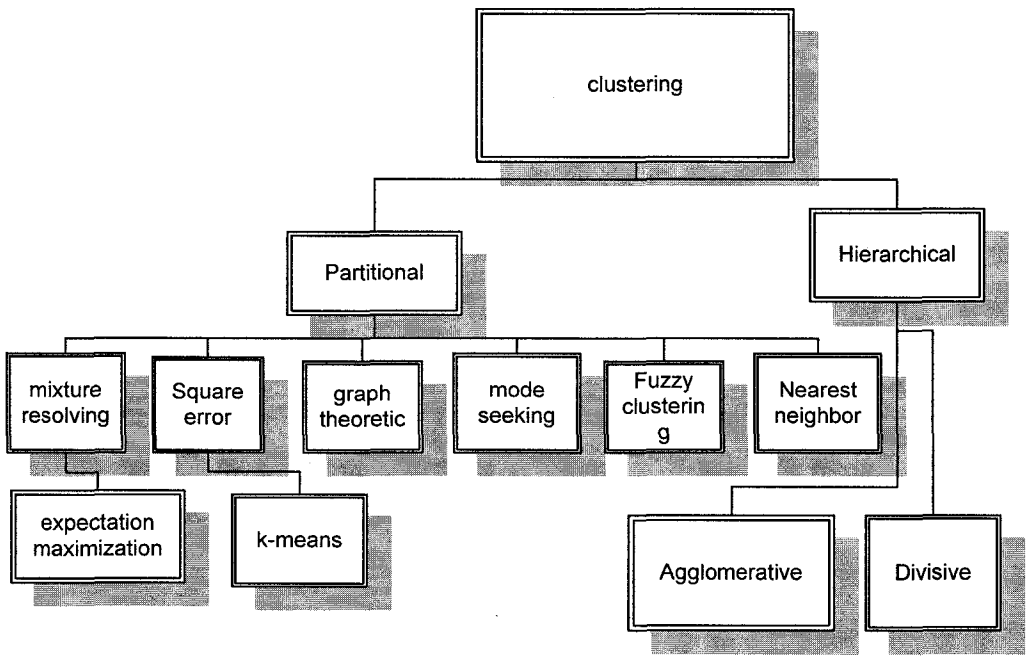


Fig. 3. Classification of clustering algorithms.

- ***Clustering algorithm***: Clustering algorithms are the schemes that use particular similarity measures to assign data to clusters.

- ***Result validation***: It involves evaluating results to check whether they make sense or not. It is useful to check whether clusters are available at all or not (testing for clustering tendency). It is worth to mention that some clusters will be produced regardless of whether natural clusters exist or not.

Different clustering methods can be described with the aid of figure which is based on the taxonomy of the survey done by Jain et al. [41]. We will discuss these clustering algorithms in the following subsections.

2.5.1 Hierarchical Clustering

Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity. The end result of the algorithm is a tree of clusters that shows how the clusters are related. The complete process of hierarchical clustering can be summarized as shown in Fig. 4.

- *Calculate the distance between all initial clusters. In most analysis initial clusters will be made up of individual cases.*
- *REPEAT*
    - *Merge the two most similar clusters and recalculate the distances.*
- *UNTIL all cases are in one cluster.*

Fig. 4. Hierachical clustering.

Hierarchical algorithms can be further divided into divisive (top down) and agglomerative (bottom up). Agglomerative hierarchical clustering starts with every single object in a single cluster. Then, it successively agglomerates (merges) the closest pair of clusters by satisfying some similarity criteria, until all data is in one cluster.

The single link and complete link clustering algorithms are the most popular agglomerative hierarchical clustering techniques. The two methods differ in the way they

define the similarity between a pair of clusters. The single link algorithm takes the distance between two clusters as the minimum of all the distances between all pairs of patterns drawn from the two clusters, while the complete link algorithm takes the maximum distance. The complete link algorithm produces compact clusters and more useful hierarchies in many applications than the single link algorithm [42]. The single link algorithm, on the other hand, tends to produce elongated clusters. It is more versatile than the complete link algorithm where it can extract concentric clusters, for example, when the complete link fails.

On the contrary to agglomerative algorithms, divisive algorithms starts with a single cluster containing all objects, and then successively splits resulting clusters until only clusters of individual objects remain. All possible partitions of clusters are checked at every step to find the partition that would yield two clusters of minimum similarity (the most dissimilar sets of items need to be in separate clusters as soon as possible).

The main disadvantage of both the agglomerative and the divisive hierarchical techniques is that once a merge or a split is done, it can not be reversed or refined.

2.5.2 Partitional Clustering

Partitional clustering algorithms attempt to directly decompose the data set into a set of disjoint clusters. The decomposition is achieved by minimizing a criterion function that is defined either locally (on part of the data) or globally (over all the data).

2.5.2.1 Square error clustering methods

Square error methods are the most commonly used clustering methods which are based on the square-root error criterion. The square error is the sum of the Euclidean distances between every pattern and the center of its cluster. The objective of square error clustering methods is to obtain a partition that minimizes the square error for a fixed number of clusters. The algorithm is shown in Fig. 5 [41].

K-means algorithm is the simplest and most commonly used algorithm that employs the square error criterion. It starts with a randomly selected initial partition and keeps reassigning the patterns to clusters based on the similarity between the pattern and the cluster centers until it meets a convergence. This algorithm has been applied to large

scale data sets due to the linearity of its time complexity. However, it can converge to a local minimum if the initial partition is not chosen carefully [41].

- *Select an initial partition with k clusters.*
- *REPEAT*
  - *REPEAT*
    - *Generate a new partition by assigning each pattern to its closest cluster center.*
    - *Compute new cluster centers as the centroids of the clusters.*
  - *UNTIL an optimum value of the criterion is found.*
  - *Adjust the number of clusters by merging and splitting existing clusters or by removing small or outlier clusters.*
- *UNTIL the cluster membership stabilizes.*

Fig. 5. Square error clustering.

2.5.2.2 Mixture-Resolving Clustering Methods

Mixture-resolving methods assume that the data items in a cluster are drawn from one of several distributions (usually Gaussian) and the goal is to estimate the parameters and the number of all these distributions. Traditional approaches to solve the parameters estimation problem involve obtaining a maximum likelihood estimate of the component densities parameter vectors. The expectation maximization (EM) algorithm, which is a general purpose maximum likelihood algorithm, can be used to solve this problem. The EM algorithm initially assumes a parameter vector and iteratively re-calculates the data items against the density produced by the parameter vector. The new data items are then used to update the parameter vector [41].

Mixture-resolving methods have a high computational complexity and make rather strong assumptions regarding data distribution. They usually view each cluster as a single simple distribution and thus strongly constrain the shape of the clusters.

2.5.2.3 Graph Theoretic Clustering Methods

These algorithms treat the patterns as nodes in a graph and the dissimilarity between two patterns is the length of the edge between the two corresponding nodes. Each pattern is connected with all its neighbors to form a complete graph. Most graph theoretic algorithms are based on constructing the minimal spanning tree (MST) of the data where the longest edges are deleted to generate clusters. The MST-based clustering algorithm is summarized in Fig. 6 [41].

- *Construct the MST of a set of patterns.*
- *Identify inconsistent edges whose weight is significantly larger than the average weights of the neighbor edges.*
- *Remove the inconsistent edges and make clusters from the connected components.*

Fig. 6. Graph theoretic clustering.

The MST approach, however, suffers from a great deficiency when it works in more than two dimensions. This deficiency involves the complexity of identifying the inconsistent edges in three or more dimensional data. Proper heuristics need to be used in complex data which in turn requires a prior knowledge of the shapes of the clusters to be able to choose the prior heuristic.

2.5.2.4 Mode Seeking Clustering Methods

In mode seeking clustering algorithms, clusters are viewed as dense sets of data items separated by less dense regions; clusters may have arbitrary shape and data items can be arbitrarily distributed [41]. Clusters can be identified by searching for regions of high density, called modes, in the data space where each mode is associated with the

cluster center and each item is assigned to the cluster with the closest center. The simplest way to identify modes is to search for bins with large number of items in a multidimensional histograms of the input data set. Histograms can be constructed by partitioning the data space into a number of non-overlapping regions.

2.5.2.5 Fuzzy Clustering Methods

In this type of clustering, an object can be a member of more than one cluster. The relationship between objects and clusters are maintained through a membership function. Fuzzy clustering algorithms can be summarized as in Fig. 7 [41].

- *Select an initial fuzzy partition of N objects into K clusters.*
- *REPEAT*
  - *Select the N x K membership matrix such that members have values in [0,1].*
  - *Compute a fuzzy criterion function which is a weighted squared error criterion function associated with the corresponding partition.*
  - *Reclassify patterns to reduce the value of the criterion function.*
- *UNTIL the cluster memberships do not change significantly.*

Fig. 7. Fuzzy clustering.

Fuzzy clustering avoids local minima of the criterion function and it is useful in cases where the clusters are touching or overlapping.

2.5.2.6 Nearest Neighbor Clustering Methods

In this type of clustering, each unlabeled data item is assigned to the cluster of its closest labeled neighbor item, provided that the distance between the unlabeled item and that labeled neighbor is below a certain threshold. The assignment continues until all items are labeled or no additional labeling occurs [41].

2.5.3 Discussion

The **k-means** clustering algorithm is a well known, understood, and a simple algorithm that has been adapted to many problem domains. The linearity of its time complexity O (n), where n is the number of objects to be clustered, is what makes it attractive to researchers to use in many clustering problems. In designing our hierarchical navigation service, we used k-means clustering algorithm to group 3D models together for faster navigation. Though the clustering was done offline, it took us about 20 minutes only to cluster 1800 3D models. In addition to having a linear time complexity, k-means has another advantage that the number of clusters, k, needs to be specified in advance. In designing our hierarchical navigation service, we needed to specify a maximum number of clusters that a user can view at a time and this is one of the reasons we chose k-means to implement.

2.6 Digital Libraries

2.6.1 Definition of a Digital Library

The term "digital library" has been defined in literature by many different ways. It can be defined as a data repository, a collection of digital objects, or to be an extension of the traditional library. Being an extension of the traditional library, a digital library will carry out the traditional library functions of collection, preservation and access provision, in addition to integrating digital media and remotely accessible digital services [18].

According to Terry Smith [76], a digital library is an enhanced version of a traditional library where digital technology has been applied to enhance many of traditional libraries' aspects. These aspects include the organization and management of library collections, accessing library items, processing and communicating the information of the different items. We define a digital library to be a set of collections and services. Collections can contain text documents, image or any type of document that can be stored in digital format. Services provided in the digital library should –at a minimum- support storage, search, retrieval, publishing and management of documents. In the following section, we will provide sample of existing digital libraries including those who have text document, images or 3D models.

2.6.2 Sample Digital Libraries

2.6.2.1 Text Digital Libraries

Arc [50] is one of the first federated searching services based on the OAI protocol. "Arc harvests metadata from several OAI compliant archives, normalizes them, and stores them in a search service based on a relational database (MYSQL or Oracle). The *Arc* architecture is based on the Java Servlets-based search service that was developed for the Joint Training, Analysis and Simulation Center (JTASC) ([58],[59]). This architecture is platform independent and can work with any web server. Moreover, the changes required to work with different databases are minimal. Arc's implementation supports two relational databases, one in the commercial domain (Oracle), and the other in the public domain (MySQL). The architecture improves performance by employing a three-level caching scheme" [50] .

Archon ([2],[57]) is a federation of physics collections with varying degrees of metadata richness. "Archon provides services on these metadata that make the federation a learning environment useful not only for the researcher but for students as well. Archon uses the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) to harvest metadata from several Physics collections that are geographically distributed. The architecture of Archon [2], [57] is largely based on Arc [50], the first OAI-PMH compliant service provider to provide end-user search services across OAI-PMH repositories. However, Archon provides some services that are specifically tailored for research and education in the physics community. Archon currently supports searching and browsing of equations and formulae and a citation linking service for arXiv and American Physical Society (APS) archives. The architecture of Archon provides the following basic services: a storage service for the metadata of collected archives; a harvester service to collect data from other digital libraries using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [47]; a search and discovery service; and a data provider service to expose the collected metadata to other OAI-PMH harvesters. A range of services were especially developed for physics collections based on metadata available from the participating. For example, a service to allow searching on equations embedded in the metadata was provided. It also provides a cross-archive

citation service has been developed to integrate heterogeneous collections into one unified linking environment" [2],[57].

2.6.2.2 Image and Multimedia Digital Libraries

The **American Memory project** contains the historical collections for the National Digital Library at the Library of Congress. It contains multimedia collections of digitized documents, photographs, recorded sound, moving pictures, and text. There are currently over 40 collections in American Memory digital library.

The **Astronomy Digital Image Library (ADIL)** [66] was supported by NASA and the National Science Foundation (NSF) to address some of the challenges of distributing scientific data over the network. The purpose of ADIL is to collect fully processed astronomical images in FITS format (a standard astronomical image format) and make them available for searching, browsing, and downloading to the research community and the interested public via the World Wide Web. ADIL also allows researchers to add images to its growing collection, thus providing them with a place to archive and share their fully processed images with the community.

The **Columbia's Content-Based Visual Query Project** was developed at Columbia University [13] to establish large digital image and video archives. It provides content based retrieval tools by allowing users to specify image queries by giving examples, drawing sketches, selecting visual features like color and texture, and by arranging spatial structure of features.

The **Informedia Digital Video Library** at Carnegie Mellon University is one of the NSF/DARPA/NASA jointly funded Digital Library Initiative projects [85], established in 1995. This particular effort focuses on the achievement of machine understanding of video and film media, including all aspects of search, retrieval, visualization and summarization in all collections via desktop computers and metropolitan area networks. The library has been deployed in education, health care, defense intelligence and the coordination of human activity. Additionally, it has been extended to include multilingual, multi-cultural and cross-lingual versions across Europe and Asia.

2.6.2.3 3D Digital Libraries

The **Visible Human Project** [8] from National Library of Medicine (NLM) produced complete, anatomically detailed, three-dimensional representations of the male and female human body by collecting transverse CT, MRI and cross section images of representative male and female cadavers at one millimeter intervals.

Princeton 3D Model Search Engine [60] provides a repository and search engine for 3D models. Its goal is to investigate issues in shape-based retrieval and analysis of 3D models. The Princeton 3D search engine has about 36,000 models and they provide text search interface as well as various 3D and 2D query interfaces.

3DESS: A 3D Engineering Shape Search System [40] was developed at Purdue Research and Education Center for Information Systems in Engineering (PRECISE). The primary goal of this project is to develop a search system for 3D engineering shapes. This required the extraction of four feature vectors which are moment invariants, geometric ratios, principal moments, and eigenvalues of the adjacency matrix of skeletal graph. The system consists of two modules: multidimensional index and database. Multidimensional indexes is hard to be implemented in current commercial databases, therefore it was built on top of the system's database (Oracle 8i). Whenever a shape is inserted in the database, an ID is generated for it, all the feature vectors are extracted and stored in the database, and the index is updated with the ID and feature vectors. The 3DESS database consists of 113 engineering shapes employed to test the effectiveness of the shape descriptors. The 113 shapes are manually classified into 26 similar groups and they are mixed with some noisy shapes that do not belong to any group. .

2.6.3 Discussion

The Visible Human Project produced an excellent set of anatomically detailed 2D and 3D models of the male and female human body. However, this project lacks essential 3D digital library services such as a user-friendly interface to search for models or the capability to search by characteristics of models other than text. On the other hand, the Princeton 3D Model Search Engine is the most complete digital library existing for 3D models. They provide services for search and retrieval of 3D models by text annotation or by shape. However, they do not provide specialized navigation services for large

collection. In this research, we will present details of developing a hierarchical browsing service to reduce the time for browsing large collections. Moreover, we will prove that our 3D search is more accurate than that provided by Princeton. 3DESS supports services such as searching 3D models by shape. However, 3DESS does not provide compression services specialized for 3D models. Due to lack of data, we could not compare the performance of our shape-based retrieval to that of 3DESS. However, in our research, we provide a storage service that stores 3D models in compressed format that will enable progressive reconstruction of the 3D model.

CHAPTER III
STORAGE AND RETRIEVAL SERVICES

3.1 Introduction

Our objectives include reducing the storage requirements for 3D models, searching for 3D models by shape and providing convenient tools for browsing large collections. The key to achieve this is developing a compressed representation for 3D models, which can also be used to identify or recognize the 3D model in the compressed format. We developed the surface signatures which are 2D images that capture information about the 3D model and can be used in recognizing the 3D model. The surface signatures are a compressed representation of a 3D model, and as demonstrated the original 3D model can be easily reconstructed from these signatures.

We now give details for the surface signature of a 3D model followed by the inverse signature transform for reconstructing the original model.

3.2 Surface Signature

A surface signature at a given point on the surface of an object is a descriptor that encodes the geometric properties measured in a neighborhood of the point. In early work, Besl and Jain [9] characterized surface points according to the signs of their mean and Gaussian curvatures. In recent years, more complex surface signature representation schemes have been reported in the literature. They include the splash representation of Stein and Medioni [77], the point signatures of Chua and Jarvis [17], the shape spectrum scheme of Dorai and Jain [26], the surface signatures from simplex meshes of Yamany et al., [86],[5], the harmonic shape images of Zhang and Hebert [87], and the spin-image representation introduced by Johnson and Hebert [41].

We developed a new surface signature representation based on the surface signature, developed by Yamany and Farag [86]. Yamany ([86], [5]) encoded the curvature of the surface inside the signature and computed the signature at selected points on the surface. In our version of the signature, we reduced the signature size by removing curvature information, and we extended the definition of the signature to allow it to be computed at points that might not be on the surface of the 3D model. Moreover,

we updated the calculation of the signature to be able to use it in reconstructing the original 3D model as we will show in the following subsections.

The signature image is computed as follows [4], [5]: Using an anchor point A, defined by its 3-D coordinates and a direction $\vec{U}_A$, each other point P_i on the surface can be related to A by two parameters:

(1) the distance $d_i = \|A - P_i\|$ and (1)

(2) the angle $\alpha_i = \cos^{-1}\left(\frac{\vec{U}_A \cdot (A - P_i)}{\|A - P_i\|}\right)$. (2)

Examples of some basic 3D surfaces and the signature viewed from some selected points are illustrated in the following figures. These examples are included to get an idea of how a typical signature looks like and will be used later in illustrating how to get the inverse signature which is the basis for our compression technique. Fig. 8 shows a 3D sphere with radius 1 and Fig. 10 shows the signature at any point on the surface of the sphere using the surface normal as the direction for the anchor point (any point on the sphere has the same signature).

Fig. 10 shows a 3D cone and Fig. 11, Fig. 12, and Fig. 13 show the signature of the cone calculated from 3 different points. Notice that the signature calculated from the apex of the cone is a straight line as the signature. This is due to the fact that all points on the surface of the cone have the same (α) value with respect to the apex of the cone.

3.2.1 Anchor Point Selection

The surface signature is calculated from different view points called anchor points. The strategy for selecting anchor points, greatly affects the number of signatures to be used. Hence; it controls the compression ratio and accuracy. In the following subsections, we will illustrate different approaches along with their trade-offs in acquiring the anchor points.

3.2.1.1 Highest Curvature Points

In this approach, we select points on the 3D surface having highest curvature values to be anchor points. These points are selected because they are perceived to capture the shape of the 3D models from different perspectives. Moreover, they are

independent of the orientation of the 3D model. The curvature was calculated at each point on the surface using an approximation approach developed by Yamany [86].

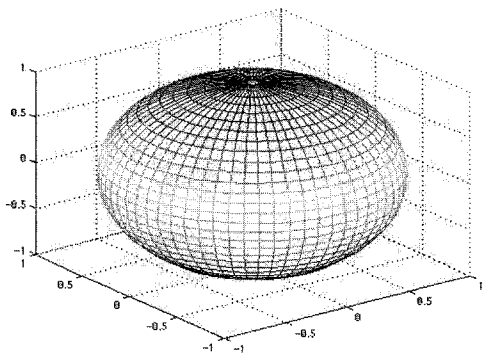


Fig. 8. Sphere.

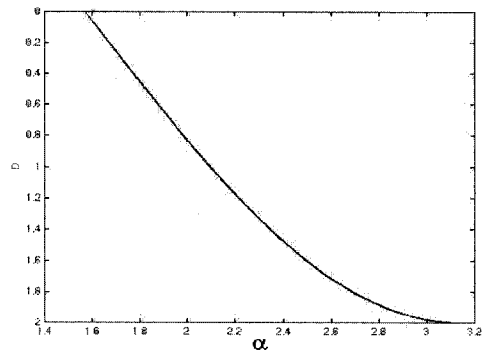


Fig. 9. Signature of a sphere.

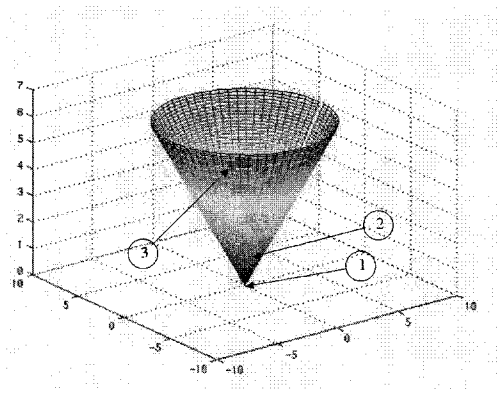


Fig. 10. Cone.

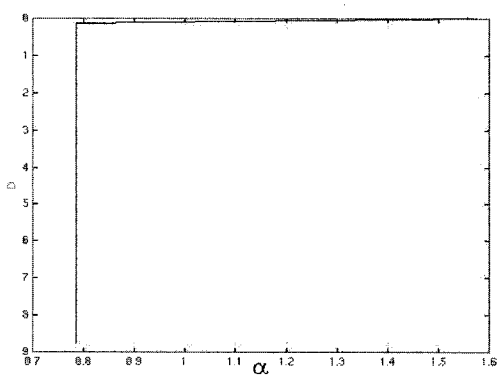


Fig. 11. Signature of cone (from apex).

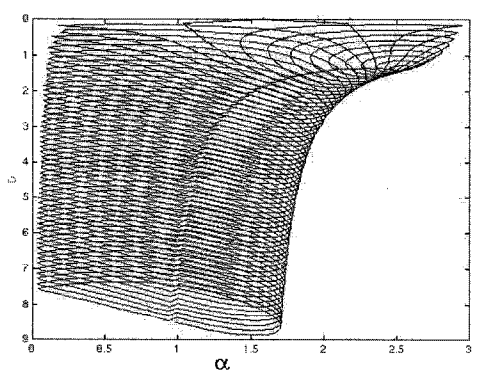


Fig. 12. Signature of cone (from point 2).

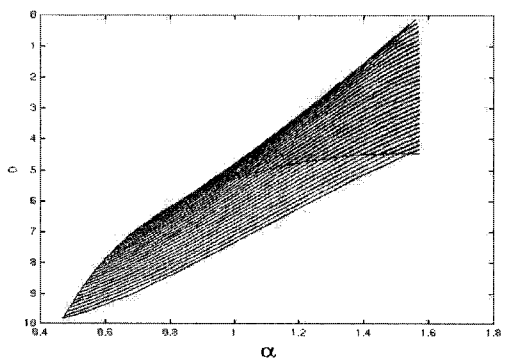


Fig. 13. Signature of cone (from point 3).

We conducted some experiments to see if using highest curvature points as anchor points will be effective. In the experiments, we used a 3D model and calculated the signature at an anchor point on the surface that has the highest curvature and we used the normal direction as the surface normal at the anchor point then we calculated the surface signature. We then repeated the experiment with variations of the same model after removing some parts of the model or applying Gaussian noise to the points of the model.

Table 1 shows Variations of a 3D chair with signatures calculated at an anchor point that has highest curvature. Table 1 shows variations of 3D chair with signatures calculated at an anchor point that has the highest curvature.

Table 1 shows that removing parts of the 3D model, has minimal effects on the shape of the signature. However, adding noise significantly affects the shape of the signature.

3.2.1.2 Using the Signature Area to Select Anchor Points

When calculating the signature for the sphere, we found that any point on the surface of the sphere have the same signature. This it is enough to use only one anchor point for representing a sphere. Looking at Fig. 10- Fig. 13, we can see signatures of the cone from different points on its surface. Fig. 11 shows the signature of the cone where the anchor point is at the apex. This signature is a single line. This signature is special for the fact that it can be used (alone) to reconstruct the full 3D cone. On the other hand, the other two signatures in Fig. 12, Fig. 13 can not fully reconstruct the cone.

We looked at the special characteristic of this signature and found that it has the minimum area. We, then, conducted experiments to find is using anchor points that produce a minimal area signature will be better for reconstructing the 3D model. In the experiments, we acquired a 3D model, calculated the surrounding cube for that 3D model. Then we divided that cube into set of 15 equidistant planes.

On each plane, we calculated set of 15x15 equidistant points that cover the whole plane. We used these equidistant points as anchor points. For each anchor point, we calculated the signature.

TABLE 1

VARIATIONS OF A 3D CHAIR WITH SIGNATURES CALCULATED AT HIGHEST CURVATURE POINTS

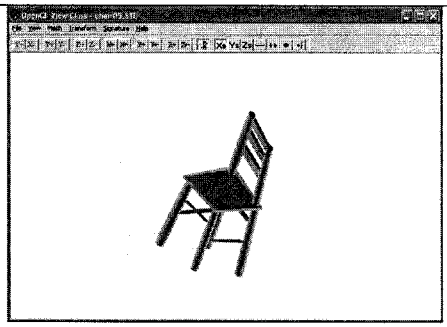
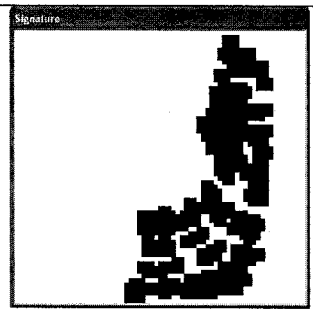
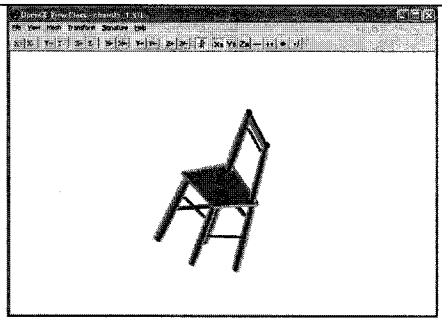
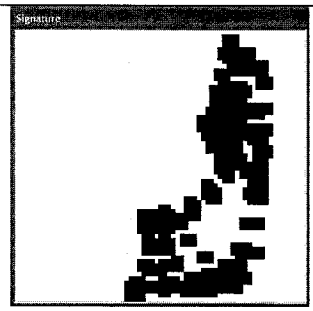
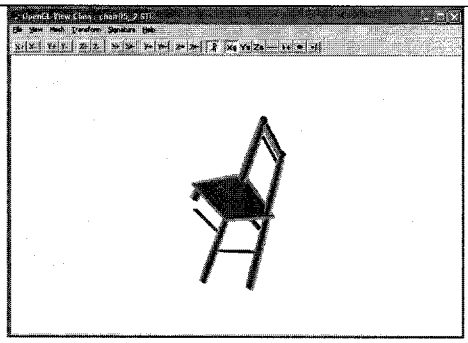
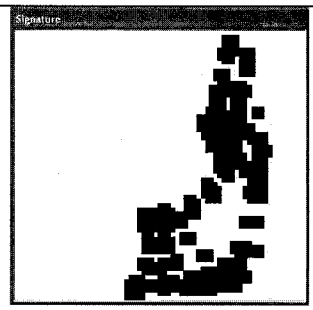
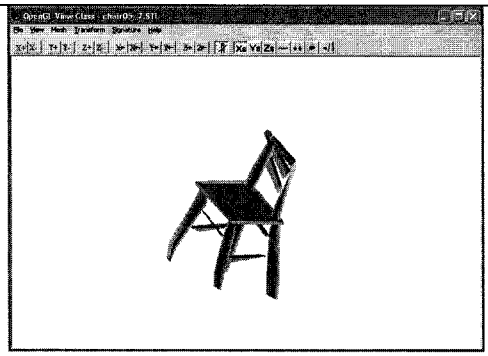
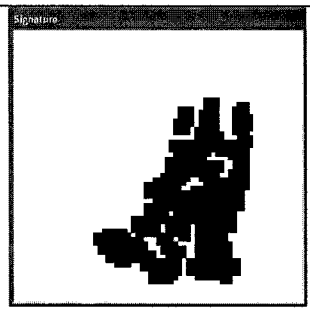
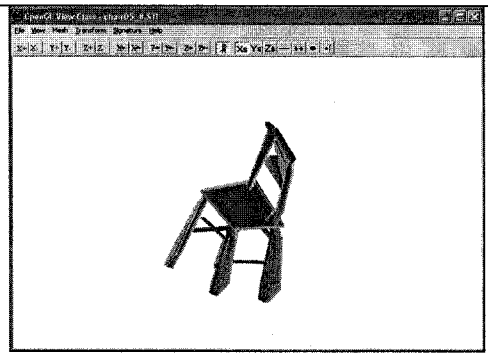
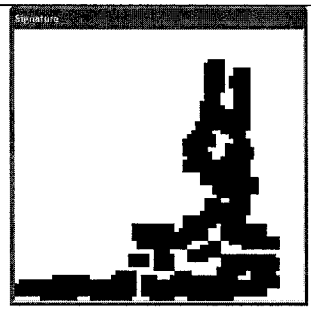
Model	Signature at Highest Curvature Point
 Fig. 14. Original 3D chair.	 Fig. 15. Signature of the original 3D chair.
 Fig. 16. 3D chair after removing some parts from the back of the chair.	 Fig.17. Signature of the chair that has missing parts from the back.
 Fig. 18. 3D chair after removing some parts from the back of the chair and two legs.	 Fig. 19. Signature of a chair that has missing parts from the back and missing legs.

TABLE 1 (continued)

Model	Signature at Highest Curvature Point
 Fig. 20. 3D chair after applying 5% noise.	 Fig. 21. Signature of 5% noisy model of a chair.
 Fig. 22. 3D chair after applying 10% noise.	 Fig. 23. Signature of 10% noisy model of a chair.

Based on the signature, we calculated the area of the signature defined as the number of pixels in the signature. Then we tried to reconstruct the 3D model from this signature and computed the reconstruction error.

Fig. 24 shows the actual 3D model and Fig. 25 - Fig. 39 show the area of signature at each anchor point along with the reconstruction error for each of the 15 planes.

Looking at the graph, we could not conclude that there is a relation between the signature area and the reconstruction error. Hence, we decided not to select anchor points based on the area of the signature.

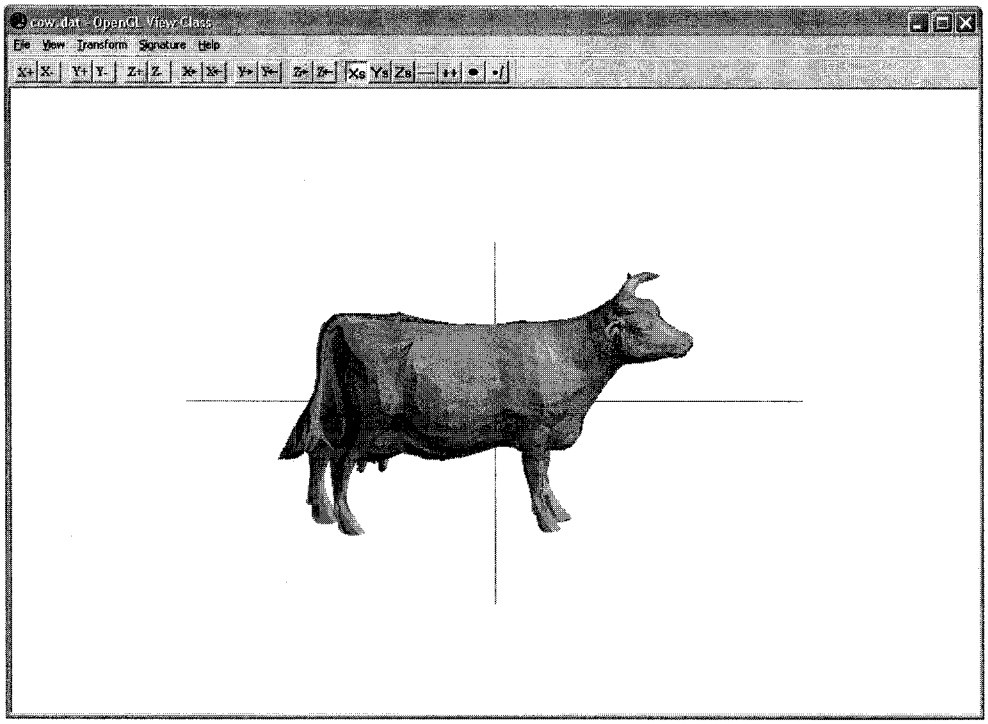


Fig. 24. A 3D model for a cow.

3.2.1.3 Medial Axis

Medial Axis Definition

The medial Axis (MA) of a shape is the locus of the centers of all the maximal inscribed circles of the shape [11].

A maximal inscribed circle does not intersect with the shape's exterior; it does however touch the shape's boundary at two or more points.

The medial axis of a 3D model is the locus of the center of a maximal sphere, as the sphere rolls around the object interior, which can be referred to as the medial surface [74]. Fig. 41 shows examples of Medial Axes for 3D models.

The distance from a medial axis point to its nearest boundary points defines the *local feature size*, which measures the local diameter of the geometry. The medial axis together with the local feature size gives a complete description of the original geometry. This description is referred to as the *Medial Axis Transformation.*

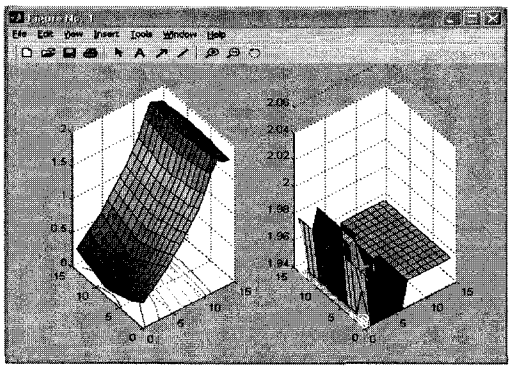


Fig. 25. Signature area (right) and reconstruction error (left) at plane 1.

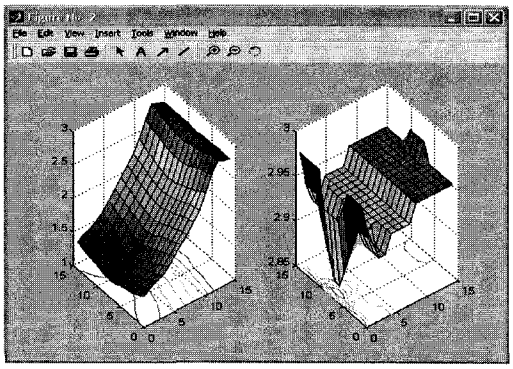


Fig. 26. Signature area (right) and reconstruction error (left) at plane 2.

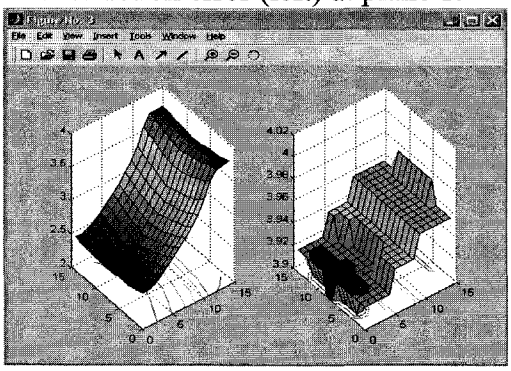


Fig. 27. Signature area (right) and reconstruction error (left) at plane 3.

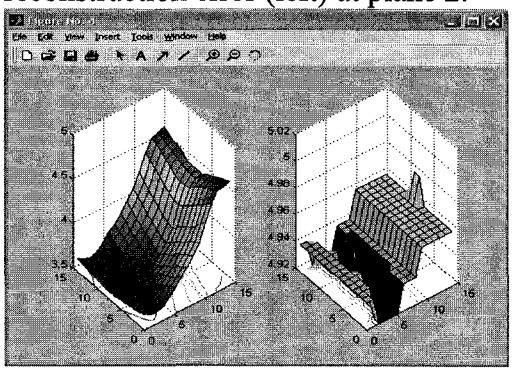


Fig. 28. Signature area (right) and reconstruction error (left) at plane 4.

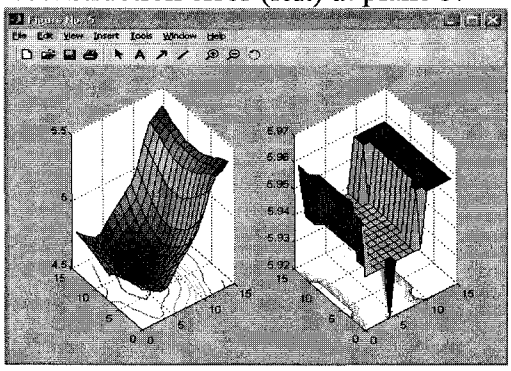


Fig. 29. Signature area (right) and reconstruction error (left) at plane 5.

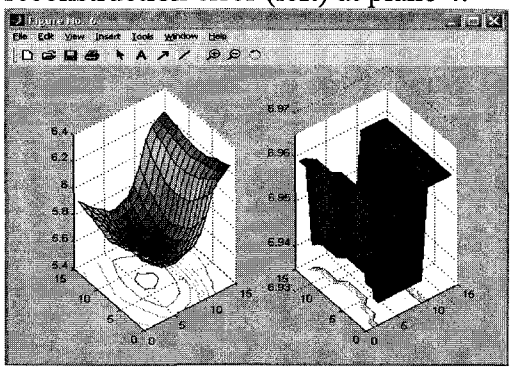


Fig. 30. Signature area (right) and reconstruction error (left) at plane 6.

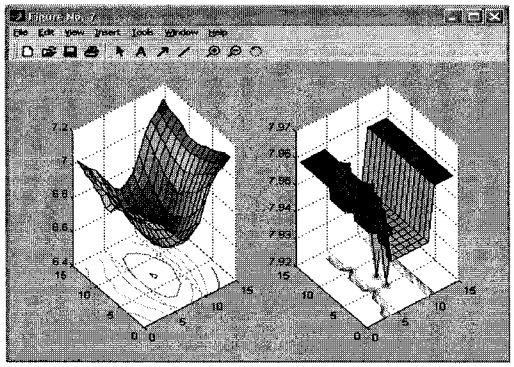


Fig. 31. Signature area (right) and reconstruction error (left) at plane 7.

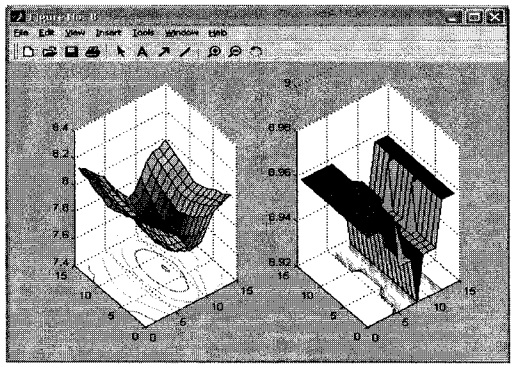


Fig. 32. Signature area (right) and reconstruction error (left) at plane 8.

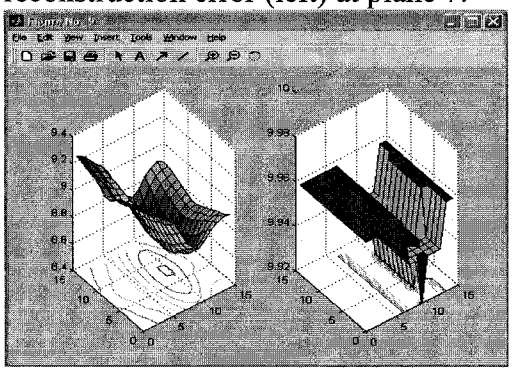


Fig. 33. Signature area (right) and reconstruction error (left) at plane 9.

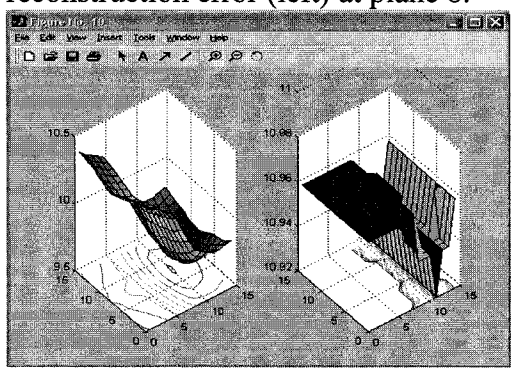


Fig. 34. Signature area (right) and reconstruction error (left) at plane 10.

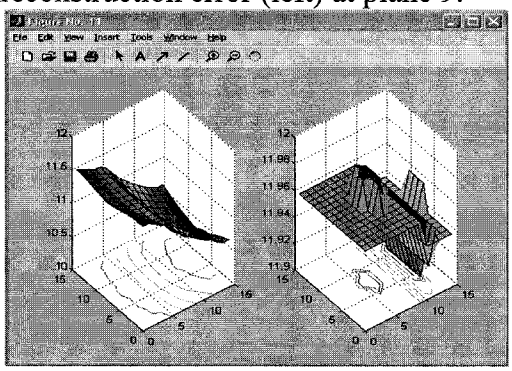


Fig. 35. Signature area (right) and reconstruction error (left) at plane 11.

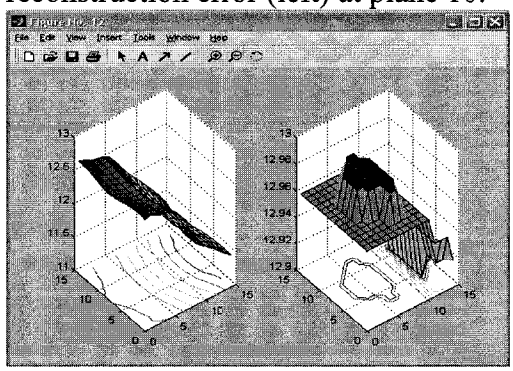


Fig. 36. Signature area (right) and reconstruction error (left) at plane 12.

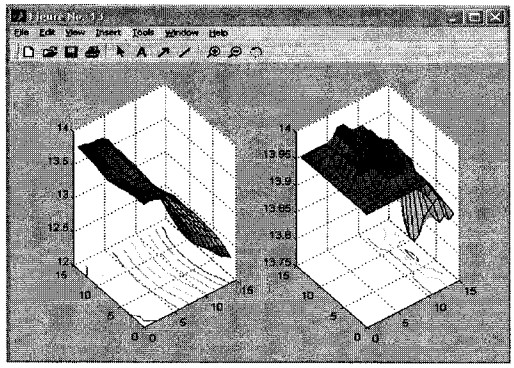


Fig. 37. Signature area (right) and reconstruction error (left) at plane 13.

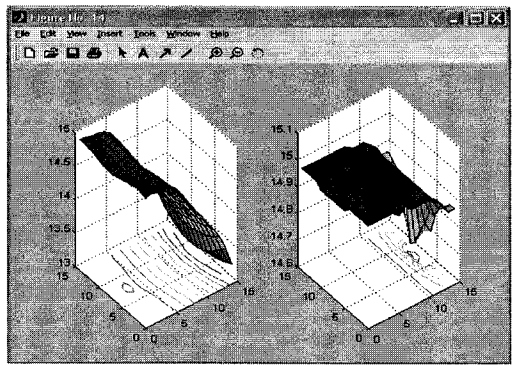


Fig. 38. Signature area (right) and reconstruction error (left) at plane 14.

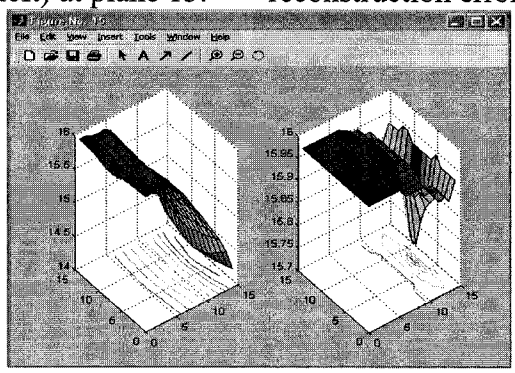


Fig. 39. Signature area (right) and reconstruction error (left) at plane 15.

The medial axis transformation (MAT) specifies the radii of the maximal circles along with their centers to define the object's skeleton. The goal of finding the skeleton is to reduce the object's dimensionality and hence extracts a simple, robust representation of the shape of an object.

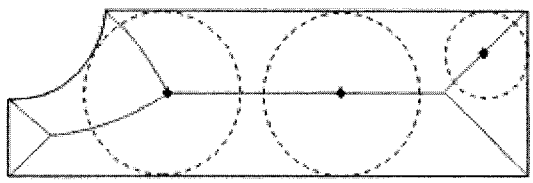


Fig. 40. Example of 2D medial axis.

The MAT however does not reflect non-geometric related topology (CAD topology), which obstructs the mesh generation of objects. This makes the MAT a useful tool geometrically, but may not be used solely to determine meshing complexity.

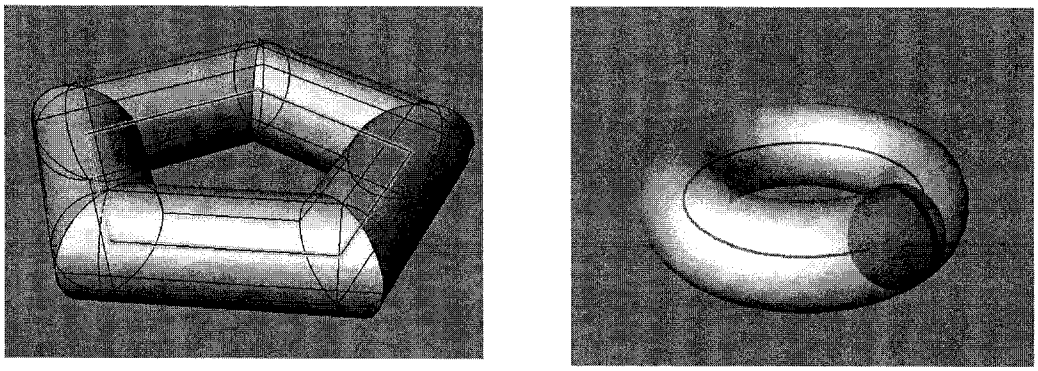


Fig. 41. Examples of 3D medial axis.

Why Use Medial Surface in positioning anchor points

The inverse signature uses the intersection of a sphere whose center is the anchor point with a cone that has the anchor point as an apex. Hence, using anchor points as the medial axis (centers of sphere tangent to the surface) will maximize the number of correct points generated by the signature. This will reduce the total number of signatures needed to reconstruct the 3D model.

Problems in using the medial surface points as anchor points

The use of the medial axis has been limited mainly by two significant drawbacks: First, it is unstable, in that small deformations in the boundary of the solid can lead to large changes in the medial axis. Second, it is difficult to compute because of the underlying algebraic complexity. Geometric computation with primitives of high degree is hard to make the medial axis calculation both reliable and fast. For example, calculating an approximate medial axis for a 3D model with about 3700 vertices may take about 10 minutes. This might work if the calculations are done offline, however, in

real time applications when a user submits a model online to retrieve similar models he is not likely to wait for 10 minutes to get the results back.

3.2.1.4 Pre-specified fixed points

In this approach, fixed points relative to the surrounding cube of the model are selected as anchor points.

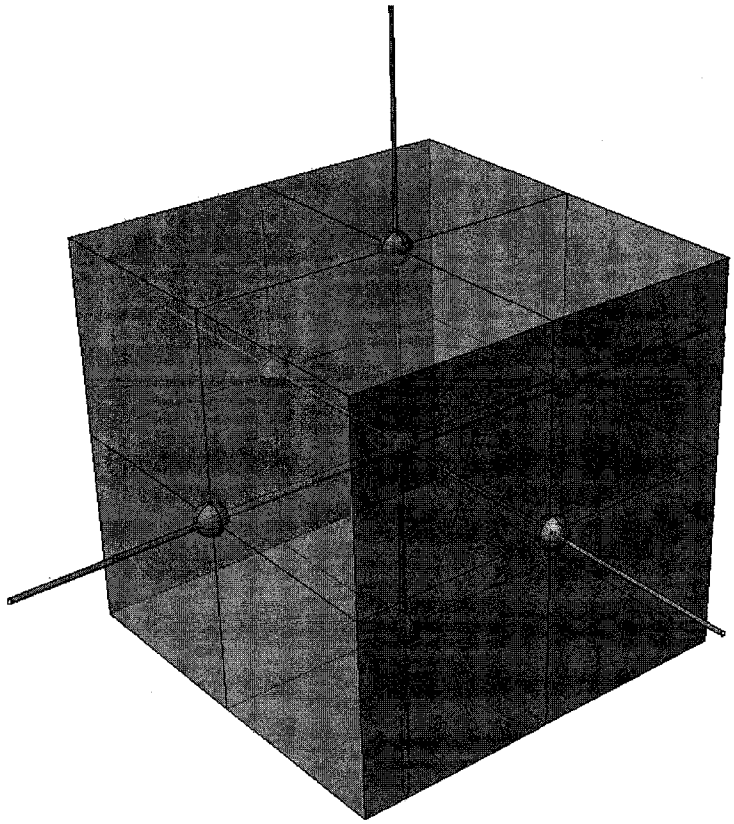


Fig. 42. Fixed anchor points using the bounding cube of the 3D model.

The points selected are the center point and the intersection of axes starting from the center point with each surface of the 6 surfaces of the cube. The drawback for this approach is that it rotation variant which means it depends in the orientation of the 3D model. To overcome this effect, 3D models should be aligned to be normalized for

rotation before selecting the anchor points. We used principal axes to normalize models for rotation[1]. The algorithm to get the anchor points is shown in Fig. 43.

3.2.1.5 Discussion

Based on the experiments we conducted, we decided not to use anchor points based on the curvature of the 3D model as minor variations in the model significantly changes the signature making it unusable for comparing shapes of 3D models. Using points on the medial axes has proved to be computationally expensive. Moreover, when using either highest curvature points or points on the medial axes, we get number of anchor points that vary depending on the complexity of the 3D model. Thus, resulting in complicating the process of comparing the shape of the 3D models using signature because in this case, we have to find what are the corresponding points to compare between the 3D models and decide what to do if we get different number of anchor points in the 2 models?

Using fixed points makes the compression ratio always predictable. However, the selection of the anchor points does not vary with the complexity of the model which might sometimes produce more points than what is needed as in the case of a sphere where one point is enough to reconstruct the whole sphere. Some times these points might not be enough to reconstruct the whole model which might result in inaccurate reconstruction of the 3D model. Moreover using fixed points is rotation variant which requires an additional process to normalize the orientation of the 3D model. Table 2 summarizes the different used in selecting the anchor points.

3.3 Progressive Compression Based Approach

3.3.1 Inverse Signature

The point A at which the signature image is calculated is called the anchor point for that signature. Let us define the normal at point A to be $\vec{N}$. A point P on the signature image as shown in Fig. 42, has distance d from A and angle α with respect to the normal $\vec{N}$ as shown in Fig. 44.

---

[1] In our experiments we used models from Princeton Shape Benchmark which had the center and principal axis pre-computed.

- *Get the center C (x_c,y_c,z_c) of the 3D model as the average (x,y,z) coordinates for all points on the surface of all polygons.*

$$C = \frac{\sum_{i=1}^{N} P_i}{N} \quad (5)$$

Where

P_i is the (x_i,y_i,z_i) coordinates of a point on the surface

N is the number of points on all polygons of the 3D model.

- *Translate all points of the 3D model to the center C.*
- *Get the covariance matrix CM*

$$CM = \begin{pmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{pmatrix} \quad (5)$$

Where

$$\text{cov}(A,B) = \frac{\sum_{i=1}^{N} (A_i - A_c)(B_i - B_{c)})}{N} \quad (5)$$

Where

N is the number of points on all polygons of the 3D model

A_c and B_c are the average values of all values of A_i and B_i respectively (center coordinates).

- *Get the eigenvalues and eigenvectors of the covariance matrix CM.*
- *Sort the eigenvectors by the values of eignenvalues in descending order.*
- *The sorted eigenvectors represent the principal axis of the 3D model.*
- *Rotate the 3D model to align it with the principal axis.*

Fig. 43. Model aligning algorithm.

TABLE 2

COMPARISON OF ANCHOR POINT SELECTION TECHNIQUES

Anchor Point Selection	Rotation Invariance	Efficiency	Robustness
Highest Curvature Points	Yes	Medium	Low
Medial Axes Points	Yes	Slow	Medium
Fixed Points (on the bounding cube)	No	Fast	High

In order to get the reverse mapping of the point P and get the corresponding point in the 3D domain, we must locate all points that have distance d from the anchor A and angle α with respect to the normal $\vec{N}$. All points that have distance d from the anchor A lie on the surface of a sphere that has center A and has radius d. On the other hand all points that has angle α lie on the surface of a cone that has its apex at the point A, has normal $\vec{N}$ at the apex and has angle α with the normal as shown in Fig. 45.

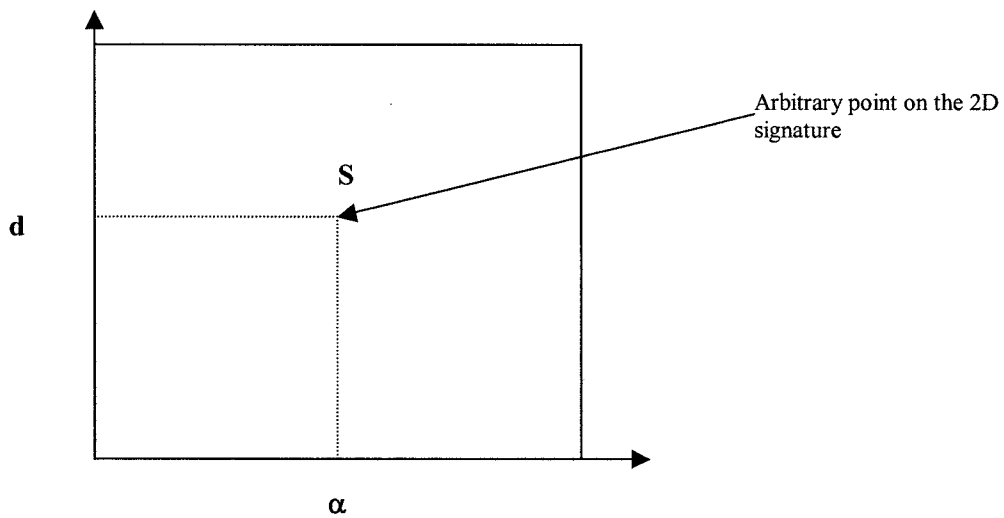


Fig. 44. 2D Signature.

The intersection of the cone and the sphere is a circle that has radius d sin (α), and perpendicular to the normal $\vec{N}$ and its center has distance d cos (α) from the anchor A as shown in Fig. 44.

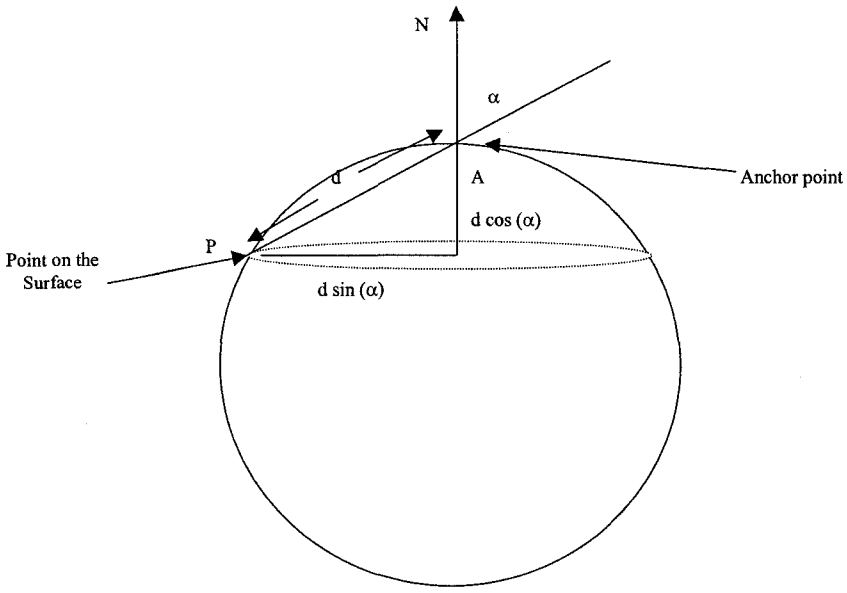


Fig. 45. Signature calculation from a point P.

This means that each point on this circle represents the inverse mapping of a single point in the 2D signature image. In other words, in the 3D domain any point that lies on this circle will map its signature in the 2D domain to a single point P. This means that the mapping from the 3D domain to the 2D signature image is many-to-one.

3.3.1.1 Derivation of the Inverse Signature

We will derive the equations of the inverse signature in the 3D- domain given:

(1) a point S = (d,α) on the 2D-surface signature as shown in Fig. 42,

(2) the coordinates of the anchor point A in the 3D domain A = (a1,a2,a3) and

(3) the normal at the anchor point $\vec{N} = \begin{bmatrix} n1 \\ n2 \\ n3 \end{bmatrix}$.

We will transform the 3D-coordinate system such that the anchor point A will be the origin and the normal $\vec{N}$ will be in the Z-axis direction.

This will result in two transformations: a translation by (a1,a2,a3), and rotation by angle θ around the vector

$$\vec{V} = \vec{N} \times \vec{Z} = \begin{bmatrix} n1 \\ n2 \\ n3 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -n2 \\ n1 \\ 0 \end{bmatrix} \tag{6}$$

where

$$\theta = \cos^{-1}(\vec{N}.\vec{Z}) = \cos^{-1}(n3)$$

When applying these two transformations on a point P = (x,y,z) , we get a point P':

$$P' = (\vec{V}.P)\vec{V} + (P - \vec{V}.P\vec{V})\cos(\theta) + \vec{V} \times p \sin(\theta) + A$$

$$P' = \begin{bmatrix} \cos(\theta) + n_2{}^2(1-\cos(\theta))x - n_1 n_2(1-\cos(\theta))y + n1\sin(\theta)z + a1 \\ -n_1 n_2(1-\cos(\theta))x + \cos(\theta) + n_1{}^2(1-\cos(\theta))y + n_2 \sin(\theta)z + a2 \\ -n_1 \sin(\theta)x - n_2 \sin(\theta)y + \cos(\theta)z + a3 \end{bmatrix} \tag{7}$$

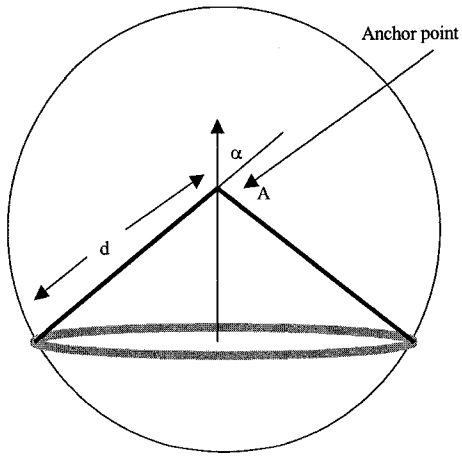


Fig. 46. Inverse signature calculation.

As we illustrated before, the inverse signature of a point S = (d,α) is a circle in the 3D-domain that results from the intersection of a cone, whose apex is at the anchor point (currently translated to the origin) and its normal is $\vec{N}$ (currently rotated to the Z-axis), with a sphere of radius d.

The parametric equations of this circle is

$x = d \sin(\alpha) \cos(2 \pi t)$

$y = d \sin(\alpha) \sin(2 \pi t)$

$z = d \cos(\alpha)$

where $t \in [0,1]$.

To get the correct coordinates in the 3D- domain, we have to apply the translation and rotation for each point in this circle as in equation (7).

3.3.2 Compression and Reconstruction Algorithms

This section illustrates how to compress the 3D model using surface signatures and restore it back from the compressed format. The compression algorithm is shown in Fig. 47.

- *Select anchor points*
- *Compute the surface signature at each anchor point*
- *Store the data of the each anchor point (including its position and normal) along with the 2D-signature image at this point*
- *Compress each 2D-signature image (using any traditional compression technique such as Huffman encoding, JPEG or GIF). Since the 2D-surface signatures are usually sparse, this is expected to have a high compression ratio.*
- *Each anchor point along with the compressed 2D-surface signature image forms a component of the new representation of the 3D object.*

Fig. 47. Signature compression algorithm.

To reconstruct the 3D model, we map the 3D space into a three dimensional grid of Voxels (Volume Elements) as shown in Fig. 48.

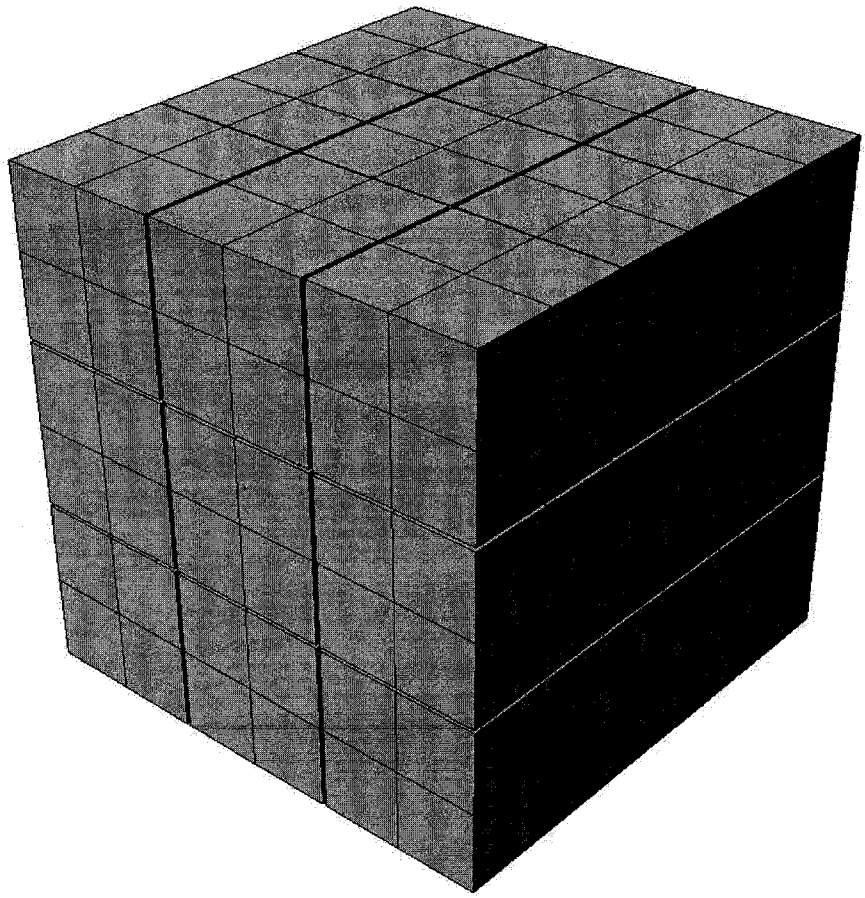


Fig. 48. 3D grid.

The reconstruction algorithm is shown in Fig. 49. For memory optimization, a single grid can be used and the calculation for each component can be computed directly for this grid (along with the intersection operation for all components except the first one).

- *FOR each signature image, do the following:*
    - *Create a (m*l*n) grid of Voxels (Volume Elements). Where m, l, and n are determined according to the available memory storage and to the required granularity in the directions of X, Y and Z respectively.*
    - *Define a mapping function from the grid coordinates to 3D coordinates that maps each Voxel cube to an (x,y,z) location in the 3D space.*
    - *Decompress the 2D-surface signature image*
    - *FOR each non-zero pixel in the 2D-surface signature image*
        - *compute the equation of the inverse signature circle*
        - *put the circle as a set of Voxels in the 3D grid*
    - *END FOR*
- *END FOR*
- *Find the intersection of all 3D grids. This will be the decompressed vertices of the 3D object.*

Fig. 49. Reconstruction algorithm.

In the previous algorithm, a circle in the 3D space is generated for each point in the 2D signature image which is computationally expensive. Moreover, the number of circles generated will depend on the number of pixels in the signature image. To speed up the computations, we devised an inverse grid approach. This approach is shown in Fig. 50.

In this approach, instead of starting by a pixel from the 2D-surface signature image and finding the corresponding circle for it in the 3D grid, we start from the 3D grid and test if a corresponding pixel exist in the signature image in the 2D domain. This will reduce the computation by eliminating the need for computing the 3D circle corresponding to a signature pixel.

- *FOR each signature image, do the following:*
  - *Create a (m*l*n) grid of Voxels (Volume Elements). Where m, l, and n are determined according to the available memory storage and to the required granularity in the directions of X, Y and Z respectively.*
  - *FOR each Voxel in the 3D grid*
    - *Compute (x,y) coordinates of the corresponding point in the 2D signature image*
    - *IF a pixel exists at position (x,y) in the signature image*
      - *Mark the current Voxel as part of the reconstructed 3D mode*
    - *END IF*
  - *END FOR*
- *END FOR*
- *Find the intersection of all 3D grids. This will be the decompressed vertices of the 3D object*

Fig. 50. Accelerated reconstruction algorithm.

CHAPTER IV
SEARCH AND DISCOVERY SERVICES

Due to the different requirements of users of 3D digital libraries, we are developing multiple types of search and discovery services. These services provide the means to locate 3D models with special requirements or based on some features of the model, as well as providing mechanisms to browse large collections of 3D models.

4.1 Metadata Search

Each 3D model has a group of metadata associated with it such as title, description, creation date, size, number of vertices, etc. (A detailed description of the metadata used will be presented in section 4.1.1).

The metadata search service is a service that allows a user to search combinations of metadata. It can also allow of the use of operators such as AND/OR to enhance the search if required. Of most importance to these queries, are queries on the text description of the model. These queries can provide accurate results if each model is accurately described, which - unfortunately - is not always done. Details on the implementation of this service are presented in section 4.1.2.

4.1.1 Metadata Repository

The repository stores the metadata for the 3D models. We used metadata that are derived from Dublin Core (DC) representation [27]. Table 3 shows a list of Dublin Core metadata elements [27] that we used in our database together with our definition.

We added metadata specific to the 3D models to the previously listed Dublin Core metadata elements, such as: the height, length, width, number of vertices, number of triangles and orientation directions. A detailed list is shown in Table 4.

These metadata fields are stored in an indexed Oracle database that provides fast search capabilities through the metadata sets.

4.1.2 Metadata Search and Retrieval

The metadata search service that we developed and implemented is derived from an implementation of a search service developed for JTASC [59] for searching text

documents and from the Arc search service [Liu2001] developed at Old Dominion University.

TABLE 3

DUBLIN CORE METADATA ELEMENTS USED IN OUR DATABASE

Data Element	Definition
Contributor	An entity responsible for making contributions to the content of the resource.
Coverage	The extent or scope of the content of the resource.
Creator	An entity primarily responsible for making the content of the resource
Date	A date associated with an event in the life cycle of the resource.
Description	An account of the content of the resource.
Format	The physical or digital manifestation of the resource.
Identifier	An unambiguous reference to the resource within a given context.
Language	A language of the intellectual content of the resource.
Publisher	An entity responsible for making the resource available.
Relation	A reference to a related resource.
Rights	Information about rights held in and over the resource.
Subject	The topic of the content of the resource.
Title	A name given to the resource.
Type	The nature or genre of the content of the resource.

We used an Oracle-based implementation using Servlet technology ([57],[59]). The Oracle database stores the index information (metadata) to speed searching for 3D models using different metadata attributes such as the author name, the title and the

abstract. In the prototype we implemented, all Dublin Core metadata are indexed to speed up searching by any of these attributes.

TABLE 4

3D MODEL METADATA

Data Element	Definition
VERTEX_NUM	Number of vertices in the 3D model
TRIANGLE_NUM	Number of triangles in the 3D model
Length, Width, Height	Dimensions of the 3D model
MinX, MaxX, MinY, MaxY, MinZ, MaxZ	Coordinates of the surrounding cube of the 3D model.
Center	Center of mass of the 3D model.
Scale	The average distance from all points on the surfaces of all polygons to the center of mass.
Principal axes	The major axes of the 3D model. They can be used to align the orientation of 3D models
Thumbnail	An image of the model rendered with colors and textures.

Fig. 51 shows the architecture of the search server. The search server is implemented in Java using Servlets. The search server has a session manager that maintains one session per user per query. It is responsible for creating new sessions for new queries (or for queries whose session has expired). Sessions had to be used because queries may return large number of results (hits) that cannot be displayed on one page. Thus sessions are used to cache some or all of the results (as will be explained later) in order to make browsing through the hits faster. The session manager receives two types of requests from the client. The session manager receives either a request to process a

new query (search), or a request to retrieve another page of results for a previously submitted query (browsing).

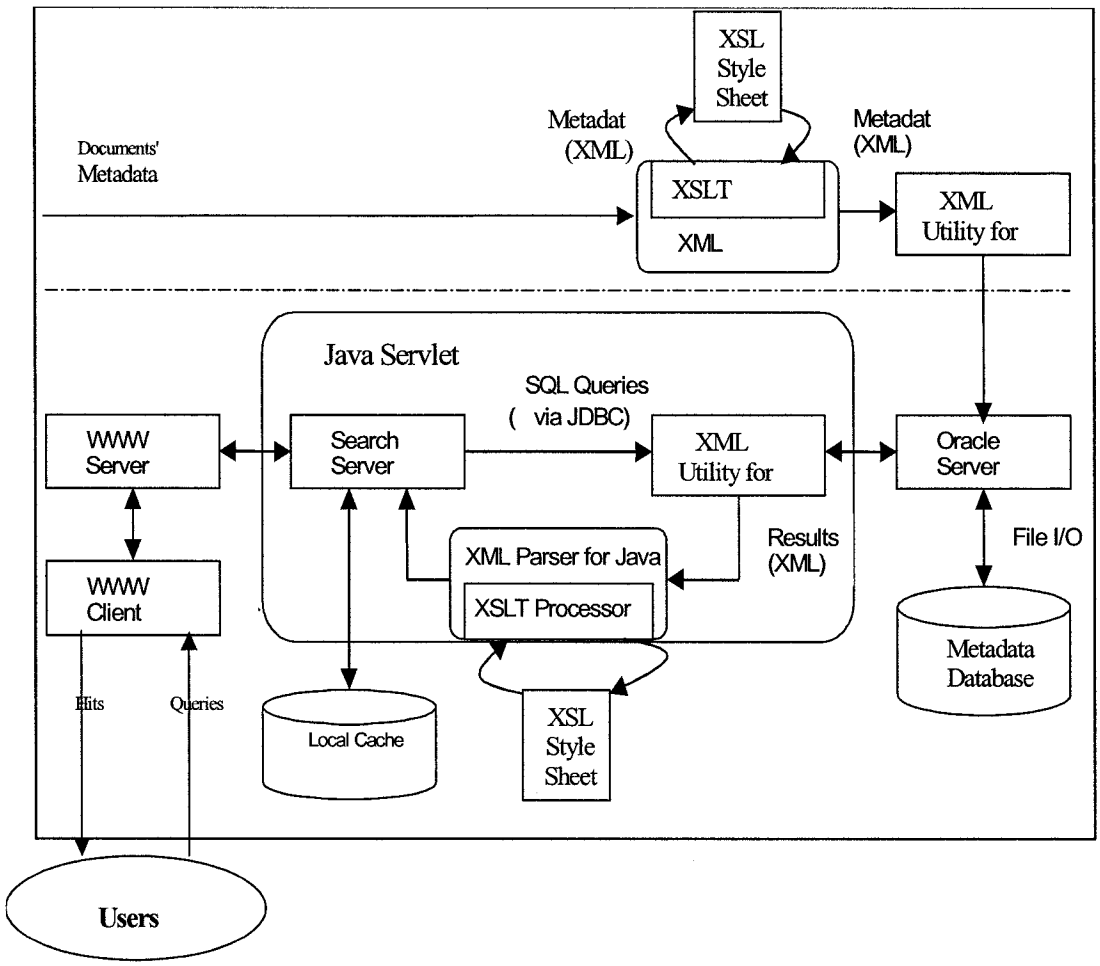


Fig. 51. Metadata search & retrieval component architecture.

For a search request, the session manager calls the index searcher that formulates a query (based on the search parameters) and submits it to the Oracle server (using JDBC) then retrieves the search results. The session manager then calls the result displayer to display the first page. For a browsing request, the session manager checks the existence of a previous session (sessions are expired after a specific time of inactivity). If a previous session is not found (expired), a new session is created and the search is re-

executed, then the required page is displayed. In the case where the previous session still exists, the required page is displayed based on the cached data (which may require additional access to the Oracle database). Different caching alternatives are used to get better response time.

4.2 Shape-Based Search of 3D Models

In the shape-based search, a 3D model is provided as input to a query and similar models are retrieved. We have multiple modes for 3D shape-based search:

A query model is selected from the digital library and similar models are retrieved. This search mode can be used in situations where users browse the digital library and find models of interest. Upon locating models of interest in the digital library, a search can be initiated to retrieve similar models.

A query model is provided as a file. In this search mode, users already have 3D models and they want to retrieve similar models. In this case, the 3D model will be uploaded to the digital library. The model will then be processed to extract features used in shape-based retrieval. These features include the anchor points and the signature images at each of the anchor points. These features are compared with those of the models in the digital library to retrieve similar models.

A query model is sketched using 3D authoring tool or a view of the model is sketched using a 2D drawing tool. This mode is helpful to users who have an idea on what the target model looks like. This mode is currently not supported in the prototype.

This requires a robust technique to recognize 3D models. The 3D recognition basically depends on extracting salient features from the models and then uses these features to identify 3D models. We used the surface signature as the feature used in computing the shape similarity of 3D models. In the following subsections, we will present details on how to compute shape similarity using the surface signature and how to search the 3D models collection based on the similarity measure that we developed.

4.2.1 Shape Similarity

In our work, we convert the process of computing shape similarity between two 3D models into the process of computing the similarity between the signature images of the 3D models. The basic assumption we used here is: "if two signatures i and j are similar

the models i and j are similar". We experimentally verified this assumption and we will provide the results of our experiments in chapter 5.

The process works by generating the signature images for the 3D models at the selected anchor points (For more details about selecting anchor points, please see section 3.2.1.). Next, we use 2D image matching techniques to compute the similarity for each 2D image pair and compute the overall average, which will be defined as the similarity between these models. We defined the similarity as the average root mean square error between all signature images of the two models [1]:

$$Sim_{i,j} = 1 - \frac{1}{k}\sum_{s=1}^{k}\sum_{m=0}^{Height}\sum_{l=0}^{Width}(P_{s,i}(m,l) - P_{s,j}(m,l))^2 \qquad (8)$$

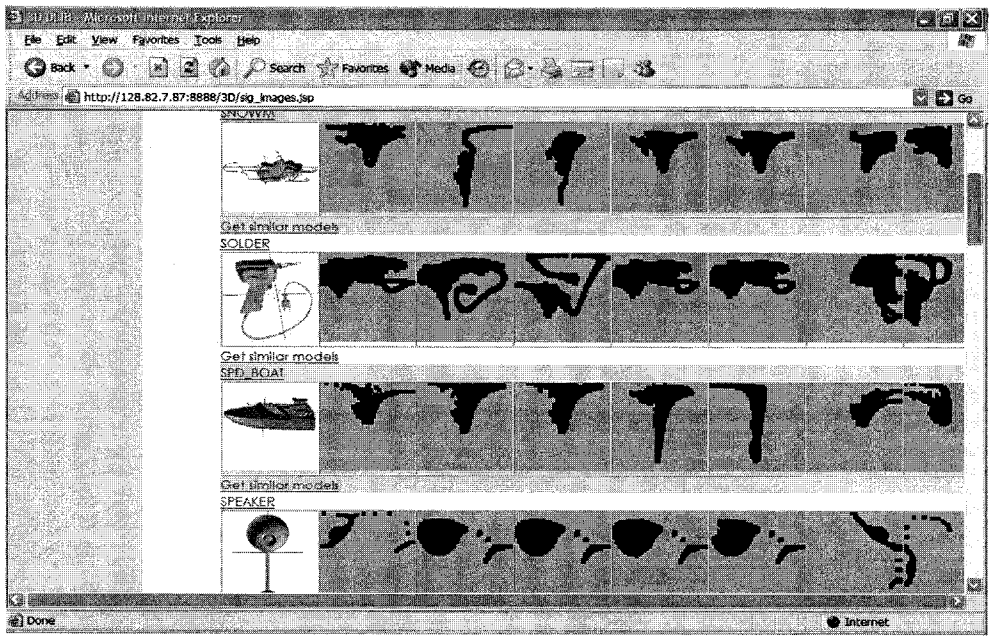


Fig. 52. Sample 3D models and their corresponding signature images.

Fig. 52 shows sample of 3D models and their corresponding signature images using 7 anchor points.

4.3 Shape-based retrieval of 3D models

In this section we will describe the technique we used for shape based retrieval where a model is provided as a query and similar models are retrieved from the digital library. The query model can be an already existing model in the digital library or an external model provided as a file[2]. In the first case, anchor points, and signature images should be already available in the digital library. In the later case, the model needs to be processed to extract vital information needed for shape-based retrieval. This processing is done as shown in Fig. 53.

- *Compute principal axes as the eigenvectors (and associated eigenvalues) of the covariance matrix obtained by integrating $x_i.x_j$ with xi ϵ {x,y,z}, over all points on the surfaces of all polygons [74]. We used these axes to normalize the models for rotations (see section 3.2.1.4).*
- *Align the model to the principal axes direction*
- *Select the anchor points as explained in section 3.2.1.*
- *Compute the signature image at each anchor point.*

Fig. 53. Model aligning and signature computation.

The steps shown in Fig. 54, can then be used in shape-based retrieval given the query model $(m)_{source}$:

The above algorithm has a time complexity of O(n) where n is the number of models in the digital library. Fig. 52 shows a snapshot of the result page where a sphere was presented as a query. Similar models of the sphere are retrieved and presented in order of similarity.

---

[2] In our prototype, we support files of STL, OFF and PLY formats.

- *FOR each 3D model (m_i) in the digital library*
  - *FOR each anchor point*
    - *Compute the similarity between each two corresponding signature images (similarity of 2D images is calculated as the percentage of matching pixels on the two images or hamming distance/total number of pixels).*
  - *END FOR*
  - *Compute the average similarity of all the signature images.*
  - *IFthe average similarity is higher than a pre-specified threshold, $(m)_{source}$ and m_i models are considered similar.*
- *END FOR*

Fig. 54. Shape retrieval algorithm.

4.4 Cluster-Based Navigation

A difficult problem even in 2D collections is to provide the searcher with mechanisms to explore the collection in a controlled manner especially when the size of the collection is large. As the size of the collection becomes larger, it comes harder for the user to look at all the models and search for models of interest. In our approach, we limit the number of models the user scans at any one time while attempting to present the user only with relevant objects. We propose an iterative process where more and more of the collection is eliminated as not being relevant to the user's search. To achieve this, we use clustering to group models based on their shape similarity.

The clustering of models is based on the shape similarity measure presented in equation (1). In the following subsections, we will introduce the optimal clustering criteria and we will follow this by the clustering algorithm used to group the models and we will explain how it can be used for *hierarchical browsing* of 3D models.

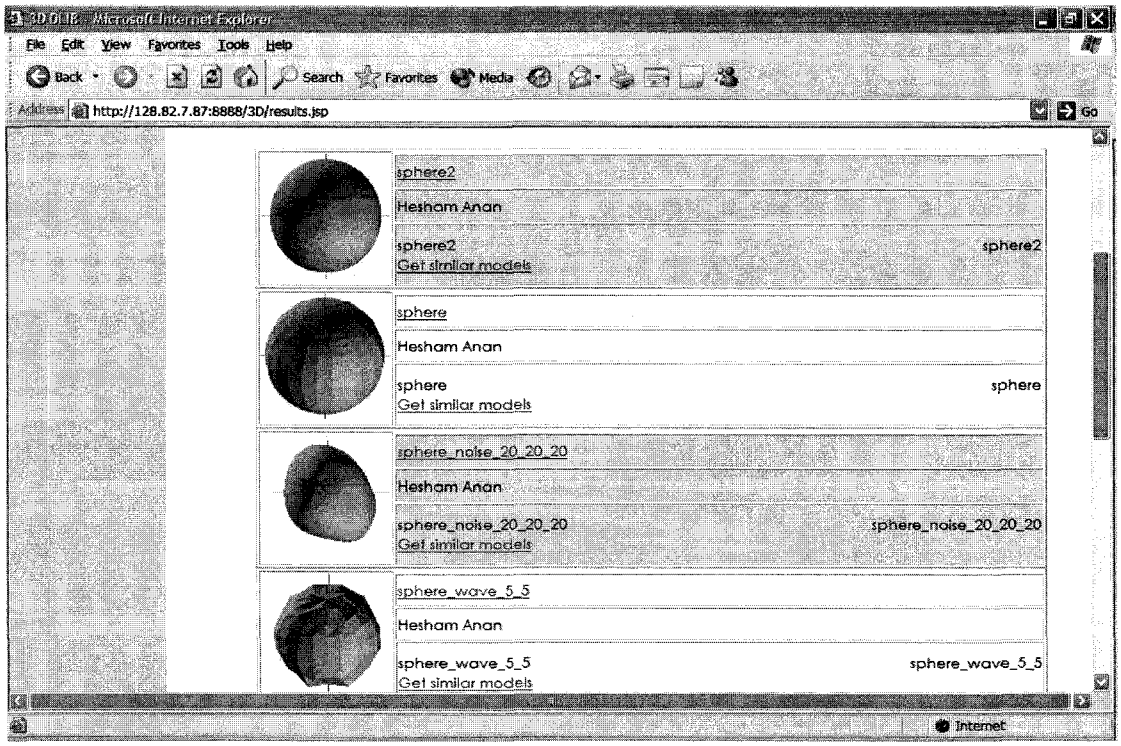


Fig. 55. Objects similar to a sphere.

4.4.1 Optimal Clustering Criteria (Clustering Distance)

The optimal clustering algorithm should select clusters such that it will maximize the similarity of members of the same cluster (internal distance) and minimize the similarity of members of different clusters (external distance).

Internal Cluster Error: Clusters should be formulated such that models in the same clusters are as close together as possible. In this case, we use the following equation to compute the error:

$$\text{Internal Cluster Error} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i^{\,2}}\sum_{j=1}^{m_i}\sum_{k=1}^{m_i}(1 - Sim_{j,k}) \qquad (9)$$

Where

N is the number of clusters,

m_i is the number of members of cluster I,

$Sim_{j,k}$ is the similarity value between model j and model k.

Equation (9) thus computes the average dissimilarity over all clusters. This works by computing the average dissimilarity between each two members of a cluster and it then computes the total average for all clusters.

External Cluster Error: Clusters should be formulated in such a way that models in different clusters are as far from each other as possible. In this case, we use the following equation to compute the error:

$$\text{ExternalClusterError} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i\sum_{p=1}^{N,p\neq i} m_p}\sum_{j=1}^{m_i}\sum_{h=1}^{N,h\neq i}\sum_{k=1}^{m_h} Sim_{j,k} \qquad (10)$$

Where

N is the number of clusters,

m_i is the number of members of cluster I,

$Sim_{j,k}$ is the similarity value between model j and model k.

As equation (3) shows, external cluster distance computes the similarity between members of a cluster and non members of the same cluster. This is averaged for all clusters. This error measure assumes that if we have two models belonging to different clusters, they should be dissimilar (otherwise they should have been in the same cluster). Thus, to compute the external error, we compute the average similarity between each members of a given cluster and each member of the rest of clusters. This is repeated for all clusters and an average is computed to indicate the overall external clustering error.

4.4.2 Hierarchical Browsing

As indicated in the introduction, to browse large collections of 3D models, we shall limit the size of the cluster at any level to a parameter (M) associated with the user capability to analyze M 2D representations (signature images) of the 3D models. For example we can set M to 30 which means, a user will not see more than 30 models (or clusters) at the same time.

There is a tradeoff between the maximum number of clusters per level and the accuracy of the cluster. When the number of clusters per level is very large, the user will be presented with the large number of models at each time which will make inconvenient

to browse and will defeat the purpose of clustering. On the other end, when the maximum number of clusters is small, this will result in higher number of members per cluster which will affect the accuracy of the clusters. In which case, we apply the clustering process to these clusters. Hence, we create hierarchy of clusters. For example, if a digital library has 3000 models and we limited M to 30, the average number of models per cluster will be 100 which is fairly large. This might result in having clusters that contain models that are not very similar. Of course this can be resolved by increasing the number of clusters. For instance, if we set M to 300, we will have an average of 10 models per cluster. But this will defeat the purpose of clustering since the user will be presented with large number of clusters to view. We resolve this by re-applying clustering process recursively to clusters that have large number of members.

When presenting the models to the user, clusters are abstracted by representative models called cluster keys. Each of the M images may represent a specific 3D model or the key model of an entire cluster. The user navigates through the hierarchy by selecting, at each level, the image(s) most similar to the desired target model.

A cluster key is defined to be the cluster member that represents the whole cluster. We select the cluster key to be the cluster member that has the max average similarity with other cluster members. Fig. 56 shows an example of hierarchical clustering. At the first level, a user is presented with cluster keys. Upon selecting a key, the user will see the actual contents of the cluster if its size is less than or equal to M. Otherwise the user will see cluster keys for members of that cluster.

Fig. 57 shows a sample of 9 models grouped into a hierarchy (remember the user sees only one level at a time). At the first level the user is presented with 2 cluster keys a fish and a plane. Upon selecting the plane, the user will be presented with two cluster key representing fighter jets and biplanes. Upon selecting the fighter jet, the user will be presented with 3 models of fighter jets.

Fig. 58 and Fig. 59 show snapshots of the digital library prototype (3DLIB). Fig. 58 shows a snapshot of the cluster keys for the first level. Fig. 59 shows a snapshot of the members of a cluster whose key is the dolphin.

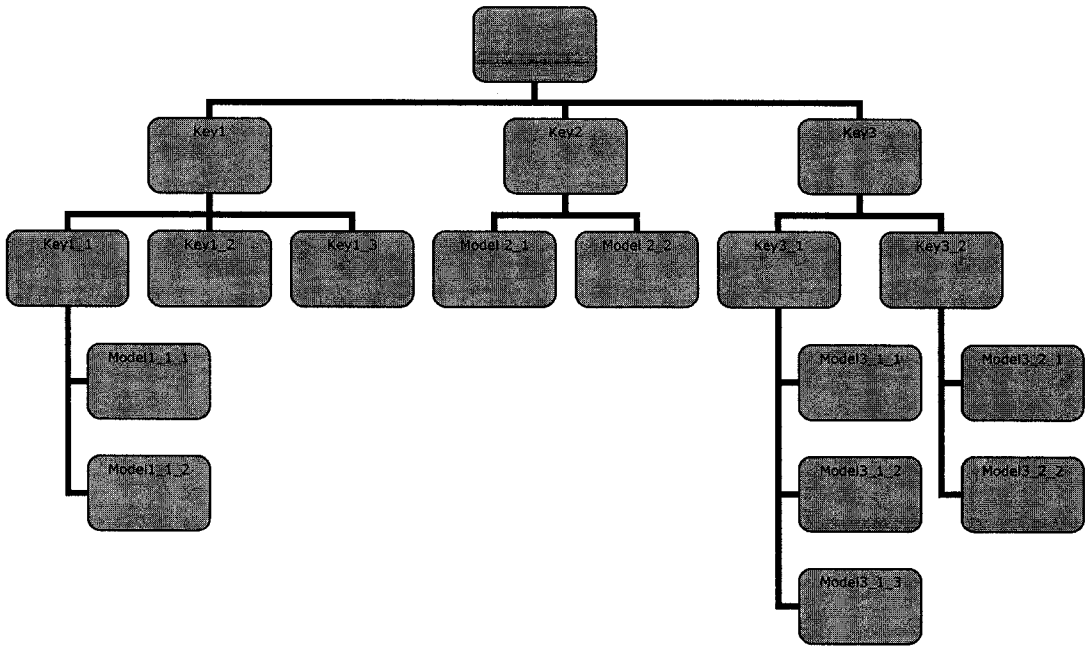


Fig. 56. Hierarchical clusters.

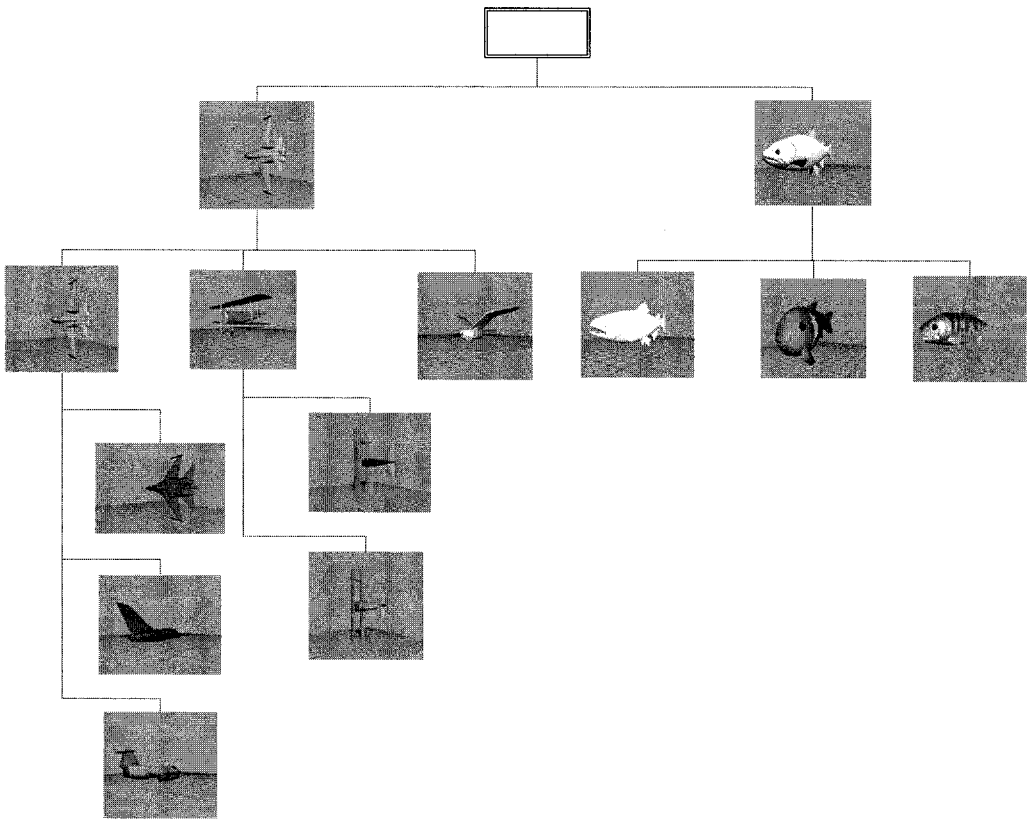


Fig. 57. Sample of hierarchical clustering.

4.4.2.1 Hierarchical Clustering Algorithm

We used k-means clustering algorithm [1], [41] to group 3D models based on the similarity measure we presented earlier. The k-means algorithm starts with a collection of models in a chosen number of clusters (n). The models are initially randomly assigned to a cluster. The k-means clustering proceeds by repeated application of a two-step process where the mean vector for all models in each cluster is computed and the models are reassigned to the cluster whose center is closest to the item. Since the initial cluster assignment is random, different runs of the k-means clustering algorithm may not give the same final clustering solution. To deal with this, the k-means clustering algorithm is repeated many times, each time starting from a different initial clustering. The sum of distances (based on our similarity metric) within the clusters is used to compare different clustering solutions.

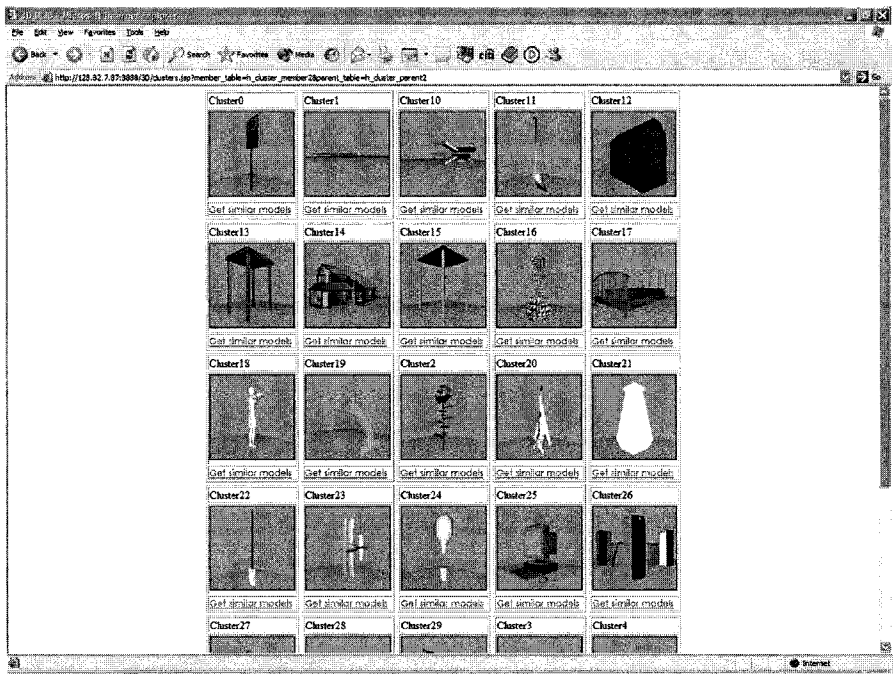


Fig. 58. Snapshot of cluster keys.

Our objective in clustering was to present the user with a view of the whole models that can show the different categories of models that are available in the database in just

on glimpse. This is why we limited the number of members of a cluster to a pre-specified number M, where M is chosen such that models can be displayed within one page.

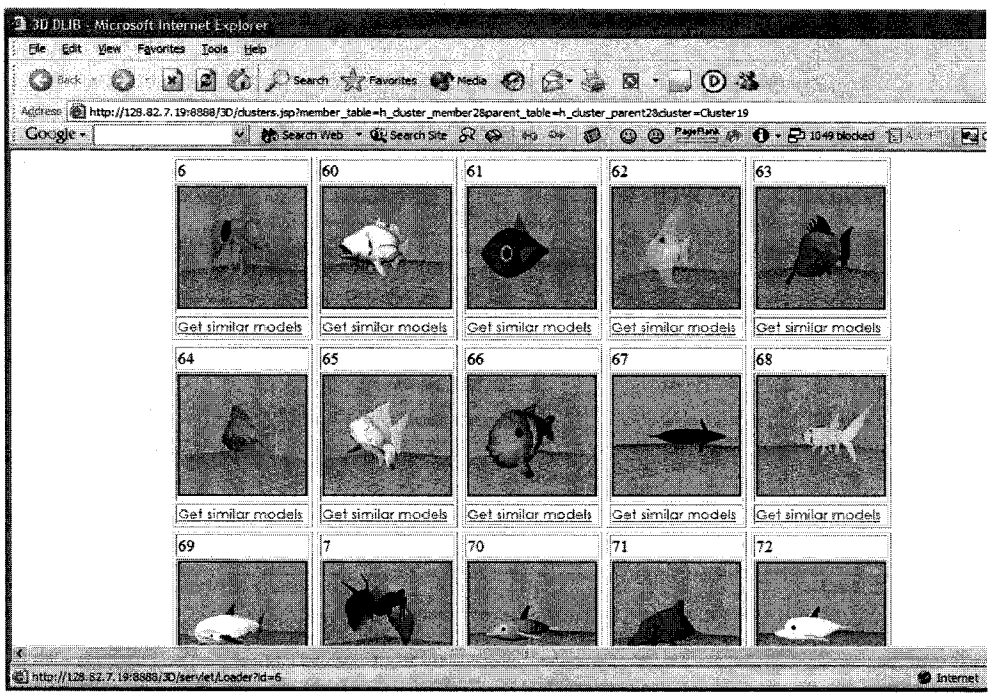


Fig. 59. Expansion of a cluster whose key is the dolphin.

- *Put all 3D models in one cluster*
- *WHILE there are clusters with cardinality > M DO*
    - *FOR All Clusters with cardinality>M*
        - *Apply k-means clustering*
        - *Select a key model that minimizes the average distance to all other models within the same cluster.*
    - *END FOR*
- *END-WHILE*

Fig. 60. K-means clustering.

Since we can not guarantee that all clusters will have a number of members that are less than M, the same clustering algorithm is re-applied again for clusters that have number of members higher than M. After applying the clustering algorithm, a model is selected out of the cluster members to be a representative (key) for that cluster. The key model is selected to be the model that has the min average distance to all other models (based on our similarity metric). Specifically, the algorithm works as shown in Fig. 60.

- *Set current level to 1*
- *IF the current cluster has cluster keys*
  - *FOR all cluster key models (CK_i) in the current level*
    - *Compute the similarity between model $(m)_{source}$ and (CK_i)*
    - *IF the similarity higher than a pre-specified threshold*
      - *Repeat the same algorithm recursively for cluster i*
    - *END IF*
  - *END FOR*
- *ELSE IF the current cluster has 3D models*
  - *FOR each 3D model (m_k) in the digital library*
    - *Compute the similarity between $(m)_{source}$ and $(m)_k$*
    - *IF the similarity is higher than a pre-specified threshold,*
      - *$(m)_{source}$ and $(m)_k$ models are considered similar.*
    - *END IF*
  - *END FOR*
- *END IF*

Fig. 61. Hierarchical clustering.

4.4.3 Hierarchical Shape-Based Retrieval

Our shape-based retrieval algorithm, which was illustrated in section 4.3, has a time complexity of O(n) where n is the number of models in the digital library. Though the time complexity is linear, this would be a considerable time when the number of models is large. Thus, we will show a more scalable version of this algorithm using the hierarchical clusters we created for browsing [1].

In this algorithm, given a model as a query, we will retrieve similar models. Instead of comparing the similarity of the query model with all models in the digital library, we will compare the similarity of the query model with the cluster keys first. If the result of comparison shows a great difference between the query model and the cluster key, we will not need to compare the similarity of the query model and all member of this clusters based on the assumption that members of the same cluster are similar. The algorithm works as shown in Fig. 61 given the query model $(m)_{source}$.

This algorithm has a time complexity of O(log(n)) where n is the number of models in the digital library. This has a significant effect on the time of processing the shape-based retrieval query especially if n is very large. That makes this algorithm more scalable than the one that searches all the models.

CHAPTER V
PERFORMANCE EVALUATION

5.1 Overview

In the previous chapters, we illustrated the architecture of 3DLIB and how this architecture helps in managing 3D models by providing services for storage, discovery and retrieval of 3D models. In this chapter we will evaluate the performance of the services provided by 3DLIB and we will compare the performance of some of the services to previously published similar services. In this chapter we will focus on quantitative evaluation of these 3DLIB services. We will not, however, illustrate subjective evaluations of our contributions such as the digital library architecture or the ease of use of the interface.

We will first start by describing the test bed we used in evaluating the performance. We will follow this by a description of performance measures and experiments we conducted to evaluate the performance of progressive compression, shape-based retrieval and hierarchical browsing.

5.2 Test Bed Description

In order to evaluate the performance of our 3DLIB services, we needed to build a collection that has a number of 3D models sufficient for valid performance data. Moreover, we needed the models to be of various sizes and shapes. In the following subsections, we will describe the test beds we used and how they were acquired.

5.2.1 Test Bed 1

We constructed a test bed that contained about 350 3D-models. Some of the models were designed using 3D studio. Others were collected from the Internet from different archives such as 3D café (http://www.3Dcafe.com). We used some of the models that were publicly available by 3D research groups such as Stanford 3D Scanning Repository (http://www-graphics.stanford.edu/data/3Dscanrep/) from Stanford University and GIT Large Geometry Models Archive (http://www.cc.gatech.edu/projects/large_models/) from Georgia Institute of Technology.

5.2.2 Test Bed 2

Though we spent some time in collecting and constructing test bed 1, we decided to use Princeton Shape Benchmark (PSB) which was released in 2004 [74] as our test bed to conduct performance comparisons. The advantage of using PSB is that it is a standardized benchmark which makes it easy to compare the performance of various services without the bias of using a privately collected test bed.

The 3D models in PSB were collected from the World Wide Web in two years. The PSB collection we used contained a database of 1,814 manually 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. For instance, the following metadata is provided to help identify the source and object type for each model:

Model URL [74]:" the web address where the model was found on the web"

Referring URL [74]: "the address of the Web page containing a link to the model".

Thumbnail image [74]: "an image of the model rendered with colors and textures".

In addition PSB provides useful data for normalizing 3D models for differences in translation, scale and orientation such as:

Center of mass [74]: "the average (x,y,z) coordinates for all points on the surfaces of all polygons".

- Scale [74]: "the average distance from all points on the surfaces of all polygons to the center of mass".
- Principal axes [74]: "the eigenvectors (and associated eigenvalues) of the covariance matrix obtained by integrating the quadratic polynomials $x_i.x_j$ with $x_i \in \{x,y,z\}$, over all points on the surfaces of all polygons. We used these axes to normalize the models for rotations".

The 3D models were manually partitioned by Shilane et. al. [74] into classes based on shape attributes of the models into 161 classes each containing at least four models. Fig. 62 shows the main categories of models in the PSB database. It shows that models are distributed among different categories such as vehicles, buildings, household items,

animals, furniture and plants. The wide variety of model categories available in this benchmark reduces the bias in results.

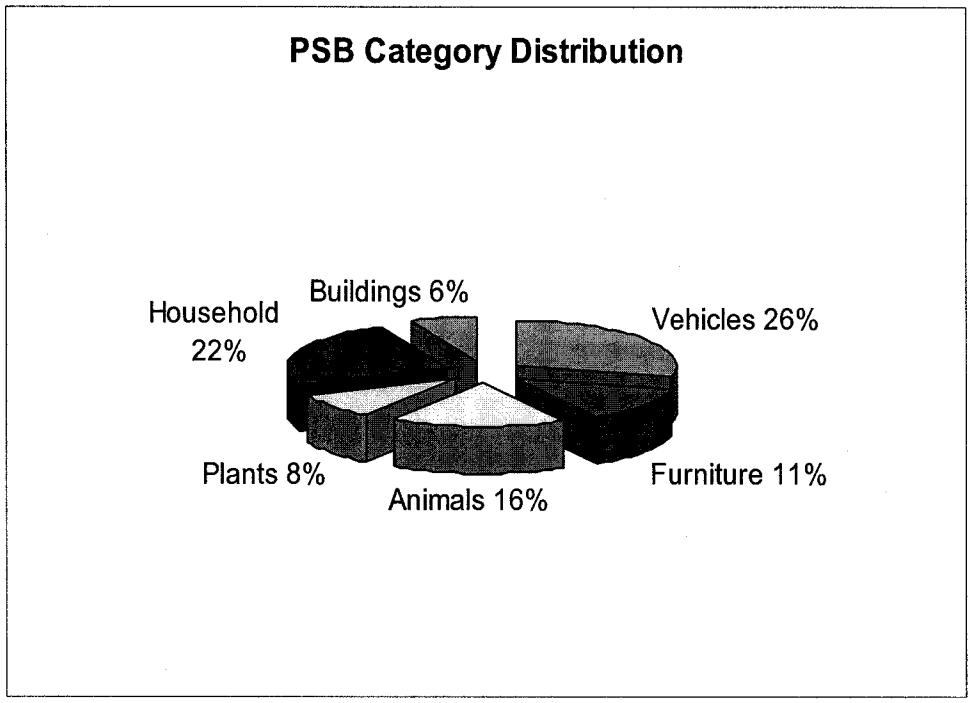


Fig. 62. Princeton shape benchmark categoy description.

5.3 Compression Technique Evaluation

5.3.1 Compression Ratio

The compression ratio is defined as the ratio of the size of the original 3D model to the size of the compressed 3D model. We use this ratio to judge the amount of storage we save by applying our compression technique.

5.3.2 Compression Ratio Experiments

We conducted experiments to compute the compression ratio for our signature image representation of the 3D model. The experiment to compute the compression ratio is as shown in Fig. 63.

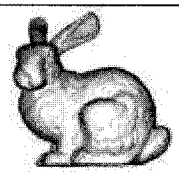
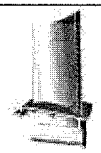
Using 7 signatures in representing the 3D model, we computed the average compression ratio which was about 18. This means that our signature representation uses, on the average, $1/18^{th}$ of the original model size.

- *FOR each 3D model in 3DLIB*
  - *Get the actual size of the 3D model*
  - *Get the size of the sum of the sizes of signatures used to represent the model*
- *END FOR*
- *Divide the sum of actual model sizes by the sum of all signature sizes to compute the compression ratio*

Fig. 63. Compression ratio computation.

TABLE 5

SAMPLE COMPRESSION RATIO RESULTS FOR SOME 3D MODELS

Model		Compression Ratio
Stanford Bunny		54
Dragon		129
Turbine Blade		158

We believe we can get even higher compression ratio if the signature images themselves are compressed using one of the existing 2D image compression techniques such as those listed in [18]. This ratio is significantly better than Deering [22] who reported a lossy compression ratio ranging from 6 to 10. However for relatively large models we could achieve higher compression ratios as shown in Table 5.

Though we could get high compression ratios, the compression technique is a lossy compression technique and we cannot guarantee that we will recreate the original 3D model. However, using the signatures we reconstruct models that are similar to the original model. Fig. 64 shows a 3D view of the Stanford bunny, which was developed by Stanford University and is widely used in testing 3D applications.

Fig. 64. 3D view of the original Stanford bunny.

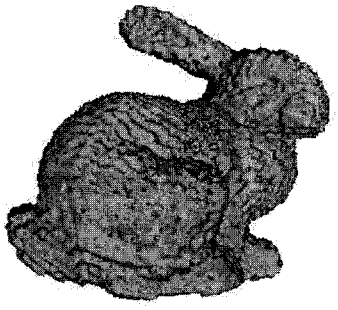


Fig. 65. Reconstructed Stanford bunny.

Fig. 65 shows the same model reconstructed using the signature images. Fig. 66 shows the reconstruction of the Stanford bunny using seven signature images. We can see that after seven signatures we reconstructed a model that looks like the original Stanford bunny but it is not 100% accurate.

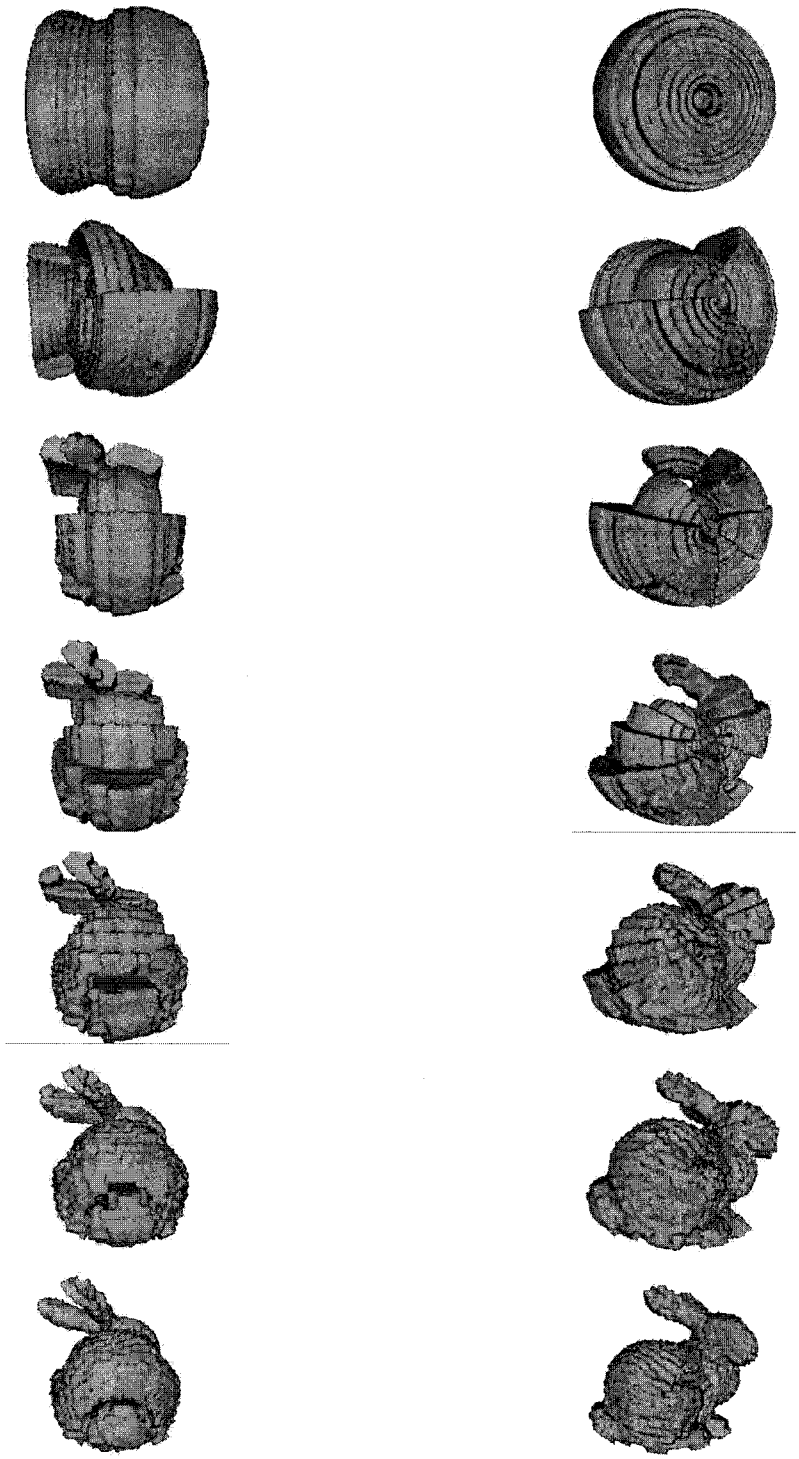


Fig. 66. Progressive reconstruction of the Stanford bunny (rear view on left and side view on right).

5.3.3 Parameters Affecting the Compression

There are some parameters that affect the quality of the reconstructed 3D model. These parameters include the signature image size and the size of the 3D grid used in reconstructing the model.

The signature image is a 2D N*N image that is used to represent the relation between the distance and the angle of points on the surface relative to the anchor point where N is represents the width and height of the signature image in pixels.

N is selected prior to the compression process. The angle ranges from 0 to π and the distance is normalized to range from 0 to 1. When the value of N becomes higher, we will have less quantization error in representing the angle and the distance but the size of the signature will increase.

On the other hand, the grid used in reconstructing the model does not affect the signature representation or the storage of the 3D model. However, it has a great effect on the quality of the reconstructed model. This grid is simply a temporary storage allocated on the computer where the model is reconstructed. When is the size of the grid is higher, more memory will be needed for reconstructing the model and the process will become slower.

We studied these effects on some models using 2.6GH computer with 1GB ram and the results are shown in Table 6. This table shows that as the grid size increases, the time needed for reconstructing increases as well. For instance, we need 14 seconds to reconstruct the Stanford bunny using 128*128*128 grid. This time is comparable to that obtained by Valette [81] which reported 14.5 seconds to reconstruct the same model. However, it is impractical for a user to wait for 14 seconds to view a 3D model over the web. This time includes the transmission time of the 3D model from the server to the client's computer.

5.4 Shape-Based Retrieval Evaluation

We will start by defining the measures we used in evaluating the shape-based retrieval. We will follow this by describing the experiments we conducted and their results.

TABLE 6

EFFECT OF SIGNATURE SIZE AND GRID SIZE ON RECONSTRUCTION

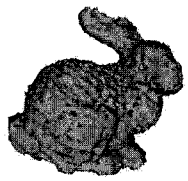
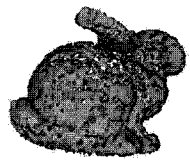
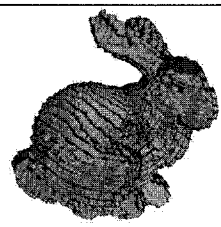
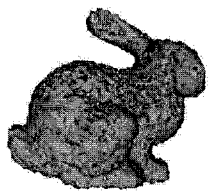
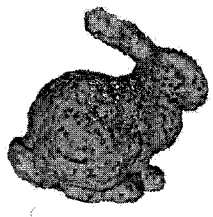
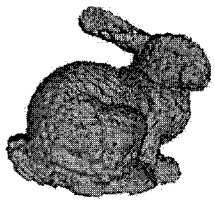
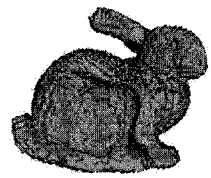
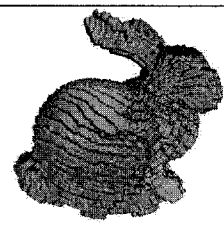
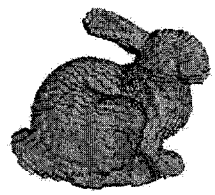
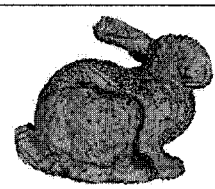
Signature Size	Grid Size	Reconstruction Time (sec)	Reconstructed Model
64*64	100*100*100	3	
128*128	100*100*100	3	
64*64	200*200*200	14	
100*100	128*128*128	4	
128*128	128*128*128	5	
100*100	200*200*200	14	

TABLE 6 (continued)

Signature Size	Grid Size	Reconstruction Time (sec)	Reconstructed Model
128*128	200*200*200	14	
64*64	250*250*25	28	
100*100	250*250*250	27	
128*128	250*250*250	27	

5.4.1 Precision and Recall

In traditional digital libraries, precision is defined as the ratio of the number of relevant documents retrieved to the total number of documents retrieved.

$$\text{Precision} = \frac{|\text{Rel} \cap \text{Ret}|}{|\text{Ret}|} \tag{11}$$

Where:

Rel : is the set of relevant documents to the query model

Ret : is the set of models retrieved by the query

Recall is defined as the ratio of relevant documents retrieved to the total number of relevant documents.

$$\text{Recall} = \frac{|\text{Rel} \cap \text{Ret}|}{|\text{Rel}|} \tag{12}$$

Where:

Rel : is the set of relevant documents to the query model

Ret : is the set of models retrieved by the query

For 3DLIB we simply replace the word document by model to obtain analogous metrics.

5.4.2 E-Measure

E-Measure is a composite measure of the precision and recall for a fixed number of retrieved results [69]. This measure takes into consideration that a user in a search engine is more interested in the first page of query results rather than in later pages. So this measure considers only the models retrieved in the first page (depending on the max size of the page) in precision and recall results. The E-Measure is defined as ([69], [74] and [47]):

$$E = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \tag{13}$$

5.4.3 Precision and Recall Experiments

The objectives of these experiments were to test the accuracy of the search service as well as its capability to retrieve the correct models in shape-based search queries. We used the precision and recall measures, which we defined earlier, to quantitatively evaluate the shape-based retrieval technique. Precision and recall are usually two contradicting measures. For example, if in any query we will return all the models existing in the database, we will get a 100% recall but very low precision. On the other hand if we are more aggressive and we limit the number of models retrieved, we will get a high precision but we may get a low recall value. We can control the values of

precision and recall by using a threshold T that specifies the cutoff value for similarity. We conducted the experiments as shown in Fig. 67.

The average precision and the average recall for different values of threshold are shown in Fig. 68.

Fig. 68 shows that the bigger the threshold becomes, the better the recall. However the precision becomes poorer. On the other hand, as the threshold becomes lower, the more the precision becomes and the less the recall. The value of the threshold can be tuned according to the user community requirements.

- *FOR a value of cutoff threshold (T) ranging from 0 to 1*
  - *FOR Each 3D model in 3DLIB*
    - *Provide a 3D model from 3DLIB as a query, and compute the similarity between this model and all the models in 3DLIB using the signature similarity measure described in earlier chapters.*
    - *Retrieve all models that have a similarity value higher than a cutoff threshold T.*
    - *Compute the actual number of relevant results using the manual classification of models*
    - *Compute precision using equation (11)*
    - *Compute recall using equation (12)*
  - *END FOR*
  - *Compute the average precision*
  - *Compute the average recall*
- *END FOR*

Fig. 67. Precision and recall experiments.

Based on the results of the previous experiments, we have drawn a curve that shows the relationship between precision and recall in Fig. 69. For perfect shape-based retrieval, the precision should be 1.00 for every recall value. This means that the closer a curve approaches a constant 1, the better the retrieval algorithm is.

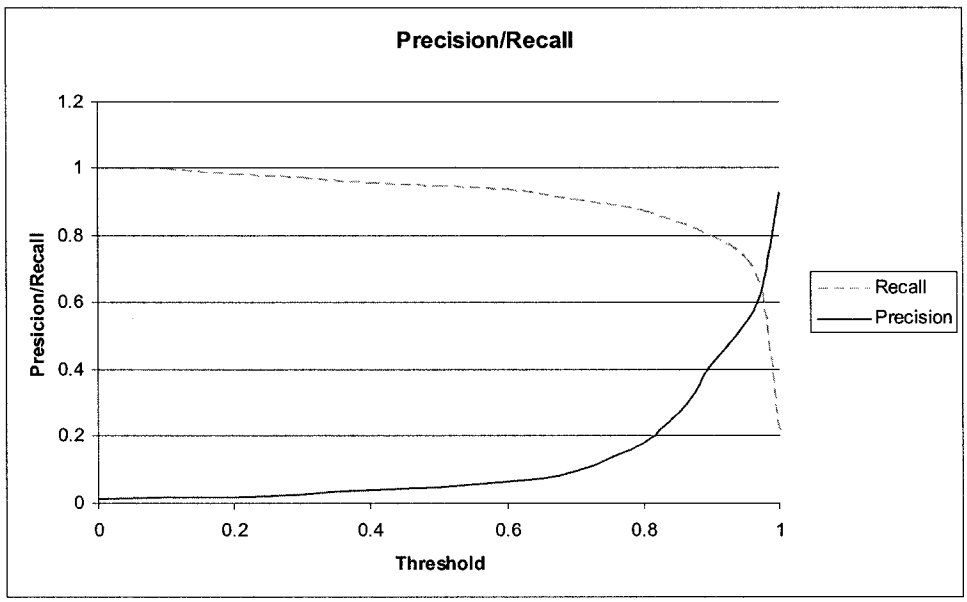


Fig. 68. Average precision & recall.

We also compared our results to the results of 12 shape matching algorithms reported in [74]. The algorithms reported in [75] use shape descriptors to compare the similarity of 3D models. The following are the descriptors used in these algorithms:

- D2 Shape Distribution (D2): "a histogram of distances between pairs of points on the surface " [63].
- Extended Gaussian Image (EGI): "a spherical function giving the distribution of surface normals" ([39] and [74]).
- Complex Extended Gaussian Image (CEGI): "a complex-valued spherical function giving the distribution of normals and associated normal distances of points on the surface "[45] and [74]).

- Shape Histogram (SHELLS): "a histogram of distances from the center of mass to points on the surface " ([6] and [74])
- Shape Histogram (SECTORS): "a spherical function giving the distribution of model area as a function of spherical angle" ([6] and [74]).
- Shape Histogram (SECSHEL): "a collection of spherical functions that give the distribution of model area as a function of radius and spherical angle" ([6] and [74]).
- 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" ([72] and [74]).
- 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" ([83] and [74]).
- 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" ([46] and [74]).
- Spherical Harmonic Descriptor (SHD): "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" ([46] and [74]).
- 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 difference, taken over all rotations and all pairings of vertices on two dodecahedra". ([15] and [74]).

The results are shown in Fig. 69. This figure shows that our signature shape based retrieval method (SSBR) for computing the similarity performance exceeds the other 12 methods. The other methods show all a typical concave characteristic and the distance between our curve to any other curve is greater than the distance between any curve of the other methods. For example our method provides about 40% recall at 80% precision which is nearly double of the value of any other curve.

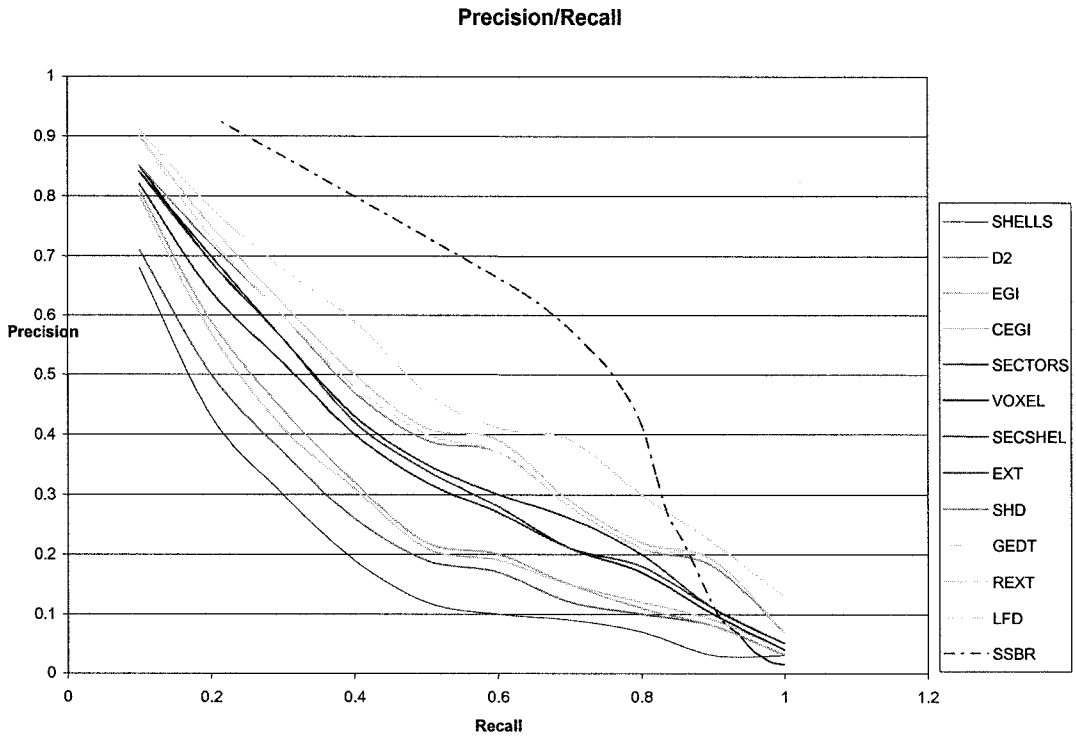


Fig. 69. Precision and recall curves.

5.4.4 E-measure Experiments

Precision and recall are good measures to quantify search queries performance. However, these measures assume that a user will retrieve all the results of a search query. Such assumption is not practically true especially if the number of results returned by a search query is large. The fact is that most users will probably look at a limited set of results.

Consider for example a search engine like Google (http://www.google.com). When using text as search query, you can get millions of hits. But the reality is that no one will look at all the results. Most of the users will probably browse the first couple of pages of search results. This is why another measure is needed to quantify the quality of search results taking into consideration the limited capability of users to browse large number of search results. We used the e-measure [69] which is a composite measure of precision

and recall considering only a limited number of search results. We conducted experiments to compute the e-measure as shown in Fig. 70.

- *FOR Each 3D model in 3DLIB*
  - *Provide the 3D model as a query to retrieved similar models*
  - *Get the results and order them by similarity.*
  - *Consider only the first (N) results and discard the rest*
  - *Compute the actual number of relevant results using the manual classification of models*
  - *Compute precision using equation (11)*
  - *Compute recall using equation (12)*
  - *Compute e-measure using equation (13)*
- *END FOR*
- *Compute the average e-measure*

Fig. 70. E-measure experiments.

We repeated the previous experiment for values of N ranging from 0 to 100 and the values of e-measure are shown in Fig. 71.

Fig. 71 shows that the value of e-measure tends to decline as the max retrieval size increases. For example for a max retrieval size of 20, the e-measure is 0.65. For a max retrieval size of 80, the e-measure is 0.38. In an ideal search, the precision will be always 1. On the other hand the recall will start from a value less than 1 and will increase till it reaches 1 when the max retrieval size is equal to the max cluster size. Which means the ideal e-measure will start from a value less than 1, then climbs till it reaches 1 then it will remain fixed. So far, there is no such similarity search that always gets the ideal results. However, the fact that our e-measure value decays when we increase the max retrieval size, indicates that the results retrieved first have higher relevance than those

retrieved later. This indicates that ordering the results by similarity measure coincides with the relevance of the similarity of the models resulting in the user getting the relevant models first. We did not find anything in the literature of reported shape-based retrieval techniques that shows how the e-measure changes for different values of maximum retrieval size. However, in [74], we found, the value of e-measure for a maximum retrieval size of 32 computed for 12 shape-based retrieval techniques.

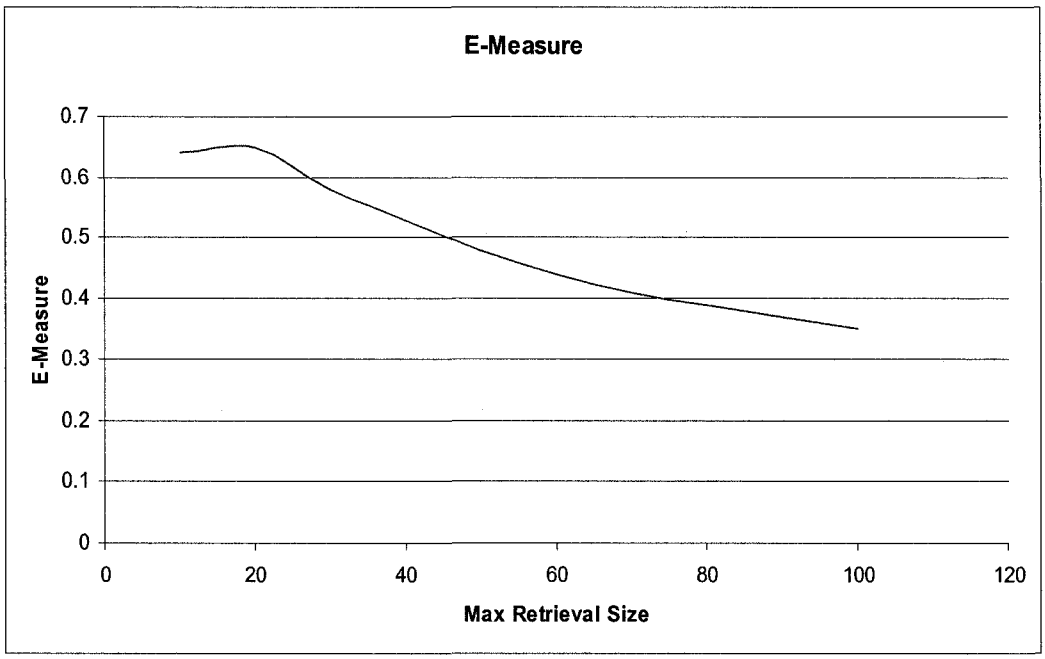


Fig. 71. E-measure vs retrieval size.

We compared the value e-measure of our similarity measure to the e-measure from other algorithms and the results are shown in Table 7. Table 7 shows that the e-measure for our SSBR method is 57% which is twice as good as the best reported technique.

5.5 Hierarchical Browsing Evaluation

In our research we implemented a hierarchical browsing and navigation services using clustering techniques. To evaluate the accuracy of the hierarchical browsing, we will evaluate the clusters we generated.

TABLE 7

E-MEASURE COMPARISON

Content-Based Retrieval Algorithm	E-Measure
LFD	28.0%
REXT	25.4%
SHD	24.1%
GEDT	23.7%
EXT	21.9%
SECSHEL	20.9%
VOXEL	20.7%
SECTORS	19.8%
CEGI	17.0%
EGI	16.5%
D2	13.9%
SHELLS	10.2%
SSBR	57.0%

The effectiveness of clustering depends mainly on the calculation of the similarity between cluster members. In previous sections we have demonstrated the effectiveness of our shape similarity computation method; in this section we will specify additional measures to evaluate the performance of our clusters Generated for hierarchical browsing.

Another factor that affects the quality of clustering is the number of clusters and the cluster size. If the number of clusters is large, the cluster size is small, we will have high quality clusters but the large number of clusters will defeat the purpose of clustering. On the other hand if the number of clusters is small, the number of members in a cluster becomes large. Hence, the small number of clusters may lead to the inclusion of not-so-similar models in the same cluster. To allow us to study this trade-off quantifiably, we

introduce two error metrics: the internal cluster error and the external cluster error. The first measures the average distance (similarity) of any two objects in a cluster and the second measures the average distance of objects in different clusters. The first we obviously try to minimize and the second we want to maximize. To evaluate the clustering, we used the internal and external error measures defined in section 4.4.1. The experiments to get the cluster error are shown in Fig. 72. The clustering errors are shown in Fig. 73.

Performing the clustering of a collection is a time consuming process and can not be done in real time to allow a user to change number of clusters on the fly. However we can produce a few clusters in advance and give the user the choice of the ones available. This will need updating as new models are published and change the clustering (we need to satisfy the size constraint). In Fig. 73, we show the relation of the internal and external cluster errors for a particular collection for varying cluster size. This kind of graph can be useful to the user in making a choice of cluster size, given that users are aware of their own capacity for handling a number of 3D objects simultaneously.

- *Start with the set of all models in the database*
- *FOR number of clusters (N) changing from 2 TO 30*
    - *Apply clustering algorithm to cluster model in a maximum of N clusters*
    - *Compute the internal clustering error*
    - *Computer the external clustering error*
    - *Compute a weighted average of internal and external clustering errors*
- *END FOR*

Fig. 72. Clustering error computation.

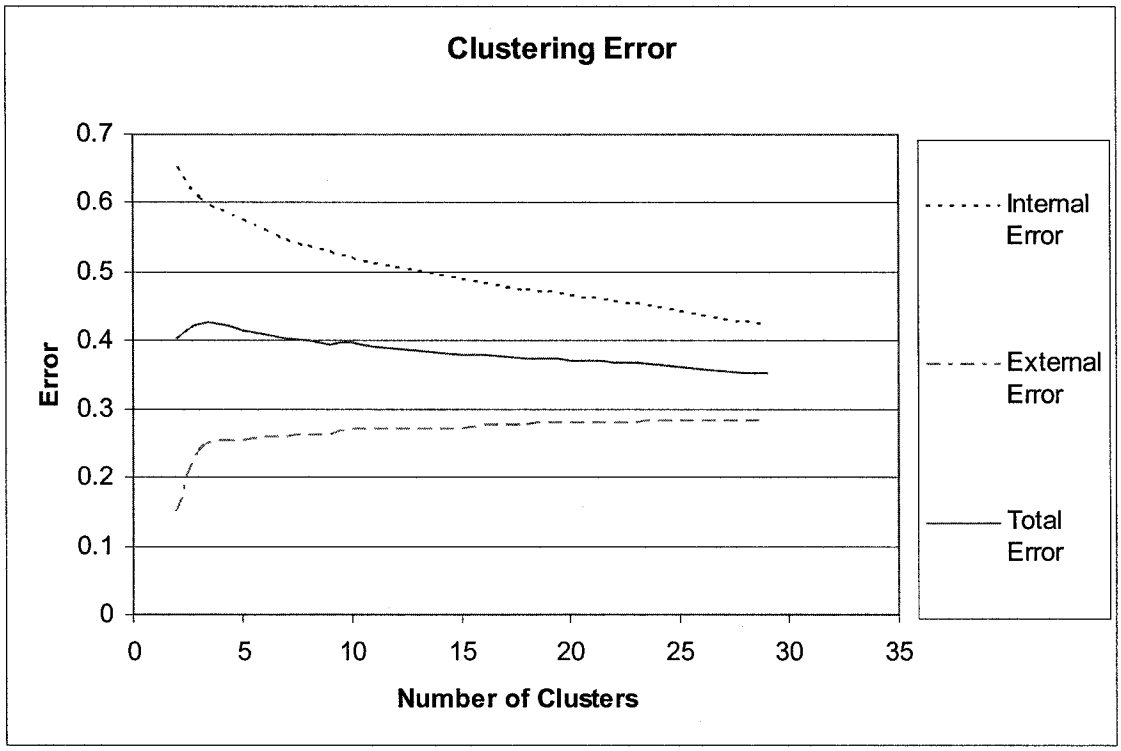


Fig. 73. Clustering errors

In Fig. 73, we also show an overall error that is the combination of the two: overall error = 0.5*Internal Cluster Error+ 0.5*External Cluster error (different weight can be used according to the significance of these errors for the application). The direction of the curve is intuitive. For example, if we have only one cluster, the external error will be zero and the internal error will be high. The curve shows that the error tends to decrease as the cluster size increases. This suggests that selecting a large cluster size will likely produce better results. However, we should take into consideration that we want to limit the number of clusters that a user can see at a time.

5.5.1 Time Analysis Experiments

We ran experiments to measure the average time to compute similarity between two models. The average time was about 0.007 seconds on a 1.8MGH PC, 512 MB of memory and running Windows XP Professional. This time is relatively small. However

the actual time of processing a shape-based retrieval query depends on the number of models in the digital library. In out example where we have 1814 models in 3DLIB, a similarity query would require comparing the signature of the query model with the signatures of all models in the digital library. Hence, it will require 12 seconds to process one similarity query.

On the other hand, based on clustering the 3D models, which was used in hierarchical browsing, we developed a scalable shape-based search in which similarity is computed between the query model and cluster keys. This reduces the total number of similarity computation to O ($\log_k$ (n)) instead of O (n) where n is the number of models in the digital library and k is the maximum number of clusters per level.

TABLE 8

SIMULATED NUMBER OF SIMILARITY COMPUTATIONS BASED ON THE NUMBER OF MODELS IN THE DIGITAL LIBRARY

Number of Models	Linear Shape-Based Retrieval	Hierarchical Shape-Based Retrieval
200	200	37
400	400	43
600	600	50
800	800	57
1000	1000	61
2000	2000	62
3000	3000	63
4000	4000	64
5000	5000	66
10000	10000	71
100000	100000	94
1000000	1000000	121

Table 8 shows the result of simulating the number of similarity computations based on the number of models in the digital library where the maximum cluster size is 30. It

shows the relationship between the number of models in the digital library and the number of signature comparison needed in shape-based retrieval queries using cluster assisted search (hierarchical shape-based retrieval) or non-cluster assisted search (linear shape-based retrieval). In cluster-assisted search, the signature of the query model is compared with the signatures of cluster keys in the first level of the hierarchical clusters instead of comparing it with signatures of all models of the digital library. This process is repeated recursively for clusters that has a key similar to the query model. For more details about this algorithm, please refer to section 4.4.3.

This clearly shows that using clusters reduces the number of comparisons needed to process a similarity query which in turn reduces the time needed to process such query.

CHAPTER VI
CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this research, we have developed an architecture for a digital library that supports 3D models. We have demonstrated its feasibility by implementing a prototype containing about 1800 3D models, which are part of the standard benchmark developed at Princeton University, and which we used to evaluate our 3D models search engine. We have reduced the storage requirements for 3D models by developing a novel compression technique with an average compression ratio of 18, in which 3D models are represented as sequences of two-dimensional images called surface signatures. We have extended the traditional metadata-based search functionality by developing a 3D shape-based search technique that uses surface signature to compute the similarity between 3D models. We have compared the performance of our 3D shape-based search with other well-known algorithms and we have demonstrated by extensive experiments that our algorithm outperforms other algorithms in precision and recall. In order to add to the usability of our 3D digital library, we have developed a hierarchical browsing mechanism to browse large collections of 3D models by recursively clustering them into groups. This hierarchical browsing technique provides the user of the digital library with a limited set of key models that are easier to view, enables the user to browse the digital library contents in a shorter time, and helps the user to locate the required 3D models faster. Accordingly, we expect that our hierarchical browsing feature will be beneficial to the users of 3D digital libraries in fields like engineering, medicine and electronic commerce. Finally, we have studied scalability issues with respect to the number of 3D models in the digital library and we have improved the shape similarity search time using the hierarchical clustering. This was achieved by comparing a query model against key models of clusters, instead of comparing a query model with all the models of the database. This means that the search time complexity will be logarithmic (O(log n)) instead of being linear (O(n)), where n is the number of models in the digital library.

6.2 Future Work

Though our search proved to outperform existing techniques in terms of precision and recall, we believe that feedback from users (relevance feedback) can be used to adaptively enhance the search precision. This can be done by monitoring responses of users to query results. This monitoring can be either explicit or implicit. Explicit monitoring works by providing users with forms to provide feedback on models they did not expect to find in the search results. Implicit feedback can work by monitoring which models the user has selected from the query result. Implicit feedback will be more convenient from the user's point of view and will be totally transparent. However, it may have less accuracy than explicit feedback. The results of feedback from users can be used later to update the similarity of the 3d models.

It will be beneficial if an artificial intelligence technique be used to implement a semantic-based search. For example, queries like "retrieve all living-room furniture models" will save the users hours of searching for individual items. In addition, 2D sketches and drawings can be used in searching for models instead of using pre-existing 3D models. This means that a user should be able to sketch a 2D image of what he is searching for, and the search engine should use this sketch to retrieve a 3D model similar to what the user has in mind. This will be a very useful research since the 2D environment remains the simplest way of sketching, especially to non professional users.

REFERENCES

[1] H. Anan, K. Maly, and M. Zubair, "Navigating and Browsing 3D Models in 3DLIB," in *Proc. Fifth National Russian Research Conference,* 2003, pp. 216-223.

[2] H. Anan, M. Nelson, K. Maly, M. Zubair, X. Liu, J. Gao, J. Tang, and Z. Yang, "Archon: building learning environments through extended digital library services," in *Proc. ICNEE*, 2003, pp. 27-35.

[3] H. Anan, X. Liu, K. Maly, M. Nelson, M. Zubair, J. C. French, E. Fox, and P. Shivakumar, "Preservation and Transition of NCSTRL Using an OAI-Based Architecture," in *Proc. JCDL*, 2002, pp. 181-182.

[4] H. Anan, K. Maly, and M. Zubair, "Digital Library Framework for Progressive Compressed 3D Models," in *Proc. of SPIE*, 2002, pp. 138-147.

[5] H. Anan and S. Yamany, "Free-Form 3D Objects Compression using Surface Signature," in *Proc. of SPIE*, 2000, pp. 202-210.

[6] M. Ankerst, G. Kastenm¨uller, H. P. Kriegel, and T. Seidl, "Nearest neighbor classification in 3D protein databases," in *Proc. ISMB*, 1999, pp.34-43.

[7] C. Bajaj, V. Pascucci, and G. Zhuang, "Single Resolution Compression of Arbitrary Triangular Meshes with Properties," in *Proc. IEEE Data Compression Conference*, 1999, pp. 247-256.

[8] R. A. Banvard, "The Visible Human Project Image Data Set From Inception to Completion and Beyond," in *Proc. CODATA 2002: Frontiers of Scientific and Technical Data*, 2002.

[9] P. Besl and R. Jain, "Three-Dimensional Object Recognition," ACM *Computing Surveys*, vol. 17, no. 1, March 1985.

[10] B. J. Besl and N. D. McKay, "A Method for Registration of 3D Shapes," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, no. 2, pp. 239-256, 1992.

[11] H. Blum and R. N. Nagel, "Shape Description Using Weighted Symmetric Axis Features," *Pattern Recognition*, vol. 10, no. 3, pp. 167-180, 1978.

[12] L. G. Brown, "A Survey of Image Registration Techniques," *ACM Computing Surveys*, Vol. 24, N. 4, December 1992.

[13] R. J. Campbell and P. J. Flynn, "A Survey of Free-Form Object Representation and Recognition Techniques," *Computer Vision and Image Understanding*, vol. 81, no. 2, pp. 166–210, 2001.

[14] S. Chang, J. Smith, H. Meng, H. Wang, and D. Zhong, "Finding Images/Video in Large Archives. Columbia's Content-Based Visual Query Project," *D-Lib Magazine*, February 1997.

[15] D. Y. Chen, M. Ouhyoung, X. P. Tian, and Y.T. Shen, "On Visual Similarity Based 3D Model Retrieval," *Computer Graphics Forum*, pp. 223–232, 2003.

[16] H Chiyokura, *Solid Modeling with Designbase: Theory and Implementation*, Addison-Wesley Publishing Company, 1988.

[17] C. S. Chua and R, Jarvis, "Point Signatures: A New Representation for 3D Object Recognition," *International Journal of Computer Vision*, vol. 25, no. 1, pp. 63-85, 1997.

[18] R. J. Clarke, "Image and Video Compression: A Survey," *International Journal of Imaging Systems & Technology*, vol.10, no.1, pp.20-32, 1999.

[19] G. Cleveland, "The Challenge of the Digital Library," *National Library News*, vol. 28, no. 5, May 1996.

[20] R. Crawfis, and N. Max, "Direct Volume Visualization of Three-Dimensional Vector Fields," in *Proc. of the 1992 Workshop on Volume Visualization*, 1992, pp. 55 – 60.

[21] C. M. Cyr and B. Kimia.,"3D Object Recognition Using Shape Similarity-Based Aspect Graph," in *Proc. ICCV01*, 2001, pp. 254-261.

[22] M. Deering, "Geometry Compression," in *Proc. ACM Computer Graphics SIGGRAPH'95*, 1995, pp. 13-20.

[23] M. DeHaemer, and M. Zyda, "Simplification of Objects Rendered by Polygonal Approximations", Computer & Graphics, vol. 15, no. 2, pp. 175-184, 1991.

[24] H. Delingette, M. Herbert, and K. Ikeuchi, "Shape Representation and Image Segmentation Using Deformable Surfaces," *Image and Vision Computing*, pp. 132-144, April 1992.

[25] H. Delingette, "Simplex Meshes: A General Representation for 3D Shape Reconstruction," Tech. Rep. 2214, Unite de recherché INRIA Sophia-Antipolis, 2004 route des Luciols, BP93, 06902 Sophia-Antipolis Cedex (France), March 1994.

[26] C. Dorai and A. K. Jain, "Shape Spectrum Based View Grouping and Matching of 3D Free-Form Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 19, no. 10, October 1997.

[27] Dublin Core Metadata Initiative DCMI, http://dublincore.org/.

[28] M. Elad, A. Tal, and S. Ar, "Content Based Retrieval of VRML Objects- an Iterative and Interactive Approach," in *Proc. Eurographics Multimedia Workshop*, 2001, pp. 97-108.

[29] M. El-Mehalawi and R. A. Miller, "A Database System of Mechanical Components Based on Geometric and Topological Similarity. Part I: Representation", *Journal of Computer-Aided Design*, vol. 35, no. 1, pp. 83-94, January 2003.

[30] C. Erikson, "Polygonal Simplification: An Overview", Technical Report, UNC-Chapel Hill, TR96-016, 1996.

[31] J. Foley, A. Dam, S. Feiner, and J. Hughes, *Computer Graphics Principles and Practice*, Addison-Wesley Publishing Company, 1990.

[32] B. Hamann, "A Data Reduction Scheme for Triangulated Surfaces," *Computer Aided Geometric Design*, vol. 11, no. 2, pp. 197-214, April 1994.

[33] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang, "Voxel Based Object Simplification," in *Proc. of Visualization '95*, 1995, pp. 296-303.

[34] M. Herbert, K. Ikeuchi, and H. Delingette, "A Spherical Representation for Recognition of Free-Form Surfaces," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 17, no. 7, pp. 681, 1995.

[35] P. Hinker, and H. Charles, "Geometric Optimization," in *Proc. of Visualization*, 1993, pp. 189-195.

[36] H. Hoppe, T. DeRose, Tduchamp, J. McDonald, and W. Stuezle, "Surface Reconstruction from Unorganized Points," in *Proc. ACM Computer Graphics SIGGRAPH '92*, 1992, pp. 71-78.

[37] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization," in *Proc. ACM Computer Graphics SIGGRAPH'93*, 1993, pp. 19-26.

[38] H. Hoppe, "Progressive Meshes," in *Proc. ACM Computer Graphics SIGGRAPH'96*, 1996, pp. 99-108.

[39] B. Horn. "Extended Gaussian Images," in *Proc. of the IEEE*, 1984, vol. 72 no. 12, pp. 1671–1686.

[40] N. Iyer, Y. Kalyanaraman, K. Lou, S. Jayanti, and K. Ramani, "Early Results With a 3D Engineering Shape Search System," in *Proc. International Symposium on Product Lifecycle Management (PLM 03)*, 2003.

[41] A. K. Jain, M. N. Murthy and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Reviews*, Nov 1999.

[42] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[43] A. Johnson and M. Herbert, "Surface Matching for Object Recognition in Complex Three-Dimensional Scenes," *Image and Vision Computing*, vol. 16, pp. 635-651, 1998.

[44] A. Kalvin and R. Taylor, "Superfaces: Polygonal Mesh Simplification With Bounded Error," Technical Report RC 19808 (#87702), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10958, 1994.

[45] S. Kang and K. Ikeuchi, "Determining 3-D Object Pose Using the Complex Extended Gaussian Image," in *Proc. Computer Vision and Pattern Recognition (CVPR'91)*,1991, pp. 580–585.

[46] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors," *Symposium on Geometry Processing*, June 2003.

[47] C. Lagoze and H. Van de Sompel, "The Open Archives Initiative: Building a Low-Barrier Interoperability Framework," in *Proc. of the First ACM/IEEE Joint Conference on Digital Libraries*, 2000, pp. 54-62.

[48] G. Leifman, S. Katz, A. Tal, and R. Meir, "Signatures of 3D Models for Retrieval," in *Proc. The 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, 2003, pp. 159–163.

[49] J. Li and C. C. Kuo, "Progressive Compression of 3D Graphic Models," in *Proc. of the 1997 International Conference on Multimedia Computing and Systems (ICMCS'97)*, 1997, pp. 135-142.

[50] X. Liu, K. Maly, M. Zubair, and M. L. Nelson, "Arc - An OAI Service Provider for Digital Library Federation", *D-Lib Magazine*, vol. 7, no. 4, April 2001.

[51] K. Lou, S. Jayanti N. Iyer, Y. Kalyanaraman, K. Ramani, and S. Prabhakar, "A Reconfigurable, Intelligent 3D Engineering Shape Search System Part II: Database Indexing, Retrieval and Clustering," in *Proc. of ASME DETC' 03 23rd Computers and Information in engineering (CIE) Conference*, September 2003.

[52] W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *Proc. ACM Computer Graphics SIGGRAPH '87*, 1987, vol. 21, no. 3, pp. 163-169.

[53] M. Lounsbery, T. DeRose, and J. Warren, "Multiresolution Analysis for Surfaces of Arbitrary Topological Type," Technical Report 93-10-05b, Department of Computer Science and Engineering, University of Washington, January 1994.

[54] D. Luebke, "Hierarchical Structures for Dynamic Polygonal Simplification," Technical Report TR 96-006, University of North Carolina at Chapel Hill, 1996.

[55] D. Luebke and C. Erikson, "View-Dependent Simplification of Arbitrary Polygonal Environments," in *Proc. of the 24th Annual Conference on Computer Graphics & Interactive Techniques*, August 1997.

[56] C. Lynch and H. Garcia-Molina, "Interoperability, Scaling, and the Digital Libraries research Agenda: A Report on the May 1995 IITA Digital Libraries Workshop," *IITA Workshop*, August 1995.

[57] K. Maly, M. Zubair, M. Nelson, X. Liu, H. Anan, J. Gao, J. Tang, and Y. Zhao "Archon - A Digital Library that Federates Physics Collections," in *Proc. DC 2002*, 2002, pp. 27-35.

[58] K. Maly, M. Zubair, and H. Anan, "An Automated Classification System and Associated Digital Library Services," in *Proc. NDDL2001*, July 2001, pp.113-127.

[59] K. Maly, M. Zubair, H. Anan, D. Tan, and Y. Zhang, "Scalable Digital Libraries based on NCSTRL/DIENST,' in *Proc. of ECDL2000*, 2000, pp. 168-180.

[60] P. Min, J. A. Halderman, M. Kazhdan, and T. A. Funkhouser, "Early Experiences with a 3D Model Search Engine", *Web3D Symposium*, 2003, pp. 7-18.

[61] B. Naylor, J. Amanatides, and W. Thibault, "Merging BSP Trees Yields Polyhedral Set Operations," in *Proc. ACM Computer Graphics SIGGRAPH '90*, 1990, vol. 24, no. 4, pp 115-124.

[62] M. Novotni and R. Klein, "3D Zernike Descriptors for Content Based Shape Retrieval," *Solid Modeling*, 2003.

[63] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Matching 3D Models With Shape Sistributions," in *Proc. Shape Modeling International*, 2001, pp. 154–166.

[64] R. Pajarola and J. Rossignac, "Compressed Progressive Meshes," *IEEE Trans. on Visualization and Computer Graphics*, vol. 6, No. 1, January 2000.

[65] S. M. Pizer, C. A. Burbeck, J. M. Coggins, D. S. Fritch, and B. S. Morse, "Objects Shape Before Boundary Shape: Scale-Space Medial Axis," *Journal of Mathematical Imaging and Vision*, vol. 4, pp. 303-313, 1994.

[66] R. L. Plante, "The NCSA Astronomy Digital Image Library: The Challenges of the Scientific Data Library," *D-Lib Magazine*, October 1997.

[67] A. R. Pope, "Model-Based Object Recognition: A Survey of Recent Research," Technical report TR-94-04, University of British Columbia, Computer Science Department, January 1994.

[68] R. Ramamoorthi and J. Arvo, "Creating Generative Models from Range Images," in *Proc. ACM Computer Graphics SIGGRAPH,99*, Los Angles, CA, 1999.

[69] C.K. van Rijsbergn, *Information Retrieval*, Butterworths, 1975

[70] J. Rossignac and A. Requicha, *Solid Modeling*, Chapter in the Encyclopedia of Electrical and Electronics Engineering, Ed. J. Webster, John Wiley & Sons. 1999.

[71] J. Rossignac and P. Borrel, "Multi-resolution 3D Approximations for Rendering Complex Scenes," Technical Report RC 17697 (#77951), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10958, 1992.

[72] D. Saupe and D. V. Vranic, "3D Model Retrieval with Spherical Harmonics and Moments," in *Proc. DAGM2001*, 2001, pp. 392–397.

[73] W. Schroeder, J. Zarge, and W. Lorensen, "Decimation of Triangle Meshes," in *Proc. ACM Computer Graphics SIGGRAPH'92*, 1992, vol. 26, no. 2, pp. 65-70.

[74] D. J. Sheehy, C.G. Armstrong, and D.J. Robinson, "Computing the Medial Surface of a Solid from a Domain Delaunay Triangulation," in *Proc. ACM Sym. Solid Modeling and Applications*, 1995, pp. 201-212.

[75] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," in *Proc. of Shape Modeling International*, 2004.

[76] T. Smith, "Report of the Multimedia Perspective Working Group," *IITA Digital Libraries Workshop*, May 1995.

[77] F. Stein and G. Medioni, "Structural Indexing: Efficient 3D Object Recognition," *IEEE Tran. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, no. 2, pp. 125-145, 1992.

[78] J. Tangelder and R. Veltkamp, "A Survey of Content Based 3D Shape Retrieval Methods," in *Proc. Shape Modeling International*, 2004, pp 145-156.

[79] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus, "Progressive Forest Split Compression," *ACM Computer Graphics SIGGRAPH '98*, July 1998.

[80] G. Turk, "Re-Tiling Polygonal Surfaces," in *Proc. ACM Computer Graphics SIGGRAPH '92*, 1992, vol. 26, no. 2, pp.55-64.

[81] S. Valette, A. Gouaillard, and R. Prost, "Compression of 3D Triangular Meshes with Progressive Precision", Computer & Graphics, Special Issue on Compression, January 2004.

[82] A. Varshney, "Hierarchical Geometric Approximations," PhD Thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1994.

[83] D. V. Vranic, "An Improvement of Rotation Invariant 3D Shape Descriptor Based On Functions on Concentric Spheres," *in IEEE International Conference on Image Processing (ICIP 2003)*, 2003, vol. 3, pp. 757–760.

[84] D. V. Vranic and D. Saupe, "3D Shape Descriptor based on 3D Fourier Transform," in *Proc. The EURASIP Conference on Digital Signal Processing for Multimedia communications and services (ECMCS 2001)*, 2001, pp. 271-274.

[85] H. Wactlar, "Informedia Digital Video Library," *D-Lib Magazine*, July/August 1996.

[86] S. Yamany and A. A. Farag, "Free-Form Surface Registration Using Surface Signatures," in *Proc. IEEE International Conference of Computer Vision*, 1999, pp. 1098-1104.

[87] D. Zhang and M. Hebert, "Harmonic maps and their applications in surface matching," in *Proc. Computer Vision and Pattern Recognition (CVPR'99)*, July 1999.

VITA

Hesham Anan

Department of Computer Science

Old Dominion University

Norfolk, VA 23529

EDUCATION

Ph. D Computer Science, December 2004, Old Dominion University
M.Sc. Computer Science, December 1997, Alexandria University, Egypt
B.Sc. Computer Science, June 1993, Alexandria University, Egypt