Navigating and Browsing 3D Models in 3DLIB

Hesham Anan, Kurt Maly, Mohammad Zubair

Computer Science Dept. Old Dominion University, Norfolk, VA, 23529
(anan, maly, zubair @cs.odu.edu)

Abstract

Traditional digital libraries have proved to work well with one-dimensional data such as reports and two-dimensional data such as images. However, handling three-dimensional (3D) data in digital libraries is relatively new and faces many challenges such as: efficient storage, fast retrieval, and a user-friendly search and discovery process. We have previously developed a prototype for a digital library framework for 3D models (3DLIB) that supports compressed storage, along with progressive retrieval of 3D models. In addition, the prototype supports search and discovery services that are tuned for 3-D models. In this paper we introduce a new service to browse large number of 3D models based on their shapes. For this we develop a clustering algorithm based on a similarity measure to group similar models. We present results of using our browsing scheme on a test bed digital library.

1 Introduction

Recently there have been rapid improvements in modelling, acquiring, and visualizing of 3D models. As a result, the volume of 3D models available in digital form is rapidly growing, which is creating a need for a digital library to manage publishing, storing, and discovering of 3D models. In (anan02) we described a digital library framework (3DLIB) that manages the storage, retrieval and discovery of 3D models. In this paper, we introduced signature transform of a 3D model and illustrated how to use this representation in compression and progressive retrieval of 3D models. The progressive retrieval enables a low-quality rendering of a 3D model quickly from the first few signatures with improving quality as more data (signatures) is received. In (anan2002) we described the details of the surface signatures and how we could use them to reconstruct 3D models using the inverse signature transform. Being able to reconstruct models from signatures provides more accurate rendering of the 3D object the more signatures are used. Even using one signature is often sufficient to identify the shape of the 3D model. Details on experiments analysing the trade-off between the size of the model, retrieval time of full object, and signature will be presented in a future paper. In this paper we present details of a browsing service that uses this similarity measure to group similar models. Our objective is to provide a mechanism for a user to effectively browse collections of 3D models. Since it is difficult for a user to view many models at once, we group models into clusters based on their shape, where each cluster is represented by a key model. When browsing, a user initially sees a view of key models. By selecting a key model, the user can browse the selected cluster and repeat the process till a model has been found (if there is one).

The rest of this paper is organized as follows; section 2 presents a brief background on signature computation and the overall architecture of the 3D Digital Library (3DLIB). Section 3 describes the browsing service, and we provide results from experimentation on a test bed collection in Section 4. Finally, we present a discussion on future work in section 5.

2 Background

In Figure 1 we summarize the architecture of 3DLIB as it was described in (anan02). It consists of the following modules:

Repository: a database that is divided into two parts; one for storing metadata and signatures of 3D models, and the other for storing features for the similarity search as explained below. Original models can be reconstructed from the signatures.

Publishing service: a service used to upload 3D models and their metadata to the repository. The publishing service accepts a 3D model from the client, determines anchor points (points at which signatures are calculated), computes the signatures and uploads the signatures along with the metadata of the 3D models to the repository.

Progressive retrieval service: a streaming service to progressively retrieve the 3D model. The compression format has the characteristic that not all data are required for an initial view of the model. This makes the compression technique more amenable to progressive retrieval, and as well as for fault-tolerance in network applications. In other words, the response time to view

Proceedings of the 5th Russian Conference on Digital Libraries RCDL2003, St.-Petersburg, Russia, 2003

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.

Search and discovery service: A simple search service is used to search the metadata for keywords, for example, a search can be initiated to retrieve an object with the word 'chair' in the model description. A more advanced search service is also available that can retrieve models that are similar (in shape) to a given model.

We implemented the repository and simple search services based on the architecture of the Java servlet-based Arc (http://arc.cs.odu.edu) (Liu2001) with an Oracle database at the backend.

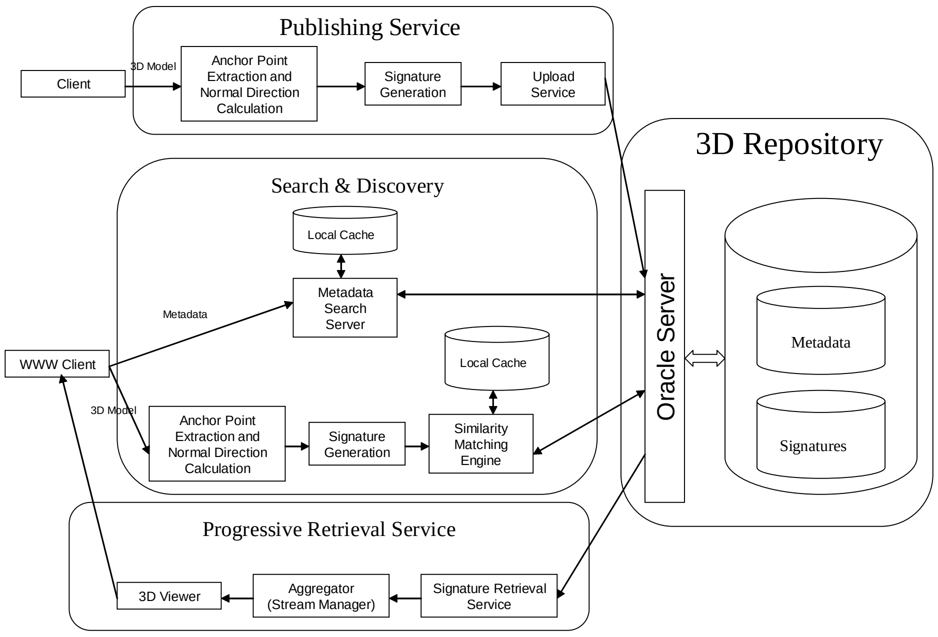


Figure 1. Digital Library Framework

The representation of the 3D models is a sequence of 2D images called surface signatures. Details about the computation of signatures and derivations of inverse signature transform (that is used in reconstructing the 3D models) can be found in anan02. For completeness we will summarize the key points in the following subsection.

2.1 *Signature Computation*

The signature image is generated as follows: For each point A, defined by its 3D coordinates and normal $\vec{N}$, each other point P_i on the surface can be related to N by two parameters as shown in :

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

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

2.2 *Inverse Signature (Signature Transform Basis)*

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 S on the signature image has distance d from A and angle α with respect to the normal $\vec{N}$. In order to get the reverse mapping of the point S 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 have angle α lie on the surface of a cone that has its apex at the point A, have a normal $\vec{N}$ at the apex and have angle α with the normal.
The intersection of the cone and the sphere is a circle that has radius d sin (α), is orthogonal to the normal $\vec{N}$, and its center has distance d cos (α) from the anchor A . 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 S. This means that the mapping from the 3D domain to the 2D signature image is many-to-one. Details for the derivation of the inverse signature equations can be found in anan2002.

2.3 Reconstruction

The decompression algorithm is as follows:

- *For each component of the 3D object representation, do the following:*
    - *Create a (m*l*n) grid of Voxels. 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 from the grid coordinates to 3D coordinates.*
    - *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*
    - *Find the intersection of all 3D grids. This will be the decompressed vertices of the 3D object*

3 Browsing and Navigation Service

A difficult problem even in 2D collections is to provide the searcher with mechanisms to explore the collection in a controlled manner. One major approach is to cluster images according to some similarity metric and present the user with the choice of providing a sample image or traversing a hierarchical set of clusters. We have extended both approaches to 3DLIB and will describe in this section the hierarchical approach. The three factors to be considered in designing a browsing and navigation service are: how many objects can the viewer absorb to make a selection (*N*); how do we define similarity (*sim*); and how do we select a representative object from a cluster (*key*). We cluster the collections using *sim* and the constraint that the average cluster size should be *N*. For each cluster select a *key* and iteratively apply the algorithm to the collections of keys until only one cluster is left. Navigation starts at the top and the user selects the most appropriate key model shown and traverses recursively down (with possible back-ups) until the desired model has been found or the search fails.

Clustering methods can be divided into two basic types (Kaufman90): partitional and hierarchical clustering. Partitional clustering attempts to directly decompose the data set into a set of disjoint clusters. The criterion function that the clustering algorithm tries to minimize may emphasize the local or the global structure of the data. The global criteria involve minimizing some measure of dissimilarity in the samples within each cluster, while maximizing the dissimilarity of different clusters. Partitional techniques include square error methods, mixture decomposition, graph theory, nearest neighbor, and fuzzy clustering. K-means algorithm is the most common partioning algorithm where the clustering strategy is based on the square root error criteria. 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 [1]. 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. Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity [1]. The end result of the algorithm is a tree of clusters called a dendrogram, which shows how the clusters are related. By cutting the dendrogram at a desired level a clustering of the data items into disjoint groups is obtained. Hierarchical algorithms can be further divided into divisive (top down) and agglomerative (bottom up). Divisive algorithms starts with a single cluster containing all objects, and then successively splits resulting clusters until only clusters of individual objects remain. Agglomerative hierarchical clustering on the other hand starts with every single object in a single cluster. Then in every successive iteration, it agglomerates (merges) the closest pair of clusters by satisfying some similarity criteria, until all of the data is in one cluster. We chose to use agglomerative clustering technique called direct join (Jain 1999) because it allows us to decide the number of clusters that are generated besides it is fast and simple.

A fundamental issue to any clustering algorithm is how to compute the similarity. For this purpose, we computed the similarity of two 3D models based on the similarity of their signatures using Equation (1).

$$Sim_{i,j} = \frac{1}{7}\sum_{s=1}^{7}\sum_{m=0}^{Height}\sum_{l=0}^{Width}(P_{s,i}(m,l) - P_{s,j}(m,l))^2$$

--------------------- (1)

Where

7 is the number of signatures used in computing similarity for each model

$P_{s,i}(m,l)$ is the value for pixel that has coordinates m,l in signature s.

This similarity measure is computed based on the signature images of the 3D models, which we have shown earlier that they can be used to reconstruct the 3D models. Based on this similarity measure, we used a direct join clustering algorithm to cluster 3D models as follows:

- *Compute the similarity between each two models.*
- *Place each model in its own cluster.*
- *While (Number of Clusters > N) where N is the desired number of clusters*
    - *Merge the two closest clusters*

Selecting the number of clusters (*N*) is a key issue in this algorithm. This number should be as small as possible because only keys of these clusters are used in the top level browsing of the model. The smaller this number gets, the user will view a small number of keys. However, we do not want it to be too small to have un-correlated models grouped together. In the next section we will show some results of applying this clustering technique with varying number of clusters.

In section 4 we describe experiments to study the effect of varying N. Note that a smaller value of N (few cluster) may lead to the inclusion of two not-so-similar models in the same cluster. To allow us to study this trade-off quantifiably, we introduce two error metrics: the intra cluster error and the inter cluster error. The first measures the average distance (similarity) of any two objects in a cluster and the second the average distance of objects in different clusters. The first we obviously try to minimize and the second we want to maximize.

Intra 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:

$$IntraClust\ erError = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i^2}\sum_{j=1}^{m_i}\sum_{k=1}^{m_i}(1 - Sim_{j,k})$$

----------------- (2)

Where

N is the number of clusters,

m_i is the number of member of cluster I,

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

Inter Cluster Error: Clusters should be formulated such 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:

$$InterClust\ erError = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i\sum_{p=1}^{N,p\neq i}m_p}\sum_{j=1}^{m_i}\sum_{h=1}^{N,h\neq i}\sum_{k=1}^{m_h}Sim_{j,k}$$

----------------- (3)

N is the number of clusters,

m_i is the number of member of cluster I,

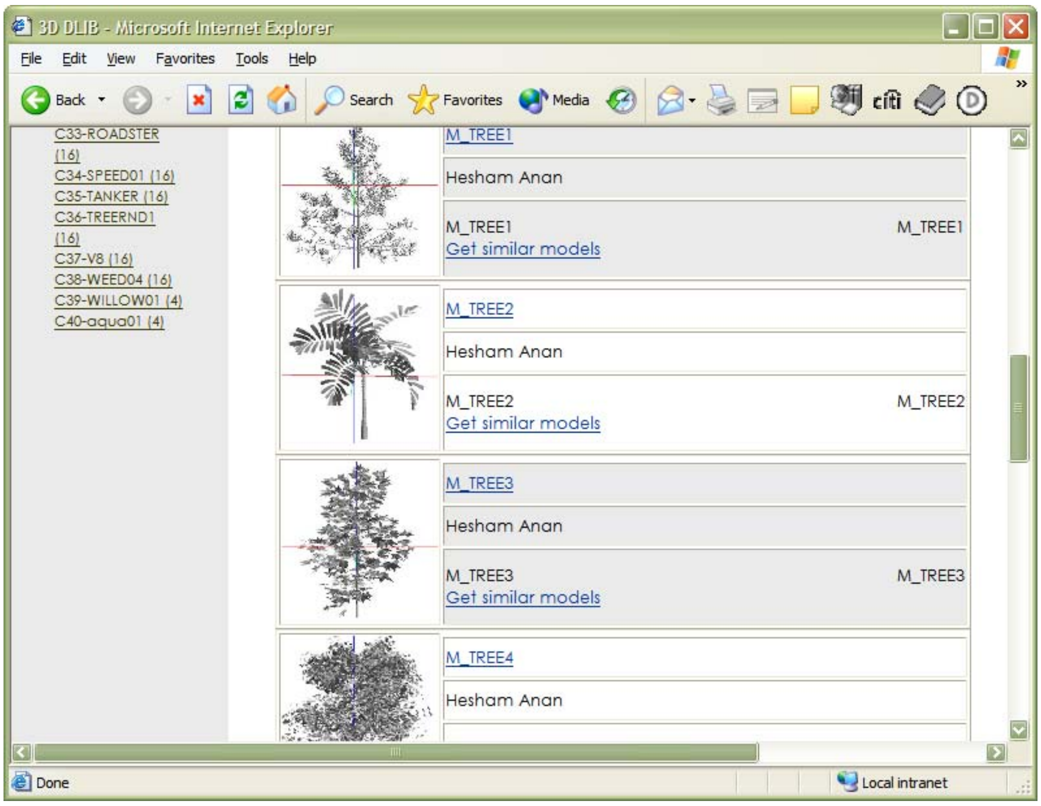
$Sim_{j,k}$ is the similarity value between model j and model k.

4 *Experimental Results*

We used 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).

Figure 2 shows examples of contents of clusters. The right panel shows a partial list of the keys of the clusters (this will be replaced later by graphical view of the keys). The clusters in Figure 2.(a) and Figure 2.(b) contain models that subjectively are indeed similar.

However, in clusters such as those in Figure 2.(c) and Figure 2.(d) some models that are different in shape or category. This is due to the fact that isolating these models into separate clusters will increase the number of clusters used and in our approach we have fixed the numbers of clusters in advance. Also some models have different granularity in their representation (some models might be using a high density number of polygons to represent then others might use a very small number which affects the accuracy of computing the similarities for the models.


3D DLIB - Microsoft Internet Explorer

C33-ROADSTER (16)
C34-SPEED01 (16)
C35-TANKER (16)
C36-TREERND1 (16)
C37-V8 (16)
C38-WEED04 (16)
C39-WILLOW01 (4)
C40-aqua01 (4)

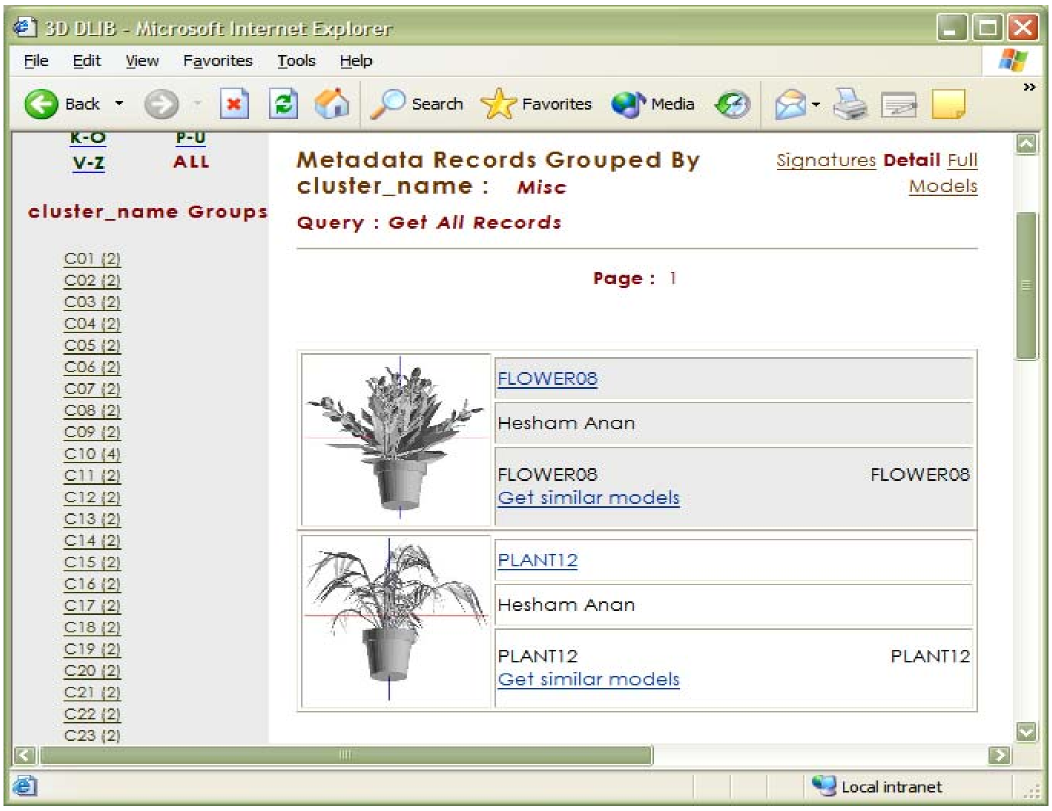
M_TREE1
Hesham Anan
M_TREE1 M_TREE1
Get similar models

M_TREE2
Hesham Anan
M_TREE2 M_TREE2
Get similar models

M_TREE3
Hesham Anan
M_TREE3 M_TREE3
Get similar models

M_TREE4
Hesham Anan

(a)


3D DLIB - Microsoft Internet Explorer

K-O P-U
V-Z ALL

cluster_name Groups

C01 (2)
C02 (2)
C03 (2)
C04 (2)
C05 (2)
C06 (2)
C07 (2)
C08 (2)
C09 (2)
C10 (4)
C11 (2)
C12 (2)
C13 (2)
C14 (2)
C15 (2)
C16 (2)
C17 (2)
C18 (2)
C19 (2)
C20 (2)
C21 (2)
C22 (2)
C23 (2)

Metadata Records Grouped By cluster_name : Misc

Query : Get All Records

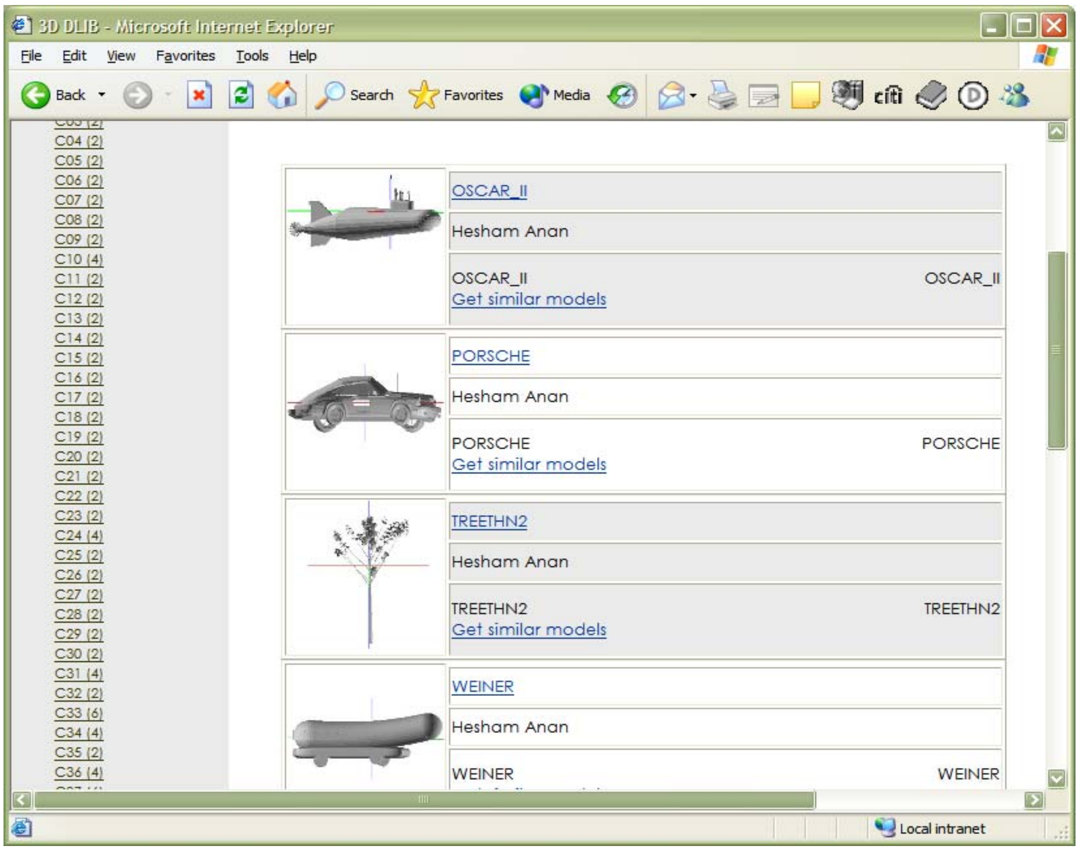
Signatures Detail Full Models

Page : 1

FLOWER08
Hesham Anan
FLOWER08 FLOWER08
Get similar models

PLANT12
Hesham Anan
PLANT12 PLANT12
Get similar models

(b)



3D DLIB - Microsoft Internet Explorer

C04 (2) C05 (2) C06 (2) C07 (2) C08 (2) C09 (2) C10 (4) C11 (2) C12 (2) C13 (2) C14 (2) C15 (2) C16 (2) C17 (2) C18 (2) C19 (2) C20 (2) C21 (2) C22 (2) C23 (2) C24 (4) C25 (2) C26 (2) C27 (2) C28 (2) C29 (2) C30 (2) C31 (4) C32 (2) C33 (6) C34 (4) C35 (2) C36 (4)

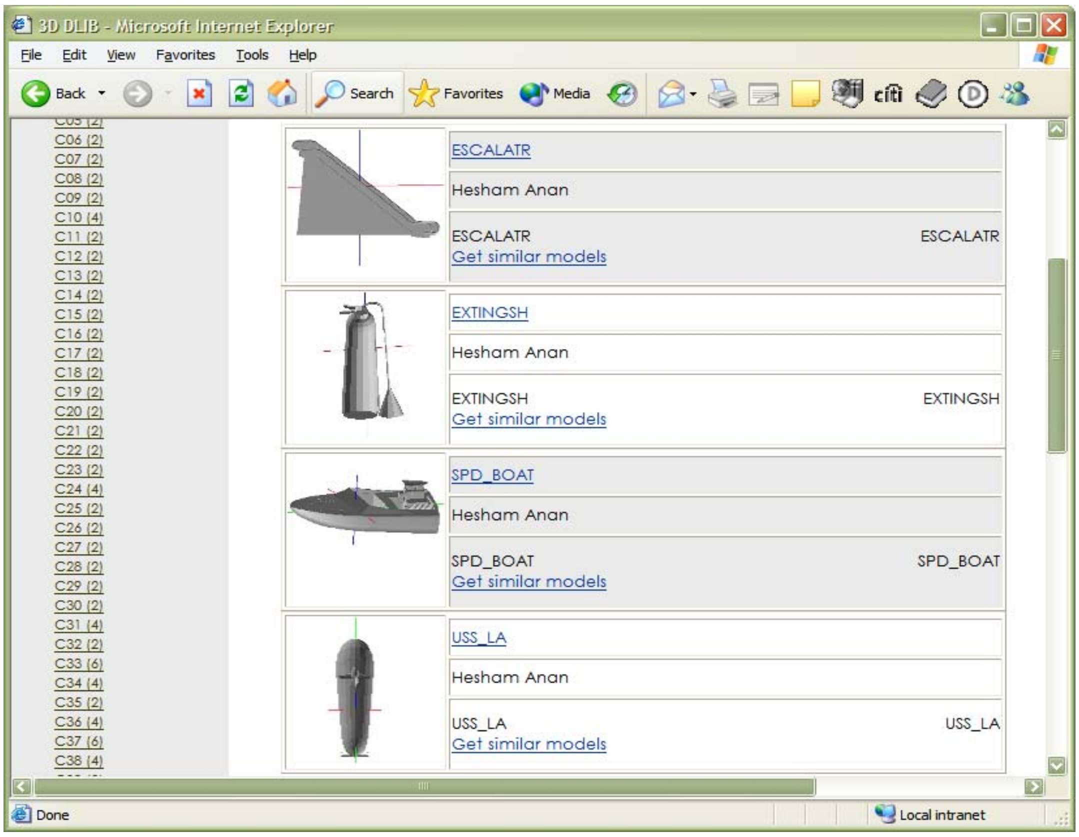
OSCAR_II
Hesham Anan
OSCAR_II OSCAR_II
Get similar models

PORSCHE
Hesham Anan
PORSCHE PORSCHE
Get similar models

TREETHN2
Hesham Anan
TREETHN2 TREETHN2
Get similar models

WEINER
Hesham Anan
WEINER WEINER

(c)



3D DLIB - Microsoft Internet Explorer

C06 (2) C07 (2) C08 (2) C09 (2) C10 (4) C11 (2) C12 (2) C13 (2) C14 (2) C15 (2) C16 (2) C17 (2) C18 (2) C19 (2) C20 (2) C21 (2) C22 (2) C23 (2) C24 (4) C25 (2) C26 (2) C27 (2) C28 (2) C29 (2) C30 (2) C31 (4) C32 (2) C33 (6) C34 (4) C35 (2) C36 (4) C37 (6) C38 (4)

ESCALATR
Hesham Anan
ESCALATR ESCALATR
Get similar models

EXTINGSH
Hesham Anan
EXTINGSH EXTINGSH
Get similar models

SPD_BOAT
Hesham Anan
SPD_BOAT SPD_BOAT
Get similar models

USS_LA
Hesham Anan
USS_LA USS_LA
Get similar models

Done

(d)

Figure 2. Examples of Clusters

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 *N* 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 figure 3 we show the relation of the inter and intra cluster errors for a particular collection for varying *N*. This kind of graph can be useful to the user in making a choice of *N*, given that the user is aware of her own capacity for handling a number of 3D objects simultaneously. Since the extra cluster error reaches a limit at around 30 clusters, we have computed a number of data points between 1 and 30 to show the detail of the curves. In figure 3, we also show an overall error that is the combination of the two: overall error = 0.5*Intra Cluster Error+ 0.5*Inter Cluster error (different weight can be used according to the significance of these errors for the application). To allow for the influence of the models in the collection (clearly the error is dependent on the specific objects in the collection), we have repeated the experiment for random sub collections of the test collections (shown in Figures 4 and 5).

The directions of the curves is intuitive, the shape is not. Our current belief is that this is directly dependent on the nature of the collection and less on the specific clustering algorithm we use. As our test bed collection grows we will be able to experimentally describe the changing nature of these graphs.

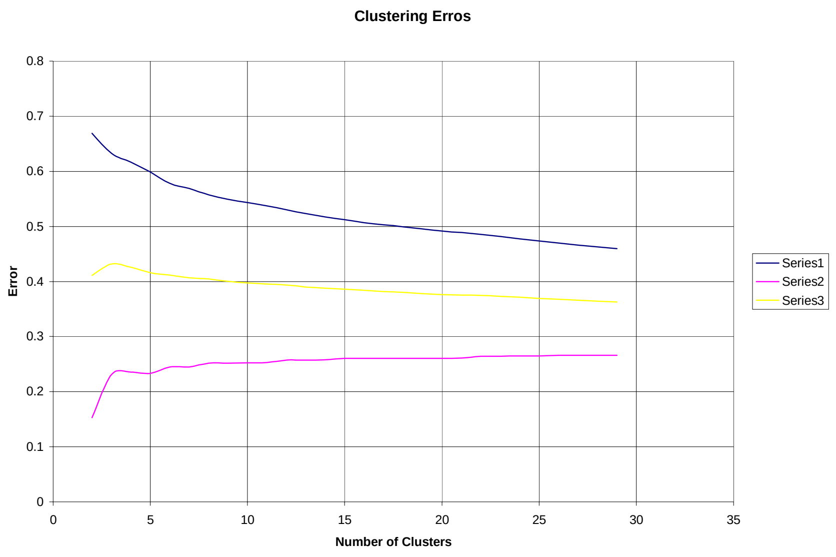


Figure 3. Clustering Erros

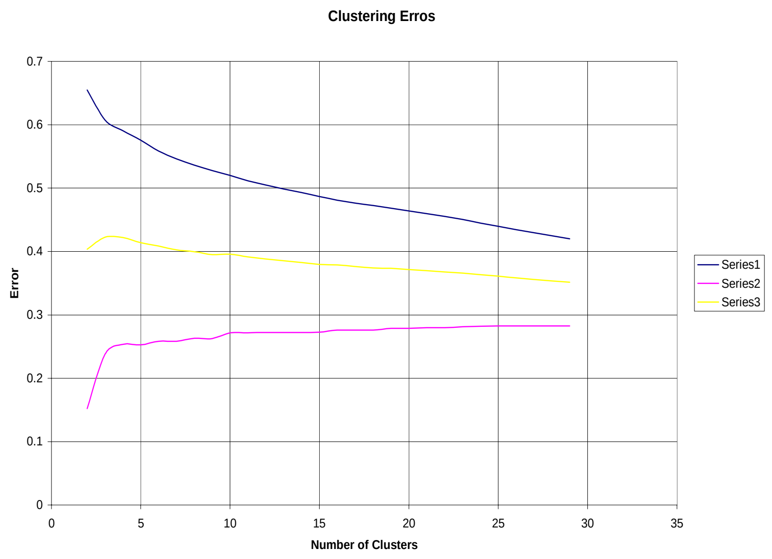


Figure 4. Clustering Error for Random Clusters

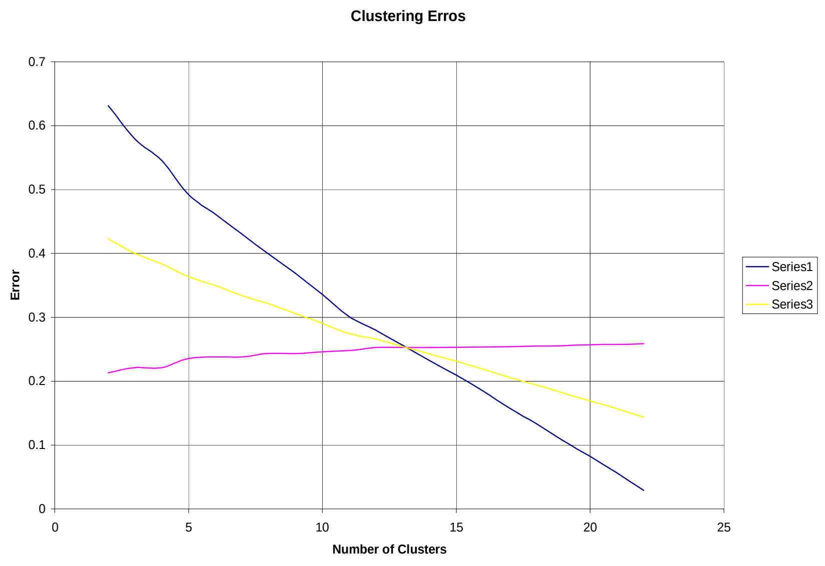


Figure 5. Clustering Error for Random Clusters

5 Summary and Future Work

In this paper we presented a digital library framework that has services to support storage, retrieval and discovery of 3D models. We described a navigation service based on hierarchical clustering of similar 3D models. In this paper we used shape similarity to compute the clusters of the 3D models used in the navigation service.

In the future we are planning to use additional similarity measure that combine the shape similarity along with other metadata similarity measures (e.g. using the text description of the 3D model). We are also planning to use clustering of similar 3D models to speed up the similarity search process using the same methodology that we implemented for browsing the 3D models. This will be done by computing the similarity of models with only the representative “key” model of a cluster instead of computing the similarity with all members of the cluster.

References

[1] H. Anan, K. Maly and M. Zubair, “Digital Library Framework for Progressive Compressed 3D Models, Proceedings of SPIE Vol. 4661, January 2002.

[2] H. Anan and S. Yamany, “Free-Form 3D Objects Compression using Surface Signature”, Proceedings of SPIE Vol. 4197, November 2000.

[3] R. Campbell and P. Flynn. A survey of free-form objects representation and recognition techniques. CVIU, 81(2):166–210, 2001.

[4] C. S. Chua and R, Jarvis, “Point Signatures: A new representation for 3D object recognition”, International Journal of Computer Vision 25(1), pp. 63-85, 1997.

[5] A. Johnson and M. Herbert, “Surface Matching for Object Recognition in Complex three-dimensional scenes”, Image and Vision Computing 16, pp. 635-651, 1998.

[6] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data, Prentice-Hall advanced reference series. Prentice-Hall, Inc., 1988.

[7] A. K. Jain, M. N. Murty and P. J. Flynn, “Data Clustering: A Review”, ACM Computing Surveys, Vol. 31, No. 3, September 1999.

[8] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data. An Introduction to Cluster Analysis*, John Wiley, 1990.

[9] K. Maly, M. Zubair, H. Anan, D. Tan, and Y. Zhang, ”Scalable Digital Libraries based on NCSTRL/Dienst”, ECDL2000, pp.168-180, Lisbon, September 2000.

[10] 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”, DC 2002, Florence, October 2002.

[11] X. Liu, K. Maly, M. Zubair, and M. L. Nelson, “Arc - An OAI Service Provider”, JCDL2001,

[12] X. Liu, K. Maly, M. Zubair, and M. L. Nelson, “Arc - An OAI Service Provider for Digital Library Federation”, D-Lib Magazine 7(4), April 2001.

[13] F. Stein and G. Medioni, “Structural Indexing: Efficient 3D Object Recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), pp. 125-145, 1992.