

Shape-based searching for product lifecycle applications

Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou,
Yagnanarayanan Kalyanaraman, Karthik Ramani*

*Purdue Research and Education Center for Information Systems in Engineering, PRECISE, School of Mechanical Engineering,
Purdue University, 1288 Mechanical Engineering Bldg, West Lafayette, IN 47907, USA*

Received 31 January 2005; accepted 2 February 2005

Abstract

Estimates suggest that more than 75% of engineering design activity comprises reuse of previous design knowledge to address a new design problem. Reusing design knowledge has great potential to improve product quality, shorten lead time, and reduce cost. However, PLM systems, which address the issue of reuse by searching for keywords in filenames, part numbers or context attached to CAD models, do not provide a robust tool to search reusable knowledge. This paper presents a brief overview of a novel approach to search for 3D models. The system is built on a client–server–database architecture. The client takes in the query input from the user along with his search preferences and passes it to the server. The server converts the shape input into feature vectors and a unique skeletal graph representation. Details of the algorithms to perform these steps are presented here. Principal advantages of our graph representation are: (i) it preserves geometry and topology of the query model, (ii) it is considerably smaller than the B-Rep graph, and (iii) it is insensitive to minor perturbations in shape, but sensitive enough to capture the major features of a shape. The combined distance of feature vectors and skeletal graphs in the database provide an indirect measure of shape similarity between models. Critical database issues such as search system efficiency, semantic gap reduction and the subjectivity of the similarity definition are addressed. This paper reports our initial results in designing, implementing and running the shape search system.

© 2005 Elsevier Ltd. All rights reserved.

1. Introduction

Traditional Computer Aided Design (CAD) software has undergone rapid transformation and has evolved into an industry commodity. Technological progress enabled CAD software to incorporate engineering know-how into the design process and develop Product Data Management (PDM) systems. CAD models were further integrated forward in the design cycle into analysis and manufacturing to develop Computer Aided Engineering (CAE) tools. Rapid development in each of these areas resulted in a lot of ‘pockets’ of automation. Currently, all of these ‘pockets’ are being integrated by Product Lifecycle Management (PLM) systems thereby producing a completely digital design through manufacturing solution.

Conservative estimates suggest that more than 75% of design activity comprises case-based design, i.e. reuse of previous design knowledge to address a new design problem [1]. Most PLM systems address the issue of reuse by searching for keywords in filenames, part numbers or context attached to the CAD model. However, this method is not robust primarily because of the following reasons:

1. All models will not have a well-defined attached context.
2. Keywords such as project names or part names may be unknown to the user.
3. Context may be too narrow or too broad to retrieve relevant models.
4. Context changes with time—such as when designers or naming conventions change.

Design reuse spans across the entire product lifecycle as shown in Fig. 1. Design reuse includes not only the CAD design but also of physical inventories such as tooling or other knowledge such as cost data and lead time information. Clearly, effective deployment of a knowledge mining system that will enable knowledge-reuse would require a combination of text and shape-based searching.

* Corresponding author. Tel.: +1 765 494 5725; fax: +1 765 494 0539.

E-mail addresses: iyern@purdue.edu (N. Iyer), jayantis@purdue.edu (S. Jayanti), kuilou@purdue.edu (K. Lou), yagna@purdue.edu (Y. Kalyanaraman), ramani@purdue.edu (K. Ramani).

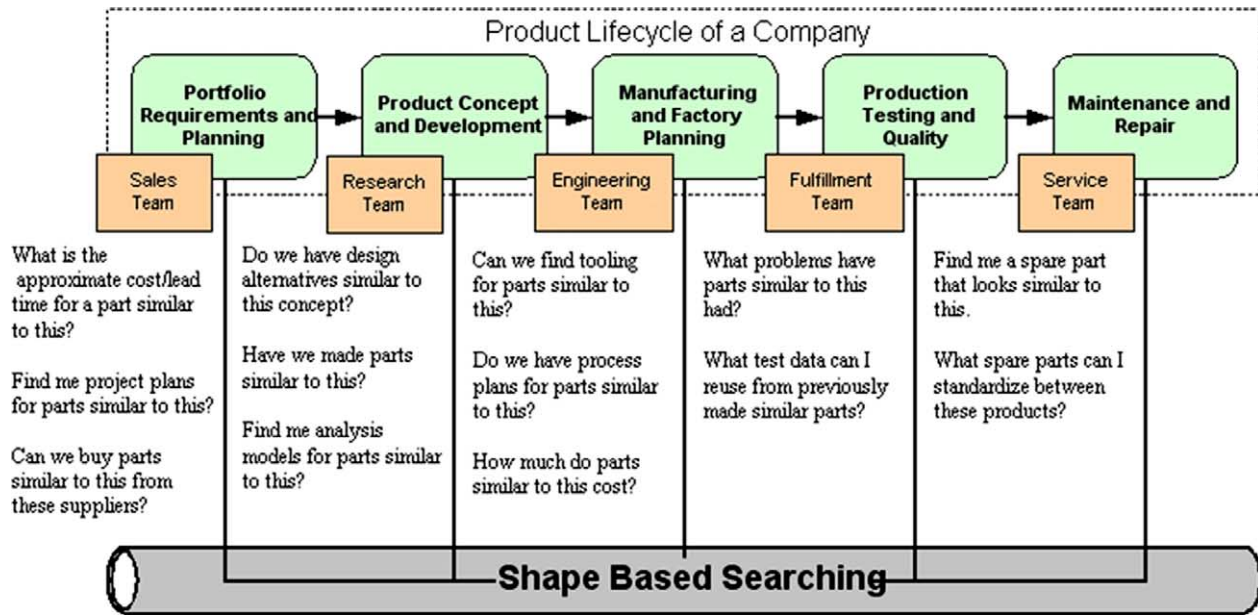


Fig. 1. Knowledge-reuse across the Product Lifecycle.

Significant benefits of design reuse include:

1. Reduction of variability, leading to lesser quality and warranty problems.
2. Eliminating unnecessary parts, thus increasing inventory turnovers.
3. Elimination of downstream activities such as testing and verification and prototyping.
4. Uniform engineering specifications, leading towards standardization.

A significant amount of knowledge generated during design and manufacturing is associated with 3D CAD data. Most of this information (context) is in the form of documents, images and text. From the viewpoint of design engineers who want to reuse design knowledge, 'similar products' usually means those designs that have 'similar' attributes such as shape, materials, processes, tolerances and applications. Among these attributes, shape is the most complex to compare and evaluate. In addition, the 3D shape of an object transcends language and is an unambiguous representation of an engineering artifact and is tightly coupled with other engineering information such as analysis and manufacturing information. Hence, a search system which is capable of retrieving similar 3D models based on their shape will retrieve shape and related knowledge that would not be discovered by other means. Furthermore, shape is one of the factors that significantly influence decision-making and analysis. Thus, we focus our research on developing efficient and effective 3D engineering shape search for design reuse. This paper describes the related literature and experimental results of a 3D engineering shape search system.

2. Literature survey

The past years have seen limited attempts in searching for 3D CAD models. Almost all related methods for matching 3D shapes decompose a shape into a signature. An extensive review of related methods is available in [2,3]. Based on the methods used to convert a shape to a signature, they can be classified into the following categories:

1. *Invariant/descriptor-based*. These methods use invariants or descriptors of the 3D shape such as volume, surface area, aspect ratio, higher order moments or moment invariants as signatures [4–6].
2. *Harmonics-based*. These approaches use a set of harmonic functions of a shape as its signature. Spherical or Fourier functions are usually used to decompose a discrete 3D model into an approximate sum of its (first n) harmonic components [7–9].
3. *Statistics/probability-based*. Osada et al. [10] use shape functions and construct a shape distribution by random sampling of points. Ankerst et al. [11] use shape histograms to approximate and search for a 3D model.
4. *3D object recognition-based*. Some 3D object recognition approaches that have been used for 3D shape searching are Aspect Graphs [12], Spin Images [13], and Geometric Hashing [14].
5. *Graph-based*. Graph-based approaches have employed subgraph isomorphism for matching B-Rep graphs [15, 16], matching eigenvalues of a model signature graph [17] constructed from the B-Rep graph [18–21]. A multi-resolution Reeb graph that captures the topology of a part is proposed in [22].
6. *Feature recognition (FR)-based*. Ramesh et al. [23] decompose a part into cells which are further processed

to identify machining features and their spatial relationships to calculate similarity between parts. Ascher et al. [24] simplify parts into maximal feature subgraphs (MFSG) which are matched to obtain a similarity measure. Other FR-based methods are discussed in [25–28].

7. *Group Technology (GT)-based.* A two-step Group Technology method was developed in [29] to compare similarity between parts. A Type Abstraction Hierarchy (TAH) was proposed in [30] as an alternative to GT.

Invariant/descriptor-based and Harmonic-based methods, while being computationally efficient, are unable to discriminate among dissimilar shapes. Statistics/probability-based methods have a tradeoff between computational cost and number of sampled points. Sampling a lower number of points leads to very low accuracy. 3D object recognition-based methods have been tested for limited shapes and have high storage/computational costs. Graph-based methods are intractable for large graphs because of the NP-completeness of graph/subgraph-matching problems. Feature Recognition-based methods do not decompose shapes uniquely. Additionally, they may require human intervention. Group Technology-based methods are not robust or require extensive human input. In summary, most previous approaches are either granular (categories 4–6) or lumped (categories 1, 2, 3, 7). Granular approaches represent the shape in great detail making the search intractable, while lumped approaches combine all shape characteristics into a single quantity. Furthermore, they do not address the problem of reducing the semantic gap and are not scalable. The input to most systems is a detailed model, which is unrealistic in an engineering design situation. Depending on the stage of design, the designer may not know the detailed shape of the model he/she is searching for. Feature vectors and similarity metrics are predefined within most systems, thus making it impossible for the user to define a custom similarity measure. Importantly, an application domain is not considered in most methods. The notion of similarity for engineering shapes is cognitively different than that for other shapes.

Based on our discussions with personnel from design, manufacturing, and cognitive psychology, we confirmed that a universal similarity metric is impossible to formulate. Similarity is an application-driven, user-perceived notion. Our discussions also helped us to generate important criteria for a shape-based search system as shown below:

1. *Sensitivity.* The system must be sensitive to large variations in shape. However, it must also be invariant to trivial perturbations in shape.
2. *Similarity measure.* We treat similarity as a customizable user decision. A user can construct a similarity metric ‘on-the-fly’ and be able to change it real-time after looking at preliminary results.

3. *Efficiency.* The algorithms developed for performing similarity searches must be computationally efficient. Although database indexing could take time, subsequent processing and searching should be efficient.
4. *Effectiveness.* The system must be effective enough to find the relevant results in a limited number of retrieved models. This is related to reducing the ‘semantic gap’ between the user’s similarity notion and the database similarity metric.
5. *Global vs. local.* We also found that most users would like to know which local areas of two parts were similar or dissimilar. This requires that the representation should capture and allow search based not only on the global shape attributes, but also local attributes.
6. *Query interface.* The query interface must allow for multiple modes of inputs—quick, approximate shape creation through sketching, selecting a query model from a cluster of parts and construction of a user-driven similarity metric.

3. System architecture and search process

Based on the criteria described in Section 2, we developed a three tier Client–Server–Database architecture (Fig. 2) for a shape-based search system that uses two types of shape representations (global feature vectors and skeletal graphs) to capture the global and local shape of the 3D model. The Client end consists mainly of the user interface (UI), which enables query creation and display of results. The Server side takes in the shape input from the Client and converts it into multiple representations (voxel, skeleton, feature vectors and skeletal graphs), which are then stored into the Database for subsequent search and reuse.

3.1. User interface

The function of the user interface (UI) is to allow a user to browse the database, compose a custom query and to display the search results. The primary function of the UI is to enable the following user interactions:

1. *Query-by-example (QBE).* QBE is a method of query creation that allows a user to search for 3D models based on an example in the form of a 3D model selected from the database. A query is initiated by choosing a part most similar to that the user desires. Part choice will be enabled by a user interface that will enable a user to ‘drill down’ through clusters of similar parts to find a part to initiate the search process (shown by the dashed-dotted line in Fig. 2).
2. *Query-by-sketching (QBS).* QBS is a query initiated by sketching an approximate 3D model, the user is interested in searching for (shown by the thick dashed line in Fig. 2). Any commercial CAD system can be used to generate the rough sketch. The rough sketch from a user only needs to capture the important features of the overall shape of

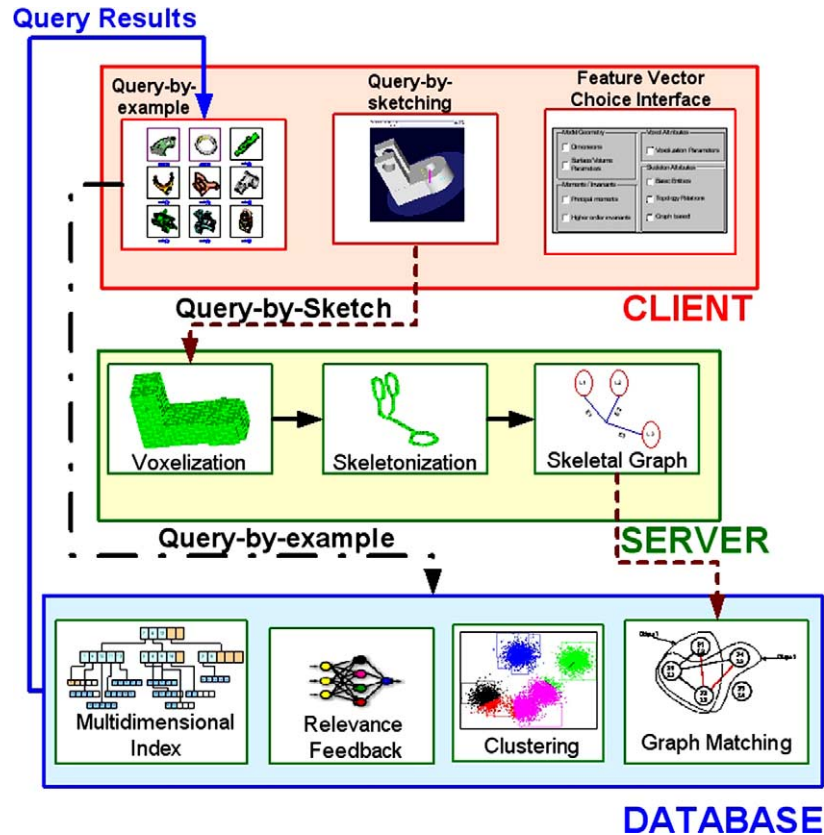


Fig. 2. System architecture showing various query processes and components of the search system.

the 3D model. The reason for allowing a rough 3D sketch interface is for the fact that engineering models tend to be complex and are often difficult to reconstruct from 2D projection-based sketch-interfaces as provided in most 3D shape search systems (such as in [7]).

3. *Feature vector choice interface.* The search system generates multiple feature vectors for the same 3D model. Each feature vector captures different characteristics of the 3D shape. A user can combine or switch between different feature vectors dynamically during the search process. Depending on the user's notion of similarity, the search can be 'reconfigured' until the user is satisfied.
4. *Relevance feedback interface.* The search system also allows the user to provide feedback to the system regarding the relevant and irrelevant models from a given search result by clicking on the images. A neural network-based algorithm is used to learn the weights of the similarity metric for subsequent searches, thereby reducing the semantic gap between the user and the system.

3.2. Server

In this section we explain the functionalities of the server. The first step in converting 3D models to a searchable representation is to normalize the model into a canonical form independent of position, orientation and

scaling. Subsequently, two types of shape representations are extracted from the normalized model: Global Feature vectors and Skeletal graph representation. The skeleton graph is obtained through a series of operations on the normalized model: voxelization (Section 3.2.1), skeletonization (Section 3.2.2), and skeletal graph extraction (Section 3.2.3). However, the feature vectors are extracted from the model after each such operation. While skeletal graphs are explained in detail in Section 3.2.3, the feature vectors used in this paper are explained in Section 3.2.4.

3.2.1. Voxelization

Voxelization is defined as the process of converting a geometric representation of a synthetic model into a set of voxels (volume elements) that best represents the synthetic model within the discrete model space. In recent years, a number of curve, surface and polygon mesh voxelization algorithms have been proposed. Most algorithms provide efficient ways to extend 2D scan conversion methods to the 3D domain. The important difference between 2D scan conversion and voxelization is that voxelization is decoupled from the rendering process and hence is a one-time process. The most dominant solid representation methods are Boundary Representation (B-Rep) and Constructive Solid Geometry (CSG). B-Rep describes a part in terms of its vertices, edges and faces. CSG describes the part

in terms of a set of Boolean operations applied to primitive geometric entities such as cubes and cylinders.

It is difficult to voxelize a B-Rep solid because the interior is not explicitly defined. Except for the point-classification method proposed in [32], there has not been any efficient voxelization method proposed for B-Rep solids. However, advances in parallel computation, hardware and computational power have made voxelization of B-Rep solids almost real-time. Most 3D CAD systems such as Pro/Engineer™, SolidWorks™ and IDEAS™ use a B-Rep representation as their internal data structure. Furthermore, there is no unique CSG representation standard for a part. Therefore, in this paper, we only consider B-Rep models since most engineering parts are either available in B-Rep formats or can be easily translated into a neutral B-Rep representation such as STEP.

3.2.2. Skeletonization

The notion of a skeleton was introduced by Blum [33]. Skeletonization is the process of extracting a skeleton from a 3D binary image (voxel model). The model can be converted into a binary 3D digital model. However, in digital spaces, only an approximation to the ‘true skeleton’ can be extracted. The two requirements for a skeleton are:

1. *Topological*. A skeleton must retain the topology of the original object.
2. *Geometrical*. A skeleton must be in the ‘middle’ of the original object and must be invariant to translation, rotation and scaling.

The three major skeletonization techniques are:

1. *Voronoi-based*. The Voronoi diagram of a set of discrete points (generating points) is the partition of the given space into cells. If the density of boundary points of the 3D image goes to infinity, then the corresponding Voronoi diagram converges to the skeleton. The exact Euclidean skeleton is usually approximated by a subgraph of the Voronoi diagram [34].
2. *Distance transform-based*. Distance criteria such as four-neighbor (city block), eight-neighbor (chessboard) or (3,4) chamfer distance are used to convert a 3D image into a distance map representing distances from object boundaries. The local extrema on the distance map appear as ridges and they are detected as skeletal points [35].
3. *Thinning*. Thinning is an iterative object reduction technique for extracting skeletons in digital spaces. It can be used to generate approximate medial surfaces or medial axes of the 3D image [36]. Points in the image are deleted if they satisfy deletion conditions that preserve topology.

We choose thinning as our preferred skeletonization method because it offers advantages over distance transform

and Voronoi-based skeletons. Distance transform skeletons satisfy ‘geometrical’ conditions but may not satisfy ‘topological’ conditions. Voronoi-based skeletons satisfy both requirements, but are very expensive to compute especially for realistic engineering models. Furthermore, Voronoi skeletons have unwanted appendages, which require pruning as an additional process. Thinning satisfies the topological requirements but does not always satisfy geometrical requirements. However, for our application, topological correctness is more important than geometrical correctness. We use the skeletonization algorithm described in [22].

3.2.3. Skeletal graph generation

The skeleton is converted into a hierarchical skeletal graph for storing in the database. This graph can be analyzed at different levels depending on the information content. Previous approaches have converted a skeleton into a voxel graph [37] or into a skeleton tree [38]. We convert the skeleton into a skeletal graph made up of the following basic skeletal entities:

1. Node—the voxel situated at the ends of the edges
2. Edge—set of voxels forming a single geometric entity
3. Loop—one or more edges forming a closed entity

Each edge in the skeleton translates into an independent geometric entity giving shape to the model in physical space. Similarly, each loop represents a hole in the 3D model. For example, if a user wants to search for models with holes, it can be completed by searching for graphs with loops. Thus the skeletal graph is more physically relevant than feature relationship graphs, but simpler than B-Rep graphs.

A skeleton marching algorithm is used to identify the basic entities form the skeleton and to construct the skeletal graph. The algorithm identifies edges and simple loops in the skeleton. Loops, which do not share any edge or node with the other loops, are called simple loops. To get non-simple loops in the skeleton, the skeleton is analyzed based on the depth first technique. Thus, the geometry of the 3D model is captured in the individual entities of the skeleton and the topology in the skeletal graph.

3.2.4. Feature vector extraction

A 3D model is converted into a set of feature vectors for database searching. We extract the following shape descriptors for a 3D model:

- (a) *Moments/invariants*. The three principal moments of the 3D model are extracted using the ACIS API and stored as a feature vector. Further, the three second-order moment invariants for the model are also stored as a feature vector. Moment/invariants are by nature independent of orientation. For every voxel in the model translation, rotational and scale invariant second-order moments are calculated as described

below

$$\kappa_{lmn} = \frac{\int \int_{-\infty}^{\infty} \int x^l y^m z^n \rho(x, y, z) dx dy dz}{\mu_{000}^{5/3}}, \quad l + m + n = 2$$

where μ_{lmn} are central moments after translation. The three moment invariants that are calculated are:

$$I_1 = \kappa_{200} + \kappa_{020} + \kappa_{002},$$

$$I_2 = \kappa_{002}\kappa_{200} + \kappa_{002}\kappa_{020} + \kappa_{200}\kappa_{002} - \kappa_{101}^2 - \kappa_{011}^2 - \kappa_{110}^2,$$

$$I_3 = \kappa_{200}\kappa_{020}\kappa_{002} + 2\kappa_{110}\kappa_{011}\kappa_{101} + \kappa_{101}^2\kappa_{020} - \kappa_{011}^2\kappa_{200} - \kappa_{110}^2\kappa_{002}$$

- (b) *Geometry parameters.* The geometry parameters stored as a feature vector are the two aspect ratios for the bounding box, model surface area, model volume and the ratio of surface area to volume.
- (c) *Voxelization parameters.* The voxel size and the scaling factor obtained during the normalization process are stored as a feature vector.
- (d) *Graph parameters.* The number of loops, edges and nodes in a skeletal graph is characteristic of the shape of the 3D model. We store them as a feature vector. Further, the different topological relationships in the skeletal graph like the number of loop–loop, edge–edge and loop–edge connections are also stored as a feature vector.

3.3. Database

A *similarity measure* is a function for quantifying the similarity between two models. It takes the feature vectors of the query model and that of a model in the database and outputs a real number that reflects the degree of similarity between the two models. It is preferable that similarity measures satisfy the metric axioms [39,40] as follows.

Let S be a set of objects, a metric on S is a function $d : S \times S \rightarrow \mathfrak{R}$, which satisfies the following metric axioms, for all $x, y, z \in S$

$$d(x, x) = 0, \quad d(x, y) = 0 \Rightarrow x = y,$$

$$d(x, y) + d(x, z) \geq d(y, z), \quad d(x, y) \geq 0, \quad d(x, y) = d(y, x)$$

We use the Euclidean distance between points in feature space to indirectly represent the similarity measure. In other words, larger the distance between two models the lesser the similarity. Clearly, this similarity measure is a metric and is the primary measure of similarity for models based on features.

3.3.1. Database indexing

Index structures are used to speed up searching in large databases, and the design of these structures is a critical

issue for any realistic 3D shape search system. The logarithmic query time of B+ trees is in part responsible for the success of relational databases [41]. However, the B+ tree index is basically a one-dimensional index structure, which orders the data records by one attribute at a time. This one-dimensional index structure is not sufficient for 3D shape search systems. The fundamental problem is that the feature vectors representing 3D models are complex data types. Searching is usually based on overall similarity (*similarity query*) rather than the similarity of individual attributes (*attribute query*). We have to use all the attributes simultaneously to determine the similarity. In other words, we cannot discard a model from the candidate list only because some attributes do not match the query model. Thus, in order to search for similar 3D models efficiently, we need an index structure with properties such that:

1. It is a multi-dimensional index. The multi-dimensional index arranges the models using all the feature elements, and
2. It groups similar models and puts them in the same node or contiguous index nodes. This grouping can reduce the disk I/O time during search operations.

R-tree-based multi-dimensional indexes have been extensively studied for content-based image retrieval. In such an index, points in feature space are clustered into groups, while a group is represented by a bounding rectangle/hyper-rectangle containing the points. The bounding hyper-rectangle is a tight bounding box that is represented by the coordinates of its corners. To answer a query, the query point is compared with the bounding box in order to prune the sub-tree rooted at this hyper-rectangle, using a nearest-neighbor search. Therefore, the R-tree index satisfies the requirements of the similarity index listed above. The leaf nodes contain pointers to 3D models. The search starts at the root and is directed by internal node to the leaf node. The tree structure is similar to a B+ tree; however, the criterion to arrange the records is totally different from that of B+ trees.

3.3.2. Graph-based indexing

In addition to similarities between feature vectors, we also investigate similarity based on *skeletal graphs*. Skeletal graphs are undirected entity graphs and are represented as adjacency matrices. The adjacency matrix is formulated to capture the structural properties of the skeleton, such as loop–loop, edge–loop connections, etc. This representation reflects the topology and the high-level geometry of the skeletons.

The goal of our search system is to retrieve all models that are ‘similar’ and not only those that match exactly with the query model. In graph similarity search, this translates to finding models whose skeletons have isomorphism as well as subgraph isomorphism with the query model. Traditional exact approaches [42] for graph comparison lead to an

NP-complete problem. It is infeasible to efficiently find a solution except for small graphs (typically 30–50 nodes). Therefore, large graphs have to be preprocessed before comparison [43]. Since the skeletonization of 3D models produces graphs of very small size, we are able to considerably reduce indexing and search time for a large number of parts. More than 70% of the 150 models in our database have skeletons with eight entities or less.

Skeletal graphs are indexed in the database based on a decision-tree-based approach described in [43]. The small sizes of our *skeletal graphs* prove to be inexpensive in terms of indexing time and space requirements for this algorithm. Our system uses a rudimentary similarity measure, which is defined as the ratio of the number of entities that match between two graphs (say, G_A and G_B) to the size of the larger of the two graphs

$$\text{Similarity}(G_A, G_B) = \left(\frac{\# \text{ of Matching Entities}}{\text{Max}(|G_A|, |G_B|)} \right) \times 100$$

This similarity measure is a metric since it satisfies all the metric axioms described earlier. However, this measure does not take into account the relative sizes of the skeletal entities in determining similarity, and is not refined enough for finer comparison. We are studying the use of additional geometric measures to describe the local properties of the 3D model. Some of these measures include the lengths of the edge entities, average circumference and diameters of loop entities, and the tangential angle between connected entities at the point of intersection.

3.3.3. Relevance feedback

The semantic gap and subjectivity of similarity make it difficult to correctly retrieve similar models for a user. Relevance feedback [44] is used to train the system to reconfigure its similarity definition to reflect user's preference so as to bridge the 'semantic gap of similarity'. This reconfiguration is applied on the low-level feature vectors. However, it is impossible for a user to directly adjust the weights of low-level feature elements. The method to circumvent this problem is to present search results graphically to a user, ask the user to evaluate the search results, and give feedback to the system. The system, in turn, identifies patterns in the user's feedback and adjusts the weights accordingly. Thus a user just deals with the 'high-level' models instead of 'low-level' features.

The procedure of relevance feedback consists of the following steps:

- (a) *Database query*. When the system retrieves some models based on the query example, it will order them by similarity measure of feature vectors and presents them to the user.
- (b) *Relevance feedback*. Users classify them into relevant and irrelevant groups and inform the system.

- (c) *Query reinterpretation and reconfiguration*. The system, in turn, reconfigures itself to reflect the user's preference, reinterpret user's intention and performs the search again.
- (d) *Repeat query*. This process iterates until either the user is satisfied or the system cannot improve the results further.

We utilize user's relevance feedback for updating the query point to represent user's query intent. After being presented the search results, the user is given an option to identify the relevant and irrelevant models. In *query reinterpretation*, the query neighborhood is moved to the centroid of the relevant models. The new query better represents the user's search intention. The search based on the new query can find more relevant models that were not identified by the previous search. We are also investigating other approaches such as weight reconfiguration in conjunction with neural networks.

3.3.4. Clustering

In order to enable a user to quickly search a model, our system is designed to provide an easy to browse hierarchical Query-by-example interface as described earlier. This interface will use unsupervised clustering algorithms to form hierarchical similarity-based clusters. Traditionally, there are three major types of clustering algorithms: (1) *hierarchical* (2) *K-means* clustering and (3) *Self-Organizing Maps (SOMs)*. We have used SOMs for forming the cluster maps because they: (1) are amenable to visualization, (2) allow easy interpretation, (3) allow dynamic updating, and (4) are simple to implement. A self-organizing map (SOM) is a clustering algorithm based on a competitive learning approach [45]. It consists of a number of interconnected computational units called *neurons*, which are laid in either a *hexagonal* or *rectangular* topology. As new instances of 3D models are presented to the network, the positions of neurons are updated such that similar neurons move closer while dissimilar neurons move away from each other. Also, similar models are grouped in neighboring neurons, and dissimilar models are indexed in distant neurons.

4. Results

We designed and implemented experiments to test our approaches to deal with challenges for 3D shape search systems, i.e. subjectivity of shape similarity, semantic gap and efficiency. The tests were conducted on a DELL Pentium 2.66 GHz PC with 1.0 GB RAM. The voxelization and skeletonization algorithms were implemented in C++ with ACIS libraries, while the database systems were implemented in Java. In actual deployment, the database will be constantly updated if a change is made to the CAD model, and the server side processes of voxelization,

skeletonization, skeletal graph and feature vector extraction are repeated.

We tested the ability of the multi-dimensional R-tree index as well as SOMs to cluster similar models with a dataset of real-world 3D CAD models, while the efficiency of search operations using the R-tree index was tested using both synthetic and the real-world datasets. The real-world dataset currently consists of about 150 engineering CAD models. Although this database is being expanded, the size is still relatively small as compared to typical design repositories. We therefore generated synthetic datasets for which the sizes vary from 50 to 1,000,000 data records. The effects of database *size*, *dimensionality* of feature vector, and *node volume* on the efficiency of the feature vector-based search were studied with the synthetic datasets. The synthetic database was created using a random number generator. The utility of relevance feedback to enable a customizable similarity definition was tested with the real-world dataset.

4.1. Skeletal graph generation and matching

The decision-tree-based index for graph matching was populated with skeleton graphs of the real-world dataset. Fig. 3 shows results for a query on the database. The search results are presented with the rank and similarity values for each retrieved model. Although, the system retrieves all the relevant models in the database, it also retrieves irrelevant parts, since there is no filtering mechanism to reduce the search space. Enriching the *skeletal graph* structure with more information such as local properties (volume, parametric equations and moments) and classification of edges into *straight* and *curved* types will increase the precision of our system. As the database is scaled to have more parts, a filtering mechanism needs to be designed. Due to the presence of rich geometric information in our skeletal graphs, we will be able to reduce the search space.

4.2. Feature-based database indexing

An R-tree-based multi-dimensional index was also built for the real-world dataset. We used the moment invariants as the feature vector for search. For this database, the index has 62 nodes and four levels, with a maximum of five models in

 Query	 Rank: 1 Simil: 100%	 Rank: 2 Simil: 83.33%	 Rank: 3 Simil: 80%
 Rank: 4 Simil: 71.43%	 Rank: 5 Simil: 60%	 Rank: 6 Simil: 60%	 Rank: 7 Simil: 60%

Fig. 3. Query results for skeletal graph-based searching.

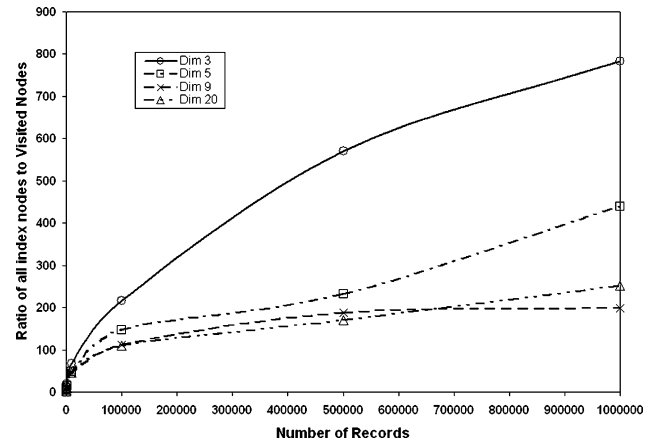


Fig. 4. Ratios of visited nodes at different dimensionality and database size.

each node. The results of a query for this database are presented in Fig. 6 of Section 4.3. For a synthetic database with 1,000,000 records, the R-tree has 386,829 nodes and 11 levels.

The primary indicators of the search efficiency are the number of visited nodes and ratio of visited nodes during a search operation. The visited nodes are loaded into the temporary memory (RAM) from the database during search operations. The query model is compared with the models in the retrieved node to determine the search direction or to compute similarity. The *ratio of visited nodes* is the ratio of the total number of nodes to the number of *visited nodes*. Since disk I/O is the bottleneck for current computer systems, the number of nodes visited directly translates to the number of disk I/O operations, which directly impacts overall performance. The *ratio of visited nodes* indicates the improvement that R-tree achieves compared to an exhaustive search. Figs. 4 and 5 show some results from testing synthetic data sets with different *sizes* and *dimensionality*, respectively, when the *node volume* is five. *Dimensionality* is the number of feature elements that describe a 3D model.

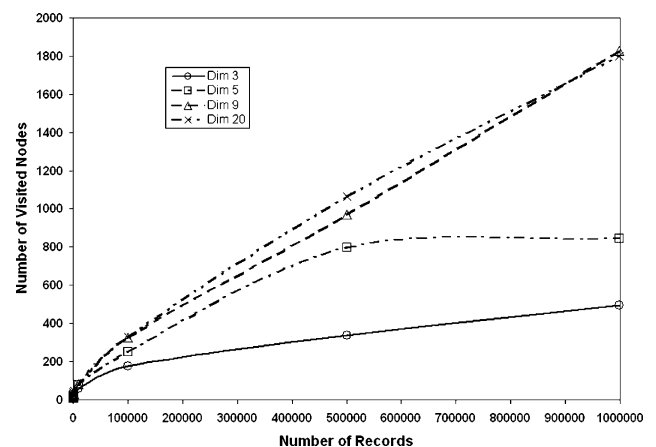


Fig. 5. Numbers of visited nodes vs. dimensionality and database size.

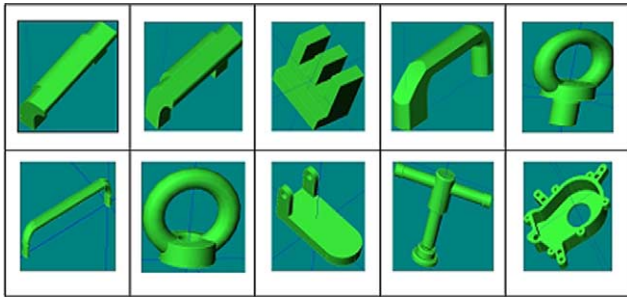


Fig. 6. Search results before relevance feedback. The top left model is the query.

4.3. Relevance feedback

Figs. 6 and 7 are the models retrieved by the search system before and after query reconstruction based on relevance feedback. The top-left model is the query model. These results were obtained by using moment invariants as the feature vector. It can be seen from Fig. 7 that the similarity is recomputed and the system retrieves a larger number of similar models after relevance feedback than before. These results illustrate the effectiveness of relevance feedback in retrieving similar parts and reducing the semantic gap between the user and the feature space.

4.4. Database clustering

The multi-dimensional R-tree structure indexes models based on their similarity into the same or contiguous nodes. Fig. 8 presents a few examples of the 3D models clustered in the nodes of the R-tree. However, it must be noted that R-tree is not optimized for clustering. Therefore, in this study, self-organizing maps were studied for clustering 3D engineering parts based on the second-order moment invariants. The architecture of the SOM consists of 64 nodes in an 8x8 hexagonal topology, while training was continued until 800 iterations.

Fig. 9 illustrates three sample nodes that were generated by the Self-Organizing Map (SOM). The models grouped by

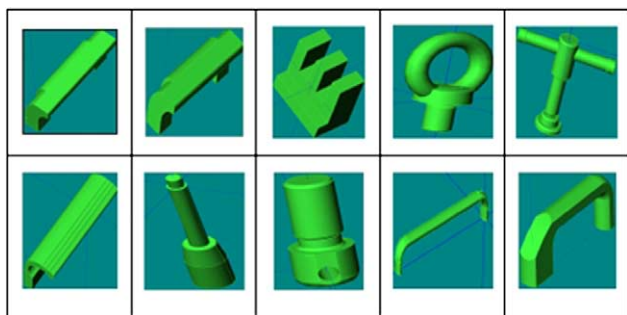


Fig. 7. Search results after relevance feedback. The top left model is the query.

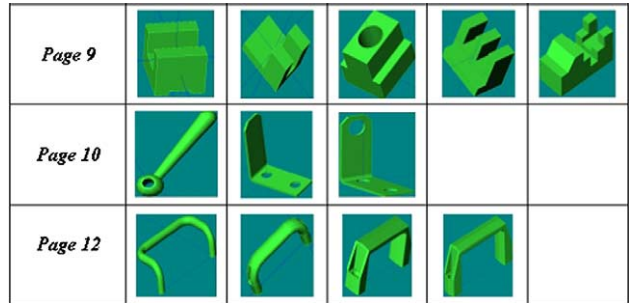


Fig. 8. Models clustered in the R-tree nodes.

the SOM in each of the three sample nodes are similar. Due to space constraints we are unable to present all the results from the two methods for comparison. In the future, we will use data clustering algorithms to group similar models into clusters prior to insertion in the R-tree. This will improve the performance of the multi-dimensional index. We will also explore methods to cluster models based on the skeletal graph representations.

5. Discussion

Moment-based shape descriptors and aspect ratio have been used for 3D shape search systems [5,9], however, skeletal graph-based techniques have not been explored in this context. We have used these two representations to capture the global and local aspects of the shape, respectively. While the former representations describe the global shape, they do not capture the global topology of the 3D model. Skeletal graphs, on the other hand, are topology-preserving and also have the ability to capture the local geometric properties thereby satisfying *Criteria 4* and *5* as observed in Section 2. In addition, the skeletal graphs are invariant to trivial perturbations in the shape of the 3D model (*Criterion 1*).

One of the principal advantages of skeletal graph-based approach over traditional global methods is

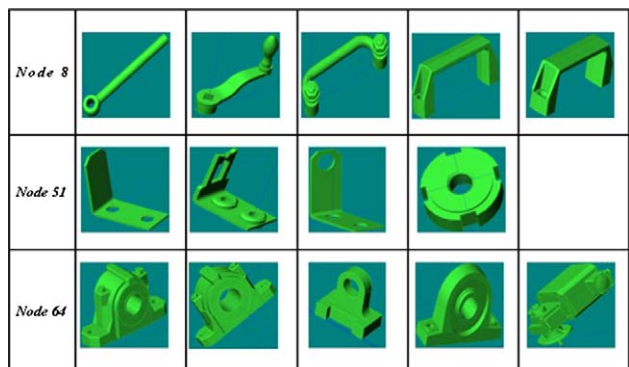


Fig. 9. Models clustered in SOM nodes.

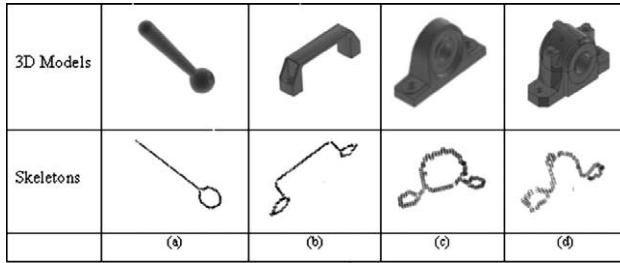


Fig. 10. Example skeletons for 3D part models.

that skeletal graphs preserve the topology of the 3D model, thereby providing meaning to shape search in the engineering context. For example, the handle model in Fig. 10(a) is reduced to two entities, namely, a loop and an edge. The length of the edge characterizes the length of the handle. Similarly, in Fig. 10(b)–(d) the two loop entities at both ends in the skeleton indicate bolt-holes for securing the part. It must, however, be noted that the resolution of voxelization has serious impact on the skeleton graph-based matching. For example, although parts in Fig. 10(c) and (d) have similar overall shape and functionality, the resolutions are different, thereby producing ‘un-intuitive’ skeleton for Fig. 10(d). In the future, we will develop a multiple Level of Detail (Multi-LOD) approach, where the skeleton of the part is derived at different resolutions in order to capture increasing levels of detail in the part, thereby increasing the effectiveness of skeleton-based search. We observed that at a higher resolution, the model in Fig. 10(d) produces a skeleton graph with the same topology as for the part in Fig. 10(c). This fact explains the reason for the results in Fig. 3, where the part with rank 3 has higher similarity (80%) to the query model than the other two parts with ranks 4 and 5.

The R-tree-based index structure (Section 3.2.1) addresses the efficiency criterion for feature vector-based search (*Criterion 3* in Section 2), while the decision-tree-based index structure (Section 3.2.2) addresses this issue for skeletal graph representation. However, it must be noted that graph-based structures generally have higher computational requirements with regards to time and space for indexing as compared to feature vector-based indexing.

The relevance feedback mechanism reduces the semantic gap between the user and the system, thereby addressing the effectiveness criteria (*Criterion 4*).

The Feature Vector Choice interface addresses *Criteria 2* and *6*, while the clustering mechanism was developed to explicitly address *Criterion 6*.

The framework developed in this paper allows us to easily incorporate the different shape representation available in literature for testing and evaluation. In the future, we will benchmark the effectiveness of other shape

representation methods available in literature against our approach.

6. Conclusions

The paper has introduced the database and related techniques to support a shape search system for PLM systems. Since our system level representation is based on a neutral voxel format, changes in standards such as CAD formats and parametric models will not affect the search process.

Skeletal graphs, which capture topological properties of 3D models, were used for searching 3D models utilizing graph-matching algorithms. Parts retrieved from various queries were ranked based on a similarity measure developed in the study. To the best of our knowledge, this paper is the first to demonstrate the feasibility of using *skeletal graphs* for 3D shape search.

An R-tree-based multi-dimensional indexing technique was used to speed up the content-based 3D shape retrieval effectively. Preliminary testing on a synthetic database showed the scalability of the system. To improve the similarity indexing ability of the R-tree, we investigated the use of clustering algorithms to pre-classify the models. Preliminary results show that the clustering algorithms perform better than the R-tree in grouping similar models. A relevance feedback mechanism, namely query reconstruction, was used to improve the effectiveness of the search system. Additional tests on large databases are required before more conclusions can be drawn.

The system performance will be tested against other available approaches in the 3D shape search domain. Furthermore, a precision vs. recall study will be performed. Additionally, methods for searching for local shape features will be developed.

Acknowledgements

The initial funding for this project came from the 21st Century Research and Technology Fund award from the state of Indiana. We also acknowledge the University Faculty Scholar Award from Purdue University for Professor Karthik Ramani, which seeded this project. Supplemental support from the e-Enterprise Center at Discovery Park, Purdue University is acknowledged. We also thank the Innovation Realization Laboratory at Purdue University for supporting Natraj Iyer. Dr Christoph Hoffmann and Dr Sunil Prabhakar, Department of Computer Sciences, Purdue University deserve a special vote of thanks for their valuable discussions.

References

- [1] Ullman DG. The mechanical design process. 2nd ed. New York: McGraw-Hill; 1997.
- [2] Cardone A, Gupta SK, Karnik M. A survey of shape similarity assessment algorithms for product design and manufacturing applications. *ASME J Comput Inf Sci Eng* 2003;3(2):109–18.
- [3] Iyer N, Lou K, Jayanti S, Kalyanaraman Y, Ramani K. (2005). Three dimensional shape searching: state-of-the-art review and future trends. *Comput Aided Des*;37(5) 509–30.
- [4] Cybenko G, Bhasin A, Cohen K. Pattern recognition of 3D CAD objects. *Smart Eng Syst Des* 1997;1:1–13.
- [5] Elad M, Tal A, Ar S. Content based retrieval of VRML objects—an iterative and interactive approach. *Eurographics multimedia workshop*; 2001. p. 97–108.
- [6] Rea H, Corney J, Clark D, Pritchard J, Breaks M, MacLeod R. Part sourcing in a global market. *Proceedings of ICeCE'01*. Beijing: China Machine Press; 2001.
- [7] Kazhdan M, Funkhouser T, Rusinkiewicz S. Rotation invariant spherical harmonic representation of 3D shape descriptors. *Proceedings of ACM/eurographics symposium on geometry processing*; 2003. p. 167–75.
- [8] Vranic D, Saupe D, Richter J. Tools for 3D object retrieval: Karhunen–Loeve transform and spherical harmonics. *Proceedings of IEEE 2001 workshop on multimedia signal processing*; 2001. p. 293–8.
- [9] Saupe D, Vranic D. 3D Model retrieval with spherical harmonics and moments. In: Radig B, Florczyk S, editors. *Proceedings of the DAGM 2001*, Munich, Germany, 2001. p. 392–7.
- [10] Osada R, Funkhouser T, Chazelle C, Dobkin D. Shape distributions. *ACM Trans Graph* 2002;21:807–32.
- [11] Ankerst M, Kastenmüller G, Kriegel H-P, Seidl T. 3D shape histograms for similarity search and classification in spatial databases. *Proceedings of sixth symposium on large spatial databases*; 1999. p. 207–26.
- [12] Cyr CM, Kimia B. 3D object recognition using shape similarity-based aspect graph. *Proceedings of ICCV'01*; 2001. p. 254–61.
- [13] Ruiz-Correa S, Shapiro L, Meila M. A new signature based method for efficient 3D object recognition. *Proceedings of CVPR'00*, SC; 2000.
- [14] Lamdam Y, Wolfson HJ. Geometric hashing: a general and efficient model based recognition scheme. *Proceedings of ICCV'88*; 1988.
- [15] El-Mehalawi M, Miller R. A database system of mechanical components based on geometric and topological similarity. Part I. Representation. *Comput Aided Des* 2003;35:83–94.
- [16] El-Mehalawi M, Miller R. A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment. *Comput Aided Des* 2003;35:95–105.
- [17] Sun T-L, Su C-J, Mayer R, Wysk R. Shape similarity assessment of mechanical parts based on solid models. *Proceedings of ASME DFM conference*, Boston, MA; 1995. p. 953–62.
- [18] McWherter D, Peabody M, Regli WC, Shoukofandeh A. Solid model databases: techniques and empirical results. *ASME J Comput Inf Sci Eng* 2001;1(4):300–10.
- [19] Peabody M, Regli WC. Clustering techniques for databases of CAD models. Technical report DU-MCS-01-01. Department of Mathematical and Computer Science, Drexel University, Philadelphia, PA; June 2001.
- [20] McWherter D, Peabody M, Regli WC, Shoukofandeh A. An approach to indexing databases of graphs. Technical report DU-MCS-01-01. Department of Mathematical and Computer Science, Drexel University, Philadelphia, PA; June 2001.
- [21] McWherter D, Peabody M, Regli WC, Shoukofandeh A. Transformation invariant shape similarity comparison of solid models. *ASME DETC, computers and information in engineering conference*, September 9–12, Pittsburgh, Pennsylvania; 2001.
- [22] Hilaga M, Shinagawa Y, Kohmura T, Kunii T. Topology matching for fully automatic similarity estimation of 3D shapes. *Proceedings of SIGGRAPH'01*; 2001. p. 203–12.
- [23] Ramesh M, Yip-Hoi D, Dutta D. Feature based shape similarity measurement for retrieval of mechanical parts. vol. 1; 2001. p. 245–56.
- [24] Ascher M, Marefat M, Fasse E. A methodology for automatic retrieval of similarly shaped machinable components. *IEEE international conference on systems, man, and cybernetics*. vol. 2; 2001. p. 2840–5.
- [25] Cicirello V, Regli WC. Machining feature-based comparisons of mechanical parts. *ACM international conference on shape modeling and applications*, Genova, Italy. p. 176–85; 2001.
- [26] Cicirello V. Intelligent retrieval of solid models. MS Thesis. Drexel University, Philadelphia, PA; 1999.
- [27] Cicirello V, Regli WC. Resolving non-uniqueness in design feature histories. In: Anderson D, Bronsvort W, editors. *Proceedings of fifth ACM symposium on solid modeling and applications*, New York, NY.
- [28] Elinson A, Nau D, Regli WC. Feature-based similarity assessment of solid models. In: *Proceedings of fourth ACM/SIGGRAPH symposium on solid modeling and applications*, Atlanta; 1997. p. 297–310.
- [29] Iyer S, Nagi R. Automated retrieval and ranking of similar parts in agile manufacturing. *IIE Trans Des Manufact* 1997;29: 859–76.
- [30] Srinivas G, Fasse E, Marefat M. Retrieval of similarly shaped parts from a CAD database. *IEEE international conference on systems, man, and cybernetics*. vol. 3; 1998. p. 2809–14.
- [32] Lee Y, Requicha A. Algorithms for computing the volume and other integral properties of solids. *Commun ACM* 1982;25: 642–50.
- [33] Blum H. A transformation for extracting new descriptors of shape. In: Dunn E, editor. *Models for the perception of speech and visual form*; 1967. p. 362–80.
- [34] Pudney C. Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images. *Comput Vis Image Understand* 1998; 72:404–13.
- [35] Niblack C, Gibbons P, Capson DW. Generating skeletons and centerlines from the distance transform. *Graph Models Image Process* 1992;54:420–37.
- [36] Palagy K, Kuba A. A 3D 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognit Lett* 1998;19:613–27.
- [37] Reinders F, Jacobson M, Post F. Skeleton graph generation for feature shape description. In: *Proceedings of data visualization 2000*; p. 73–82.
- [38] Park J, Chang D. A hierarchical skeleton-based shape description. *KSPC '99*. Korea: POSTECH; 1999. p. 597–600.
- [39] Copson ET. Metric spaces. Cambridge: Cambridge University Press; 1968.
- [40] Santini S, Jain R. Similarity measures. *IEEE Trans Pattern Anal Mach Intell* 1999;21(9):871–80.
- [41] Hellerstein JM, Koutsoupias E, Papadimitriou CH. On the analysis of indexing schemes. *SIGMOD-SIGART symposium on principles of database systems*; 1997. p. 249–56.
- [42] Ullman JR. An algorithm for subgraph isomorphism. *J Assoc Comput Mach* 1976;23:31–42.
- [43] Messmer B, Bunke H. Subgraph isomorphism in polynomial time. Technical report TR-IAM-95-003; 1995.
- [44] Efthimiadis EN. Interactive query expansion: a user-based evaluation in a relevance feedback environment. *J Am Soc Inf Sci* 2000;51(11): 989–1003.
- [45] Kohonen T. Self-organizing maps. Springer series in information sciences. Berlin: Springer; 1995.



Natraj Iyer obtained his BE (1997) in Production Engineering from the University of Bombay, India. Thereafter he worked as a Research Engineer at the Industrial Design Centre, IIT Bombay. He obtained his Masters in Mechanical Engineering (2000) from University, West Lafayette IN, where he is also currently a PhD candidate. His research primarily focuses on developing the architecture and new shape representations for 3D shape searching. His other interests are in

Multi-scale Design and Manufacturing, Information Technology, and Knowledge Management.



Subramaniam Jayanti received his BTech degree in Civil Engineering (1999) from the Regional Engineering College, Warangal, India and MS (2001) degree in Mechanical Engineering from Purdue University, Indianapolis. He is currently pursuing his PhD in Mechanical Engineering, at Purdue University, West Lafayette. His research is primarily in the areas of Engineering Design, Engineering Information Retrieval and Data Mining. He is also interested in design software for enabling mass customization of products.



Kuiyang Lou is a Research Associate at Purdue University. His research interests include scientific and engineering problem solving, CAD/CAM, and computing and information systems in engineering. He received a BS in Material Science and Engineering from Changsha Institute of Technology, China (1987), and an MS in Material Science and Engineering from Tianjin University, China (1994). He continued his study at Purdue University and received an MS in Computer Science and a PhD in Mechanical Engineering both in 2004.



Yagna Kalyanaraman obtained his BE in Mechanical Engineering (2001) from Anna University, India. He is currently pursuing an MS in Mechanical Engineering at Purdue University, West Lafayette. His work focuses on developing algorithms for obtaining feature vectors for 3D shape searching. His interests include Engineering Design, Computer Graphics, and Artificial Intelligence.



Karthik Ramani is a Professor in the School of Mechanical Engineering at Purdue University and the director of PRECISE. He earned his BTech from the Indian Institute of Technology, Madras (1985), an MS from The Ohio State University (1987), and a PhD from Stanford University (1991), all in Mechanical Engineering. He was awarded the Research Initiation and CAREER Awards from NSF, and the Outstanding Young Manufacturing Engineer Award from SME. In 2002, he was

recognized by Purdue University through a University Faculty Scholar Award. His research interest includes development of frameworks and architectures for advice during early design, product shape search, next generation product configuration, and mass customization.