Proceedings of DETC' 03
ASME 2003 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
Chicago, Illinois, USA, September 2-6, 2003

DETC2003/CIE-48180

A RECONFIGURABLE 3D ENGINEERING SHAPE SEARCH SYSTEM PART I: SHAPE REPRESENTATION

Natraj Iyer
Yagnanarayanan Kalyanaraman
Kuiyang Lou
Subramaniam Jayanti
Karthik Ramani
Purdue Research and Education Center for Information Systems in Engineering (PRECISE)
School of Mechanical Engineering
Purdue University, West Lafayette IN 47907-2024, USA

ABSTRACT

This paper presents an approach for a reconfigurable shape search system for 3D engineering models using a client-server-database architecture. The current paper focuses on the server functionality, while a subsequent paper will focus on the database issues. The server takes the shape query as input from the client and converts it into feature vectors and a new skeletal graph representation which we have developed. The algorithms such as voxelization, skeletonization, and skeletal graph extraction for accomplishing these are described in detail. The principal advantages of the skeletal graph representation are that (i) it preserves the 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, while sensitive enough to capture the major features of a shape. Our representation is also synergistic with the human cognitive representation of shape. The results indicate that the skeletal graph is considerably smaller than the B-Rep graph even for complicated shapes.

1. INTRODUCTION

Designers spend about 60% of their time searching for the right information, which is rated as the most frustrating of engineers' activities [1]. Often designers have to make "assumptions" while designing, which may lead to problems at a later stage of the design. Such unforeseen errors can lead to allocation of scarce resources for solving unanticipated problems or putting out "fires." In the product development context, fire-fighting describes the unplanned allocation of engineers' time and other resources to fix problems discovered late in a product's development cycle. Re-learning from errors is a costly method of learning. Nevertheless, fire-fighting has become the *de facto* process for developing new products in industry [2]. For example, in engineering, it is conservatively estimated that more than 75% of design activity comprises case-based design – i.e. reuse of previous design knowledge to address a new design problem [3]. However, physical inventories such as tooling and associated knowledge are often not located or reused resulting in significant losses. Twenty-first century product development will mainly be driven by the need to reduce cycle times and costs. Design and associated knowledge reuse is the key to reducing new product development time as well as fire-fighting.

Engineering design and manufacturing has progressed extensively from 2D to 3D in the last decade. This includes development of large-scale computer-aided design (CAD) and manufacturing software used throughout a product's lifecycle [4]. Advances in 3D graphics hardware have also contributed to the widespread use of 3D models. The use of 3D models is especially high in net-shape manufacturing processes such as injection molding and casting. In these processes, it is critical to visualize and understand the part accurately before building expensive tooling, including dies and molds. For example, approximately 66% of CAD modeling in the mold making industry was being done in 3D in 2001. This was expected to increase to 80% in 2003 [5]. As a result, the shapes of products and associated tooling have increased considerably in complexity and number, contributing to the *3D model explosion*. For example, a cursory study reveals that the Boeing 777 has about 3 million parts, of which about 132,500 parts are uniquely engineered [6, 7].

A significant amount of knowledge generated during the design and manufacturing is attached to the 3D model. Most of this knowledge (context or meta-data) is geometry-related

(manufacturing process details) or even geometry-dependent (analysis results). The simplest form of searching is by keyword in filenames, part numbers, or context attached to the CAD model. Current Product Lifecycle Management (PLM) systems allow part-name based searching of 3D models [8]. However, this method is not robust primarily for 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.

The Internet has facilitated newer business models along with geographically-distributed design and manufacturing. Hence, 21st century designers may not be familiar with design history and context, making a keyword-based search an unattractive option. Although large parts of our brains are devoted to the processing of shape, the visual channel has not been exploited for information retrieval in engineering.

Thus, 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, designers relate products to 3D shapes more than to meta-data and, hence, only a shape-based system can provide an answer when other methods fail.

Within the past few years, research in 3D shape searching has been recognized as important. There have been a limited number of attempts in the search for 3D CAD models. Attempts made at developing search techniques for 3D models, to the best of our knowledge, are direct approaches based on a single, predefined, and often oversimplified model of similarity. However, 3D search techniques focused on realistic engineering models have very different considerations than do search techniques for 3D geometry in other domains. The motivation for this research is based on enabling new possibilities and associated applications, and specifically

1. Allowing for the retrieval of past knowledge and previous parts that are similar.
2. Enabling better quotation support, including reducing risk and improving time of response.
3. Being able to locate suppliers through neutral secure locations where geometry-related manufacturing capabilities can be determined without revealing part geometry.
4. Being able to overcome limitations of knowing part history, part names, project names, and context that are often forgotten.
5. Using distributed repositories at other locations whose histories one may not be familiar with.
6. Reducing the search time for parts and even finding unknown relations or knowledge from earlier projects across the extended enterprise.
7. Enabling the association of context by relating the shape and text-based index structures in multi-dimensional queries.

The S3 system developed at TU-Munich [9] attempts to solve the shape search problem in 2D by providing different algorithms for shape searching. It was also found that that similarity depends not only on the application domain but also on the user. Based on our discussions with personnel from design, manufacturing, and cognitive psychology, we confirmed that similarity is very much an application-driven and user-perceived notion. A universal similarity metric is impossible to formulate. Our discussions also helped us generate a needs checklist for a shape 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. Although these two statements appear conflicting, we believe that there is a trade-off between perturbations in shape and similarities 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 we store local and global shape attributes or information.
6. *Query Interface*: The query interface must allow for multiple modes of inputs – quick and approximate shape creation through sketching, selecting a query model from a cluster of parts, and construction of a user-driven similarity metric.

Constructing a shape search system in itself consists of the following sub-tasks:

a. Representations of a shape in the database, and
b. Database indexing, retrieval, and clustering

This paper discusses the representation task at length, while the indexing task is presented in a subsequent paper. Section 2 discusses *Related Work* and *Unaddressed Issues;* Section 3 presents our *Approach* and associated research issues; and finally, Sections 4 and 5 present *Results* and *Conclusions* respectively.

2. RELATED WORK

A significant amount of work has been done in the past two decades on text-based document retrieval [10]. More recently content-based retrieval systems have been developed for images [11], audio [12] and video [13]. One of the first search systems

focusing on the mechanical engineering domain to match 2D images was developed by Berchtold and Kriegel [9]. Shape-based retrieval of 3D data has been an area of research in disciplines such as computer vision [14], mechanical engineering [15], and molecular biology [16]. Almost all of the related methods for matching 3D shapes decompose a shape into a *signature*. 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. [17, 18, 19]. Primary limitations with these methods are that they fail to capture the specific details of a shape and are not very robust. Often they fail to discriminate among dissimilar shapes.
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. This method is also not always robust especially for engineering shapes, as seen in Figure 1. Clearly, similar parts may not always fit in the same spheres [20, 21, 22].

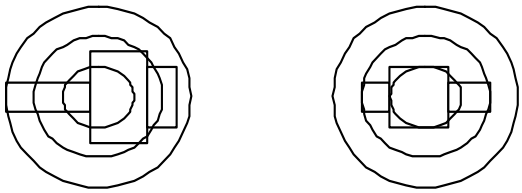


Figure 1 Spherical harmonics for engineering shapes. Similar parts need not always have similar bounding spheres.

3. **Statistics/Probability-based**: A probability distribution usually sampled from a set of representative points that measure geometric properties of a model is used as a signature. Osada et al. [23] and Ip et al. [24] use shape functions, predefined functions which capture geometry, such as lengths, angles, areas, or volumes. A random sampling of points from the 3D model is used to generate samples for constructing a shape distribution. Ankerst et al. [25] use shape histograms to approximate and search for a 3D model. The bounding sphere for the 3D model is divided into a number of sectors, depending on the accuracy desired. A histogram is constructed based on the volume of the 3D model lying in each sector. A limitation with statistics/probability type approaches is that fairly detailed models are required for finding similar parts. Besides, they can only calculate global part similarity and not local part similarity.
4. **3D Object Recognition-based**: Significant work has been done by the computer vision community to detect specific 3D models from a scene. Some approaches used are Aspect Graphs [14], Spin Images [26, 27], Extended Gaussian Images [28], and Geometric Hashing [29]. Limitations with these approaches are that they are not robust and consume a lot of time and storage space.
5. **Graph-based**: These approaches are based on the graph representation for a 3D model. One approach uses a subgraph isomorphism algorithm for matching B-Rep graphs [30, 31]. Regli et al. use eigenvalues of a model signature graph (MSG) constructed from the B-Rep graph as feature vectors [32]. Another approach constructs a graph for all patches in the 3D model and compares these graphs by reducing the graph to a tree [33]. Hilaga et al. use a geodesic distance function to construct a multi-resolution graph that captures the topology of the part. The similarity among parts is calculated by comparing these graphs [34]. A limitation with these methods is that B-Rep graphs are quite complex and are sensitive to very small variations in shape. A simple engineering part such as a bracket usually has on the order of 30 nodes. Moreover, subgraph isomorphism is NP-complete. Graph-based approaches are thus intractable for realistic engineering parts.
6. **Feature Recognition-based**: Ramesh et al. [35] decompose a part into cells which are further processed to identify machining features. Next, spatial and dimensional relationships among features are used to calculate similarity among parts. Ascher et al. [36] simplify a part into a maximal feature subgraph (MFSG), which consists of features that cannot be subsumed by other features. Maximal subgraphs of components are matched to obtain a similarity measure. The limitations with feature recognition techniques are that they cannot identify complex features and often require human interaction.
7. **Group Technology-based**: Group Technology (GT) coding is used to reduce a part into a representative code based on characteristics such as shape, material, dimensions, surface finish, etc. A two-step method was developed in [37] to compare the similarity among parts. The first step evaluates global similarities among parts by comparing their GT Codes. The second stage evaluates more specific part characteristics. A Type Abstraction Hierarchy (TAH) was proposed in [38] as an alternative to GT. The shape is differentiated from the application domain unlike in GT. Thus, similar parts can be retrieved across all applications. Limitations with coding approaches are that only superficial shape information is captured. Further, it is a human-error prone approach.

2.1. Unaddressed Issues

The previous approaches are bipolar - too granular (categories 4, 5, 6) or too lumped (categories 1, 2, 3, 7). Granular approaches represent the shape in great detail thus making the search intractable, while lumped approaches combine all shape characteristics into a single quantity. Furthermore, they also overlook the issues listed below:

1. Most systems do not address the problem of reducing the semantic gap.
2. Most systems are not scalable with respect to database disk I/O time. Furthermore, database index structures for 3D shape searching have not been fully explored.
3. The input to most systems is a detailed model or a detailed drawing, which is unrealistic in an engineering design situation. The designer, depending on the stage of the design, may not know the detailed shape of the model he/she is searching for or wants to learn about.
4. Feature vectors and similarity metrics are predefined within most systems, thus making it impossible for the user to define a custom similarity measure.
5. Importantly, an application domain is not considered in most methods. Most approaches search for 3D objects such as dinosaurs, bananas, or cars for similarity comparisons. Searching for engineering shapes requires a different consideration of shape descriptors, feature vectors, and user interfaces. Furthermore, the notion of similarity in engineering is cognitively different than that for other domains [9].

3. A PROTOTYPE FOR EXPERIMENTATION

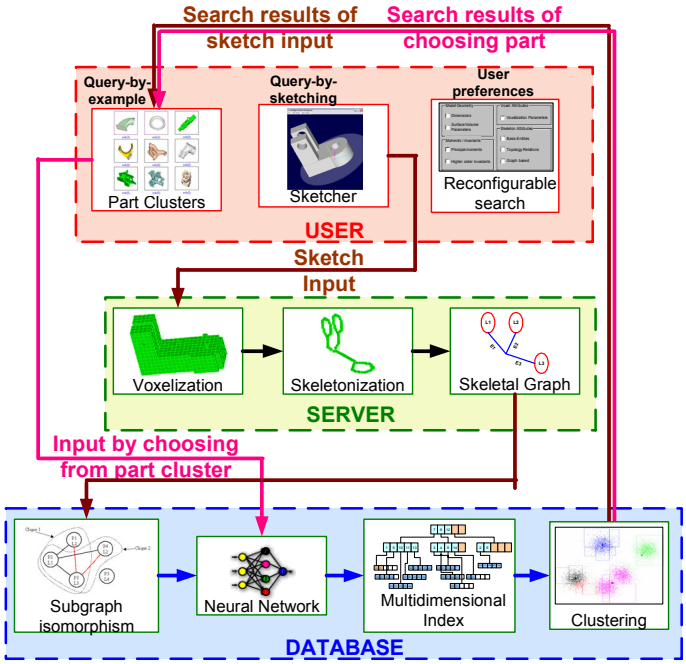


Figure 2 Architecture showing different query processes and the components of the search system

As seen in Figure 2, the search system consists of a Client-Server-Database architecture. The Client end consists mainly of the user interface, which enables query creation and display of results. The Server side essentially takes in the shape input from the Client and converts it into multiple representations (voxel, skeleton, feature vectors, and skeletal graph), which are then stored in the Database. The primary focus of this paper is to discuss the Server side functionality, while the Database and associated functions are explained in a subsequent paper. A brief overview of the Client user interface is provided in the next section.

Our approach makes the following major contributions:

1. Customizable user query construction.
2. Capturing only the "essential" features in a shape.
3. Representing shape as a combination of both geometry and topology.
4. Multi-level and multi category representation of shapes.
5. Accuracy and computational efficiency.

3.1. User Interface

The function of the user interface is to allow a user to create a custom query, either by example or by sketching, and to display the search results. The user interface consists of three main components: (1) Quick Shape creation interface, (2) Cluster Map interface, and (3) Feature vector choice interface. The quick shape creation interface is a web-based collaborative system implemented in Java™ and uses an ACIS modeling kernel [39]. The geometry of the shape is stored as a B-Rep model and is passed on to the Server side. The Cluster Map and Feature Vector Choice Interfaces are described in detail in Part II of the paper. The primary function of the user interface is to enable the following user queries:

1. *Query-by-example*: A query is initiated by choosing a part most similar to the one the user desires. Part choice will be enabled by the Cluster Map Interface that will allow a user to drill down through clusters of similar parts to find a part to start searching with.
2. *Query-by-sketching*: The user initiates a query by sketching an approximate 3D mock-up of the model he/she is interested in searching for. Alternatively, a user could also specify the location of a CAD file at the Client end.
3. *Feature Vector Choice Interface*: The user can customize the feature vectors to be used in the search. This enables the user to "reconfigure" the search at any time.

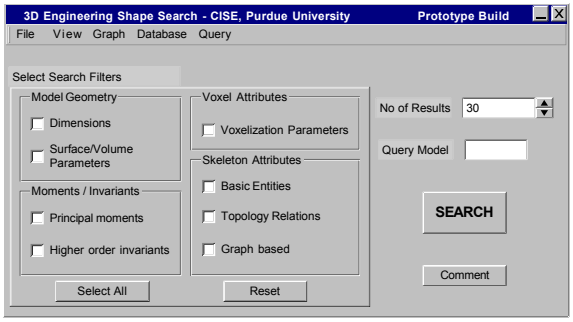


Figure 3 Feature Vector Choice Interface

3.2. Server Side

Prior to converting a 3D model into a voxel model, the model is normalized into a canonical form independent of position and scaling. This will be used in calculating moment invariants which by nature are independent of orientation. The

model is made invariant to position by translating it to its centroid. Further, the model is made invariant to scale by scaling it with respect to a fixed volume V. The calculation of moment invariants is discussed in Section 4.5.

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 represent the synthetic model within the discrete model space. The terminology used in voxelization is defined below [40].

Let the continuous 3D space be marked by $\mathbb{R}^3$ and the discrete 3D voxel space, which is a (regular) array of grid points by $\mathbb{Z}^3$. A voxel, or the region contained by a 3D discrete point *(x,y,z)* is the continuous region *(u,v,w)* such that:
$x - 0.5 < u \le x + 0.5$, $y - 0.5 < v \le y + 0.5$, $z - 0.5 < w \le z + 0.5$.
This assumes that a voxel occupies a unit cube centered at the grid point *(x,y,z)* and the array of voxels tessellates $\mathbb{Z}^3$. Voxels that belong to the model have a value of 1, while the remaining voxels have a value of 0. A voxel has three kinds of neighbors as shown in section 3.2.3.

In recent years, a number of curve [41], surface [42] and polygon mesh [43] 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.

Efficient voxelization of CSG solids has been studied extensively, and approaches such as point sampling [44], volume sampling [45] and distance volumes [46] have been developed. It is difficult to voxelize a B-Rep solid because the interior is not explicitly defined. Except for the point-classification method proposed in [47], 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 easily be translated into a neutral B-Rep representation such as STEP [48].

3.2.2. Categorization of shapes

Most engineering shapes can be grouped into three categories as described below and in Fig. 4.

a. *Solid-like*: These shapes are frequently encountered in processes such as turning or forging. Solid-like shapes are shapes that can be reduced to a line skeleton. From here on, these shapes are referred to as "prismatic" shapes.

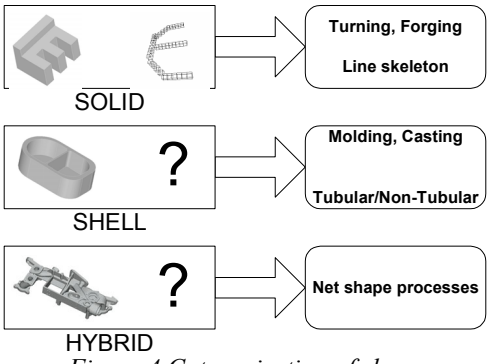


Figure 4 Categorization of shapes

b. *Shell-like*: These shapes are frequently made by net-shape processes such as molding or casting. Shell-like shapes are further classified as "tubular" and "non-tubular." Shell-like shapes cannot be reduced to a line skeleton because they are already in a skeletal (surface skeleton) form.

c. *Hybrid*: Hybrid shapes have some solid-like areas and some shell-like areas. They are usually manufactured by net-shape processes.

The results from the experimental prototype in the current paper are only for prismatic shapes. However, the same architecture would work for non-prismatic (Shell and Hybrid) parts as well.

3.2.3. Skeletonization

The notion of a skeleton was introduced by Blum [49]. 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. Each cell in the Voronoi diagram contains exactly one generating point and the locus of all points which are nearer to this generating point than other generating points. If the density of the boundary points of the 3D image (as generating points) 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 [50].

2. **Distance transform-based**: The original 3D image is converted into feature and non-feature elements. All points belonging to the boundary of the 3D image are assigned as feature elements. A distance map is then generated for the entire 3D image, where each element gives the distance to the nearest feature element. This is usually done by using distance criteria such as 4-neighbor (city block), 8-neighbor (chessboard), or (3, 4) chamfer distance. The local extrema on the distance map appear as ridges and are detected as skeletal points. Distance transformation can be executed in linear time as the number of voxels [51].
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 [52]. Points in the image are deleted if they satisfy deletion conditions that preserve topology. The advantages of thinning are that it preserves the general shape of the part and produces a one-voxel width skeleton.

We choose thinning as our preferred skeletonization method because it offers advantages over Distance transform and Voronoi-based skeletons. Distance transform skeletons satisfy "geometrical", 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", but does not always satisfy the "geometrical" requirements. However, for our application, topological correctness is more important than geometrical correctness. An overview of the thinning algorithm we use is explained in this section. We first present a background of 3D digital topology.

Let $x=(x_1,x_2,x_3)$ and $y=(y_1,y_2,y_3)$ be two points with integer coordinates in the 3D discrete space $\mathbb{Z}^3$ and let us consider the Euclidean distance

$$\|x-y\| = \sqrt{\sum_{i=1}^{3}(x_i-y_i)^2} .$$

Points x and y are said to be *6-adjacent* if $\|x-y\| \le 1$, *18-adjacent* if $\|x-y\| \le \sqrt{2}$ and *26-adjacent* if $\|x-y\| \le \sqrt{3}$. Let $N_j(p)$ denote the set of points *j-adjacent* to point p, for $j=6,18,26$ as shown in Fig. 5. The point p is said to be *j-adjacent* to the non-empty set of points X if there is a point $x \in X$, such that $x \in N_j(p)$. The sequence of distinct points $\langle x_0, x_i, ..., x_n \rangle$ is a *j-path* of length $n \ge 0$ from point x_0 to point x_n in a non-empty set of points X if each point of the sequence is in X and x_i is *j-adjacent* to x_{i-1} for each $1 \le i \le n$. A single point is a *j-path* of length 0. Two points are *j-connected* in the set X if there is a *j-path* in X between them. A set of points X is *j-connected* if any two points in X are *j-connected* in X.

The 3D binary (m,n) digital picture Π is a quadruple $\Pi = (\mathbb{Z}^3, m, n, B)$, where (m,n) is a pair of adjacencies ((m,n)=(6,26),(26.6),(6,18) or (18,6) are generally considered). Each element in $\mathbb{Z}^3$ is a voxel or a point of Π. A point in $B \subseteq \mathbb{Z}^3$ is called black point; a point in $\mathbb{Z}^3 \setminus B$ is called white point. Picture Π is finite if B is a finite set. Value 1 is assigned to each black point and value 0 is assigned to each white point. Equivalences classes of B under *m-connectivity* are called black *m-components* or objects. A singleton black component is called an isolated point. The *n-connectivity* relation and the white n-*component* can be defined in the same way. A finite picture has a unique infinite white component, which is called the background. A finite white component is called a cavity in a picture. A black point is a border point if it is *6-adjacent* to at least one white point (only for m=26 and m=18). Let us suppose that the number of holes H in a picture Π denoted as $H(\Pi)$ is determined. Denote by $\Theta(\Pi)$ and $C(\Pi)$ the number of objects (black components) and the number of cavities of Π respectively. Then, the 3D Euler characteristic $X(\Pi)$ is defined as $X(\Pi) = \Theta(\Pi) - H(\Pi) + C(\Pi)$.

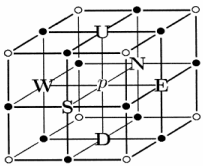


Figure 5 The three types of adjacencies for a voxel in $\mathbb{Z}^3$: Points in $N_6(p)$ are marked U,D,N,S,E *and* W. *Points in $N_{18}(p)$ but not in $N_6(p)$ are marked* "•". *Points in $N_{26}(p)$ but not in $N_{18}(p)$ are marked* "○". *Point 'p' is in all three sets* (from [52]).

Iterative thinning algorithms delete border points satisfying conditions of topology preservation. The entire operation is repeated until there are no more black points to be deleted. Thus, thinning is a reduction operation. A concept of a simple or deletable point was proposed in [53]. A simple point is one whose deletion does not change the topology of the picture. In general, 3D thinning algorithms are based on the following criterion for topology preservation proposed in [54].

Criterion: Let p be a black point of a 3D picture $(\mathbb{Z}^3, m, n, B)$. The point p is simple if and only if all of the following conditions hold:

a. p is m-adjacent to only one m-component of $N_{26}(p) \cap (B \setminus \{p\})$
b. p is n-adjacent to only one n-component of $N_{26}(p) \setminus B$
c. $X\left(\left(\mathbb{Z}^3, m, n, B \cap N_{26}(p)\right)\right) = X\left(\left(\mathbb{Z}^3, m, n, (B \setminus \{p\}) \cap N_{26}(p)\right)\right)$

Most thinning algorithms only operate with (26,6) pictures. Extensive research has been done on such pictures, and a number of conditions have been proposed for (26,6) pictures [55, 56]. We use a thinning algorithm proposed by Palagyi and Kuba [52]. The algorithm directly extracts a line skeleton without extracting a surface skeleton. Deletion conditions for a

voxel are assigned based on the 3x3x3 neighborhood of the voxel. Only black points can be changed, but white points cannot. Deletion conditions are described by a set of six masks or templates, and a black point is to be deleted if and only if its 3x3x3 neighborhood matches at least one of the six masks. Each mask is a characterization of simple points for 3D thinning. The masks are constructed according to the six deletion directions, **U**, **D**, **N**, **S**, **E** and **W**. For instance, the masks corresponding to the direction **U** as shown in Fig. 6 describe the condition to delete certain **U**-border points [52]. Additionally, all rotations (90°, 180° and 270°) around the vertical axis (according to the **U** and **D** directions) of the base masks **M1**–**M6** are also masks corresponding to the deletion direction **U**. Let Σ be the set of masks assigned to the deletion direction **U** (Σ contains the base masks **M1**–**M6** and their rotated versions).

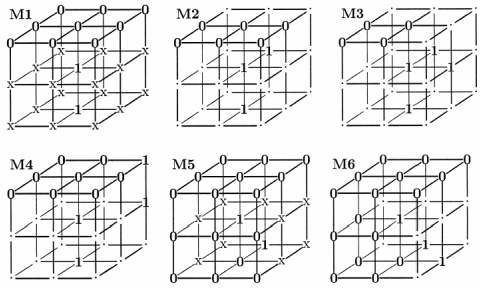


*Figure 6 Base masks for deletion direction **U*** (from [52]).

Deletion conditions for the other directions **D**, **N**, **E**, **S**, and **W** can be derived from the appropriate rotations and reflections of the masks in Σ. A proof for topology preservation of this algorithm can be found in [52].

3.2.4. Graph generation

The skeleton obtained after skeletonization is converted into a hierarchical skeletal graph for storing in the database. The skeletal graph can be viewed at different levels depending on the level of information to be retained. The main approaches in the past have been to convert the skeleton into a voxel graph [57] or into a skeleton tree [58]. In the voxel graph approach, each voxel is considered as a node, then the skeleton is pruned into two graphs, one for the topology and one for the geometry of the skeleton. In the skeleton tree approach, the 2D skeleton is processed to identify terminal and non-terminal nodes.

In our approach, we convert the skeleton into a skeletal graph made up of the following basic skeletal entities:

1. Vertex/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

The skeleton is converted into a higher level abstraction by representing it using the skeletal graph of basic skeletal entities. Each edge in the skeleton translates into an independent geometric entity giving the shape to the model in the physical space. Similarly, each loop represents a hole in the 3D model. For example, if a user wants to search for models with holes, this can be done by searching for models with holes. Thus, the skeletal graph is more physically relevant than the model signature graphs and feature relationship graphs, but simple compared to the voxel graph and the B-Rep graph.

A skeleton-marching algorithm is used to identify the basic entities from the skeleton (Fig. 7) and to construct the skeletal graph. First, the algorithm selects a starting voxel for marching. The algorithm then applies a set of masks to identify the types of neighboring voxels for each voxel in the skeleton. An ordinary voxel has two neighbors. When a voxel has more than two neighbors, it is a potential node candidate. Further processing is done at the region near these voxels to identify the correct nodes. Also, when a voxel has only one neighbor in its 26 voxel neighborhood, it is a terminal node. The set of voxels visited between two nodes forms an edge. This set of voxels forming an edge is then assigned an entity ID and stored in the entity stack. A curve-fitting subroutine is used to approximate the geometry of the edges. This can be used for exact matching in the search process.

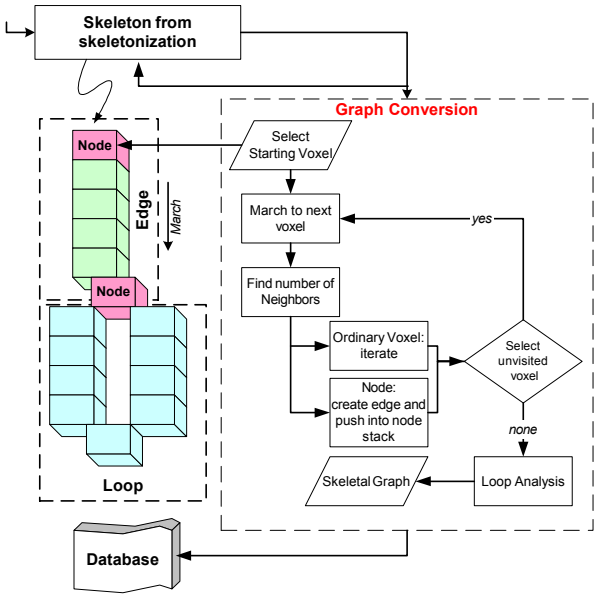


Figure 7 Skeletal graph extraction algorithm

When the marching algorithm revisits any node, it means that a loop has been traversed. The entity stack is then processed to identify the edge or set of edges which form the loop. The algorithm maintains a node stack and processes the branches at each node one by one. Once all the voxels are visited and the related edge/loop processing is done, the algorithm is complete. In the case of multiple loops in the same skeleton, some of the loops share one or many edges with other loops. The loops which do not share any edge or node with the other loops are called simple loops. The marching algorithm identifies all the simple loops. To get all the non-simple loops in the skeleton, the skeletal graph 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. This graph is stored in the database for a future search. Also, the graph avoids the possibility of zero loops.

3.2.5. 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 api_body_mass_pr() and stored as a feature vector. Further, the three second order moment invariants for the model are also stored as a feature vector. For every voxel in the model translation and scale invariant, the second order moments κ_{lmn} are calculated as described below.

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

where μ_{lmn} are central moments after translation, given by,

$$\mu_{lmn} = \int_{-\infty}^{\infty}\int\int (x-\hat{x})^l (y-\hat{y})^m (z-\hat{z})^n \rho(x-\hat{x}, y-\hat{y}, z-\hat{z})\, dx\, dy\, dz$$

$$l,m,n = 1,2,3,\ldots$$

The integrals above are approximated by summation of the contribution of every voxel to the moment. Since the characteristic function is invariant under rotation, the characteristic function of the matrix M of translation and scale invariant second order moments is *RST (Rotation-Scale-Translation) invariant*.

$$\mathrm{M} = \begin{bmatrix} \kappa_{200}-\Lambda & \kappa_{110} & \kappa_{101} \\ \kappa_{110} & \kappa_{020}-\Lambda & \kappa_{011} \\ \kappa_{101} & \kappa_{011} & \kappa_{002}-\Lambda \end{bmatrix}$$

After evaluating the characteristic function for this matrix, 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.

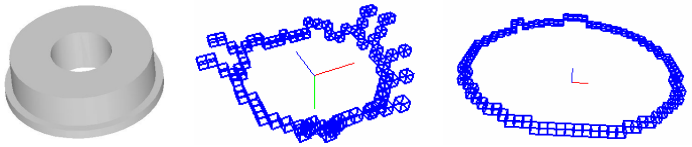
c. *Voxelization parameters*: The voxel size and the scaling factor obtained during the normalization process are stored as a feature vector.

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.

4. RESULTS AND DISCUSSION

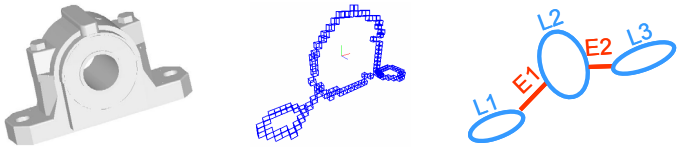
The voxelization, skeletonization, and skeletal graph extraction algorithms were implemented in C++ using ACIS as the geometric modeling kernel, and the user interface was implemented in Java. The prototype was tested only for prismatic parts since skeletonization methods for prismatic parts have been covered well in the literature. The same architecture can be used to search for non-prismatic parts. We are currently developing skeletonization algorithms for non-prismatic parts using voxel-based representations.

Figure 8 shows a bearing and its associated skeleton. It can be seen that the skeleton has unwanted appendages as noise for a voxel size of 1/8, while a voxel size of 1/32 produces a skeleton with no noise. The choice of the voxel size is an important decision. We observed that a voxel size of 1/16-1/32 seems to produce relatively noise-free skeletons for engineering parts.



(a) (b) (c)

Figure 8 (a) CAD model for a bearing, (b) Skeleton for a voxel size of 1/8 (c) Skeleton for a voxel size of 1/32



(a) (b) (c)

Figure 9 (a) CAD model, (b) Skeleton, and (c) Skeletal graph for a bearing block containing 182 faces

(a) (b) (c)

Figure 10 (a) CAD model, (b) Skeleton, and (c) Skeletal graph for a clamp containing 82 faces

Figures 9 and 10 illustrate the skeletal graphs for two models at a voxel size of 1/32. For the parts shown in Figs. 8 and 9, the B-Rep graph consists of 182 and 82 nodes respectively, while the skeletal graph consists of only 5 nodes for both parts. It is evident that the skeleton only captures the significant features of a 3D model. For instance in Fig. 9, minor details such as the bolt-heads are not captured. Thus, skeletal graph-based searching, using algorithms such as subgraph isomorphism, is more practical and more efficient than B-Rep graphs.

Furthermore, the skeletal graph data structure can be extended to store local attributes. This will enable searching for

local features in 3D models. We envision that users will soon be able to construct queries such as "Find me all 3D models having the selected feature."

5. CONCLUSIONS

In this paper, we describe an architecture and an efficient shape representation for shape-based searching of 3D models. A skeleton captures only the essential features of a 3D model. We show that the skeletal graph representation is simple and more efficient than B-Rep graphs for the purposes of searching. A prototype for prismatic parts was implemented and tested. This will be extended to include non-prismatic parts in the future. Searching based on local features can be made possible by extending the skeletal graph data structure. Additional feature vector attributes will be developed to improve the search capabilities.

ACKNOWLEDGEMENTS

The initial funding for this project came from the 21st Century Research and Technology Fund award from the state of Indiana. We acknowledge the University Faculty Scholar Award from Purdue University for Professor Karthik Ramani, which seeded this project. We also thank the Innovation Realization Laboratory at Purdue University for supporting Natraj Iyer. We thank Dr. Christoph Hoffmann, Department of Computer Sciences, Purdue University for his discussions that helped advance this project further.

REFERENCES

[1] Leizerowicz, W., Lin, J., and Fox, M.S., (1996), "Collaborative Design Using WWW," Proceedings of WET-ICE '96, CERC, University of West Virginia.
[2] Repenning, N.P., "Understanding Fire Fighting in New Product Development," Journal of Product Innovation Management, Vol. 18, pp.285-300.
[3] Ullman, D.G., (1997), T*he Mechanical Design Process.*, 2nd Edition, McGraw-Hill.
[4] http://www.enovia.com/, (2002), Enovia Corp, Charlotte NC.
[5] Christman, A., (2001), "It's all about the geometry," Moldmaking Technology Magazine, May 2001, Communication Technologies, Inc., PA.
[6] http://www.boeing.com/commercial/777family
[7] http://www.geocities.com/CapeCanavarel/6864/
[8] PTC Windchill, (2002), Parametric Technologies Corp., Boston MA.
[9] Berchtold, S., and Kriegel, H.P., (1997), "S3: Similarity Search in CAD Database Systems," Proceedings of SIGMOD'97, pp.564-567.
[10] Brin, S., and Page, L., (1998), "The Anatomy of a Large Scale Hypertextual Web search engine," Proceedings of the 7th International World Wide Web Conference (WWW7), Vol. 30, pp.107-117.
[11] Castelli, V., and Bergman, L., (2001), *Image Databases: Search and Retrieval of Digital Imagery.* John Wiley & Sons.
[12] Foote, J., (1999), "An Overview of Audio Information Retrieval," ACM Multimedia Systems, Vol. 7, No. 1, pp.2-11.
[13] Veltkamp,R.C., Burkhardt, H., and Kriegel, H-P., (2001), *State-of-the-Art in Content-Based Image and Video Retrieval*, Kluwer Academic Publishers.
[14] Cyr, C.M., and Kimia, B., (2001), "3D Object Recognition Using Shape Similarity-Based Aspect Graph," Proceedings of ICCV'01, pp.254-261.
[15] McWherter, D., Peabody, M., Shokoufandeh, A. and Regli, W., (2001), "Database Techniques for Archival of Solid Models," Proceedings of 6th ACM/SIGGRAPH Symposium on Solid Modeling and Applications, Ann Arbor, MI, pp.78-87.
[16] Ankerst M., Kastenmüller G., Kriegel H.-P., Seidl T., (1999), "Nearest Neighbor Classification in 3D Protein Databases," Proceedings of ISMB'99, Heidelberg, Germany, pp.34-43.
[17] Cybenko, G., Bhasin, A., and Cohen, K., (1997), "Pattern Recognition of 3D CAD Objects," Smart Engineering Systems Design, Vol. 1, pp.1-13.
[18] Rea, H., Corney, J., Clark, H., Pritchard, J., Breaks, M., and MacLeod, R., (2001), "Part Sourcing in a Global Market," Proceedings of ICeCE' 01, China Machine Press, Beijing.
[19] Elad, M., Tal, A. and Ar, S., (2001), "Content Based retrieval of VRML Objects - an iterative and interactive approach," Eurographics Multimedia Workshop, pp.97-108.
[20] Kazhdan, M., Chazelle, B., Dobkin, D., Finkelstein, A., and Funkhouser, T., (2002), "A Reflective Symmetry Descriptor," Proceedings of 7th European Conference on Computer Vision, Copenhagen, Denmark, pp. 642-656.
[21] Vranic, D., Saupe, D., and Richter, J., (2001), "Tools for 3D Object Retrieval: Karhunen-Loeve Transform and Spherical Harmonics," Proceedings of IEEE 2001 Workshop on Multimedia Signal Processing, pp. 293-298.
[22] Vranic, D., and Saupe, D., (2001), "3D Shape Descriptor Based on 3D Fourier Transform," Proceedings of the ECMCS01, Budapest, Hungary, pp. 271-274.
[23] Osada, R., Funkhouser, T., Chazelle, C., and Dobkin, D., (2002), "Shape Distributions," ACM Transactions on Graphics, Vol. 21, pp. 807-832.
[24] Ip, C.Y., Lapadat, D., Sieger, L., and Regli, W, (2002), "Using Shape Distributions To Compare Solid Models," Proceedings of ACM Symposium on Solid Modeling and Applications, pp.273-280.
[25] Ankerst M., Kastenmüller G., Kriegel H-P., Seidl T., (1999), "3D Shape Histograms for Similarity Search and Classification in Spatial Databases," Proceedings of 6th Symposium on Large Spatial Databases, pp. 207-226.
[26] Johnson, A. and Hebert, M., (1999), "Using Spin Images for Efficient Multiple Model Recognition in Cluttered 3D Scenes," IEEE PAMI, Vol. 21, pp.433-449.
[27] Ruiz-Correa, S., Shapiro, L. and Meila, M., (2000), "A New Signature Based Method for Efficient 3D Object Recognition," Proceedings of CVPR'00, SC.
[28] Horn, B.K.P, (1984), "Extended Gaussian Images," Proceedings of the IEEE, Vol. 72, pp.1671-1686.

[29] Lamdam, Y. and Wolfson, H.J., (1988), "Geometric Hashing: a General and Efficient Model Based Recognition Scheme," Proceedings of ICCV'88.
[30] El-Mehalawi, M., and Miller, R., (2003), "A Dtabase Sstem of Mechanical Components Based on Geometric and Topological Similarity. Part I: Representation," Computer Aided Design, Vol. 35, pp. 83-94.
[31] El-Mehalawi, M., and Miller, R., (2003), "A Database Sustem of Mechanical Components Based on Geometric and Topological Similarity. Part II: Indexing, Retrieval, Matching, and Similarity Assessment," Computer Aided Design, Vol. 35, pp. 95-105.
[32] Sun, T-L., Su, C-J., Mayer, R., and Wysk, R., (1995), "Shape Similarity Assessment of Mechanical Parts Based on Solid Models," Proceedings of ASME DFM Conference, Boston, MA, pp. 953-962.
[33] United States Patent Application No. 20020004710.
[34] Hilaga, M., Shinagawa, Y., Kohmura, T., and Kunii, T., (2001), "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes," Proceedings of SIGGRAPH' 01, pp. 203-212.
[35] Ramesh. M., Yip-Hoi, D., Dutta, D., (2001), "Feature Based Shape Similarity Measurement for Retrieval of Mechanical Parts", Vol. 1, pp. 245-256.
[36] Ascher, M., Marefat, M, and Fasse, E, (2001), "A Methodology for Automatic Retrieval of Similarly Shaped Machinable Components," IEEE International Conference on Systems, Man, and Cybernetics, Vol. 2, pp. 2840-2845.
[37] Iyer, S. and Nagi, R., (1997), "Automated Retrieval and Ranking of Similar Parts in Agile Manufacturing", IIE Transactions on Design and Manufacturing, Vol. 29, pp. 859-876.
[38] Srinivas, G.., Fasse, E., Marefat, M., (1998), "Retrieval of Similarly Shaped Parts from a CAD Database," IEEE International Conference on Systems, Man, and Cybernetics, Vol. 3, pp. 2809-2814.
[39] Agrawal, A., Ramani, K., and Hoffmann, C., (2002), "CADDAC: Multi-Client Collaborative Shape Design System with Server-Based Geometry Kernel", Proceedings of ASME DETC' 02 CIE Conference.
[40] Kaufman, A., (1987), "An Algorithms for 3D Scan Conversion of Polygons," Proceedings of Eurographics, G. Marechal, Ed., North Holland, Amsterdam, pp. 197-208.
[41] Cohen, D. and Kaufman, A., (1991), "Scan Conversion Algorithms for Linear and Quadratic Objects," Volume Visualization, ed. Kaufman, A., pp.280-301.
[42] Cohen, D. and Kaufman, A., (1995), "Fundamentals of Surface Voxelization", Computer Vision, Graphics and Image Processing, Vol. 56, pp.453-461.
[43] Huang, J., Yagel, R., Filippov, V., and Kurzion, Y., (1998), "An Accurate Method To Voxelize Polygonal Meshes", Proc. IEEE Symposium on Volume Visualization, pp. 119-126.
[44] Breen, D., (1991), "Constructive Cubes: CSG Evaluation for Display Using Discrete 3D Scalar Data Sets," in Proceedings of Eurographics' 91, pp.127-142.
[45] Wang, S., and Kaufman, A., (1994), "Volume Sampled 3D Modeling," IEEE Computer Graphics and Application, Vol. 14, pp.26-32.
[46] Breen, D., Mauch, S., and Whitaker, R., (1998), "3D Scan Conversion of CSG Models into Distance Volumes," Proceedings of IEEE/ACM Symposium on Volume Visualization, pp.7-14.
[47]Lee, Y., and Requicha, A., (1982), "Algorithms for Cmputing the Volume and Other Integral Properties of Solids," Communications of the ACM, Vol. 25, pp. 642-650.
[48] ANS US PRO/IPO-2000-042-1994, (1994), "Product Data Exchange Using STEP PDES Part 42: Integrated Generic Resources: Geometric and Topological Representation, US PDA.
[49] Blum, H., (1967), "A Transformation for Extracting New Descriptors of Shape," Models for the Perception of Speech and Visual Form, eds. E. Dunn, Cambridge, MA, pp. 362-380.
[50] Pudney, C., (1998), "Distance-Ordered Homotopic Thinning: A Skeletonization Algorithm for 3D Digital Images", Computer Vision and Image Understanding, Vol. 72, pp. 404-413.
[51] Niblack, C., Gibbons, P., Capson, D. W., (1992), "Generating Skeletons and Centerlines from the Distance Transform," Graphical Models and Image Processing, Vol. 54, pp. 420-437.
[52] Palagyi, K., and Kuba, A., (1998), "A 3D 6-subiteration Thinning Algorithm for Extracting Medial Lines," Pattern Recognition Letters, Vol. 19, pp. 613-627.
[53] Morgenthaler, D, (1981), "Three Dimensional Simple Points: Serial Erosion, Parallel Thinning and Skeletonization," Technical Report TR-1005, Computer Vision Laboratory, University of Maryland, College Park MD.
[54] Kong, T., and Rosenfeld, A., (1989), "Digital Topology: Introduction and Survey," Computer Vision, Graphics and Image Processing, Vol. 48, pp.357-393.
[55] Saha, P.K., Chaudhuri, B.B., (1994), "Detection of 3D Simple Points for Topology Preserving Transformations with Application to Thinning," IEEE Pattern Analysis and Machine Intelligence, Vol. 16, pp. 1028-1032.
[56] Ma, C.M., (1994), "On Topology Preservation in 3D Thinning," Computer Vision, Graphics, and Image Processing: Image Understanding, Vol. 59, pp. 328-339.
[57] Reinders, F., Jacobson, M., and Post, F., (2000), "Skeleton Graph Generation for Feature Shape Description," in Proceedings of Data Visualization 2000, pp. 73-82.
[58] Park, J., and Chang, D., (1999), "A Hierarchical Skeleton-Based Shape Description," KSPC '99, POSTECH, Korea, pp. 597-600.

