Dmitriy Bespalov

Ali Shokoufandeh

Department of Computer Science,
College of Engineering,
Drexel University,
3141 Chestnut Street,
Philadelphia, PA 19104

William C. Regli

Department of Mechanical Engineering and Mechanics,
College of Engineering,
Drexel University,
3141 Chestnut Street,
Philadelphia, PA 19104

Wei Sun

Department of Computer Science,
College of Engineering,
Drexel University,
3141 Chestnut Street,
Philadelphia, PA 19104

Scale-Space Representation and Classification of 3D Models

This paper presents a framework for shape matching and classification through scale-space decomposition of 3D models. The algorithm is based on recent developments in efficient hierarchical decomposition of a point distribution in metric space (p,d) using its spectral properties. Through spectral decomposition, we reduce the problem of matching to that of computing a mapping and distance measure between vertex-labeled rooted trees. We use a dynamic programming scheme to compute distances between trees corresponding to solid models. Empirical evaluation of the algorithm on an extensive set of 3D matching trials demonstrates both robustness and efficiency of the overall approach. Lastly, a technique for comparing shape matchers and classifiers is introduced and the scale-space method is compared with six other known shape matching algorithms.
[DOI: 10.1115/1.1633576]

1 Introduction

This paper presents a new technique for matching of 3D models using scale-space representations. We show how this matching algorithm can be used to encode an object classifier, which can effectively be applied to mechanical parts created by modern computer-aided design systems. Lastly, this paper addresses the important problem of establishing objective metrics and shared benchmarks for comparing shape matching and classification algorithms.

The problem of 3D object recognition is often formulated as that of matching configurations of features. Such configurations are often represented as vertex-labeled graphs, whose nodes represent 3D features (or their abstractions), and whose edges represent spatial relations (or constraints) between the features. The relations are typically geometric or hierarchical but can include other types of information. To match two 3D models means to establish correspondences between their constituting features. In this context, *features* are intrinsic properties of the 3D shape which may encompass local geometry and topology related to design or manufacturing operations. To evaluate the quality of a match, one defines an overall distance measure, whose value depends on similarities of models' graph representations.

Due to the importance of the 3D matching problem in a number of disciplines, there has been a growing interest in developing efficient algorithms for matching vertex-labeled graphs. Previous work on 3D matching (see Section 2) has typically focused on the problem of finding a correspondence between the vertices of two graphs. However, the assumption of direct graph matching is a very restrictive one, for it assumes that the primitive features (nodes) in the two graphs are easy to extract and are invariant in their level of abstraction. If there is a lesson to be learned from the feature recognition community [1,2], it is that features are not easy to extract and that they are not invariant across different domains. This paper presents a wholly new technique that goes beyond existing feature-based indexing.

Several existing approaches to the problem of constructing the graph representations of 3D models suffer from computational inefficiency and/or from an inability to handle perturbations in models. This paper seeks a solution to this problem while addressing drawbacks of existing approaches. Drawing on recently-developed techniques from the domain of spectral clustering, we have explored an *efficient* method for mapping a 3D model to a rooted tree. This mapping not only simplifies the original 3D model representation, but it *retains important information* about both local (features) as well as global structure of the model.

Matching of 3D models can now be reduced to the much simpler problem of matching rooted trees using a *recursive structural* similarity framework. We consider one such similarity measure, based on a dynamic programming framework, and show that the framework realizes the desired matching between components of the original 3D models. The result is a more efficient and more stable approach to 3D matching.

2 Related Work

Our research aims to bring information retrieval to CAD databases, enabling them to have indexing and query mechanisms like those beginning to be found in multimedia databases and knowledge management systems.

2.1 Comparing Solid Models. The brief literature in this area consists of results from engineering, computer science and, in particular, computer vision communities. Elinson et al. [3] and Cicirello and Regli [4–6] examined how to develop graph-based data structures to capture feature relationships and create heuristic similarity measures among artifacts. Work in [7,8] examined manufacturing feature-based similarity measurement. Elsewhere, automatic detection of part families [9] and topological similarity assessment of polyhedral models [10] has been examined.

Work of McWherter et al. [11–13] examined other graph-based methods for model comparison, including spectral analysis. These

Contributed by the Computer Aided Product Development (CAPD) Committee for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received August 2003; revised manuscript received October 2003. Guest Editor: V. Shapiro and G. Elber.

techniques can match models based on both topology (i.e., solid model boundary representation) and features (i.e., machining or design features).

Historically GT (Group Technology) coding was the way of indexing parts and part families [14], this facilitated process planning and cell-based manufacturing by imposing a classification scheme (a human-assigned alphanumeric string) on individual machined parts. While there have been a number of efforts to automate the generation of GT codes [15–19], none have been fully transitioned to commercial practice.

2.2 Comparing Shape Models. The computer vision and computer graphics research communities has typically viewed shape matching as a problem in 2D. This has changed in the past several years with the ready availability of 3D models (usually meshes or point clouds) generated from range and sensor data [20]. A considerable body of work has emerged to interrogate acquired datasets; we briefly review some of this work.

Thompson et al. [21,22] reverse engineered designs by generating surface and machining feature information of range data collected from machined parts. Hilaga et al. [23] present a method for matching 3D topological models using Multiresolution Reeb graphs. A Reeb graph is a skeletal structure defined by a continuous scalar function operating on an object. It shares properties similar to medial axes (in 2D) and medial surfaces (in 3D); however, Reeb graphs avoid the degeneracies that minor surface perturbations can cause to medial structures. In this work, the Reeb graphs are used to assess distances among a set of pre-categorized shape models (i.e., airplanes, cars, animals, etc). Other beneficial properties of their approach include invariance under certain transformations and robustness under variations in the quality of the models.

Osada and Funkhouser et al. [24–26] developed a method that creates an abstraction of the 3D models as probability distribution of samples from a shape function acting on the model. Specifically, measure the similarity between two models by measuring the similarity between their shape distributions. Shape distributions are generated by random sampling of points on the surface of the model. In their paper, they empirically study five different shape functions and conclude (experimentally) that a function they call $D2$ (which measures the distance between two random points on the surface of a model) results in the best shape classification method. Their database is a set of 130 shape models in VRML format gathered from the Internet. Recent work by the same investigators includes results of using shape harmonics and symmetry for shape matching [27–29]. Elad et al. [30] introduced both iterative and interactive approach to the problem of searching a database of 3D models in VRML format.

Ip et al. [31,32] propose several extensions to the basic shape distribution method, most significant of which is a method that uses machine learning to enhance recognition of desired classes. Based on training data in the form of classified models, a weighted distance is created using a set of shape distributions acquired from the CAD model. These distributions capture the internal, external and global shape properties of the object. These are combined into a single distance measurement that is weighted to maximize the separation of the models across classes.

Cyr and Kimia [33] proposed a similarity measure for comparing two projected 2D views of 3D objects. In fact, they used the same 2D measure to decompose the viewing sphere of 3D objects in terms of groups of similar views that in turn can be used to generate the aspect graph characterization of a 3D objects. They showed the performance of their system for the task of 3D object recognition in computer vision, where the main objective is to identify the 3D pose of an object using a set of characteristic 2D views (view-based 3D object recognition). It is not clear how this framework can be generalized to measure the similarity of two distinct objects with similar components.

In the computer vision community, the problem of object recognition is often reformulated as that of matching feature graphs. Several researchers have developed algorithms that find one-to-one correspondences between graph nodes. Shapiro and Haralick [34] proposed a matching algorithm based on comparing weighted primitives (weighted attributes and weighted relation tuples) using a normalized distance for each primitive property that is inexactly matched. Kim and Kak [35] used a combination of discrete relaxation and bipartite matching in model-based 3-D object recognition. Pellilo et al. [36] devised a quadratic programming framework for matching association graphs using a maximal clique reformulation, while Gold and Rangarajan [37] used graduated assignment for matching graphs derived from feature points and image curves. Siddiqi et al. combined a bipartite matching framework with a spectral decomposition of graph structure to match shock graphs [38], while Shokoufandeh et al. [39] extended this framework to directed acyclic graphs that arise in multi-scale image representations. Hancock and his colleagues have also proposed numerous frameworks for graph matching, including [40].

For a recent survey on vision and graphics-based matching techniques, and their limitations, readers are referred to [41]. In general, shape matching-based approaches operate only on the gross-shapes of single parts (not applicable to considering assembly structures) and does not operate directly on the solid models or consider semantically meaningful engineering information (i.e., manufacturing or design features, tolerances). Retrieval strategies are usually based on a query-by-example or query-by-sketch paradigm. The Princeton 3D shape database that has been used in a number of these studies [23,24,42] contains mainly models from 3D graphics and rendering, and not any models that are specifically engineering, solid modeling or mechanical CAD oriented.

3 Feature Decomposition

During the last decade, hierarchical segmentation has become recognized as a powerful tool for designing efficient algorithms. The most common form of such hierarchical segmentations is the scale-space decomposition in computer vision. Intuitively, an inherent property of real-world objects is that they only exist as meaningful entities over certain ranges of scale. The fact that objects in the world appear in different ways depending on the scale of observation has important implications if one aims at describing them. Specifically, the need for multi-scale representation arises when designing methods for automatically analyzing and deriving information from real-world measurements.

In the context of solid models, the notion of scale can be simplified in terms of the levels for the 3D features. The notion of *feature* in this sense draws from the computer vision literature rather than the CAD literature. Namely, given an object $\mathcal{M}$, we are interested in partitioning $\mathcal{M}$, into k features $\mathcal{M}_1, \ldots, \mathcal{M}_k$, with $\mathcal{M}_i \cap \mathcal{M}_j = \emptyset$, for $1 \leq i < j \leq k$, and $\mathcal{M} = \cup_i \mathcal{M}_i$ subject to maximization of some coherence measure, $f(\mathcal{M}_i)$, defined on the 3D elements forming each $\mathcal{M}_i$. At a finer scale, each feature $\mathcal{M}_i$ will be decomposed into $j = 1, \ldots, k_i$ sub-features, subject to the maximization of some coherence measures.

There are three central components in the aforementioned process: the number of components at each scale of decomposition, k; the feature coherence function $f(.)$; and the number of scales of decomposition process, ℓ. In most pattern recognition applications, k is a control parameter. If models $\mathcal{M}$ and $\mathcal{M}'$ are topologically similar, the k major components at every scale should also be similar. The coherence function $f(\mathcal{A})$ will assign an overall metric to the quality of 3D elements participating in the construction of feature $\mathcal{A}$. Finally, the depth of decomposition will be controlled depending on the quality of a feature in comparison to all its sub-features. Specifically, assume $\mathcal{A}$ represents a feature at scale i, and $\mathcal{A}_1, \ldots, \mathcal{A}_j$, for $j \leq k$ represent its sub-features at scale $i+1$. The decomposition process should proceed to scale $i+1$ with respect to feature $\mathcal{A}$ if and only if $f(\mathcal{A}) \leq f(\mathcal{A}_1) + f(\mathcal{A}_2) + \ldots + f(\mathcal{A}_j)$. This simple criteria for expansion of

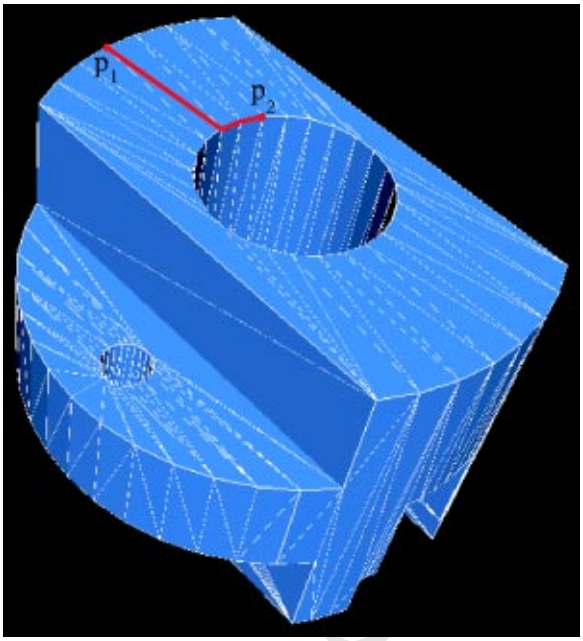

Fig. 1 Distance Metric for two points of a GOOD-COUPLING

scale-space at every feature has its roots in information theory. It is in fact motivated by linear form similar to entropy of feature $\mathcal{A}$ as opposed to its sub-features $\mathcal{A}_1, \ldots, \mathcal{A}_j$.

Our interest in scale-space feature decomposition is motivated by its ability to transform the problem of matching 3D models into hierarchical matching problems in rooted trees. Specifically, let $\mathcal{M}_1$, $\mathcal{M}_2$ denote two 3D models under coherence metrics $f_1(.)$ and $f_2(.)$, respectively. Ideally, we seek a scale-space decomposition that can map each 3D model to the set of primitive features, in which the two feature decompositions can be directly compared. However, in general, this is not possible without introducing unacceptable discrepancies in choosing $f_1(.)$ and $f_2(.)$.

We will, therefore, tackle the problem in two steps. First, we will seek a systematic scale-space decomposition of $\mathcal{M}_1$ and $\mathcal{M}_2$ that maps the elements in 3D models into correlated sub-features across a coarse to fine hierarchy of 3D features. Next, we will align the two scale-space feature decompositions, so their hierarchical representations can be directly compared using a hierarchical matching framework. Using these procedure, the problem of measuring coherence between $\mathcal{M}_1$ and $\mathcal{M}_2$ will be reduced to that of computing a mapping $\mathfrak{P}$ between the corresponding scale-space representations.

3.1 Decomposition Algorithm. We are given a 3D model $\mathcal{M}$ in polyhedral representation (in our experiments we used models in VRML format). Before we can proceed with the scale-space decomposition of model $\mathcal{M}$, we must choose a suitable distance function to capture the affinity structure of $\mathcal{M}$, i.e., we must define a distance between any two 3D points in $\mathcal{M}$. Let $\{v_1, \ldots, v_n\}$ denote the set of points in the 3D model $\mathcal{M}$. We will say that $\mathcal{D}$ is a metric function for $\mathcal{M}$ if, for any three points $v_1, v_2, v_3 \in \mathcal{M}$, $\mathcal{D}(v_1, v_2) = \mathcal{D}(v_2, v_1) > 0$, where $v_1 \neq v_2$, $\mathcal{D}(v_i, v_i) = 0$, and $\mathcal{D}(v_1, v_3) \leqslant \mathcal{D}(v_1, v_2) + \mathcal{D}(v_2, v_3)$. In general, there are many ways to define metric distances on a 3D model. One of the best-known metric functions is the shortest-path metric $\delta(.,.)$ (geodesic distance) on the triangulation of $\mathcal{M}$ with respect to points $\{v_1, \ldots, v_n\}$; i.e., $\mathcal{D}(u,v) = \delta(u,v)$, the shortest path distance on the triangulated surface between u and v for all $u, v \in \mathcal{M}$ (see Fig. 1). Observe that by construction the matrix $\mathcal{D}_{\mathcal{M}} = [\mathcal{D}(v_i, v_j)]_{n \times n}$ is symmetric, and the ith row (or column) in $\mathcal{D}$, p_i, is an n-dimensional vector, characterizing the distance structure of point v_i in model $\mathcal{M}$. We use All Pairs Shortest Path algorithm [43] for $\mathcal{D}_{\mathcal{M}}$ construction in our experiments.

The problem of decomposing model $\mathcal{M}$ into k most significant features $\mathcal{M}_1, \ldots, \mathcal{M}_k$ is closely related to k-dimensional subspace clustering (k-DSC). In k-DSC, we are given a set of distance vectors $p_1, \ldots, p_n$, and the objective is to find a k-dimensional subspace $\mathcal{S}$ that minimizes the quantity:

$$\sqrt{\sum_{1 \leqslant i \leqslant n} d(p_i, \mathcal{S})^2},$$

where $d(p_i, \mathcal{S})$ corresponds to the smallest distance between p_i and any member of $\mathcal{S}$. In practice, if $\mathcal{S}$ is given, then $\mathcal{M}_1, \ldots, \mathcal{M}_k$ can be computed using the principle components $\{c_1, \ldots, c_k\}$ of k-dimensional subspace $\mathcal{S}$ [44]. Observe that, these k vectors will also form a basis for $\mathcal{S}$. Specifically, p_i will belong to the feature $\mathcal{M}_j$ if the angle between p_i and c_j is the smallest among all basis vectors in $\{c_1, \ldots, c_k\}$, i.e., the point p_i will belong to the feature vector $\mathcal{M}_j$ iff the angle between these vectors is the smallest compared to all other basis vectors.

To construct the subspace $\mathcal{S}$, the optimal solution of k-DSC, we will use the technique commonly known as singular value decomposition (SVD) clustering [44]. First, observe that the symmetric matrix $\mathcal{D} \in \mathbb{R}^{n \times n}$ has a SVD-decomposition of the form

$$\mathcal{D} = U \Sigma V^T, \tag{1}$$

where $U, V \in \mathbb{R}^{n \times n}$ are orthogonal matrices and

$$\Sigma = \mathrm{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_n), \tag{2}$$

with $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma'_n > 0$, $\sigma_{n'+1} = \ldots = \sigma_n = 0, n' \leqslant n$. Let us define the order k compression matrix $\mathcal{D}^{(k)}$ of $\mathcal{D}$, for $k \leqslant n'$ as:

$$\mathcal{D}^{(k)} = U \mathrm{Diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0) V^T. \tag{3}$$

Then,

Theorem 1 [Eckart-Young].

$$\|\mathcal{D} - \mathcal{D}^{(k)}\|_2 = \min_{rank(H)=k} \|\mathcal{D} - H\|_2. \tag{4}$$

That is, matrix $D^{(k)}$ is the best approximation to $\mathcal{D}$ among all matrices of rank k. In fact, this result can be generalized to many other norms, including *Forbenius* norm:

Corollary 2 *For* $A \in \mathbb{R}^{n \times n}$, *let*

$$\|A\|_F = \left(\sum_{i,j} A_{i,j}^2 \right)^{1/2}, \tag{5}$$

then,

$$\|\mathcal{D} - \mathcal{D}^{(k)}\|_F = \min_{rank(H)=k} \|\mathcal{D} - H\|_F. \tag{6}$$

Next, assume $\mathcal{S}$ is the range of matrix $\mathcal{D}^{(k)}$ (the subspace spanned by the columns of matrix $\mathcal{D}^{(k)}$), and let c_j, for $1 \leqslant j \leqslant k$, denote the jth column of $\mathcal{D}^{(k)}$. Let $\mathcal{S}' \neq \mathcal{S}$ be any k-dimensional subspace of $\mathbb{R}^n$. For every $p_i \in \mathcal{M}$ let $q_i \in \mathcal{S}'$ be the closets point in $\mathcal{S}'$ to p_i. Define $\mathcal{Q} \in \mathbb{R}^{n \times n}$ with i^{th} column equal to q_i. Clearly, $rank\,\mathcal{Q} \leqslant k$. Using Corollary 2 we have:

$$\begin{aligned} \sum_{i=1}^{n} d(p_i, \mathcal{S}')^2 &= \sum_{i=1}^{n} d(p_i, q_i)^2 \\ &= \|\mathcal{D} - \mathcal{Q}\|_F^2 \geqslant \|\mathcal{D} - \mathcal{D}^{(k)}\|_F^2 \\ &= \sum_{i=1}^{n} d(p_i, c_i)^2 \geqslant \sum_{i=1}^{n} d(p_i, \mathcal{S})^2. \end{aligned}$$

Consequently;

Proposition 3. *The set* $\mathcal{S} = range(\mathcal{D}^{(k)})$ *is the optimal solution to* k*-DSC problem.*

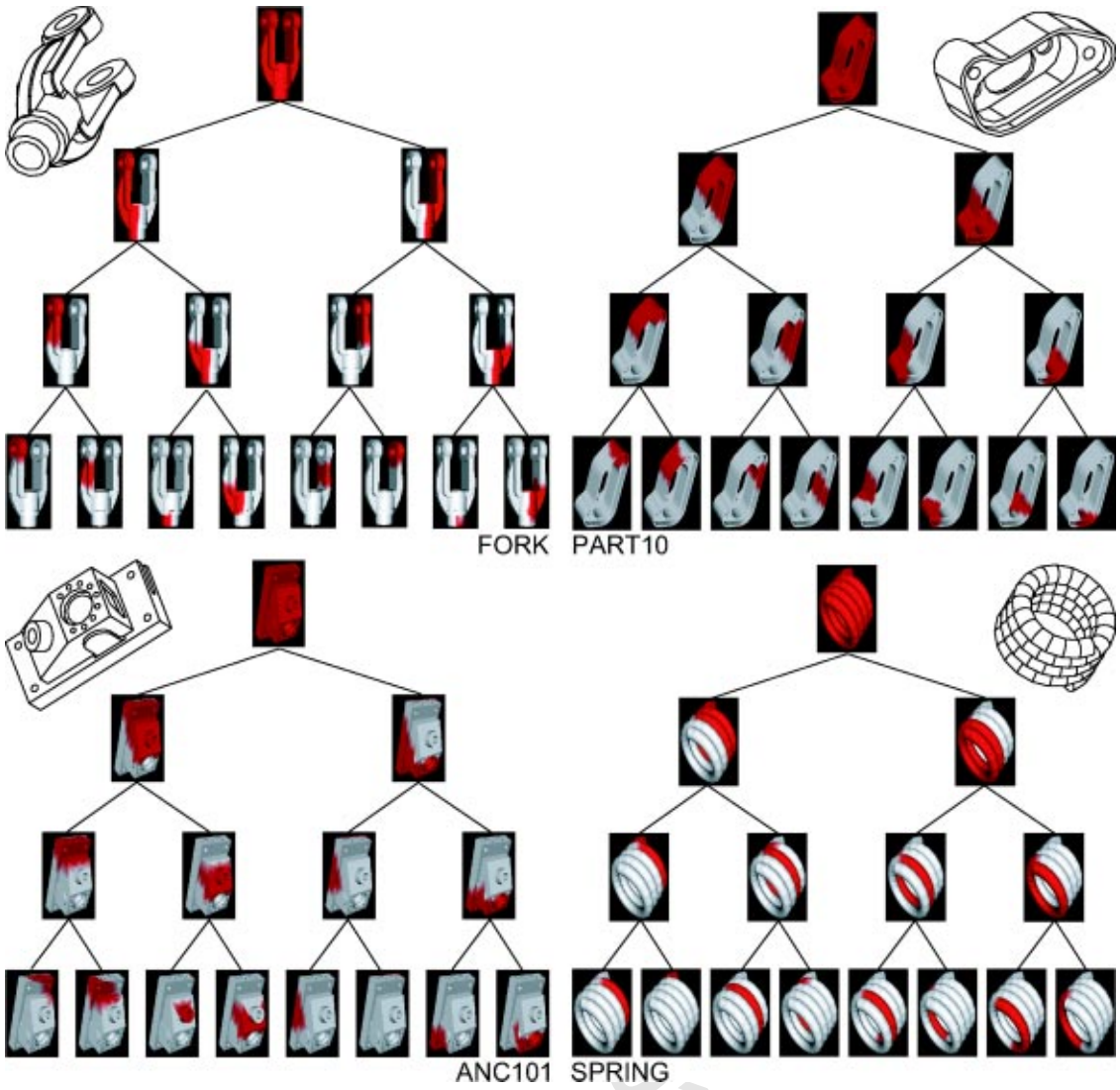

Fig. 2 Scale-space bisection of the sample models. Binary trees were obtained by recursively applying FEATURE-DECOMPOSITION($\mathcal{M}$,k). Red-colored regions of child nodes represent partitions that result in the bisection of the parent.

Algorithm 1 FEATURE-DECOMPOSITION($\mathcal{M}$,k)

1: Construct the distance matrix $\mathcal{D} \in \mathbb{R}^{n \times n}$.
2: Compute the SVD decomposition $\mathcal{D} = U \Sigma V^T$, with $\Sigma = \text{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$.
3: Compute the order k compression matrix $\mathcal{D}^{(k)} = U\text{Diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)V^T$.
4: Let c_j denote the jth column of $\mathcal{D}^{(k)}$, for $j = 1, \ldots, k$, and form sub-feature $\mathcal{M}_j$ as the union of points $p_i \in \mathcal{M}$ with $d(p_i, \mathcal{S}) = d(p_i, c_j)$.
5: Return the set $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$.

Algorithm 1 summarizes one phase of scale-space decomposition of $\mathcal{M}$ into its k most significant features, $\mathcal{M}_1, \ldots, \mathcal{M}_k$. Algorithm 1 returns the partitioning of $\mathcal{M}$ by placing each point p_i in $\mathcal{M}$ into one of the partitions $\mathcal{M}_j$, such that the angle between vector p_i and basis vector c_j corresponding to the partition $\mathcal{M}_j$ is minimized. The results of scale-space decomposition for several 3D models using the recursive application of FEATURE-DECOMPOSITION($\mathcal{M}$,k), for $k = 2$ is presented in Fig. 2.

The bottleneck of Algorithm 1 is the $O(n^3)$ SVD decomposition, for an $n \times n$ matrix. Polyhedral representation of a model provides us with planar graph of a 2D manifold. If we consider only neighboring vertices in the construction of the distance matrix $\mathcal{D}$, the number of non-zero entries in $\mathcal{D}$ would be at most $6n$ (due to planarity of the graph). Computing SVD decomposition for sparse matrices is much faster and takes $O(mn) + O(m\text{M}(n))$ [45]. Where m is the maximum number of matrix-vector computations required and $\text{M}(n)$ is the cost of matrix-vector computations of the form $\mathcal{D}x$. Since M is a planner map and $\mathcal{D}$ is a sparse matrix, $\text{M}(n) = O(n)$ and $m = O(n)$.

3.2 Controlling Decomposition Process. The decomposition process as presented in Section 3.1 does not allow for an explicit mechanism to stop indefinite break up of a feature into a point-cloud. Clearly, we could use a constant to control the decomposition depth of the feature trees, i.e., decomposition process will be stopped when a root branch in feature decomposition tree reaches a prescribed depth. In this section we will present a mechanism that will control the feature decomposition through constant measurement of coherence through out the decomposi-

tion paths. Intuitively, the use of this control mechanism will terminate the decomposition process only when all significant features are extracted.

We will assign a *measurement* to each iteration of Feature-Decomposition algorithm. Specifically, Let $\mathcal{M}$ be the original model's point set and E denote the set of all edges connecting points in $\mathcal{M}$. Assume in the decomposition process a feature $\mathcal{M}_1$ in $\mathcal{M}$ can be decomposed into sub-features $\mathcal{M}_2$ and $\mathcal{M}_3$ after bisection (e.g., without loss of generality assume we are bisecting feature $\mathcal{M}_1$). The coherence measurement for $\mathcal{M}_1$ is defined as follows:

$$f(\mathcal{M}_1)=\frac{\rho(\mathcal{M}_1)}{\lambda(\mathcal{M}_1)+\rho(\mathcal{M}_1)},$$

where

$$\rho(\mathcal{M}_1)=\sum_{\substack{(u,v)\in E,\\ u\in\mathcal{M}_2,\\ v\in\mathcal{M}_3}}\mathcal{D}(u,v),$$

denotes the distance of points across sub-features, and

$$\lambda(\mathcal{M}_1)=\sum_{\substack{(u,v)\in E,\\ u\in\mathcal{M}_2,\\ v\in\mathcal{M}_2}}\mathcal{D}(u,v)+\sum_{\substack{(u,v)\in E,\\ u\in\mathcal{M}_3,\\ v\in\mathcal{M}_3}}\mathcal{D}(u,v),$$

is the sum of distances within sub-features. Intuitively, the normalized ratio in $f(\mathcal{M}_1)$ will account for coherence of $\mathcal{M}_1$ as oppose to its constituting sub-features $\mathcal{M}_2$ and $\mathcal{M}_3$. Specifically, we say that bisection of $\mathcal{M}_1$ into $\mathcal{M}_2$ and $\mathcal{M}_3$ is good if $f(\mathcal{M}_1)$ is less than a prescribed threshold τ. If this threshold condition holds, the decomposition process continues for sub-features $\mathcal{M}_2$ and $\mathcal{M}_3$, otherwise it stops at $\mathcal{M}_1$. Figures 3(*a*) and 3(*b*) are the two examples of applying the decomposition control process to Models Part 10 and Spring, respectively. Figure 3(*c*) illustrates the decomposition process for Fork model (view is presented). Pictures of the point sets for three bisections are presented. Note that the technique extracted a pin on one side of the Fork (the pin is not present on the other side).

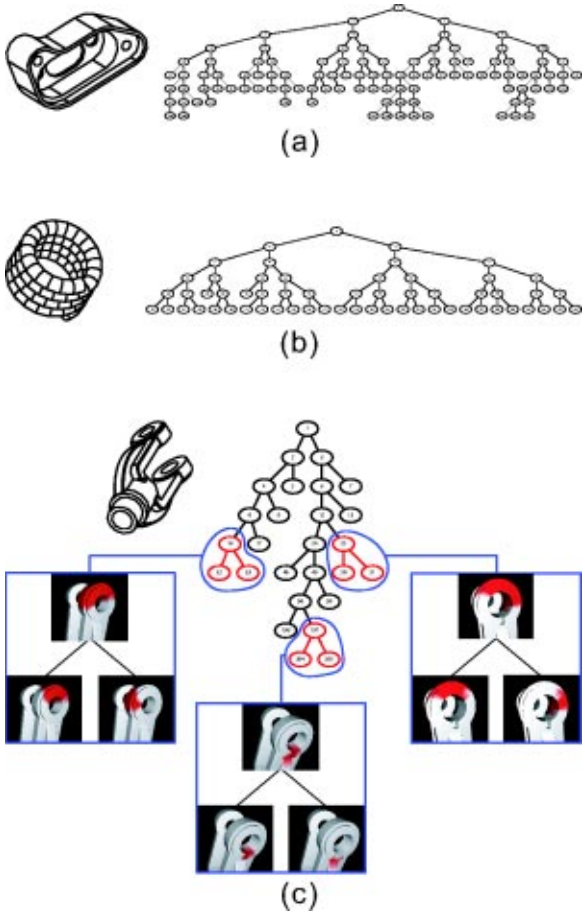

Fig. 3 Feature trees for the sample models. (*a*), (*b*) Models Part 10 and Spring respectively. Views of these models are presented on the left. (*c*) Illustrates decomposition process for Fork model (view is presented). Pictures of the point sets for three bisections are presented. Note that technique extracted a pin on one side of the Fork (pin is not present on the other side).

4 Scale-Space Matching

The scale-space decomposition of a 3D model $\mathcal{M}$ will give raise to a rooted undirected tree $T_{\mathcal{M}}=(V,E)$ in a natural way. The vertex set of this graph corresponds to the set of features produced by recursive application of Algorithm 1 to model $\mathcal{M}$. The edges of $T_{\mathcal{M}}$ will capture the decomposition relation between a feature and all its sub-features. Figure 2 shows several examples of degree-2 trees corresponding to 3D models. Using this construction the problem of comparing two 3D models $\mathcal{M}_1$ and $\mathcal{M}_2$ can be reformulated as computing a matching among the corresponding rooted trees T_1 and T_2.

Given 3D models $\mathcal{M}_1$ and $\mathcal{M}_2$, and their corresponding scale-space trees (SST's) T_1 and T_2, we will propose a matching algorithm based on the dynamic programming framework proposed by Wang [46]. Intuitively, the similarity between two features $u\in\mathcal{M}_1$ and $v\in\mathcal{M}_2$ is closely related to similarity of sub-features forming u and v. Specifically, the similarity of features u and v should be measured in terms of the similarity of subtrees $T_1(u)$ and $T_2(v)$ rooted at u and v.

The cost of matching $T_1(u)$ and $T_2(v)$ can be characterized as:

$$C(T_1(u),T_2(v))=C(F_1(u),F_1(v))+\gamma(u,v). \tag{7}$$

That is, the cost of matching trees rooted at features u and v is equal to the distance between the two features u and v ($\gamma(u,v)$) plus the cost of matching the forests $F_1(u)$ and $F_2(v)$, obtained from $T_1(u)$ and $T_2(v)$, after removing u and v, respectively. In order to deal with degenerate case, $u=\varnothing$ (and/or $v=\varnothing$) we will use:

$$C(T_1(u),\varnothing)=\gamma(u,\varnothing) \tag{8}$$

$$C(F_1(u),\varnothing)=\sum_y \gamma(y,\varnothing), \tag{9}$$

where the sum in Eq. (9) is over the roots of all trees in $F_1(u)$. Observe that for two features $u\in T_1$ and $v\in T_2$ to be matched, at some level some of their sub-features in $F_1(u)$ and $F_2(v)$ should have been matched. This phenomenon is captured in our formulation of $C(T_1(u),T_2(v))$ in Eq. (7), using term $C(F_1(u),F_2(v))$. To compute the cost of matching the two forests $F_1(u)$ and $F_2(v)$, we need to compute a maximum similarity matching among the roots of trees in these two forests. To this end, we will form a complete bipartite graph among the sub-features forming the roots of $F_1(u)$ and $F_2(v)$. Let $x\in F_1(u)$ and $y\in F_2(v)$ denote two such vertices, we will associate the following as the cost of matching x and y in aforementioned bipartite graph:

$$w(x,y)=C(x,\varnothing)+C(\varnothing,y)+C(T_1(x),T_2(y)). \tag{10}$$

Observe that the term $C(T_1(x),T_2(y))$ is the basis of the recursion in this dynamic programming framework. The chain of recursive calls will terminate at the primitive features forming the leaves of T_1 and T_2. Consequently, the cost of matching the forests $F_1(u)$ and $F_2(v)$ can be restated as

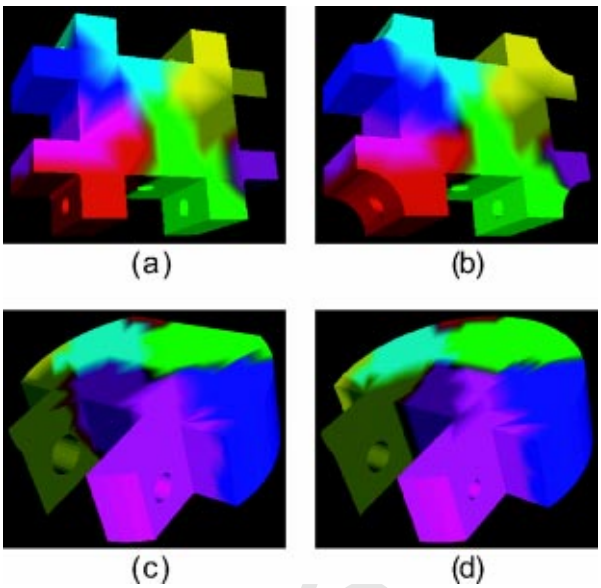

Fig. 4 Results of matching between two GOODPARTS (*a*, *b*) and two SWIVELS (*c*, *d*), with matched regions having similar colors. Most significant features are obtained for each model using FEATURE-DECOMPOSITION($\mathcal{M}$, *k*) and are used to construct binary trees T_1 and T_2. Match-Models(T_1, T_2) then generates a set of matched features which have been colored similarly for this figure.

$$C(F_1(u),F_2(v))=\frac{1}{2}\left(\sum_x C(x,\varnothing)+\sum_y C(\varnothing,y)\right)+\sum_{(x,y)\in\mathfrak{P}(u,v)} w(x,y). \quad (11)$$

The first two sums in Eq. (11) run over the roots of trees in $F_1(u)$ and $F_2(v)$, respectively, and the third sum runs over all matched pairs in bipartite matching of sub-features of u and v, $\mathfrak{P}(u,v)$.

To state our recursive algorithm we need to specify the cost function $\gamma(u,v)$ in Eq. (7). In our formulation of $\gamma(.,.)$ we will use the notion of topological similarity introduced by Hilaga et al. in [23]. In fact we will set

$$\gamma(u,v)=e^{-sim(u,v)}, \quad (12)$$

where $sim(u,v)$ is a convex combination of the form:

$$sim(u,v)=w\cdot\min(\alpha(u),\alpha(v))+(1-w)\cdot\min(\ell(u),\ell(v)), \quad \text{for} \quad 0\leqslant w\leqslant 1,$$

for $0\leqslant w\leqslant 1$, with functions $\alpha(.)$ and $\ell(.)$ representing the ratios of the area and the length of the corresponding features in the whole 3D model, respectively (for further details on the description of $\alpha(.)$ and $\ell(.)$ see [23]).

Finally, we can state the matching algorithm for two scale-space decompositions T_1 and T_2 corresponding to 3D models $\mathcal{M}_1$ and $\mathcal{M}_2$. The result of applying Algorithm 2 to two GOODPARTS parts is represented in Fig. 4. Please note that final distance function is not metric. It satisfies symmetry and non-negativity properties but it does not satisfy triangle inequality by construction. Even though we have not encountered this case in our experiments, it is theoretically possible that matching algorithm would return similarity values for the models such that they would not satisfy triangle inequality.

Based on time bound proposed by Zhang and Shasha in [46] for this framework, the number of iterations of Algorithm 2 algorithm is $O(n_1n_2hJ\log d)$, where d is the maximum degree of trees obtained in decomposition process, and n_1 and n_2 are the number of vertices in T_1 and T_2, respectively. Observe that n_1 and n_2 are much smaller than the initial sizes of $\mathcal{M}_1$ and $\mathcal{M}_2$. In the computations of running time, we will also have an additional multiplicative factor $\phi(\mathcal{M}_1,\mathcal{M}_2)$ that accounts for the complexity of vanishing any feature in models $\mathcal{M}_1$ and $\mathcal{M}_2$. Also, in order for our algorithm to meet the time bound, we precompute values of $\alpha(u)$ and $\ell(u)$ $\forall u\in T$. Complexities for computations of $\alpha(u)$ and $\ell(u)$ are linear in terms of the number of points in partition u.

5 Experimental Results

To demonstrate our approach to scale-space matching, we performed a set of matching experiments using a database of 3D models. For a given 3D model, its scale-space feature decomposition is first represented by a rooted, vertex-labeled tree,

Algorithm 2 MATCH-MODELS(T_1, T_2)

1: $C(\varnothing,\varnothing)\leftarrow 0$.
2: $\forall u\in T_1$ COMPUTE $C(T_1(u),\varnothing)$ and $C(F_1(u),\varnothing)$ using Eq. (8) and Eq. (9).
3: $\forall v\in T_2$ COMPUTE $C(T_2(v),\varnothing)$ and $C(F_2(v),\varnothing)$ using Eq. (8) and Eq. (9).
4: $\forall u\in T_1$ and $\forall v\in T_2$ do
COMPUTE $C(F_1(u),F_2(v))$ as in Eq. (11)
COMPUTE $C(T_1(u),T_2(v))$ as in Eq. (7).
5: Return the set of matched vertices:
$\cup_{(u,v)}M(u,v)$,
and the total cost of matching T_1 and T_2:
$C(\text{root}(T_1),\text{root}(T_2))$.

in which nodes represent *features* (obtained by recursive application of Algorithm 1) and edges capture the relationships between a feature and its sub-features. We will assume that each point u in the tree is labeled by two attributes, values of the functions $\alpha(u)$ and $l(u)$, defined in Section 4. These attributes were used in the Algorithm 2 to compute the distances between pairs of nodes.

Models with many features and parts lead to trees with many nodes. To reduce the size of the tree, we will impose a fixed bound on the maximum number k used in the Algorithm 1 for decomposition purposes. In practice, one can optimize on parameter k via an independent study of individual models. For the experiments, we compute the scale-space tree of every model using seven layers of decomposition (depth of the tree) with branching parameter $k=2$. This procedure results in a database of hierarchical trees, each representing a 3D model.

We tested our scale-space matching algorithm on the **CAD_40** dataset, a database of 40 3D solid models. The database consists of 10 different group of objects. A set of representative 2D views of 3D models is shown in Fig. 5. These models cover several non-trivial classes of mechanical artifacts and include parts from several industry and government sources. These models are available for download from the National Design Repository at URL `http://www.designrepository.org/SM03`.

To test our approach on the **CAD_40** dataset, we have computed distances between every 3D model in the database and each of the remaining database entries (the distance between a view and itself is always zero). These object distances are summarized in Fig. 6. The magnitudes of the distances are denoted by shades of gray, with black and white representing the smallest and largest distances, respectively.

Inter-group distances, shown along the main diagonal, are very close to zero. Also, distances between models from different groups are significantly larger than inter-group distances.

Based on the overall matching statistics, we observed that in almost 15% of the experiments, the closest match selected by our algorithm was not a member of the same group. We expect that with a more appropriate $\gamma(.,.)$ function, ensuring that for each model there exists a similar inter-group model, this error rate

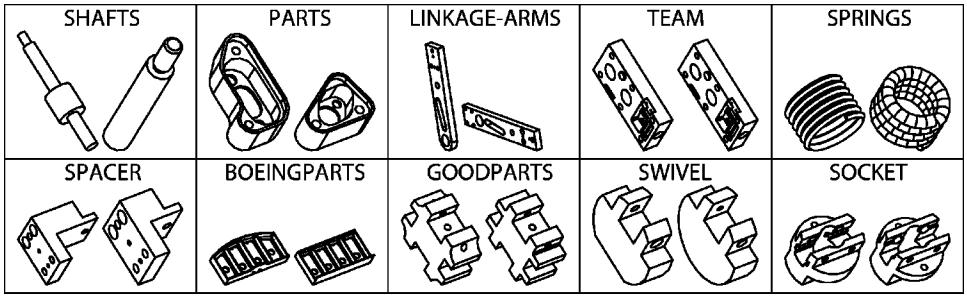

Fig. 5 Sample views from 10 Groups of 3D models in our database.

would decrease significantly. Finally, it should be noted that both the feature decomposition and matching procedures can accommodate sampling noise. This is due to the fact that the structure of trees corresponding to model are unaffected by sampling noise.

5.1 Refinement Settings. Since polyhedral representation is an approximation, different refinement settings could affect performance of the algorithms that deal with such representations. We have addressed this issue and established that scale-space decomposition is robust accross the models with different refinement settings. Figure 7 supports our claim. The graph patterns are very similar. We obtain larger graphs for the models with more points because the technique is able to extract features more precisely. Finally, similarity values prove that the models are similar.

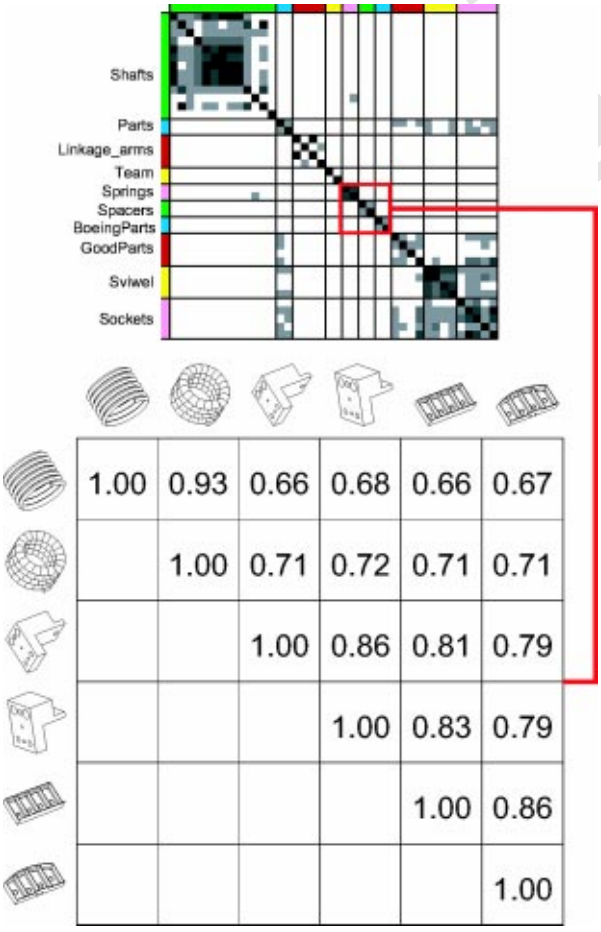

Fig. 6 Distance matrix of results for the matching process. Darker regions correspond to closer matches. Actual values for matches of a subset of 6 models is shown with pictures of the models. Most of the high-valued matches occur inside the groups, although, there are several cross-group pairs that return high similarity values.

6 Empirical Evaluation

To determine the benefits and possible shortcomings of the scale-space technique, we tested it against six (6) other approaches to shape matching and shape retrieval. To our knowledge, there is no previous work evaluating multiple matching techniques with one another using common dataset[1]. Some previous attempts to validate individual matching algorithms have tried to use *information retrieval* measures of *recall* to assess how well a technique performs a ''query-by-example.'' Studies that have adopted this approach include Funkhouser et al. [25], Tangelder and Veltkamp [42], and Klien [47]. In this context, information recall treats one's set of models as ''documents.'' *Recall* is an estimate of how well the algorithm returns *all* relevant documents related to the query, even if it returns irrelevant documents. *Recall precision* calculates how well the a system avoids finding documents that are irrelevant to a given query. The goal of traditional information retrieval is to maximize both precision and recall for normal queries entered into the system.

The goals of these studies has been to evaluate the performance of query-by-example for general classes of 3D objects (i.e., boats, blimps, planes, etc), which are more easily discriminated with global features than CAD/CAM objects. Further, our work is interested in measuring the performance for *automated classification*, which is more intricate than just processing queries. Hence, while the information retrieval approach is suitable for measuring the performance of simple query-by-example interactions, it is not sufficient for determining the overall performance of a classifier. To assess classification, our approach is to use basic statistical hypothesis testing to calculate of the number of errors made by an algorithm and combine this with evaluation strategies typically used in machine learning.

The Algorithms Used. For this evaluation, we have implemented the following known shape and solid model matching algorithms in addition to the algorithm described in this paper:

- Reeb Graphs [23];
- Shape Distributions [24–26];
- Enhanced Shape Distributions [31];
- Learning Shape Classifiers [32];
- Invariant Topology Vectors [11–13].

Procedure for Comparison. Given a set of objects $\mathcal{M}$ and a fixed set of mutually disjoint categories, $\mathcal{C}=C_1,C_2\ldots C_k$ (i.e., $\cup\mathcal{C}=\mathcal{M}$ and $C_1\cap C_2=\varnothing$ when $i\neq j$), we can use any shape matching algorithm to determine which category the query object, M_q, should be placed into by computing the model that is its

[1]The authors would like to note that the performance of any algorithm (i.e., its ability to discriminate objects and object classes) will vary greatly depending on the domain of objects. Given that the process of classification is largely subjective, here is no universal best classifier or matcher—one can construct datasets and classifications to confound any one matcher.

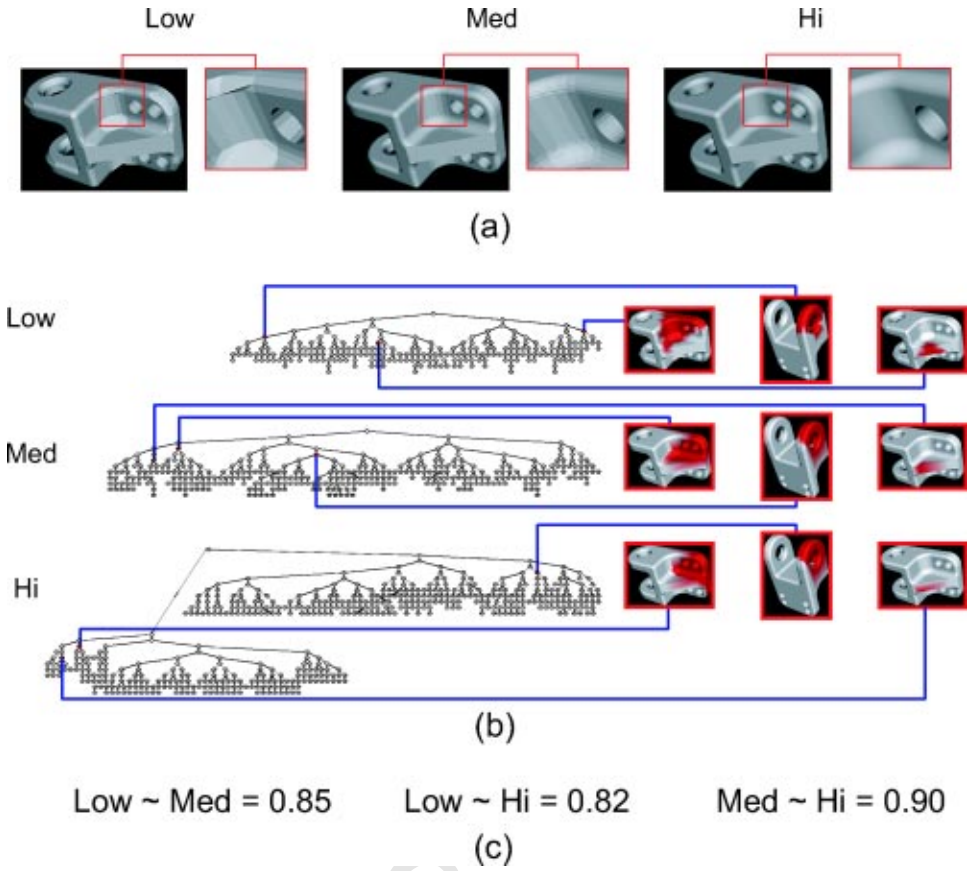

Fig. 7 Various Refinement Settings. (*a*) Sample view of the model approximated using different number of points (Low—approx 1500 points, Med—approx 2500 points, Hi—approx 6000 points). (*b*) Feature graphs for polyhedral representations that use various number of points. Overall structure of the graphs is very similar. "Zoomed-in" nodes from different graphs correspond to the same features and are placed on the same levels of the graphs. (*c*) Similarity values for these models.

nearest neighbor in the set $\mathcal{M}$. The nearest object to M_q, M_n, is the model for which the distance function over all models in $\mathcal{M}$ is minimized. The query element is then assigned to the same category as M_n.

In the case of the **CAD_40** dataset, we sequentially removed each element from the set and computed the nearest neighbor among the remaining 39 elements. In this way, the matching algorithm is testing the hypothesis that objects are in the same class as their nearest neighbors computed by the algorithm. If M_q was assigned the correct classification (i.e., M_n is in the same class as M_q), there is no error.

A *type 1* error occurs when M_q is placed into the incorrect class (i.e., because M_n is of a different class than M_q it registers as a *false negative*). In this case, *type 2* errors count the contrapositive condition (i.e., M_q is a false positive with respect to the class of M_n), hence there is one type 2 (false negative) error for each type 1 error (false positive).

Results. Table I shows the overall error accumulated by each technique across the CAD_40 dataset. To test this dataset, forty (40) classifications were performed, one for each model. The data in Table I shows the number of type 1 (and type 2) errors over these 40 classifications.

Table 1 Comparison of several matching algorithms.

Technique	Type 1 Errors	% Error
Scale-space	12	30.0%
Reeb Graph	13	32.5%
Shape Dist.	11	27.5%
Enhanced Dist.	14	35.0%
Learning	13	32.5%
ITV	15	37.5%

Tables 2 and 3 show the distribution of type 1 and type 2 errors by object class, respectively.

Discussion of Results. All of the techniques perform equally well[2] in the overall classification task for the CAD_40 set, with the error ranges falling within 10% of each other. What is interesting to note is that while, in general, 1-in-3 classifications is in error (regardless of technique), certain techniques clearly do better than others for certain classes of models. For example, on the class of ''Team'' parts, the scale-space approach has 1/2 the error rate of the original shape distribution technique. Additional examples:

- The Reeb graph approach performed best on linkage-arm dataset, where there is a strong topological consistency across shapes;
- The enhanced distribution and learning techniques performed best on the Swivel set, where both geometry and features are needed to discriminate the objects;
- The original shape distribution and ITV techniques worked best on the Housings set, which have strong geometric signatures in addition to having nearly identical boundary representation solid models.

[2]Or poorly, depending on your perspective.

Table 2 Type 1 errors by object class.

Technique	Shafts	Parts	Linkage	Team	Goodparts	Springs	Swivels	Spacers	Sockets	Housings
Category size	13	2	4	2	4	2	4	2	5	2
Scale-space	3	1	1	1	2	1	1	1	1	0
Reeb Graph	3	1	1	2	2	1	1	1	1	0
Shape Dist.	2	2	1	1	1	1	1	1	1	0
Enhanced Dist.	5	1	1	1	1	2	1	1	1	0
Learning	4	2	1	1	1	1	1	1	1	0
ITV	2	2	1	1	3	2	1	1	2	0

Table 3 Type 2 errors by object class.

Technique	Shafts	Parts	Linkage	Team	Goodparts	Springs	Swivels	Spacers	Sockets	Housings
Category size	13	2	4	2	4	2	4	2	5	2
Scale-space	0	0	2	1	1	0	1	0	4	3
Reeb Graph	0	0	0	0	1	3	1	0	4	4
Shape Dist.	0	0	1	3	0	0	1	3	1	2
Enhanced Dist.	0	0	1	6	0	0	0	1	2	4
Learning	0	0	1	5	1	0	0	2	1	3
ITV	0	0	1	1	0	1	1	2	7	0

The data indicates that one's choice of matching or classification algorithm is highly dependent on the type of data one wants to work with. There is no universal best solution, rather each algorithm identifies a slightly different set of object invariants on which to perform matching. No one technique appears better overall, however, the scale space approach introduced in this paper is among the most consistent.

Some additional observations can be made:

- The distinction between type 1 and type 2 errors lies in their distribution across the classes: some classes will be more prone to certain types of errors.
- This dataset is not sufficiently large or representative enough to enable any strong statistical generalizations. A dataset with larger (>30) classes and (perhaps) larger (>30) number of classes are needed.
- Standard data sets and test cases are needed for shape and CAD model matching, much in the same manner as has been established in machine vision. The Princeton Shape Benchmark (http://shape.cs.princeton.edu/benchark) is an important step in this direction for general shape models. In the case of CAD objects, it is simply not possible to put forth a single dataset as needs for matching and classification will vary by domain. What our studies indicate is that the choice of algorithm will depend greatly on the specifics of the dataset you wish to search and classify.

7 Concluding Remarks

We have presented a computationally efficient approach to hierarchical matching and classification of 3D models. The approach is based on a combination of scale-space feature decomposition of 3D models, rooted tree representation of these feature decompositions, and dynamic programming matching of vertex-labeled graphs. Due to the strengths of the these components, our approach is able to establish robust correspondences in the presence of sampling noise. In a series of experiments we have demonstrated the performance of our approach and introduced a method for comparing different shape matching and classification algorithms. In our evaluation, the scale-space technique performs as well, or better than, existing matching algorithms. Actual performance was shown to depend significantly on the class of models being compared and classified. We believe that the method for comparison is generalizable and can be built into a set of benchmarks for the community.

This research differs significantly from previous work on 3D model matching. The notion of feature presented here is highly tuned to the efficient identification of shape and topological categories. What we have demonstrated is that, in the context of 3D solid models of mechanical parts, the scale-space decomposition technique produces a near-exact approximation of traditional CAD features without the issues presented by traditional feature recognition and mapping. This enabled us to perform accurate matching of 3D solid models of mechanical parts which preserved meaningful engineering classifications.

Our work is in its preliminary stages, and we plan to extend its scope in several directions: 1) to improve the complexity of our matching algorithm; 2) improve the feature similarity metric $\gamma(.,.)$, to account for geometric point distribution in each feature; 3) explore techniques to add increased engineering and manufacturing data into the matching; and 4) to exploit the possibility of using the structural parameters of tree representation as signatures for indexing purposes.

Acknowledgments

This work was supported in part by National Science Foundation (NSF) CAREER Award CISE/IIS-9733545, NSF Information Technology Research Award ITR-0219176, and Office of Naval Research Grant N00014-01-1-0618. Additional support has been provided by Honeywell FM&T.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the other supporting government and corporate organizations.

References

[1] Han, J.-H., Regli, W. C., and Pratt, M. J., 2000, "Algorithms for Feature Recognition from Solid Models: A Status Report," IEEE J. Rob. Autom., **16**(6), pp. 782–796.

[2] Shah, J., Anderson, D., Kim, Y. S., and Joshi, S., 2001, "A Discourse on Geometric Feature Recognition from cad Models," ASME J. Comput. Inf. Sci. Eng., **1**(1), pp. 41–51.

[3] Elinson, A., Nau, D. S., and Regli, W. C., 1997. "Feature-based Similarity Assessment of Solid Models," In Fourth Symposium on Solid Modeling and Applications, C. Hoffman and W. Bronsvoort, Eds., ACM, ACM Press, pp. 297–310. Atlanta, GA.

[4] Regli, W. C., and Cicirello, V., 2000, "Managing Digital Libraries for Computer-aided Design," Comput.-Aided Des., **32**(2), pp. 119–132. Special Issue on CAD After 2000. Mohsen Rezayat, Guest Editor.

[5] Cicirello, V., and Regli, W. C., 1999. "Resolving Non-uniqueness in Design Feature Histories," In Fifth Symposium on Solid Modeling and Applications, D. Anderson and W. Bronsvoort, Eds., ACM, ACM Press. Ann Arbor, MI.

[6] Cicirello, V. A., 1999, "Intelligent Retrieval of Solid Models," Master's thesis, Drexel University, Geometric and Intelligent Computing Laboratory, Department of Mathematics and Computer Science, Philadelphia, PA 19104, June 1999.

[7] Cicirello, V., and Regli, W., 2001. "Machining Feature-based Comparisons of Mechanical Parts," In International Conference on Shape Modeling and Applications, ACM SIGGRAPH, the Computer Graphics Society and EUROGRAPHICS, IEEE Computer Society Press, pp. 176–187.

[8] Cicirello, V., and Regli, W. C., 2002, ''An Approach to a Feature-based Comparison of Solid Models of Machined Parts,'' Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM), **16**(5) [November], pp. 385–399.
[9] Ramesh, M. M., Yip-Hoi, D., and Dutta, D., 2000, ''A Decomposition Methodology for Machining Feature Extraction,'' In ASME Design Engineering Technical Conferences, Computers in Engineering Conference, American Association of Mechanical Engineers, ASME Press. DETC2000/CIE-14645.
[10] Sun, T.-L., Su, C.-J., Mayer, R. J., and Wysk, R. A., 1995, ''Shape Similarity Assessment of Mechanical Parts Based on Solid Models,'' In ASME Design for Manufacturing Conference, Symposium on Computer Integrated Concurrent Design, R. Gadh, Ed., ASME, pp. 953–962.
[11] McWherter, D., Peabody, M., Shokoufandeh, A., and Regli, W., 2001. ''Transformation Invariant Similarity Assessment of Solid Models,'' In ASME Design Engineering Technical Conferences, ASME, ASME Press. DETC2001/DFM-21191.
[12] McWherter, D., Peabody, M., Shokoufandeh, A., and Regli, W., 2001, ''Solid Model Databases: Techniques and Empirical Results,'' ASME J. Comput. Inf. Sci. Eng. **1**(4), pp. 300–310.
[13] McWherter, D., Peabody, M., Shokoufandeh, A., and Regli, W., 2001. ''Database Techniques for Indexing and Clustering of Solid Models,'' In Sixth ACM/SIGGRAPH Symposium on Solid Modeling and Applications, D. Dutta and H.-P. Seidel, Eds., ACM, ACM Press, pp. 78–87.
[14] Snead, C. S., 1989. *Group Technology: Foundations for Competitive Manufacturing.* Van Nostrand Reinhold, New York.
[15] Ames, A. L., 1991, ''Production Ready Feature Recognition Based Automatic Group Technology Part Coding,'' In Symposium on Solid Modeling Foundations and CAD/CAM Applications, J. Rossignac and J. Turner, Eds., ACM SIGGRAPH, ACM Press, pp. 161–169.
[16] Shah, J. J., and Bhatnagar, A., 1989, ''Group Technology Classification from Feature-based Geometric Models,'' Manufacturing Rev. **2**(3), pp. 204–213.
[17] Bond, A., and Jain, R., 1988, ''The Formal Definition and Automatic Extraction of Group Technology Codes,'' In ASME Design Technical Conferences, Computers in Engineering Conference, pp. 537–542.
[18] Henderson, M., and Musti, S., 1988, ''Automated Group Technology Part Coding from a Three-dimensional Cad Database,'' J. Eng. Ind., **110**(3), pp. 278–287.
[19] Ham, I., Marion, D., and Rubinovich, J., 1986, ''Developing a Group Technology Coding and Classification Scheme,'' Ind. Eng. (N. Y., 1922-1931), **18**(7), pp. 90–97.
[20] The Digital Michelangelo Project, http://graphics.stanford.edu/projects/mich/.
[21] Thompson, W. B., Riesenfeld, R. F., and Owen, J. C., 1996, ''Determining the Similarity of Geometric Models,'' In Proceedings of the ARPA Image Understanding Workshop.
[22] Thompson, W. B., Owen, J., de St. Germain, H., Stark, S. Jr.,, and Henderson, T., 1999, ''Feature-based Reverse Engineering of Mechanical Parts,'' IEEE J. Rob. Autom., **12**(1), pp. 57–66.
[23] Hilaga, M., Shinagawa, Y., Kohmura, T., and Kunii, T. L., 2001, ''Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes,'' In SIGGRAPH, ACM Press, pp. 203–212.
[24] Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D., 2001, ''Matching 3D Models with Shape Distributions,'' In International Conference on Shape Modeling and Applications, 154–166, Ed., ACM SIGGRAPH, the Computer Graphics Society and Eurographics, IEEE Computer Society Press.
[25] Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D., 2002, ''Shape Distributions,'' ACM Trans. Graphics, **21**(4), pp. ■■■.
[26] Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., and Jacobs, D., 2003, ''A Search Engine for 3D Models,'' ACM Trans. Graphics, **22**(1), pp. ■■■.
[27] Kazhdan, M., Chazelle, B., Dobkin, D., Funkhouser, T., and Rusinkiewicz, S., 2003, ''A Reflective Symmetry Descriptor for 3D Models,'' Algorithmica, To appear.
[28] Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S., 2003, ''Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors,'' In Symposium on Geometry Processing.
[29] Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S., 2002, ''Harmonic 3D Shape Matching,'' In SIGGRAPH.
[30] Elad, M., Tal, A., and Ar, S., 2001, ''Content Based Retrieval of VRML Objects—An Iterative and Interactive Approach,'' In The Sixth Eurographics Workshop in Multimedia, pp. 97–108.
[31] Ip, C. Y., Lapadat, D., Sieger, L., and Regli, W., 2002. ''Using Shape Distributions to Compare Solid Models,'' In Seventh ACM/SIGGRAPH Symposium on Solid Modeling and Applications, H.-P. Seidel, Ed., ACM, ACM Press, pp. 273–280.
[32] Ip, C. Y., Sieger, L., Regli, W., and Shokoufandeh, A., 2003, ''Automated Learning of Model Classifications,'' In Eighth ACM/SIGGRAPH Symposium on Solid Modeling and Applications, G. Elber and V. Shapiro, Eds., ACM, ACM Press, pp. 322–327.
[33] Cyr, C. M., and Kimia, B. B., 2001, ''3D Object Recognition Using Shape Similarity-based Aspect Graph,'' In Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), pp. 254–261.
[34] Shapiro, L. G., and Haralick, R. M., 1981, ''Structural Descriptions and Inexact Matching,'' IEEE Trans. Pattern Anal. Mach. Intell., **3**, pp. 504–519.
[35] Kim, W., and Kak, A. C., 1991, ''3D Object Recognition Using Bipartite Matching Embedded in Discrete Relaxation,'' IEEE Trans. Pattern Anal. Mach. Intell., **13**(3), pp. 224–251.
[36] Pelillo, M., Siddiqi, K., and Zucker, S., 1999, ''Matching Hierarchical Structures Using Association Graphs,'' IEEE Trans. Pattern Anal. Mach. Intell., **21**(11) [November], pp. 1105–1120.
[37] Gold, S., and Rangarajan, A., 1996, ''A Graduated Assignment Algorithm for Graph Matching,'' IEEE Trans. Pattern Anal. Mach. Intell., **18**(4), pp. 377–388.
[38] Siddiqi, K., Shokoufandeh, A., Dickinson, S., and Zucker, S., 1999, ''Shock Graphs and Shape Matching,'' Int. J. Comput. Vis., **30**, pp. 1–24.
[39] Shokoufandeh, A., Dickinson, S., Jönsson, C., Bretzner, L., and Lindeberg, T., 2002, ''On the Representation and Matching of Qualitative Shape at Multiple Scales,'' In Proceedings, 7th European Conference on Computer Vision, vol. 3, pp. 759–775.
[40] Luo, B., and Hancock, E. R., 2001, ''Structural Matching Using the EM Algorithm and Singular Value Decomposition,'' IEEE Trans. Pattern Anal. Mach. Intell., **23**, pp. 1120–1136.
[41] Veltkamp, R. C., 2001, ''Shape Matching: Similarity Measures and Algorithms,'' In International Conference on Shape Modeling and Applications, 188–197, Ed., ACM SIGGRAPH, the Computer Graphics Society and EUROGRAPH-ICS, IEEE Computer Society Press pp.
[42] Tangelder, J. W., and Veltkamp, R. C., ''Polyhedral Model Comparison and Retrieval Using Weighted Point Sets,'' In The eight annual conference of the Advanced School for Computing and Imaging, pp. 223–230.
[43] Cormen, T. H., Leiserson, C. L., and Rivest, R. L., 2001, Introduction to Algorithms, MIT Press.
[44] Thomasian, A., Castelli, V., and Li, C.-S., 1998, ''Clustering and Singular Value Decomposition for Approximate Indexing in High Dimensional Spaces,'' In The Seventh International Conference on Information and Knowledge Management Table of Contents, pp. 201–207.
[45] Golub, G. H., and van Loan, C. F., 1989, *Matrix Computations.* John Hopkins Press.
[46] Wang, J. T. L., Shapiro, B. A., Shasha, D., Zhang, K., and Currey, K. M., 1998, ''An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees,'' IEEE Trans. Pattern Anal. Mach. Intell., **20**, pp. 889–895.
[47] Novotni, M., and Klein, R., 2003. ''3D Zernike Descriptors for Content Based Shape Retrieval,'' In Proceedings of the eighth ACM symposium on Solid modeling and applications, ACM Press, pp. 216–225.