Local Feature Extraction for Partial Matching

Dmitriy Bespalov† Ali Shokoufandeh† William C. Regli†‡

Department of Computer Science†
Department of Mechanical Engineering & Mechanics‡
College of Engineering
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104

December 30, 2004

Abstract

A primary shortcoming of existing techniques for 3D model matching is the reliance on global information of model’s structure. Models are matched in their entirety, depending on overall topology and geometry information. A current open challenge is how to perform partial matching. Partial matching is important for finding similarities across part models with different global shape properties and for segmentation and matching of data acquired from 3D scanners.

This paper presents a Scale-Space feature extraction technique to support matching based on local model structure. Scale-Space decomposition has been successfully used to extract features from mechanical artifacts. Scale-Space techniques can be parameterized to generate decompositions that correspond to manufacturing, assembly or surface features relevant to mechanical design. One application of these technique is to support matching and content-based retrieval of solid models.

The new technique is computationally practical for use in indexing large models. Examples are provided that demonstrate effective feature extraction on 3D laser scanned models.

1 Introduction

In order to perform content-based indexing and retrieval of 3D objects, each model must be converted into some collection of features. Previous research on model matching and retrieval has drawn on feature definitions from mechanical design, computer graphics and computer vision literature. Many of these feature-based techniques ultimately use vertex-labeled graphs, whose nodes represent 3D features (or their abstractions) and whose edges represent spatial relations or constraints, between the features. Retrieval and matching is done using some variation of graph matching to assign a numerical value describing the distance between two models.

It is common in engineering communities for the term feature to be used to refer to machining features (i.e., holes, pockets, slots) or other local geometric or topological characteristics of interest, depending on
the domain (i.e., assembly surfaces, molding lines, etc). In the context of this work, feature will be used as an intrinsic property of the 3D shape which may encompass local geometry and topology. Depending on the choice of function to parameterize the Scale-Space decomposition, these local features could correspond to design or manufacturing operations, machining features or assembly surfaces, etc. The notion of features in this paper draws from the computer vision literature [1]; hence, the features are designed for object classification.

There are numerous surveys of feature recognition techniques for CAD [2, 3]; and similarity assessment of 3D models using feature extraction has been addressed by several efforts [4, 5]. These techniques assume the exact representation, as is obtained from a CAD system (i.e., a 3D, watertight boundary representation). However, these representations are proprietary, and their internals vary from system to system. Feature-based descriptions of models also vary by system. Hence, CAD search tools that can perform semantically effective searches using “the lowest common denominator” (e.g., shape) representation are widely applicable.

Matching 3D shape representations has been widely studied in graphics [6], computer vision [7] and engineering [8]. When shape representations are used for CAD data, there are two major shortcomings with existing work. First, the current generation of matching techniques have difficulty handling the approximate representations (i.e., polyhedral mesh, point cloud, etc) that are needed to find sub-patterns in objects or handle data created by 3D scanners. With a few notable exceptions, most researchers assume watertight VRML or shape models. Second, and more importantly, the current generation of search techniques almost exclusively focus on gross or overall shape. In the context of CAD, local features and feature patterns contribute considerably to manufacturing cost, selection of manufacturing processes, producibility and functional parameters of 3D objects. Many objects with similar gross shape can have vastly different functions, costs or manufacturing process specifications.

This paper builds on the work in [9], where it was shown how Scale-Space decompositions could be used to extract features from 3D models in a polyhedral representation. It develops a parameterizable feature decomposition method that can be tuned to extract local feature configurations of engineering significance. The approach put forth in this paper is motivated by several open problems in 3D shape matching and indexing of CAD models for useful engineering purposes:

**Parameterizable Decompositions:** Scale-Space decomposition promises to decompose a 3D model into structurally relevant components automatically. These decompositions can be parameterized with a measure function, resulting in different components and features. This allows us to perform efficient comparisons (using well studied tree-matching algorithms) of 3D models in terms of the similarity of underlying features [9]. Different measure functions can be created to tailor decompositions toward feature sets tuned to answer specific questions (i.e., cost, manufacturing process, shape, etc).

**Robustness in Presence of Noise:** Most existing work on shape and solid matching assumes an ideal world with watertight models and no noise. The reality is that objects may not be perfect, this is especially true when trying to examine 3D models acquired by laser scanning or other means (i.e., image, inspection probes, etc). In this context, one needs to be able to compare the noisy acquired data to other noisy data or to the exact geometry data that exists in a database of CAD models. The Scale-Space technique has shown consistency in its performance under noise, both with synthetic noise as well as with respect to the actual noise in data from laser scans. Hence, models can be effectively matched across representations.

**Partial Matching:** Partial matching is a major open problem in 3D indexing and matching. It manifests itself in several ways. Most evident is that acquired data is rarely complete. For example, occlu-
sion prevents scanners from getting interior points of holes and other features. In addition, obtaining a “complete” scan is time consuming, requiring manual re-positioning of artifacts on the scanning apparatus and (quite often) manual registration of the point set data acquired by these scans. In the most basic case, the scanned data may consist of only one “view” of the model—resulting in a set of points on the surfaces of the object and not a 3D shape.

From Scanned Point Cloud to Database Query: The ability to handle noise and partial data are essential in a system that can go from scanned data input directly to a database query with minimal human intervention. With even the best of present technology, it is difficult to get complete watertight solids from scanned data that precisely match the scanned artifact. In the case of CAD objects, most of which have high genus and many occluded surfaces, obtaining a complete scan that evenly samples points over the surfaces is simply impossible. Hence, matching and query mechanisms must be able to operate from limited information, i.e., point data or portions of surfaces.

Basis for Solution by Many-to-Many Matching: Many-to-Many matching aligns the corresponding decomposition from one medium (e.g., the native CAD object) with that of other media (e.g., scanned data) and similar, but slightly different, CAD objects. The decomposition process presented in this paper is consistent across these different media types; however, the exact boundaries of the segmentations may vary depending on the quality of the data, noise or differences in the underlying geometric representation. This creates a many-to-many matching problem in which subsets of segments from one object must be paired to subsets of the segments resulting from the decomposition of the other object.

2 Related Work

The work in the paper draws on concepts from several areas of computer science and engineering. We review some of this background below.

2.1 Object Recognition and Matching

This paper uses the term feature to refer to an intrinsic property of the 3D shape which may encompass local geometry and topology related to design or manufacturing operations, but may not have direct correspondence to any explicit manufacturing features. In this sense the notion of a feature draws from the computer vision literature [1].

The computer vision research community has typically viewed shape matching as a problem in 2D [10, 11, 12, 13]. These efforts address a different aspect of the general geometric/solid model matching problem—one in which the main technical challenge is the construction of the models to be matched from image data obtained by cameras.

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. While a complete survey of this area is beyond the scope of this paper, we review several notable efforts. Thompson et al. [14, 15] reverse engineered designs by generating surfaces and machining feature information from range data collected from machined parts. Jain et al. [16] indexed CAD data based on the creation of “feature vectors” from 2D images. Sipe, Casasent and Talukder [17, 18] used acquired 2D image data to correlate real machined parts with CAD models and performed classification and pose estimation. Scale-Space decomposition is very popular in Computer
Vision for extracting spatially coherent features. Most of the work in this community has focused on the Scale-Space features of 2D images using wavelets or Gaussian filters [1, 19].

Once objects are recognized, they can be segmented, decomposed and matched. Matching is frequently accomplished by encoding objects and their decompositions as a graph and doing analyses across different graph structures to identify similarity. Graphs and their generalizations are among the most common and best studied combinatorial structures in computer science, due in large part to the number of areas of research in which they are applicable. Due to space constraints, we cite only a few examples of how they are being applied to 3D object recognition and matching. Nayar and Murase extended this work to general 3D objects where a dense set of views was acquired for each object [20]. Eigen-based representations have been widely used in many areas for information retrieval and matching as they offer greater potential for generic shape description and matching. In an attempt to index into a database of graphs, Sossa and Horaud use a small subset of the coefficients of the $d_2$-polynomial corresponding to the Laplacian matrix associated with a graph [21], while a spectral graph decomposition was reported by Sengupta and Boyer for the partitioning of a database of 3D models, in which nodes in a graph represent 3D surface patches [22].

Graph matching has a long history in pattern recognition. Shapiro and Haralick’s use of weighted graphs for the structural description of objects was among the first in the vision community [23]. Eshera and Fu [24] used attributed relation graphs to describe parametric information as the basis of a general image understanding system to find inexact matches. Recently, Pelillo et al. [25] introduced a matching algorithm which extends the detection of maximum cliques in association graphs to hierarchically organized tree structures. Tirthapura et al. present an alternative use of shock graphs for shape matching [26]. The edit distance approach for finding matching in rooted trees has been studied by Zhang, Wang, and Shasha [27]. Their dynamic programming approach for degree-2 distance, when applied to unordered trees, is a restricted form of the constrained distance previously reported in [28].

### 2.2 Matching 3D Objects

With the ready availability of 3D models from graphics programs and CAD systems, there has been a substantial amount of activity on 3D object recognition and matching in the past 20 years. This body of relevant work is too large to survey in detail in this paper. Interested readers are referred to several recent survey papers [7, 8, 6].

#### 2.2.1 Comparing Shape Models.

Shape-based approaches usually work on a low-level point cloud, mesh or polyhedral model data, such as that produced by digital animation tools or acquired by 3D range scanners. Approaches based on faceted representations include that of Osada et al. [29], which creates an abstraction of the 3D model as a probability distribution of samples from a shape function acting on the model. Hilaga et al. [30] present a method for matching 3D topological models using multi-resolution Reeb graphs. A variant on this is proposed in [31]. A current trend, being pursued by several groups, is the use of different types of shape descriptors (harmonics, Zernike, etc.) to capture shape invariants [32, 33, 34, 35].

The Princeton 3D shape database [36] that has been used in a number of these studies [30, 29] contains mainly models from 3D graphics and rendering; none of these models are specifically engineering, solid modeling or mechanical CAD oriented.

In general, however, shape matching-based approaches only operate on the gross-shape of a single part and do not operate directly on 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.

2.2.2 Comparing Solid Models.

Unlike shape models, for which only approximate geometry and topology is available, solid models pro-
duced by CAD systems are represented by precise boundary representations. When comparing solid models
of 3D CAD data, there are two basic types of approaches for content-based matching and retrieval: (1)
feature-based techniques and (2) shape-based techniques. The feature-based techniques [37, 2, 38, 3, 39],
going back at least as far as 1980 [40], extract engineering features (machining features, form features, etc.)
from a solid model of a mechanical part for use in database storage, automated group technology (GT) part
coding, etc. The shape-based techniques are more recent, owing to research contributions from computa-
tional geometry, vision and computer graphics. These techniques leverage the ready availability of 3D
models on the Internet.

Feature-Based Approaches. Historically Group Technology (GT) coding was the way to index of parts
and part families [41]. This facilitated process planning and cell-based manufacturing by imposing a clas-
sification scheme (a human-assigned alphanumeric string) to individual machined parts. While there have
been a number of attempts to automate the generation of GT codes [42, 43, 44], transition to commercial
practice has been limited.

The idea of similarity assessment of 3D models using feature extraction techniques has been discussed
in [2, 3]. These techniques assume the exact representation (i.e. Brep) for the input models and there-
fore cannot be used if only an approximate representation (i.e. polyhedral mesh) is available. This is a
major shortcoming, especially in designing an archival system, where one may require partial and inexact
matching.

Elinson et al. [45] used feature-based reasoning for retrieval of solid models for use in variant process
planning. Cicirello and Regli [46, 5, 4] examined how to develop graph-based data structures and create
heuristic similarity measures among artifacts based on manufacturing features. McWherter et al. [47] in-
tegrated these ideas with database techniques to enable indexing and clustering of CAD models based on
shape and engineering properties. Other work from the engineering community includes techniques for
automatic detection of part families [48] and topological similarity assessment of polyhedral models [49].

Shape-Based Approaches. Comparing CAD models based on their boundary representations can be diffi-
cult due to variability in the underlying feature-based representations. Additional complications are created
by differences among the boundary representations used by systems (i.e., some may use all NURBS, some
may use a mix of surface types, etc). Using a shape-based approach on voxels, meshes or polyhedral models
generated from native CAD representations is one way of reducing these problems.

The 3D-Base Project [50, 51] used CAD models in a voxel representation, which were then used to
perform comparisons using geometric moments and other features. The recent work by the authors covers
several areas including shape classification, Scale-Space decomposition and classification learning [52, 53,
54, 55].

Work out of Purdue [56, 57, 58] has improved on the voxel methods of [50, 51], augmenting them with
skeletal structures akin to medial axes or shock graphs. The main accomplishment of the Purdue group is
getting these shape-only techniques in a system for query by example.
3 Sub-Space Clustering and Scale-Space 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, rather than the CAD literature. Namely, given an object $M$, we are interested in partitioning $M$, into $k$ features $M_1, \ldots, M_k$ with $M_i \cap M_j = \emptyset$, for $1 \leq i < j \leq k$, and $M = \bigcup_i M_i$ subject to maximization of some coherence measure, $f(M_i)$, defined on the 3D elements forming each $M_i$. At a finer scale, each feature $M_i$ will be decomposed into $j = 1, \ldots, k$ 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, $\ell$. In most pattern recognition applications, $k$ is a control parameter. If models $M$ and $M'$ are topologically similar, the $k$ major components at every scale should also be similar. The coherence function $f(M')$ will assign an overall metric to the quality of 3D elements participating in the construction of feature $M'$. Finally, the depth of decomposition will be controlled depending on the quality of a feature in comparison to all its sub-features. Specifically, assume $M^i$ represents a feature at scale $i$, and $M^i_1, \ldots, M^i_{i+1}$, 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 $M^i$ if and only if $f(M^{i+1}) \leq f(M^i_1) + f(M^i_2) + \ldots + f(M^i_{i+1})$. This simple criteria for expansion of scale-space at every feature has its roots in information theory. It is in fact motivated by linear form similar to entropy of feature $M^i$ as opposed to its sub-features $M^i_1, \ldots, M^i_{i+1}$. In the end, a set of the leaf nodes in a decomposition tree would correspond to the final features of a given model.

3.1 Distance Function

A 3D model $M$ is given in polyhedral representation (models in VRML format were used in the experiments). We are interested in decomposing this model into $k$ sub-features using Scale-Space decomposition. The application of Scale-Space decomposition requires some distance function $D(., .)$ that captures the affine structure of a model $M$. The shortest-path metric $\delta(., .)$ (geodesic distance [59]) on the triangulation of $M$ with respect to points $\{v_1, \ldots, v_n\}$, is one such function. In this case, the distance function $D(u, v) = \delta(u, v)$ would be the shortest path distance on the triangulated surface between $u$ and $v$ for all $u, v \in M$.

In our previous work [9] metric $\delta(., .)$ was successfully used for Scale-Space decomposition. The experimental results showed that the measure $\delta(., .)$ successfully captures affine structure of the model $M$ and produces meaningful decompositions. The problem with such a distance measure is that it captures global information of the model. Even small perturbations of the model $M$ may cause the distance function $D(., .)$ to vary significantly, which in its turn, changes extracted features. Further, using geodesic distance as distance measure for decomposition does not tolerate noise (i.e. laser-scanned data) very well. Figure 1 illustrates several distance functions that could be used in Scale-Space decomposition. Specifically, Figure 1(a) shows a geodesic distance function (weight of the shortest path between points) between points $p_1$ and $p_2$; Figure 1(b) illustrates angular shortest path (weight of the shortest path computed using an angu-
Due to the above shortcomings of the geodesic distance measure that a new distance function is introduced for use in the Scale-Space decomposition process. The new distance function is computed with respect to the triangular faces of the model \( M \{ t_1, ..., t_n \} \). Here and in the rest of the paper \( n \) denotes the number of triangles in the model. The angular shortest path between two triangular faces \( t_i \) and \( t_j \) is defined to be the shortest path on the surface of the model which is computed in terms of angular difference between faces.

Figure 1(c) shows a maximum angle on angular shortest path between the faces \( t_1 \) and \( t_2 \). Specifically, let \( t_i \sim t_j \) denote the angular shortest path \( (t_i, t_m, t_l, ..., t_j) \) between faces \( t_i \) and \( t_j \). And let \( t_m \rightarrow t_l \in t_i \sim t_j \) denote two adjacent triangular faces \( t_m \) and \( t_l \) on the angular shortest path \( t_i \sim t_j \). Then, the distance function used in this work is defined as

\[
D(t_i, t_j) = \max_{t_m \rightarrow t_l \in t_i \sim t_j} \angle(t_m, t_l).
\]  

Intuitively, distance \( D(t_i, t_j) \) is the maximum angle between adjacent faces on the angular shortest path between \( t_i \) and \( t_j \). The rationale behind such measure is to quantify the smoothness of the surface – small angle between adjacent faces correspond to smooth surface.

Observe that by construction the matrix \( D_M = [D(t_i, t_j)]_{n \times n} \) is symmetric. Also note that distance measure \( D \) is not a metric function, but it captures the geometric structure of the model \( M \). It is important to stress that such angular distance measure has the same properties as geodesic distance function used in the previous work. As a result, introduction of the new distance measure for decomposition does not violate any statements made in [9] and all of the theorems would still hold.

### 3.2 Decomposition Algorithm

Let \( v_i \) be the \( i^{th} \) row (or column) in \( D \). Then \( v_i \) is an \( n \)-dimensional vector characterizing the distance structure of face \( t_i \) in model \( M \). The problem of decomposing model \( M \) into its \( k \) most significant features \( M_1, ..., M_k \) is closely related to \( k \)-dimensional subspace clustering (\( k \)-DSC). \( k \)-DSC gives a set of distance vectors \( v_1, ..., v_n \), where the objective is to find a \( k \)-dimensional subspace \( S \) that minimizes the quantity:
Figure 2: Results of applying FEATURE-DECOMPOSITION($\mathcal{M}, k$) to a model using different distance functions $D$ for $k = 2$.

\[
\sqrt{\sum_{1 \leq i \leq n} d(v_i, S)^2},
\]

where $d(v_i, S)$ corresponds to the smallest distance between $v_i$ and any member of $S$. In practice, if $S$ is given, then $\mathcal{M}_1, ..., \mathcal{M}_k$ can be computed using the principle components $\{c_1, ..., c_k\}$ of the $k$-dimensional subspace $S$ [60]. Observe that these $k$ vectors will also form a basis for $S$. Specifically, $t_i$ will belong to the
feature $M_j$ if the angle between $v_i$ and $c_j$ is the smallest among all basis vectors in $\{c_1, ..., c_k\}$, i.e., the triangular face $t_i$ that corresponds to the vector $v_i$ will belong to the feature vector $M_j$ iff the angle between $v_i$ and $c_j$ vectors is the smallest compared to all other basis vectors.

Singular value decomposition (SVD) clustering [60] is used to construct the subspace $S$, which is the optimal solution of $k$-DSC. First, observe that the symmetric matrix $D \in \mathbb{R}^{n \times n}$ has a SVD-decomposition of the form

$$D = U\Sigma V^T,$$

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

$$\Sigma = \text{Diag}(\sigma_1, \sigma_2, ..., \sigma_n),$$

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

$$D^{(k)} = U\text{Diag}(\sigma_1, ..., \sigma_k, 0, ..., 0)V^T.$$

Then,

**Theorem 1** [follows from Eckart-Young Theorem [61]].

$$||D - D^{(k)}||_2 = \min_{\text{rank}(H)=k} ||D - H||_2.$$  \hspace{1cm} (6)

This states that matrix $D^{(k)}$ is the best approximation to $D$ among all matrices of rank $k$. In fact, this result can be generalized to many other norms, including Frobenius norm. [9] showed that the set $S = \text{range}(D^{(k)})$ ($S$ is the range of matrix $D^{(k)}$, the subspace spanned by the columns of matrix $D^{(k)}$) is the optimal solution to $k$-DSC problem.

**Algorithm 1** 

**FEATURE-DECOMPOSITION($M$, $k$)**

1. Construct the distance matrix $D \in \mathbb{R}^{n \times n}$.
2. Compute the SVD decomposition $D = U\Sigma V^T$, with $\Sigma = \text{Diag}(\sigma_1, \sigma_2, ..., \sigma_n)$.
3. Compute the order $k$ compression matrix $D^{(k)} = U\text{Diag}(\sigma_1, ..., \sigma_k, 0, ..., 0)V^T$.
4. Let $c_j$ denote the $j^{th}$ column of $D^{(k)}$, for $j = 1, ..., k$, and form sub-feature $M_j$ as the union of faces $t_i \in M$ with $d(t_i, S) = d(t_i, c_j)$.
5. Return the set $\{M_1, ..., M_k\}$.

Algorithm 1 summarizes one phase of the Scale-Space decomposition of $M$ into its $k$ most significant features, $M_1, ..., M_k$. Algorithm 1 returns the partitioning of $M$ by placing each face $t_i$ in $M$ into one of the partitions $M_j$, such that the angle between vector $t_i$ and basis vector $c_j$ corresponding to the partition $M_j$ is minimized. Figure 2 shows three decomposition trees of the model – using geodesic distance, angular shortest path and maximum angle on angular shortest path measures. Note that the presented decomposition trees are not full and the leaf nodes of the trees may not correspond to the actual final features extracted from this model.
The bottleneck of Algorithm 1 is the $O(n^3)$ SVD decomposition, for an $n \times n$ matrix. The polyhedral representation of a model provides a planar graph of a 2D manifold. If only neighboring vertices are considered in the construction of the distance matrix $D$, the number of non-zero entries in $D$ would be at most $3n$ (due to planarity of the graph). Computing the SVD decomposition for sparse matrices is much faster and takes $O(mn) + O(mM(n))$ [62]. Where $m$ is the maximum number of matrix-vector computations required and $M(n)$ is the cost of matrix-vector computations of the form $Dx$. Since $M$ is a planner map and $D$ is a sparse matrix, $M(n) = O(n)$ and $m = O(n)$.

3.3 Controlling Decomposition Process

The decomposition process as presented in Section 3.1 does not allow for an explicit mechanism to stop the indefinite subdivision of a feature. Clearly, one could use a prescribed value 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 given depth. This section provides an overview of a mechanism that will control the feature decomposition. Intuitively, the use of this control mechanism will terminate the decomposition process only when all coherent features are extracted.

Let $\mathcal{M}$ be the original model’s face set. Assume in the decomposition process a feature $\mathcal{M}_1$ in $\mathcal{M}$ can be decomposed into sub-features $\mathcal{M}_2$ and $\mathcal{M}_3$ (e.g., without loss of generality assume that feature $\mathcal{M}_1$ is being bisected). The decomposition of the feature $\mathcal{M}_1$ into sub-features $\mathcal{M}_2$ and $\mathcal{M}_3$ is said to be significant if the angular distance between components of $\mathcal{M}_2$ and $\mathcal{M}_3$ is large. Formally, this condition could be expressed as follows:

$$\forall t_i \in \mathcal{M}_2, \ t_j \in \mathcal{M}_3 \ \exists t_m \rightarrow t_i \sim t_j \ s.t.$$
\[ t_m \in \mathcal{M}_2 \land t_l \in \mathcal{M}_3 \land \angle(t_m, t_l) = D(t_i, t_j), \]

i.e. if the angular shortest path between \( t_i \in \mathcal{M}_2 \) and \( t_j \in \mathcal{M}_3 \) contains two faces \( t_m \) and \( t_l \) (from \( \mathcal{M}_2 \) and \( \mathcal{M}_3 \) respectively) with large angular distance, then \( \mathcal{M}_1 \) should be decomposed into \( \mathcal{M}_2 \) and \( \mathcal{M}_3 \).

Intuitively, if \( \mathcal{M}_1 \) is smooth it should not be bisected any further. On the other hand, if discrepancy between the neighboring triangles in \( \mathcal{M}_1 \) is significant, \( \mathcal{M}_1 \) should be bisected.

## 4 Empirical Results

The feature extraction process was performed on a number of CAD models in polyhedral representation. These models were converted from ACIS format, which is exact representation format. As a result, all of the models have nice structure (i.e. no missing faces).

In the experiments we would like to examine the qualities of features extracted using the \textsc{Feature-Decomposition}(\( \mathcal{M}, k \)) algorithm. To these ends, \textsc{Feature-Decomposition}(\( \mathcal{M}, k \)) is recursively applied to each model for \( k = 2 \). Once a decomposition tree is obtained, the last layer of the decomposition tree (leaf nodes) is said to be a set of extracted features. Note, that the union of the features (leaf nodes) is equivalent to the surface of the entire model. Refer to Figure 3 for an illustration of the feature extraction process. For illustrative purposes, only a subset of extracted features is shown in Figure 3(b); the features shown in Figure 3(a) do not correspond to the leaf nodes in Figure 3(b). The actual decomposition tree is quite large for this model.

### 4.1 Feature Decomposition on CAD Data

Figure 4 shows extracted features for several models. These images are presented in order to illustrate the type of features the technique can extract. Observe that each feature corresponds to a relatively smooth surface on the model. If there is a significant angular difference on the surface, then it gets decomposed into separate features. Any closed smooth surfaces (i.e. hole) are decomposed into two (i.e. hole) or more (i.e. surface is concave) features. We plan to address the problem of how to use the decomposition trees for matching in the future work.

In addition, partial data from these models was created. Each model was intersected with several planes and only a part of the model (on one side of the plane) was saved. As a result, a number of partial objects was obtained which enabled us to see how the \textsc{Feature-Decomposition}(\( \mathcal{M}, k \)) algorithm performs on the models where only partial data is available. Illustrations of extracted features could be found in Figure 4.

### 4.2 Feature Decomposition on Noisy Data

In order to simulate the noise produced from capturing the object using 3D laser scan, Gaussian noise was applied to each point of the models presented in Section 4.1. Gaussian Noise with standard deviation of 1% and 2% from the standard deviation of all points in the model was used. Then the features were extracted using \textsc{Feature-Decomposition}(\( \mathcal{M}, k \)) algorithm. The illustrations of the extracted features can be found in Figures 5 and 6. Similar to the CAD models presented above, partial models for this dataset were generated. Note that it is possible for separate features to be assigned visually similar colors, making them appear to be the same features. The names for the CAD models were assigned by the organizations that provided the files to us. We chose to use such names for the purpose of referencing the models within this paper.
4.3 Feature Decomposition on Acquired Models

We have established that the feature extraction procedure allows to obtain relevant subsets of a model that reflect the complexity of its 3D structure. The next experiment was aimed at assessing whether the technique is capable of handling models that were obtained using a 3D digitizer – full 3D view (Figure 7(b)) and partial 3D view (Figure 7(a)) of 3D objects. Such data is known to be very noisy, often with broken connectivity and missing faces. Ideally, one would like to be able to take a single scan of a 3D CAD model, decompose it into features, and select models from the database that contain the same feature arrangements. Three CAD parts were used to create six 3D models – full and partial (one scan) for each CAD part. Once the point clouds were obtained, they were faceted, and features were extracted using the FEATURE-DECOMPOSITION($\mathcal{M}, k$) algorithm.

Figure 8 shows correspondence between extracted features for fully and partially scanned models as well as models obtained from exact representation. Note that in some cases one feature from one model (i.e. full scan) can correspond to multiple features from another model (i.e. single scan).

The performance of the technique on noisy data is certainly not as remarkable as on the CAD dataset (Section 4.1). Although, we believe that in most cases the extracted features are meaningful and reflect the structure of the models. In addition, it is clear that there are similarities between feature decompositions of fully and partially scanned models and 3D CAD models from our database. The scanned models used for this experiment are freely available at http://edge.cs.drexel.edu/Dmitriy/Scanned.tar.bz.
<table>
<thead>
<tr>
<th></th>
<th>Full Model</th>
<th>Partial Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CIMPLEX</strong></td>
<td><img src="image1" alt="CIMPLEX Full Model" /></td>
<td><img src="image2" alt="CIMPLEX Partial Model 1" /></td>
</tr>
<tr>
<td><strong>SIMPLE BOEING</strong></td>
<td><img src="image4" alt="SIMPLE BOEING Full Model" /></td>
<td><img src="image5" alt="SIMPLE BOEING Partial Model 1" /></td>
</tr>
<tr>
<td><strong>PART 9</strong></td>
<td><img src="image7" alt="PART 9 Full Model" /></td>
<td><img src="image8" alt="PART 9 Partial Model 1" /></td>
</tr>
<tr>
<td><strong>PART 10</strong></td>
<td><img src="image10" alt="PART 10 Full Model" /></td>
<td><img src="image11" alt="PART 10 Partial Model 1" /></td>
</tr>
</tbody>
</table>

Figure 4: Extracted features from CAD models. Each extracted feature is assigned a separate color. Full models as well as partial models are presented in this figure.
<table>
<thead>
<tr>
<th></th>
<th>Full Model</th>
<th>Partial Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CIMPLEX</strong></td>
<td><img src="image" alt="Model" /></td>
<td><img src="image" alt="Model" /></td>
</tr>
<tr>
<td><strong>Simple Boeing</strong></td>
<td><img src="image" alt="Model" /></td>
<td><img src="image" alt="Model" /></td>
</tr>
<tr>
<td><strong>Part 9</strong></td>
<td><img src="image" alt="Model" /></td>
<td><img src="image" alt="Model" /></td>
</tr>
<tr>
<td><strong>Part 10</strong></td>
<td><img src="image" alt="Model" /></td>
<td><img src="image" alt="Model" /></td>
</tr>
</tbody>
</table>

Figure 5: 1% Gaussian Noise. Extracted features from CAD models with Gaussian noise applied to each point of the model. Each extracted feature is assigned a separate color. Full models as well as partial models are presented in this figure.
<table>
<thead>
<tr>
<th></th>
<th>Full Model</th>
<th>Partial Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CIMPLEX</strong></td>
<td><img src="image1" alt="CIMPLEX Full Model" /></td>
<td><img src="image2" alt="CIMPLEX Partial Model 1" /></td>
</tr>
<tr>
<td><strong>Simple Boeing</strong></td>
<td><img src="image4" alt="Simple Boeing Full Model" /></td>
<td><img src="image5" alt="Simple Boeing Partial Model 1" /></td>
</tr>
<tr>
<td><strong>Part 9</strong></td>
<td><img src="image7" alt="Part 9 Full Model" /></td>
<td><img src="image8" alt="Part 9 Partial Model 1" /></td>
</tr>
<tr>
<td><strong>Part 10</strong></td>
<td><img src="image10" alt="Part 10 Full Model" /></td>
<td><img src="image11" alt="Part 10 Partial Model 1" /></td>
</tr>
</tbody>
</table>

Figure 6: 2% Gaussian Noise. Extracted features from CAD models with Gaussian noise applied to each point of the model. Each extracted feature is assigned a separate color. Full models as well as partial models are presented in this figure.
3D DIGITIZER

scan one side of physical part
triangulate point-cloud

(a) Single Scan – take one scan from a single view

3D DIGITIZER

multiple scans of physical part
triangulate point-cloud

(b) Full Scan – take multiple scans from multiple views and register these scans together

Figure 7: Illustration of acquisition process.

<table>
<thead>
<tr>
<th>Full Scan</th>
<th>Single Scan</th>
<th>ACIS Model</th>
<th>Full Scan</th>
<th>Single Scan</th>
<th>ACIS Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Full Scan Image]</td>
<td>![Single Scan Image]</td>
<td>![ACIS Model Image]</td>
<td>![Full Scan Image]</td>
<td>![Single Scan Image]</td>
<td>![ACIS Model Image]</td>
</tr>
<tr>
<td>![Full Scan Image]</td>
<td>![Single Scan Image]</td>
<td>![ACIS Model Image]</td>
<td>![Full Scan Image]</td>
<td>![Single Scan Image]</td>
<td>![ACIS Model Image]</td>
</tr>
<tr>
<td>![Full Scan Image]</td>
<td>![Single Scan Image]</td>
<td>![ACIS Model Image]</td>
<td>![Full Scan Image]</td>
<td>![Single Scan Image]</td>
<td>![ACIS Model Image]</td>
</tr>
</tbody>
</table>

Figure 8: Correspondence of selected extracted features for the full scan models, single scan models and models obtained from exact representation (ACIS model).
5 Discussion

The next objective for this work is to introduce an efficient matching algorithm for partial similarity measures. From the above experiments we conclude that in order to perform successful matching, the technique must have the following properties: (1) tolerance for the noise that scanned data introduce; (2) ability to perform many-to-many matching, since it is possible that a feature could get divided into several features (Figure 9 gives an instance of such situation); (3) efficiency, so it can be used in the National Design Repository database 1. One of the main aspects of such matching technique is the distance function that assigns a numerical value to a pair of features. Previous work [9] successfully used such a function, which was based on area and Euclidean distance measurements within features. Please see Figure 10 for a sample view of two models with matched regions.

Other possible directions for Scale-Space work are to (1) explore techniques to extract features that resemble traditional CAD features; and (2) exploit the possibility of using Scale-Space features as signatures for indexing purposes.

6 Conclusions

This paper introduces a computationally practical approach, based on Scale-Space decomposition, to automatically segment 3D models in polyhedral representation into features that could be used for indexing, classification and matching. The decomposition is based on the local surface structure of a model. As a result similar features can be extracted in the presence of partial model information and noisy data. In this way, the technique has been shown to consistently segment partial 3D views, noisy geometry and the data (both partial and noisy) acquired by 3D laser range scanners.

One of the significant contributions of this work is to unite the notion of “feature” from the computer vision and graphics literature with the “features” of CAD/CAM. The specific measurement function behind the concept of features in this paper is highly tuned to the efficient identification of shape and topological categories. In one application, features obtained using our approach could be different from traditional CAD features and used to establish partial similarities between CAD models in polyhedral representation. We argue that the Scale-Space technique can be parameterized using different measurement functions, enabling it generate a variety of useful segmentations, including those that have semantic relevance to engineering and manufacturing properties. We believe that this locality-based feature representation can be used for 3D matching purposes, including partial matchings. Further, the Scale-Space decomposition technique is robust with respect to noise; therefore, it can be used on 3D models generated from 3D data acquisition devices, such as laser range scanners.

The Scale-Space approach developed and advanced in this paper creates a foundation for creating new approaches to existing problems in feature-based manufacturing. Foremost, one can argue that Scale-Space techniques can subsume all existing approaches to feature identification by parameterizing the decomposition of the surface on a model as a distance measure function. The concept of the measure function is highly generalizable, implying that all one needs to do is identify the measure function intrinsic to the class of features of interest and provide it as a parameter to the Scale-Space algorithm. Extending and enhancing the Scale-Space technique creates several research challenges. Because it is focused on local information, Scale-Space techniques have to be extended to capture features that have been subdivided through interactions. In addition, considerable work needs to be done to develop measure functions that map to well established engineering feature sets.

1http://www.designrepository.org
Figure 9: Illustration of many-to-many matching. Features from single scan model (on the left, shown in red) can not match features from full scan model (on the right, shown in red) individually, while the unions of the features (shown in green) match.

Figure 10: Results of matching between GOODPART and BADPART, with matched features (leafs of decomposition trees) having similar colors. Feature trees are obtained for each model using FEATURE-DECOMPOSITION($\mathcal{M}, k$) with geodesic distance function.

 Ultimately we believe that Scale-Space techniques provide important new capabilities that compliment existing approaches to feature identification and shape matching. With additional research, Scale-Space technique can become part of the solution to a number of important engineering problems.
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. 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


[56] Lou, K., Prabhakar, S., and Ramani, K., 2004. “Content-based three-dimensional engineering shape search”. In International Conference on Data Engineering.


