Clustering Solid Models for Database Storage

Mitchell Peabody, William C. Regli, David T. McWherter, Ali Shokoufandeh

Technical Report DU-MCS-01-04
Department of Mathematics and Computer Science
Drexel University
Philadelphia, PA 19104
June 2001

Clustering Solid Models for Database Storage

Mitchell Peabody [1] William C. Regli [2] David T. McWherter [3] Ali Shokoufandeh [4]

Geometric and Intelligent Computing Laboratory
Department of Mathematics and Computer Science
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104
http://gicl.mcs.drexel.edu/

[1]Email: mizhi@drexel.edu
[2]URL: http://www.mcs.drexel.edu/∼regli; Email: regli@drexel.edu.
[3]Email: udmcwher@mcs.drexel.edu
[4]Email: shokoufandeh@mcs.drexel.edu

Abstract

Computer-Aided Design (CAD) data and Solid Models are important tools in almost all the engineering disciplines, with an estimated 30 billion CAD models in the world; however, there are few good methods to manage the growing set of models.

CAD data, and the consumers of CAD data, have different requirements from traditional database media such as images, sounds, and videos. Solid models are large, contain components from multiple disciplines (mechanical, electrical, etc), and have no accepted set of readily computable features that can be used for similarity comparisons. These factors make the efficient storage and retrieval of solid models a non-trivial task.

This paper presents our approach for clustering large datasets of solid model designs with the intention of using our findings as a basis for a CAD and Solid Model database. We create a mapping of a solid model's boundary representation (B-Rep) and engineering attributes into *Model Signature Graphs* (MSG). From the MSGs, we create an *Invariant Topology Vector* (ITV) which captures shape and engineering properties of the CAD models in a vector of fixed size. We use a simple k-clustering algorithm to group the models and provide some of our empirical results to illustrate and qualitatively assess our approach.

We test our techniques against solid models in the National Design Repository; a large, publicly available resource which contains over 55,000 solid models from many different engineering disciplines and can be found at `http://www.designrepository.org`.

1 Introduction

Computer-Aided Design (CAD) data and Solid Models play an important role in the engineering disciplines with an estimated 30 billion CAD models existing in the world. While the database and data mining communities have made great strides in the management of image, audio and video media, there exists no good general method for storing, retrieving, and comparing solid models.

CAD data and solid models have different requirements from traditional multimedia databases. A solid model database will have to handle large individual data elements, components from different disciplines with unique access criteria, and a general lack of census on what features can be used for comparing solid models efficiently. These considerations make the problem a non-trivial one.

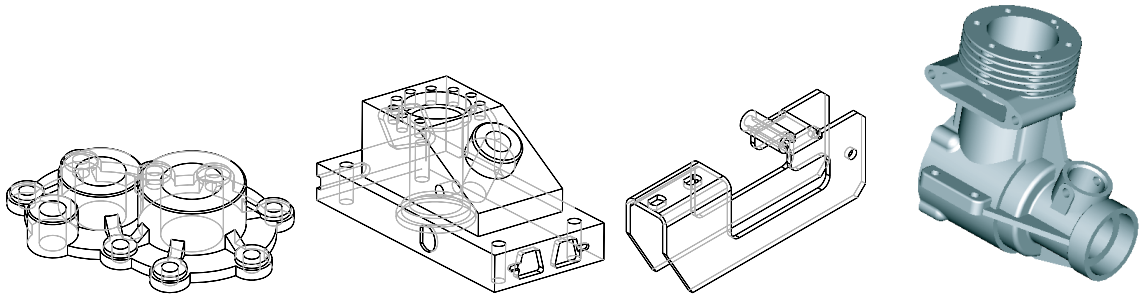
In this paper, we present our approach to clustering large databases of solid models and mechanical CAD designs for incorporation into future indexing and retrieval schemes. We describe the process by which we encode the solid models into *Model Signature Graphs* (MSG). From the MSGs, we create *Invariant Topology Vectors* (ITV), and then show the results of clustering the ITVs using a k-clustering algorithm. We believe our approach can form the basis for adapting database techniques to large engineering databases and building tools for handling 3D solid models as database media.

We use the National Design Repository (`http://www.designrepository.org`) to provide a large dataset of solid models. The repository contains over 55,000 publicly accessible solid models from multiple engineering disciplines such as those shown in Figure 1.

This paper is organized as follows: Section 2 provides a brief overview of the relevant background literature from engineering design, modeling, and databases communities. Section 3 introduces our technical approach to mapping solid models to MSGs, transforming the MSGs to ITVs, and k-clustering the ITVs. Section 4 presents some of our experimental results with the National Design Repository. Finally, Section 5 gives our conclusions and discusses our research contributions and areas for future work.

2 Background and Related Work

The scope of our work draws together several areas of study, solid modeling and CAD, engineering databases, metric spaces, graph theory, and data clustering. Section 2.1 begins with a discussion of the types of techniques used to represent CAD data and solid models. In section 2.2 we describe various techniques in use for retrieval of CAD data from databases. And finally, we discuss database clustering in Section 2.3.



(a) Machined Part (83KB) (b) ANC101 Benchmark Part (75KB) (c) Sheet Metal Part (44KB) (d) Cast Compressor Housing (637KB)

Figure 1: A few example 3D solid models from the Design Repository; (a), (b), and (c) are wire-frame representations while (d) is a rendered representation.

2.1 Solid Modeling and Computer-Aided Design

There are three broad schemes for representation of solid models [16, 17]:

1. **Decomposition** approaches model a solid as collections of connected primitive objects. Data structures used in this type of representation include quad-trees and oct-trees [18].
2. **Constructive** approaches model a solid as a combination of primitive solid objects. A common approach is *constructive solid geometry* (CSG), which represents a solid as a boolean expression on a set of primitive solids.
3. **Boundary-based** approaches model a solid using a data structure that represents the geometry and topology of its bounding faces. Recently, the *boundary-representation* (BRep) approach has become the representation of choice in solid modeling in large part due to the flexibility and power made available to developers.

A BRep solid model creates a unique and unambiguous representation of the exact shape of an object. BReps have become the dominant representation schema for modern solid modelers used in mechanical design. They are used for engineering analysis, simulation, collision detection, animation and manufacturing planning.

A BRep usually consists of a graphical structure that models an entity's topology. Connections between nodes in the BRep graph represent connections between topological components of the entity's boundary.

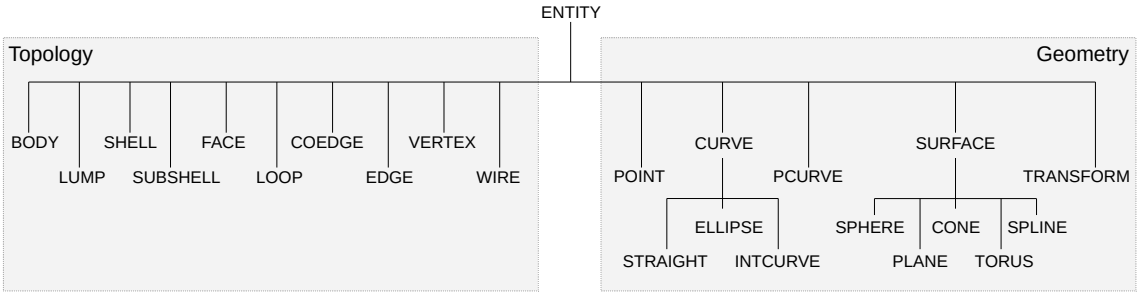


Figure 2: Distinction between geometric and topological information in the BRep used by ACIS [19].

These topology nodes contain references to their underlying geometric entities; for example, a *face* of a solid is a topological entity (represented as a collection of bounding edges) that has a *surface* associated with it. Figure 2 shows the distinction between geometric and topological information from the ACIS Solid Modeling Kernel. For more information on boundary representation data structures, interested readers are referred to [9, 16, 21, 22].

BReps and other CAD representations are distinctly different from shape models developed by the computer vision community in several important ways. Vision-based representations, such as super quadric and deformable shape models, are not designed for exact modeling of shapes—rather they are employed to reconstruct shapes based on approximate data taken from sensors and cameras. These approximated shapes can be used for a basis of comparison among 3D objects, however such comparisons are limited to analysis of geometric moments and gross shape properties. Hence, they are not directly suitable for use in answering the kinds of interrogations that design and manufacturing engineers wish to pose about CAD models.

2.2 Engineering Databases

In 2D shape matching and image retrieval, the indexing and query process follows one of four common approaches:

1. A **Textual** query based on keywords stored for each image in a database.
2. A query **by example** uses similarity measures derived off a set of query images provided as input.
3. Query **by sketch** looks for image segments matching the sketched profile.

4. **Iconic queries** use templates representing important aspects of the desired image to identify images with similar features.

Techniques from computer vision and 2D image retrieval do not directly apply to 3D solid models for several reasons of which we will address the three main ones. First, when we are using solid models, we are dealing with an explicit and exact representation of the 3D objects in CAD databases, rather than an approximate representation generated from sensor data.

Second, given a solid model A; minor perturbations in A which create A', can make A' indistinguishable from A, even though they are still very similar. Transformations such as scaling or rotation can introduce roundoff errors that exacerbate the problems posed by the inexact nature of floating point arithmetic. A simple translation of a model in 3D space by a fixed vector can make models computationally costly to compare.

Finally, there is no universally acceptable set of features on which model comparisons can be based. Evaluation metrics for solid models depend on the application intent of the engineers using the database. Process engineers may need to query based on manufacturing process features but an industrial designer may need to consider shape aspects of an object. The ill-defined semantics of the engineering design and manufacturing domain require us create a more flexible methodology for model comparisons which can be customized to consider the criteria of required by different end users.

2.3 Database Clustering

Agrawal et al. [2] describe clustering as "...task that seeks to identify homogeneous groups of objects based on the values of their attributes (dimensions)...". More formally, given a set S of data entities, a cluster is roughly defined as a subset, $T \subseteq S$ such that each of the data entities in T is "similar" or "close to" to the other data entities in T. In a clustering, $\mathcal{C} = \{T_1, T_2, T_3, ..., T_n\}$, the elements in T_j are more similar to each other than the elements in T_k.

Fasulo [6] describes a popular method of partition clustering called *k-clustering.* k-clustering takes a set of objects, S, and an integer k and outputs a partition of S into subsets T_1, T_2, . . . , T_k such that objects in a cluster are more similar to each other than to objects in different clusters. According to Jain et al [11], the general problem is known to be NP-Hard, so to make the problem tractable, heuristics are employed. These heuristics construct a representative object for a cluster and use this for the distance calculations. Some heuristics in use are the *k-means* algorithm by MacQueen [15], the *k-medioid* algorithm by Kaufman et al [13], and an algorithm by Gonzalez [7].

These all belong to a class of clustering methods that rely on iterative partitional clustering. These algorithms operate in a similar fashion to a gradient descent search. They begin with an initial partition, on each iteration they make slight modifications to the partition to improve a *criterion function.* When the algorithm has iterated a certain number of times the small clusters are merged, large clusters are split, and outlying points are removed. The criterion function indicates how good the overall partition is; a common one is the *total squared error*.

There are three major weaknesses in the described k-clustering methods. One, They favor spherical clusters, not general shapes. Two, they don't deal with noise. And finally, when points are in high dimensions, they tend to be much more spread out making the identification of clusters more difficult. These weaknesses have been addressed by Agrawal et al [2] and Ester et al [5]. The positive aspect of these methods is that they are relatively easy to implement and have been studied extensively.

3 Technical Approach

Our goal is to develop techniques to store solid models in a database and perform efficient search and retrieval of these models. In clustering the models, we accomplish two things. We show that the topological characteristics of solid models can be used as a basis for inexact similarity measures. In addition, we demonstrate the feasibility of incorporating these techniques in a traditional database setting.

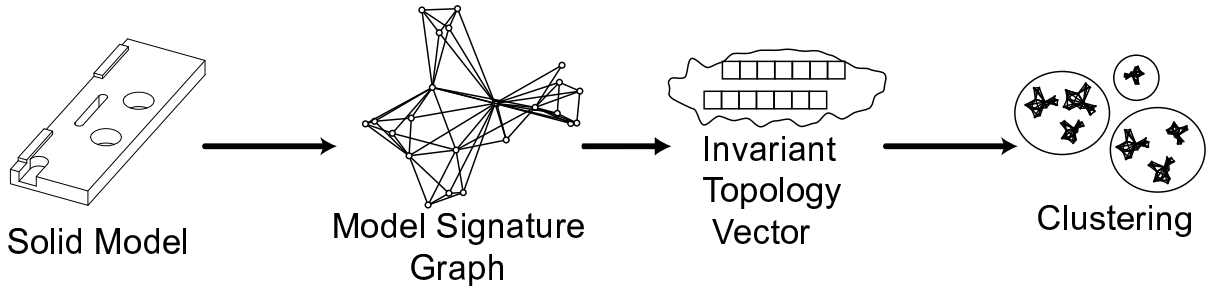
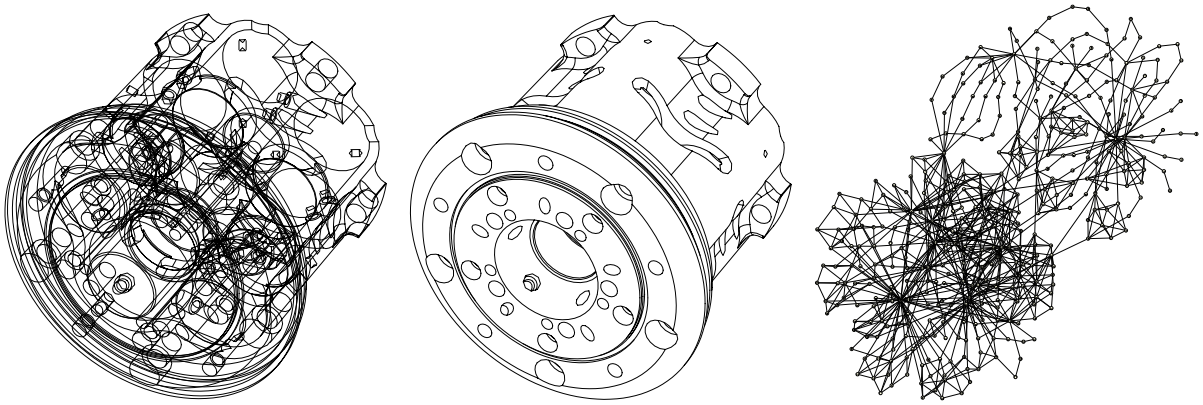


Figure 3: The transformation from solid model to Invariant Topology Vector used in clustering.



(a) Wireframe (b) Hidden Lines Removed (c) Model Signature Graph

Figure 4: Representations of a *Torpedo Motor* (768KB) as a wire solid model and MSG.

3.1 Problem Formulation

One of the main challenges in comparing solid models is finding a representation of the model in a computer usable data structure. Ideally, this data-structure will approximate the overall structure of the model as closely as possible while allowing the application of well researched and developed algorithms to perform similarity comparisons. We would further like to encode domain specific information within this data-structure to exploit any additional data available.

To this end, we have developed a data structure called a *Model Signature Graph* (MSG), the generation of which is described in Section 3.2. The MSG allows us to view the topology of a model in abstract terms and to encode information about the model itself. To prepare the MSG for clustering we generate a fixed length vector, called an *Invariant Topology Vector* (ITV), defined in Section 3.4. The ITV contains the properties we wish to exploit, and allows us to define a metric space for the models. In Section 3.5, we cluster the MSGs based on this metric space using the k-means algorithm. The overall strategy is illustrated in Figure 3.

3.2 Model Signature Graphs

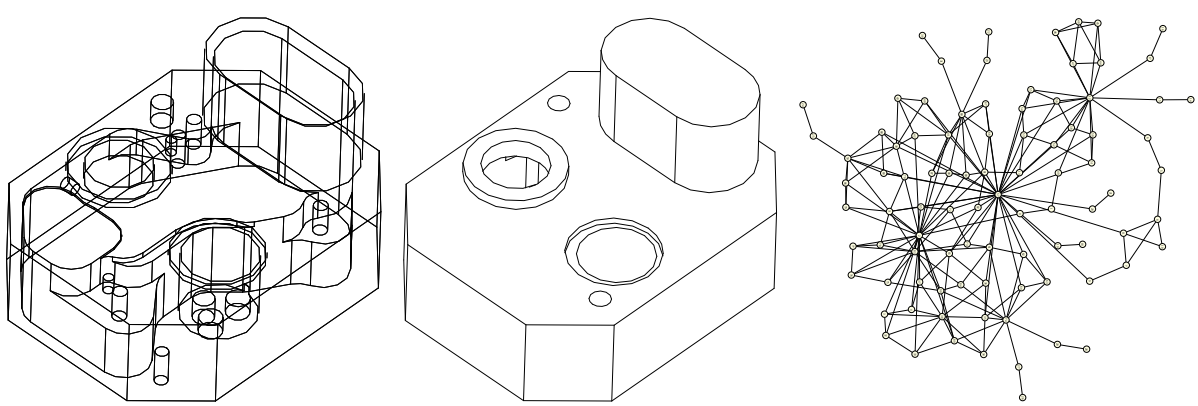
To represent solid models abstractly, we have created a data-structure called a *Model Signature Graph* (MSG), an example of which is shown in Figure 4. The MSG is constructed from the BRep of a solid model in a manner similar to that by Wysk et al. [20] and the *Attributed Adjacency Graph* (AAG) structures used to perform Feature Recognition from solid models [12].

The BRep [10] essentially consists of a set of edges and a set of faces used by the solid modeler to describe the shape of the model in 3-dimensional space. A MSG for a solid model P is defined as a labeled graph, $G = (V, E)$, where each face $f \in P$ is represented as a vertex $v \in V$, with attributes that further describe the qualities of the face in the model. Some of these attributes are listed here:

1. the surface type of the face;
2. the convexity or concavity of the face;
3. surface area, f_{area}, for the face.

A MSG edge $(f_1, f_2) \in E$ exists whenever two faces $f_1, f_2 \in P$ are adjacent—i.e., when f_1 and f_2 meet at a topological edge e in the BRep. The MSG edge $e_msg = (f_1, f_2)$ is attributed with features, some of which are listed below:

1. the type of Brep edge, e in P;



(a) Wireframe (b) Hidden Lines Removed (c) Model Signature Graph

Figure 5: Representations of an Allied Signal part as a wire solid model and MSG.

2. concavity/convexity of e;
3. the length of the curve of e.

A simple example of the encoding of a solid model into a model signature graph is shown in Figure 5. As can be seen by Figures 4 and 5, our representation can be used on a variety of models, regardless of complexity. In addition, the MSG is unambiguous with respect to a model's rotation and scale; it is only dependent on the connection of the faces of the model.

3.3 Metric Spaces of Solid Models

Once the the solid models have been transformed into MSGs, we construct a metric space over the dataset. A *metric space* is a collection of objects with a distance function, $\delta(x, y)$, known as the *metric*, which computes the distance between any two elements in the set. The distance function $\delta(x, y)$ must satisfy the following conditions:

$$\begin{array}{rcl} \delta(x,y) = 0 \iff x = y & : & \text{Identity} \\ \delta(x,y) \geq 0 & : & \text{Positivity} \\ \delta(x,y) = \delta(y,x) & : & \text{Symmetry} \\ \delta(x,y) + \delta(y,z) \geq \delta(x,z) & : & \text{Triangle Equality} \end{array}$$

Clustering techniques such as k-means or k-medioid make use of use of metric distance functions to organize datasets. Metric or near-metric distance functions for solid models have the potential for use in a wide range of applications.

To cluster the models in the database, we must construct a distance measure that operates over the space defined by the MSGs and satisfies the requirements for a metric. There is a very good chance that more than one distance function exists, leading to a number of different metric spaces that can be constructed over the same database of models. This will also lead to different clustering results, the quality of which must be examined experimentally.

To make use of distance metrics in a practical clustering system, they must be relatively easy to compute. The algorithmic complexity in computing distance metrics for arbitrarily-formed graph structures is an open question.

Proven distance metrics for graph similarity fall roughly into the category of *edit distance.* Given two graphs, G and H, edit distance is loosely defined as the number of operations needed to transform G into H. Bunke [3] and Kubicka [14] show that different forms of edit distance satisfy the criterion for a metric

distance. What this means that if $edit - distance(G, H) = 0$ then G is isomorphic H. The catch is that neither edit distance is polynomial time computable and for large graphs, which are typically generated from solid models, the metric is simply unusable in a practical setting.

3.4 Invariant Topology Vectors

Although, a measure such as edit distance provides a proven metric for graph similarity, the cost of running the algorithm is much too high for a database that could potentially consist of thousands of designs. Using graph invariants, we can form groups, or clusters of potentially isomorphic graphs, or graphs that are close enough in similarity to satisfy a query. We capture these invariants in an *Invariant Topology Vector* (ITV).

Given a graph G, a graph invariant is a property of G, denoted $X(G)$, such that if a graph H is isomorphic to G, then $X(G) = X(H)$. A complete set of invariants, $\{X_1(G), X_2(G), ..., X_n(G)\}$, determines G and H up to isomorphism. So if, $X_i(G) = X_i(H), i = 1..n$ then G is isomorphic to H. No complete set of invariants is known. A set of graph invariants can still be used to tell if two graphs have the potential to be isomorphic or not, but can not be used to give a definitive answer. In other words, if the invariants of two graphs are different, there is no way the graphs can be isomorphic; if the invariants are all equal, then the graphs may or may not be isomorphic. We believe an approximate solution can be realized through the use of selected graph invariants.

The graph invariants that we focus on are directly related to the degree sequence of a graph. The degree sequence, while not the most powerful graph invariant known, is easily computable, and can be exploited to recover general structural properties of the graph. At this stage, we have not attempted to construct these invariants such that they are normalized, so that differences of scale in the number of edges or vertices might skew results. In addition we have not investigated correlations between the invariants in order to determine their interdependencies. The following graph invariants were chosen for inclusion in our system:

1. **Vertex and Edge Count**—Trivially computable graph invariants. Unfortunately, they reveal practically no information about the graph structure, and may be weighted too strongly in the comparisons we perform, skewing our similarity metrics.

2. **Minimum and Maximum Degree**—Provide a measurement of the maximum and minimum number of adjacencies between one face and the other faces in the model.

3. **Median and Mode Degree**—Provides a measurement that represents statistically the most common and "average" cardinalities of adjacencies between faces in a model.

4. **Diameter** — The longest path of edge/face crossings between any two faces in the model.

5. **Type Histogram** — Each model BRep contains a set of faces and edges which have various types and features. The analysis and statistical breakdown of these component types is used to further generate a representation of the model.

Using graph invariants helps prune out graphs which have no similarity to a given graph and does not depend on the specific domain where the technique is applied. However, we would like to be able to tailor it in some fashion to a particular domain to aid in more precise measurements. The type histogram is an initial approach to incorporating domain specific information into the ITV.

When generating the MSG from the model, we put attributes in the vertices and edges. We use these values in our ITV to give even finer granularity in our assessment. However, since there are varying numbers of vertices and edges in the graphs, the labels can not be used directly. The ACIS Solid Modeler from Spatial Technologies is the modeling system used in our experiments. ACIS has 13 different surface types, and 8 different curve types. We modify the definition of the ITV to include 21 additional values. In effect, we create a histogram of the number of times particular surface and edge types occur in the model. With this definition, we can use the ITV to determine a position for a graph in the metric space relative to other graphs. Graphs with similar values for invariants will lie in the same region of the metric space.

Thus, the ITV is a point in 29 dimensions and when combined with a set of these points, will enable the clustering of the points in this metric space. Table 1 lists the graph invariants and type histogram used in generating the ITV for the torpedo motor illustrated in Figure 4. The ITV that results from these values is simply the result after concatenating the numbers into a single vector:

$$\vec{v} = \langle 392, 967, 1, 39, 4, 4.00, 4.04, 24, 202, 0, 0, 0, 0, 0, 119, 0, 4, 5, 0, 0, 62 \rangle$$

Invariant Rationale Until now, the discussion has been about what graph invariants were chosen for the ITV. However, no rationale has been given in regards to why the graph invariants were chosen for the ITV. Ideally, the ITV should contain invariants that are independent. This, however, is not the case. Values in some invariants will cause other invariants to be constrained to certain values. A trivial example of this is edge count and connectivity. Given $G = (V, E)$, if G is connected, then $|E| >= |V| - 1$. Another issue is that not all invariants are polynomial time computable, thus negating their usefulness for this application. Furthermore, many of those invariants that are polynomial time computable are difficult to represent in a fixed width vector. An example of this would be the degree sequence itself; the size varies directly with the number of edges in the graph.

Another characteristic of an ideal invariant is to use one that will catch dissimilarities in a unique way from the other invariants. This type of determination is rather difficult since it would involve the manual

Graph Invariants		Surface Type Counts		Edge Type Counts	
Attribute	**Value**	**Type #**	**Count**	**Type #**	**Count**
Vertex Count	403	1	231	1	464
Edge Count	939	2	0	2	246
Degree Max	1	3	0	3	0
Degree Min	46	4	0	4	0
Degree Mode	2	5	0	5	0
Degree Median	4.00	6	0	6	0
Degree Std. Deviation	4.15	7	117	7	0
Graph Diameter	13	8	0	8	229
		9	0		
		10	10		
		11	0		
		12	0		
		13	45		

Table 1: Properties in the Invariant Topology Vector for the torpedo motor in figure 4

creation of a pristine dataset. Assuming that this dataset can be created, the use of artificial neural networks could be used to determine how important the individual invariants are in differentiating models.

3.5 Clustering Method

Since the goal of this paper is to show that clustering based on invariants is a useful means to an end, a simple clustering method was chosen to illustrate feasibility. Run-time and high cluster quality are not issues at this point, the primary interest being whether or not the clustering provides suitable results for further research. The clustering method chosen is a simple version of *k-means* clustering introduced by MacQueen [15].

The algorithm seeks to produce a clustering, K, of $|K|$ clusters such that an object in a cluster c_k is more similar to the other objects within the cluster than it is to objects in other clusters.

Representative Vector The *representative vector* of a cluster, c_k, is defined to be the point that resides at the middle of all the points in c_k, or the mean of each of the points in the cluster. More formally, if

$c_k = \{\vec{v_1}, \vec{v_2}, \ldots, \vec{v_n}\}$ is a cluster with $n = |c_k|$ vectors where a vector is $\vec{v_i}$, then the representative vector, $rep(c_k)$, of c_k is $rep(c_k) = 1/n \sum_{i=1}^{n} \vec{v_i}$.

Square Error The *square error*, or within cluster variation, is a measure of how close the points are to the representative vector of the cluster. The square error is the sum of the squared Euclidean distances between each point in c_k and its representative vector. Using the square error of a cluster, we can define a measure of the overall quality for a particular clustering, or the total square error. The mathematical definition is $e_k^2 = \sum_{i=1}^{n} (\vec{v_i} - rep(c_k))^T (\vec{v_i} - rep(c_k))$, where $k = 1 \ldots |K|$ and $rep(c_k)$ is the representative vector for c_k.

Total Square Error The *total square error* of a clustering, K, is defined to be the sum of the square errors for each cluster, c_k in K. In mathematical terms, $E^2 = \sum_{k=1}^{|K|} e_k^2$.

Given this formula, the goal of the algorithm presented is to minimize the total square error for K. Since k-clustering requires a distance function in order to group the objects, we define the distance function we use to be the Euclidean distance given by the formula, $dist(\vec{u}, \vec{v}) = \sqrt{(\vec{u} - \vec{v})^T (\vec{u} - \vec{v})}$.

Where u and v are ITVs representing models in the metric space, we call this distance an $ITV Distance$. This is not a true distance metric since the identity requirement of a distance metric does not hold. There is the possibility that two ITVs are the same, but the models are not. Given this limitation, we believe that the $ITV Distance$ can still be a useful approximation.

k-clustering algorithm The k-clustering algorithm is based on the algorithm presented by MacQueen [15]. The clusters are seeded with an initial set of ITVs, next the rest of the ITVs are placed into clusters according to the Euclidean distance function, finally the clustering is improved by selecting outlying points and attempting to re-cluster them. The algorithm is presented below in greater detail.

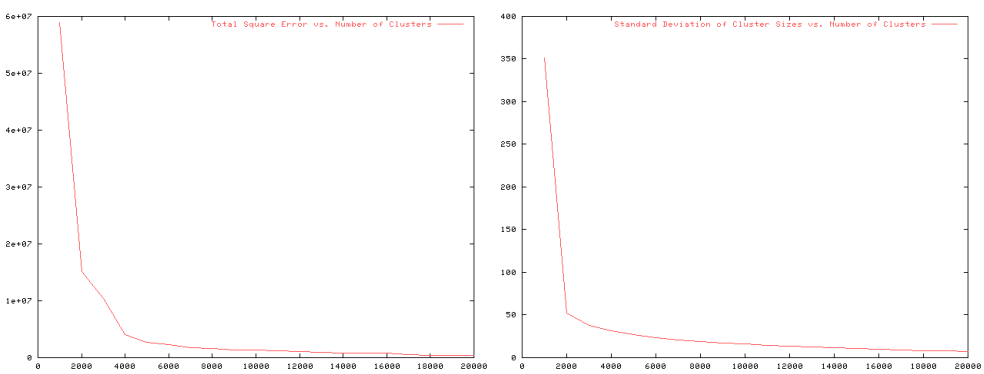
1. $|K|$ clusters are created and seeded with $|K|$ ITVs.
2. Select an unclustered ITV, $\vec{v_i}$.
3. Find cluster, c_k, whose representative object is closest to $\vec{v_i}$, and insert $\vec{v_i}$ into c_k.
4. Recalculate the *representative vector*, $rep(c_k)$, for c_k.
5. Repeat steps 3-5 until no more unclustered ITVs remain.
6. Select two clusters, c_k and c'_k with the largest *Square Error*.

7. Select two ITVs, $\vec{v_1}$ from c_1 and $\vec{v_2}$ from c_2, furthest from the *representative vector*.
8. Swap $\vec{v_1}$ and $\vec{v_2}$ iff the swap will reduce the *total square error* of the clustering.
9. Repeat steps 6-8 for s iterations.

4 Experimental Results

k	Min Size	Max Size	Standard Deviation	Time (s)	E^2
1000	1	6524	351.33	36.08	5.89e7
5000	1	1591	26.91	220.52	2.74e6
10000	1	1330	15.80	403.04	1.30e6
15000	1	1081	10.63	501.99	7.96e5
20000	1	868	7.53	532.86	4.51e5

Table 2: Results on running clustering on models in the repository for various k.



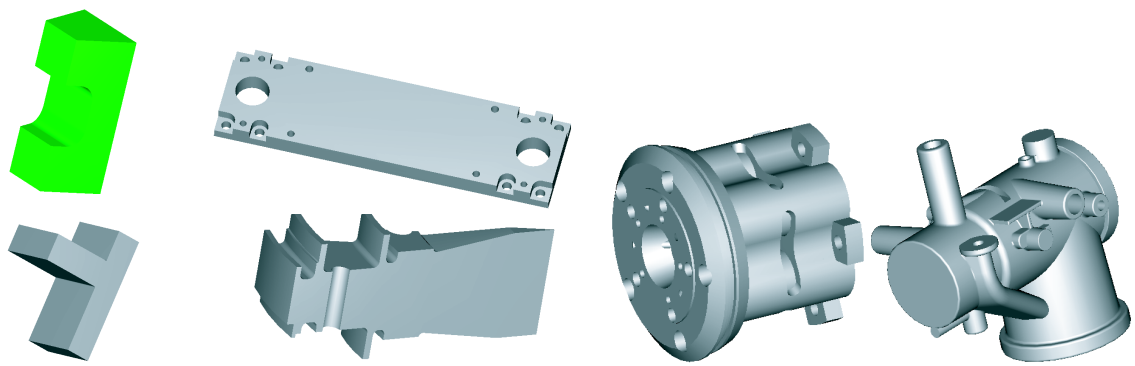
(a) Total Square Error.

(b) Standard Deviation.

Figure 6: Plots of measures taken from several clustering runs.

We ran the clustering algorithm for $k = 1000 \ldots 20000$ and took various measurements dealing with the quality of the clustering. Table 2 lists the size of the clusterings alongside relevant statistical data obtained.

Specifically, the things we were looking for were the size of the total square error and the standard deviation from the median cluster size. We are interested in the total square error, a graph of which is shown in Figure 6(a), since this measurement shows us how good the clustering is and can be used to improve the quality of the clustering. The standard deviation, shown in Figure 6(b) of the cluster size measures the uniformity of the size of the clusters; if the clustering is good, the standard deviation should be small.



(a) Two members of cluster 1000-cluster k-clustering on the Design Repository.

(b) Two members of the same cluster, after performing a 3000-cluster k-clustering on the Design Repository.

(c) Members of the cluster containing the Torpedo Motor model, after performing a 20000-cluster k-clustering on the Design Repository.

Figure 7: Examples of figures clustered together.

Interpretation of results One of the inherent difficulties in judging the quality of a similarity measure for complex objects is the definition of just what makes two distinct objects more similar to eachother than to other objects. We have already shown that the Euclidean distance between two ITVs loosely satisfies the requirements of a metric; however, this measure is without meaning if the models are not increasingly similar the closer the two ITVs lie together. This seem self-contradicting since we have already stated that this measure is inexact by its very nature. While quality of the clustering is important, our main interest lies in the clusters themselves. We used a more subjective testing methodology. We chose various values of k and selected random clusters from which we pulled random models to compare visually. Some of the results are shown in Figure 7.

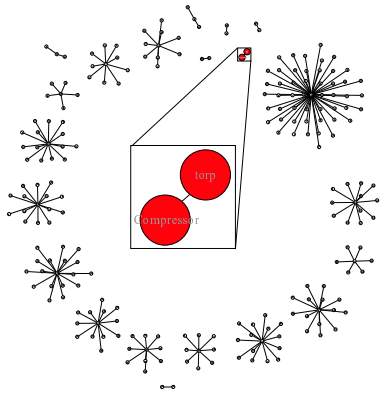


Figure 8: Graph Representation of clustering for $k = 20000$.

One model we focused particularly on is the Torpedo Motor shown in Figure 4. This model is complex and unique enough to make finding a similar model a daunting task to do manually. We were interested in what models might be clustered with it and how closely they looked to eachother visually. We chose $k = 20000$ since for that number of clusters on a relatively small dataset of 40,000 models, the models will tend to find their own clusters. As can be seen from Figure 7(b), the model the Torpedo Motor was eventually clustered with is a model of a Compressor and is not too different topologically. In this particular clustering, this cluster had only two parts. The other parts shown in Figure 7 are also very similar to eachother, but are very simple compared to the two other models.

Visualizing the clustering is a non-trivial task, we decided to represent the clusters as a graph; Figure 8 shows an example of the representation and continues the running example with the Torpedo Motor clustering with the Compressor. The cluster in which the Torpedo Motor appears with the Compressor is highlighted in red. Each cluster is represented by a connected component in the graph, where a node in a connected component represents a model in the corresponding cluster. The graph does not show the clusters' relative positions in the metric space, but rather illustrates the relative distribution of the models in the clustering.

5 Conclusions

This paper presented a method of transforming solid model representations into a data-structure that could be used as an approximation of a similarity measurement. To show the appropriateness of the technique, a

clustering method was used to demonstrate that similar models do appear close to eachother in the vector space defined by *Invariant Topology Vectors*.

Our *Model Signature Graph* approach can be an effective formalism for topological similarity assessment of solid models and enable clustering for data mining of large Design Repositories. Our empirical results show that Invariant Topology Vectors can be used to create semantically meaningful metric spaces in which solid models can be compared and clustered. We believe this work will begin to bridge the solid modeling and database communities, enabling new paradigms for interrogation of solid models.

Future Work Currently, we are considering plain solid models. In future, we believe that additional domain knowledge can be used to refine our techniques. Information about engineering tolerances, surface finishes, constraints and parametrics, etc. all can be used to augment the basic techniques presented here. Information about the feature locations, dimensions, and orientations have to be exploited fully. By leveraging some earlier work [4], we hope to incorporate additional feature attribute information into our data structures for clustering and eventual comparison and indexing of the solid models in relational databases. We believe that the attribute lists that are used in the ITVs may be further refined to eliminate interdependence of attributes and identify discriminating attributes. Other graph invariants may also be developed or exploited. Also very important will be the development of a weighting of ITV attributes to minimize the effects of attributes which contain little information.

Finally, the clustering technique used, k-means, is admittedly naive and limited. The algorithm can only generate clusters of a spherical shape and runs into problems when noise is introduced. Furthermore, according to [2], clustering high-dimensional data is ineffective with a simple distance function since the points will tend to be spread out. More effective clustering methods, such as the CURE algorithm by [8] and CLIQUE by [2] need to be explored in order to further qualify our technique.

Acknowledgments. This work was supported in part by National Science Foundation (NSF) Knowledge and Distributed Intelligence in the Information Age (KDI) Initiative Grant CISE/IIS-9873005; CAREER Award CISE/IIS-9733545 and Grant ENG/DMI-9713718.

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] National design repository. `http://www.designrepository.org`., 2000.

[2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data*. ACM, June 1998. file-name: sigmod98_clique.pdf.

[3] Horst Bunke and Kim Shearer. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, pages 689–694, 1997.

[4] Alexei Elinson, Dana S. Nau, and William C. Regli. Feature-based similarity assessment of solid models. In Christoph Hoffman and Wim Bronsvoort, editors, *Fourth Symposium on Solid Modeling and Applications*, pages 297–310, New York, NY, USA, May 14-16 1997. ACM, ACM Press. Atlanta, GA.

[5] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, 1996.

[6] Daniel Fasulo. An analysis of recent work on clustering algorithms. Web??, April 1999. http://citeseer.nj.nec.com/fasulo99analysi.html.

[7] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[8] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. ACM-SIGMOD, 1998. file-name: sigmod-1998-195.ps.

[9] Christoph M. Hoffman. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers Incorporated, CA, 1989.

[10] Christopher M. Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, Inc., California, USA, 1989.

[11] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1988.

[12] S. Joshi and T. C. Chang. Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer-Aided Design*, 20(2):58–66, March 1988.

[13] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons, Inc., 1990.

[14] Ewa Kubicka, Grzegorz Kubicki, and Ignatios Vakalis. Using graph distance in object recognition. In *Proceedings of the 1990 ACM annual conference on Cooperation*, pages 43–48, February 1990.

[15] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium*, pages 281–297, 1967.

[16] Martti Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, College Park, MD, 1988.

[17] Aristides A. G. Requicha. Representation for rigid solids: Theory, methods, and systems. *Computing Surveys*, 12(4):437–464, December 1980.

[18] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(3):287–260, 1984.

[19] Spatial Technology Inc., Three-Space Ltd., and Applied Geometry Corp., 2425 55th Street, Building A, Boulder, CO 80301. *ACIS© Geometric Modeler Application Guide*, v1.6 edition, November 1994.

[20] Tien-Lung Sun, Chuan-Jun Su, Richard J. Mayer, and Richard A. Wysk. Shape similarity assessment of mechanical parts based on solid models. In Rajit Gadh, editor, *ASME Design for Manufacturing Conference, Symposium on Computer Integrated Concurrent Design*, pages 953–962. ASME, Boston, MA. September 17-21. 1995.

[21] Kevin Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics and Applications*, 5(1):21–40, January 1985.

[22] Tony C. Woo. A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics and Applications*, pages 19–27, March 1985.