Proc. ACM SIGMOD 2003 Int. Conf. on Management of Data, San Diego, CA, 2003

Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects

Hans-Peter Kriegel, Stefan Brecheisen, Peer Kröger, Martin Pfeifle, Matthias Schubert

Institute for Computer Science
University of Munich
Oettingenstr. 67, 80538 Munich, Germany
{kriegel | brecheis | kroegerp | pfeifle | schubert}@dbs.informatik.uni-muenchen.de

ABSTRACT

In modern application domains such as multimedia, molecular biology and medical imaging, similarity search in database systems is becoming an increasingly important task. Especially for CAD applications, suitable similarity models can help to reduce the cost of developing and producing new parts by maximizing the reuse of existing parts. Most of the existing similarity models are based on feature vectors. In this paper, we shortly review three models which pursue this paradigm. Based on the most promising of these three models, we explain how sets of feature vectors can be used for more effective and still efficient similarity search. We first introduce an intuitive distance measure on sets of feature vectors together with an algorithm for its efficient computation. Furthermore, we present a method for accelerating the processing of similarity queries on vector set data. The experimental evaluation is based on two real world test data sets and points out that our new similarity approach yields more meaningful results in comparatively short time.

1. INTRODUCTION

In the last ten years, an increasing number of database applications has emerged for which efficient and effective support for similarity search is substantial. The importance of similarity search grows in application areas such as multimedia, medical imaging, molecular biology, computer aided engineering, marketing and purchasing assistance, etc. [15, 1, 24, 13, 14, 2, 7, 9, 18]. Particularly, the task of finding similar shapes in 2-D and 3-D becomes more and more important. Examples for new applications that require the retrieval of similar 3-D objects include databases for molecular biology, medical imaging and computer aided design.

Especially, the development, design, manufacturing and maintenance of modern engineering products is a very expensive and complex task. Effective similarity models are required for two- and three-dimensional CAD applications to cope with rapidly growing amounts of data. Shorter product cycles and a greater diversity of models are becoming decisive competitive factors in the hard-fought automobile and aircraft market. These demands can only be met if the engineers have an overview of already existing CAD parts. In this paper, we introduce an effective and flexible similarity model for complex 3-D CAD data, which helps to find and group similar parts. This model is particularly suitable for voxelized data, which often occur in CAD applications. It is not based on the traditional approach of describing one object by a single feature vector but instead we map an object onto a *set of feature vectors*, i.e. an object is described by a *point set*.

The remainder of the paper is organized as follows: In Section 2 we shortly review already existing spatial similarity models and provide a classification of the techniques into feature-based models and direct geometric models. Section 3 provides the basis for similarity models based on voxelized CAD objects. We address the issues of translation, rotation, reflection and scaling invariances. Furthermore we adapt three known similarity models to voxelized 3-D data. Based on the most promising of these three models, we explain in Section 4 our new approach based on sets of feature vectors. In Section 5, we analyze the different similarity models by means of hierarchical clustering. We show that our new similarity approach efficiently generates more significant results compared to the traditional approaches based on single feature vectors. The experiments are based on two real-world test data sets of our industrial partners, a German car manufacturer and an American aircraft producer. The paper concludes in Section 6 with a short summary and a few remarks on future work.

2. RELATED WORK

In recent years, considerable work on similarity search in database systems has been published. Many of the previous approaches, however, deal with 1-D or 2-D data, such as time series, digital images or polygonal data. Most of them do not support 3-D objects or are not suitable for voxelized data. In this section, we shortly list different approaches to establish similarity measures. We provide a classification of the techniques into feature-based models and direct

SIGMOD 2003, June 9-12, 2003, San Diego, CA.

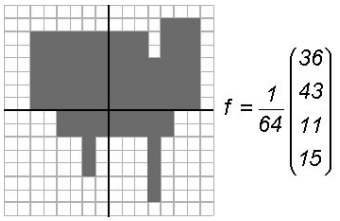


Figure 1: Space partitioning with 4 cells. The feature vector generated by the volume model is depicted on the right hand side.

geometric models.

2.1 Feature-Based Similarity

A widely used class of similarity models is based on the paradigm of feature vectors. The basic idea is as follows: Using a feature transform, the objects are mapped onto a feature vector in an appropriate multidimensional feature space. The similarity of two objects is then defined as the proximity of their feature vectors in the feature space: The closer their feature vectors are located, the more similar two objects are considered.

The paradigm of feature-based similarity has been successfully applied to the retrieval of similar spatial objects. Examples include structural features of 2-D contours [24], angular profiles of polygons [5], rectangular covers of shapes [15], algebraic moment invariants [13], 2-D section coding [9, 7], and 3-D shape histograms for biomolecular objects [4]. Non-geometric applications include similarity search on time series [1, 14, 2], and on color histograms in image databases [26, 13], among several others.

2.2 Geometry-Based Similarity

A class of models that is to be distinguished from the feature-based techniques are the similarity models that are defined by directly using the geometry of the objects. Two objects are considered similar if they minimize a distance criterion that is purely defined by the geometry of the objects. Examples include the similarity retrieval of mechanical parts [28], the difference volume approach [19, 18], and the approximation-based similarity model for 3-D surface segments [21].

3. SIMILARITY MODELS FOR VOXELIZED CAD OBJECTS

In this section, we describe three established similarity models. The first two models (the *volume* and the *solid-angle* approach) are based on an axis parallel, equi-sized space partitioning. The voxel approximations of the objects are then transformed into shape histograms. These histograms are used as intuitive feature vectors. In the third model (the *cover sequence* approach), we do not need this space partitioning but obtain our feature vectors directly from the rectangular covers which approximate our object by minimizing the symmetric volume difference. This third model forms the starting point for our new approach based on vector sets which is introduced in Section 4.

3.1 Shape Histograms

Histograms are usually based on a complete partitioning of the data space into disjoint cells which correspond to the bins of the histograms.

We divide the data space into axis parallel, equi-sized partitions (cf. Figure 1). This kind of space partitioning is especially suitable for voxelized data, as cells and voxels are of the same shape, i.e. cells can be regarded as coarse voxels.

Each of these partitions is assigned to one or several bins in a histogram, depending on the specific similarity model. By scaling the number of partitions, the number of dimensions of the feature vector can be regulated (cf. Figure 1). Obviously, the more partitions we use, the more smaller differences between the objects become decisive.

By means of the resulting feature vectors, the similarity of two objects can be defined as follows.

Definition 1. (Feature-Based Object Similarity)
Let O be the domain of the objects and $F : O \to \mathbb{R}^d$ be a mapping of the objects into the d-dimensional feature space. Furthermore, let $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a distance function between two d-dimensional feature vectors. Then $simdist$: $O \times O \to \mathbb{R}$ is defined as follows:

$$simdist(Obj_1, Obj_2) = dist(F(Obj_1), F(Obj_2)).$$

There exist a lot of distance functions which are suitable for similarity search. In the literature, often the L_p-*distance* is used, as for instance the Manhattan distance ($p = 1$) or the Euclidean distance ($p = 2$). Throughout our experiments (cf. Section 5), the common Euclidean distance was used.

3.2 Normalization

Similarity models for CAD data should recognize similar parts, independently of their spatial location. The four, respectively five, tires of a car are similar, although they are located differently. Furthermore, reflected parts, e.g. the right and left front door of a car, should be recognized as similar as far as design is concerned. If we look at the production, reflected parts are no longer similar and have to be treated differently. Likewise, the actual size of the parts may or may not exert influence on the similarity model. To sum up, a similarity model for CAD data should take translation and rotation invariances into account whereas reflection and scaling invariances have to be tunable.

CAD objects are designed and constructed in a standardized position, normalized to the center of the coordinate system. We store each object normalized with respect to translation and scaling in the database. Furthermore, we store the scaling factors for each of the three dimensions, so that we can (de)activate scaling invariance depending on the users needs at runtime. In the case of CAD applications, not all possible rotations are considered, but only 90°-rotations. This yields 24 different possible positions for each object. For similarity search, where we are not confined to 90°-rotations, we can apply principal axis transformation in order to achieve invariance with respect to rotation. Taking also reflection into account, we may obtain $24 \cdot 2 = 48$ varying positions.

We could achieve 90°-rotation and reflection invariance by storing 48 different feature vectors for each object in the database or by carrying out 48 different permutations of the query object at runtime. As we want to decide at runtime whether we want to consider reflection invariance or not, we chose the second variant. Throughout our experiments, we considered invariance with respect to translation, reflection, scaling and 90°-rotation.

Taking all these transformations into account, we get the following extended similarity definition.

Definition 2. (Extended Feature-Based Object Similarity) Let O be the domain of the objects, $F : O \to \mathbb{R}^d$ a mapping of the objects into the d-dimensional feature space, and $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a distance function between two d-dimensional feature vectors. Furthermore, let $\mathcal{T}$ be a set of all user-dependent combinations of translation, scaling, rotation and reflection transformations. Then *simdist*: $O \times O \to \mathbb{R}$ is defined as follows:

$$simdist(Obj_1, Obj_2) = \min_{T \in \mathcal{T}} \{dist(F(Obj_1), F(T(Obj_2)))\}.$$

3.3 Spatial Features

After partitioning the data space, we have to determine the spatial features of the objects for each grid cell depending on the chosen model. In order to do that we first have to introduce some notations:

The data space is partitioned in each dimension into p grid cells. Thus, our histogram will consist of $k \cdot p^3$ bins where $k \in \mathbb{N}$ depends on the model which specifies the kind and number of features extracted from each cell. For a given object o, let $V^o = \{V_i^o \mid 1 \leq i \leq p^3\}$ be the set of voxels that represents o where V_i^o are the voxels covered by o in cell i. $\bar{V}^o \subseteq V^o$ denotes the set of voxels at the surface of the objects and $\dot{V}^o \subseteq V^o$ denotes the set of the voxels inside the object, such that $\bar{V}^o \cup \dot{V}^o = V^o$ and $\bar{V}^o \cap \dot{V}^o = \emptyset$ holds.

Let f_o be the computed feature vector of an object o. The i-th value of the feature vector of object o is denoted by $f_o^{(i)}$.

Let r be the number of voxels of the dataspace in each dimension. In order to ensure a unique assignment of the voxels to a grid cell, we assume that $\frac{r}{p} \in \mathbb{N}$.

3.3.1 The Volume Model

A simple and established approach to compare two objects is based on the number of the object voxels $| V_i^o |$ in each cell i of the partitioning. In the following, this model is referred to as *volume model*. Each cell represents one dimension in the feature vector of the object. The i-th dimension of the feature vector ($1 \leq i \leq p^3$) of object o can be computed by the normalized number of voxels of o lying in cell i, formally:

$$f_o^{(i)} = \frac{|V_i^o|}{K} \qquad \text{where} \qquad K = (\frac{r}{p})^3$$

Figure 1 illustrates the volume model for the 2-D case.

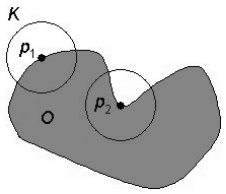


Figure 2: A sample object with different shapes at surface-points p_1 and p_2.

3.3.2 The Solid-Angle Model

The *solid-angle* method [11] measures the concavity and the convexity of geometric surfaces. Let K_c be a set of voxels that describes a 3-D voxelized sphere with central voxel c. For each surface-voxel $\bar{v}$ of an object o the so called solid-angle value is computed as follows. The voxels of o which are inside $K_{\bar{v}}$ are counted and divided by the size of $K_{\bar{v}}$, i.e. the number of voxels of $K_{\bar{v}}$. The resulting measure is called the solid-angle value $\text{SA}(\bar{v})$ and can be computed as follows:

$$\text{SA}(\bar{v}) = \frac{| K_{\bar{v}} \cap V^o |}{| K_{\bar{v}} |} \quad \text{, where:}$$

$K_{\bar{v}} \cap V^o =$
$\{w \in K_{\bar{v}} \mid \exists v \in V^o : \ w.x = v.x \ \wedge \ w.y = v.y \ \wedge \ w.z = v.z\}$

A small solid-angle value $\text{SA}(\bar{v})$ indicates that an object is convex at voxel $\bar{v}$. Otherwise, a high value of $\text{SA}(\bar{v})$ denotes a concave shape of an object at voxel $\bar{v}$. Figure 2 illustrates this behavior.

The solid-angle values of the cells are transferred into the according histogram bins as described in the following. We distinguish between three different types of cells:

1. Cell i contains surface-voxels of object o, i.e. $\bar{V}_i^o \neq \emptyset$. The mean of all SA-values of the surface-voxels is computed as the feature value of this cell:

   $$f_o^{(i)} = \frac{1}{m} \sum_{j=1}^{m} \text{SA}(\bar{v}_{i_j})$$

   where $\bar{V}_i^o = \{\bar{v}_{i_1}, \ldots, \bar{v}_{i_m}\}$.

2. Cell i contains only inside-voxels of object o, i.e. $\bar{V}_i^o = \emptyset$ and $V_i^o \neq \emptyset$. The feature value of this cell is set to 1 (i.e. $f_o^{(i)} = 1$).

3. Cell i contains no voxels of object o (i.e. $V_i^o = \emptyset$). The value of the according bin of the histogram is 0 (i.e. $f_o^{(i)} = 0$).

3.3.3 The Cover Sequence Model

The two models described above are based on a complete partitioning of the data space into disjoint cells. In this section, we adapt a known model [15, 16] to voxelized 3-D data which is not restricted to this rigid space partitioning but rather uses a more flexible object-oriented partitioning approach. This model is in the following referred to as *cover sequence model*.

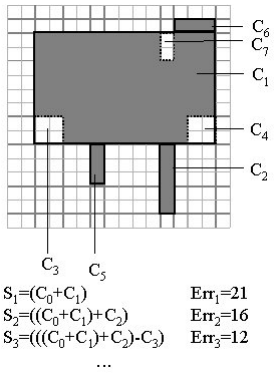


Figure 3: Cover sequence model.

As depicted in Figure 3 each edge of an object can be extended infinitely in either direction, to obtain a grid of lines. Each rectangle in this grid is called a *grid primitive*, and is located either entirely inside the object, or entirely outside of the object. Furthermore, any pair of adjacent grid primitives must also form a rectangle, respectively a cuboid in the 3-D data space. The basic idea of this model is to find large clusters of grid primitives, called *covers*, which approximate the object as good as possible [16].

The quality of such a cover sequence S_k is measured by the symmetric volume difference Err_k between the object O and the sequence S_k (cf. Figure 3). Formally, let the covers be drawn from the set $\mathcal{C}$ of all possible rectangular covers. Then each unit i of the cover sequence comprises a pair $(C_i \in \mathcal{C}, \sigma_i \in \{+, -\})$, where "+" represents set union and "−" represents set difference. The sequence after k units is:

$$S_k = (((C_0\sigma_1C_1)\sigma_2C_2)\dots\sigma_kC_k),$$

where C_0 is an initial empty cover at the zero point.

The symmetric volume difference after k units is:

$Err_k = |\, O \; XOR \; S_k \,|$, where O is the approximated object.

Jagadish and Bruckstein [16] suggest two algorithms for the retrieval of S_k: a *branch and bound* algorithm with exponential runtime complexity, and a *greedy* algorithm with polynomial runtime complexity which tries to minimize Err_i in each step $i \leq k$. Throughout our experiments we used this second algorithm.

In [15], Jagadish sketches how a 3-D cover sequence $S_k = (((C_0\sigma_1C_1)\sigma_2C_2)\dots\sigma_kC_k)$ of an object O, can be transformed into a $6 \cdot k$-dimensional feature vector. Thereby, each cover C_{i+1} with $0 \leq i \leq k-1$ is mapped onto 6 values in the feature vector f_o in the following way:

$$\begin{aligned} f_o^{6i+1} &= x\text{-position of } C_{i+1} \\ f_o^{6i+2} &= y\text{-position of } C_{i+1} \\ f_o^{6i+3} &= z\text{-position of } C_{i+1} \\ f_o^{6i+4} &= x\text{-extension of } C_{i+1} \\ f_o^{6i+5} &= y\text{-extension of } C_{i+1} \\ f_o^{6i+6} &= z\text{-extension of } C_{i+1} \end{aligned}$$

If an object O can be described by a sequence S_j with $j < k$ covers and $Err_j = 0$, we assign $((S_j\sigma_{j+1}C_0)\dots\sigma_kC_0)$ to S_k. These dummy-covers C_0 do not distort our similarity notion (cf. Definition 2), but guarantee that all feature vectors are of the same dimensionality. Thus we can use common spatial index-structures [8, 23, 6] in order to accelerate similarity queries.

4. USING SETS OF FEATURE VECTORS FOR SIMILARITY QUERIES

As proposed in [15] a data object is now represented as a feature vector. For similarity queries this method yields a major problem. Always comparing the two covers having the same ranking according to the symmetric volume difference, does not make sense in all cases. Two objects can be considered very different, because of the order of their covers, although they are very similar by intuition. The reason for this effect is that the order of the covers does not guarantee that the most similar covers due to size and position will be stored in the same dimensions. Especially for objects generating two or more covers having almost the same volume, the intuitive notion of similarity can be seriously disturbed. Thus, the possibility to match the covers of two compared objects with more degrees of freedom, might offer a better similarity measure. Figure 4 displays a 2-dimensional example of a comparison between a query object and a very similar database object. The first sequence (cf. Figure 4(a)) represents the covers of the query object in the order given by the symmetric volume difference. Let us note that cover 2, 3 and 4 are not very similar to the corresponding covers of the database object and therefore, the calculated similarity is relatively weak. By rearranging the order of these covers the total distance between the query object and the database object is considerably decreasing, which is displayed in Figure 4(b). Thus, the new order preserves the similarity between the objects much better.

To overcome the problem, the author in [15] proposes to generate several good representations of the query object and then process a query for each of the representations. Afterwards the union of the returned database objects is taken as a result. We can obtain different representations by permuting the order of the found covers and choose the most "promising" orderings to create the query vectors. Though, the method may offer reasonable results in many cases, there is no guarantee that the ordering offering the minimum distance is included within this selection. Thus, the whole similarity measure is dependent on the criteria used to select the most "promising" orderings. Since there is no well defined selection criterion known so far, the solution does not necessarily offer a precisely defined similarity measure.

Another solution for the problem is to consider all possible permutations. Since the distance between two objects can now be considered as the minimum distance over all possible orderings, the distance is defined precisely this way.

Definition 3.
Let $exch : \mathbb{N} \times \mathbb{N} \times \mathbb{R}^{(d \cdot k)} \rightarrow \mathbb{R}^{(d \cdot k)}$ be a function, where $exch(i, j, \vec{x})$ exchanges the d successive components beginning with dimension $i \cdot d + 1$ $(0 \leq i \leq k-1)$ with the d successive components beginning with dimension $j \cdot d + 1 (0 \leq j \leq k-1)$ of a vector $\vec{x} \in \mathbb{R}^{(k \cdot d)}$.

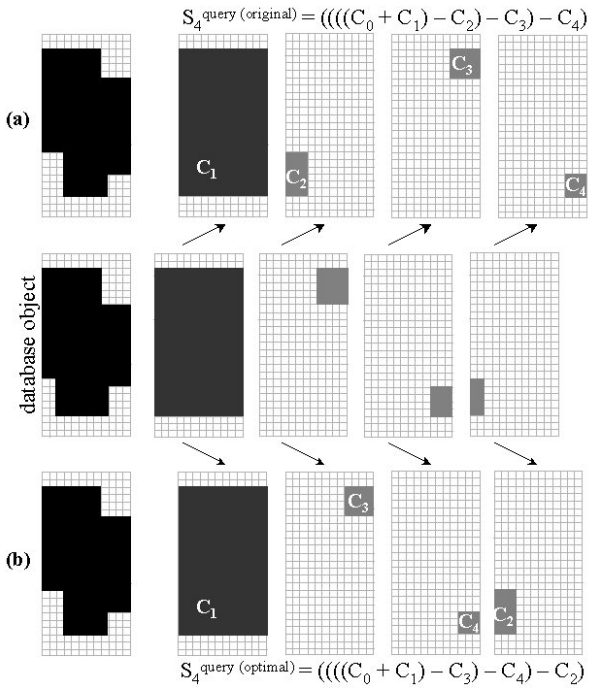


Figure 4: Examples demonstrating the advantage of free permutations.

$Exch : \mathbb{R}^{(k \cdot d)} \rightarrow 2^{\mathbb{R}^{(k \cdot d)}}$ is the function, that generates the set of all vectors that can be generated by applying $exch(i, j, \vec{x})$ arbitrary many times to a vector $\vec{x}$ using any combination for i and j.

Definition 4.
(minimum Euclidian distance under permutation)
Let O be the domain of the objects, let $F : O \rightarrow \mathbb{R}^{(k \cdot d)}$ be a mapping of the objects into the $k \cdot d$-dimensional feature space, and let $dist : \mathbb{R}^{(k \cdot d)} \times \mathbb{R}^{(k \cdot d)} \rightarrow \mathbb{R}$ be a distance function between two $k \cdot d$-dimensional feature vectors. Then $dist_{\pi-eucl.} : O \times O \rightarrow \mathbb{R}$ is defined as follows:

$$dist_{\pi-eucl.}(Obj_1, Obj_2) = \min_{\vec{y} \in Exch(F(Obj_2))} \{dist(F(Obj_1), \vec{y})\}$$

With a growing number of describing covers k, the processing time of considering all possible permutations increases exponentially, since there are $k!$ many permutations. With computation cost rising this rapidly, it is obvious that the description length k has to be kept low, which is not acceptable for all applications.

To guarantee that the permutation with the minimal distance is used, our approach does not work with one single feature vector, but with a set of feature vectors in lower dimensions. By treating the data objects as sets of d-dimensional feature vectors with a maximum cardinality of k, we introduce a new model for representing data objects in similarity search systems, the so called *vector set model*. In the following sections, we will discuss the concept of vector set representation in detail, with the goal of high quality distance measures and efficient query processing.

4.1 Reasons for the Use of Vector Set Representation

The representation of extracted features as a set of vectors is a generalization of the use of just one large feature vector. It is always possible to restrict the model to a feature space, in which a data object will be completely represented by just one feature vector. But in some applications the possibilities of vector set representation allow us to model the dependencies between the extracted features more precisely. As the development of conventional database systems in the recent two decades has shown, the use of more sophisticated ways to model data can enhance both the effectiveness and efficiency for applications using large amounts of data. In our application the vector set representation is able to avoid the problems that occur by storing a set of covers according to a strict order. Therefore, it is possible to compare two objects more intuitively, causing a relatively small rise of calculation cost compared to the distance calculation in the one-vector model. Another advantage of our new approach is the better storage utilization. It is not necessary to force objects into a common size, if they are represented by sets of different cardinality. For our current application there is no need for dummy covers to fill up the feature vectors. If the quality of the approximation is optimal with less than the maximum number of covers, only this smaller number of vectors has to be stored and loaded. In the case of a one-vector representation avoiding dummies is not possible without further modifications of the access structures used. Furthermore, we are able to distinguish between the distance measure used on the feature vectors of a set and the way we combine the resulting distances between the single feature vectors. For example, this possibility might be useful when defining partial similarity, where it is only necessary to compare the closest $i < k$ vectors of a set.

4.2 Distance Measures on Vector Sets

There are already several distance measures proposed on sets of objects. In [12] the authors survey the following three, which are computable in polynomial time: the Hausdorff distance, the sum of minimum distances and the (fair-) surjection distance. Furthermore, they introduce the link distance, which is computable in polynomial time, too. The Hausdorff distance does not seem to be suitable as a similarity measure, because it relies too much on the extreme positions of the elements of both sets. The last three distance measures are suitable for modelling similarity, but are not metric. This circumstance makes them unattractive, since there are only limited possibilities for processing similarity queries efficiently when using a non-metric distance function. In [12] the authors also introduce a method for expanding the distance measures into metrics, but as a side effect the complexity of distance calculation becomes exponential. Furthermore, the possibility to match several elements in one set to just one element in the compared set, is questionable when comparing sets of covers like in our application.

A distance measure on vector sets that demonstrates to be suitable for defining similarity in our application is based on the *minimum weight perfect matching* of sets. This well known graph problem can be applied here, by building a complete bipartite graph $G = (S_1 \cup S_2, E)$ between the vec-

tor sets S_1 and S_2. The weight of each edge $(x, y) \in E$ with $x \in S_1$ and $y \in S_2$ in this graph G is defined by their distance $dist(x, y)$. A perfect matching is a subset $M \subseteq E$ that connects each $x \in S_1$ to exactly one $y \in S_2$ and vice versa. A minimum weight perfect matching is a matching with a minimum sum of weights of its edges. Since a perfect match can only be found for sets of equal cardinality, it is necessary to introduce weights for unmatched nodes when defining a distance measure.

Definition 5. (enumeration of a set)
Let S be any finite set of arbitrary elements. Then π is a mapping that assigns $s \in S$ a unique number $i \in \{1, .., |S|\}$. This is written as $\pi(S) = (s_1, .., s_{|S|})$. The set of all possible enumerations of S is named $\Pi(S)$.

Definition 6. (minimal matching distance)
Let O be the domain of the objects and X be a set with $|X| \leq k$ and $X \subseteq 2^V$ with $V \subset \mathbb{R}^d$. Furthermore, let $F : O \to X$ be a mapping of the objects into X, and $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a distance function between two d-dimensional feature vectors. We assume w.l.o.g. $|F(Obj_1)| = m \geq n = |F(Obj_2)|$, $F(Obj_1) = \{x_1, .., x_m\}$ and $F(Obj_2) = \{y_1, .., y_n\}$. Then $dist_{mm}^{w,dist}$: $O \times O \to \mathbb{R}$ is defined as follows:

$$dist_{mm}^{w,dist}(Obj_1, Obj_2) =$$

$$\min_{\pi \in \Pi(F(Obj_1))} \left(\sum_{i=1}^{n} dist(x_{\pi(i)}, y_i) + \sum_{l=n+1}^{m} w(x_{\pi(l)}) \right)$$

where $w : \mathbb{R}^d \to \mathbb{R}^+$ is a weight function for the unmatched elements.

The weight function w provides the penalty given to every unassigned element of the set having larger cardinality. Let us note that *minimum matching distance* is a specialization of *netflow distance* which is introduced in [27]. In [27] it is proven that netflow distance is a metric and that it is computable in polynomial time. Therefore, we derive the following lemma without further proof.

LEMMA 1. *The minimal matching distance is a metric if $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a metric and $w : \mathbb{R}^d \to \mathbb{R}^+$ meets the following conditions:*

- $w(\vec{x}) > 0$, *for each* $\vec{x} \in V$
- *for* $\vec{x}, \vec{y}$, *with* $\vec{y}, \vec{x} \in V$ *the following inequality holds :* $w(\vec{x}) + w(\vec{y}) \geq dist(\vec{x}, \vec{y})$

In our application the *minimum Euclidian distance under permutation* can be derived from the *minimum matching distance*. By selecting the squared Euclidian distance as distance measure on V and taking the squared Euclidian norm as weight function, the distance value calculated by the minimum matching distance is the same as the squared value of the minimum Euclidian distance under permutation. This follows exactly from the definitions of both distance measures. Let us note that it is necessary to extract the square root from this distance value to preserve the metric character.

Though it was shown that the netflow distance can be calculated in polynomial time, it is not obvious how to achieve it. Since we are only interested in the minimum matching distance, it is enough to calculate a minimum weight perfect matching. Therefore, we apply the method proposed by Kuhn [22] and Munkres [25]. The method is based on the successive augmentation of an alternating path between both sets. Since it is guaranteed that this path can be expanded by one further match within each step taking $O(k^2)$ time and there is a maximum of k steps, the all over complexity of a distance calculation using the method of Kuhn and Munkres is $O(k^3)$ in the worst case. Let us note that for larger numbers of k this is far better than the previously mentioned method on $k!$ many permutations.

4.3 Answering Similarity Queries on Vector Set Data Efficiently

Though we discussed the time for a single distance calculation, the problem of efficiently processing similarity queries in large databases is still unanswered. Since it is necessary here, to locate the objects belonging to the result in comparably short time, the use of index structures that avoid comparing a query object to the complete database is mandatory. For one-vector-represented data objects there exists a wide variety of index structures that are suitable for answering similarity queries efficiently like the TV-Tree [23], the X-Tree [8] or the IQ-Tree [6]. But unfortunately, those index structures cannot be used directly to retrieve vector-set-represented objects.

To accelerate similarity queries on vector-set-represented objects, the simplest approach is the use of more general access structures. Since the minimal matching distance is a metric for the right choice of distance and weight function, the use of index structures for metric objects like the M-Tree [10] offers a good possibility here. Another approach is the use of the above mentioned high dimensional index structures, for querying sub tasks of the complete similarity query. In the following we will introduce a filter step that is based on the relation between a set of d-dimensional vectors and its extended centroid.

Definition 7.
Let $V \subset \mathbb{R}^d$ be a set of d-dimensional vectors. Then $w_{\vec{\omega}} : V \to \mathbb{R}$ denotes a set of weight functions having the following properties: $\vec{\omega} \in \mathbb{R}^d \backslash V$ and $w_{\vec{\omega}}(\vec{x}) = \|\vec{x} - \vec{\omega}\|_2$, where $\|\vec{x} - \vec{y}\|_2$ denotes the Euclidian distance between $\vec{x}, \vec{y} \in \mathbb{R}^d$.

Definition 8. (extended centroid)
Let $V \subset \mathbb{R}^d$ and $X \subset 2^V$ with $|X| \leq k$ be a set. Then the *extended centroid* $C_{k,\vec{\omega}}(X)$ is defined as follows:

$$C_{k,\vec{\omega}}(X) = \frac{\sum_{i=1}^{|X|} x_i + (k - |X|) \cdot \vec{\omega}}{k},$$

where $X = \{x_1, .., x_{|X|}\}$ and $\vec{\omega} \in \mathbb{R}^d \backslash V$

LEMMA 2. *Let $V \subset \mathbb{R}^d$ be a set and $\vec{\omega} \in \mathbb{R}^d \backslash V$. Furthermore, let X,Y be two vector sets with $\vec{x}_i \in X$,$\vec{y}_i \in Y$,*

let $C_{k,\vec{\omega}}(X)$, $C_{k,\vec{\omega}}(Y)$ be their extended centroids and let $dist_{mm}^{dist_{Eucl.},w_{\vec{\omega}}}$ be the minimal matching distance using $w_{\vec{\omega}}$ as weight function defined on V. Then the following inequality holds:

$$k \cdot \|C_{k,\vec{\omega}}(X) - C_{k,\vec{\omega}}(Y)\|_2 \leq dist_{mm}^{dist_{Eucl.},w_{\vec{\omega}}}(X,Y)$$

PROOF. Let π be the enumeration of the indices of X that groups the x_i to y_i according to the minimum weight perfect matching. w.l.o.g. we assume $|X| = n \geq m = |Y|$ and $n - m = \delta$.

$$k \cdot \|C_{k,\vec{\omega}}(X) - C_{k,\vec{\omega}}(Y)\|_2 =$$

$$k \cdot \| \frac{\sum_{i=1}^{n} x_{\pi(i)} + \sum_{i=1}^{k-n} \vec{\omega}}{k} - \frac{\sum_{i=1}^{m} y_i + \sum_{i=1}^{k-m} \vec{\omega}}{k} \|_2$$

$$= \| \sum_{i=1}^{m+\delta} x_{\pi(i)} + \sum_{i=1}^{k-m-\delta} \vec{\omega} - \sum_{i=1}^{m} y_i - \sum_{i=1}^{k-m} \vec{\omega} \|_2$$

$$= \| \sum_{i=1}^{m} x_{\pi(i)} - \sum_{i=1}^{m} y_i + \sum_{i=m+1}^{m+\delta} x_{\pi(i)} - \sum_{i=m+1}^{m+\delta} \vec{\omega} \|_2$$

$$\overset{\text{tri. ineq.}}{\leq} \| \sum_{i=1}^{m} (x_{\pi(i)} - y_i) \|_2 + \| \sum_{i=m+1}^{m+\delta} (x_{\pi(i)} - \vec{\omega}) \|_2$$

$$\overset{\text{tri. ineq.}}{\leq} \sum_{i=1}^{m} \|x_{\pi(i)} - y_i\|_2 + \sum_{i=m+1}^{m+\delta} \|x_{\pi(i)} - \vec{\omega}\|_2$$

$$= \sum_{i=1}^{m} \|x_{\pi(i)} - y_i\|_2 + \sum_{i=m+1}^{m+\delta} w_{\vec{\omega}}(x_{\pi(i)})$$

$$= dist_{mm}^{dist_{Eucl.},w_{\vec{\omega}}}(X,Y)$$

□

The lemma proves that the Euclidian distance between the extended centroids multiplied with the cardinality of the larger set is a lower bound for the minimal matching distance under the named preconditions. Therefore, when computing e.g. ε-range queries, we do not need to examine objects whose extended centroids have a distance to the query object q that is larger than ε/k. A good choice of $\vec{\omega}$ for our application is $\vec{0}$, since it has the shortest average distance within the position and has no volume. Since there are no covers having no volume in any data object, the conditions for the metric character of minimum matching distance are satisfied.

To implement the filter step, we stored the extended centroids in a 6-dimensional X-Tree [8]. Since this index structure provides high performance for similarity queries, it offers an efficient way to determine the keys of the candidate sets of feature vectors. Afterwards we loaded the vector sets itself to determine the membership of the object within the result. Since there exist established algorithms for ε-range [19] and knn-queries [29] using filter steps, the method is able to speed up both kinds of queries.

5. EVALUATION

In this section, we present the results of an exhaustive evaluation based on nearest-neighbor queries and clustering. We also apply the hierarchical clustering algorithm OPTICS [3] for a more objective evaluation of similarity models than sample k-nearest-neighbor queries [20].

5.1 Data Sets

We evaluated the three proposed models on the basis of two real-world datasets. The first one – in the following referred to as *Car Dataset* – contains approximately 200 CAD objects from a German car manufacturer. The Car Dataset contains several groups of intuitively similar objects, e.g. a set of tires, doors, fenders, engine blocks and kinematic envelopes of seats.

The second dataset contains 5,000 CAD objects from an American aircraft producer and in the following is called *Aircraft Dataset*. This dataset contains many small objects (e.g. nuts, bolts, etc.) and a few large ones (e.g. wings).

Using the cover sequence model and the vector set model, the data space of both datasets contains objects represented as voxel approximations using a raster resolution of $r = 15$. For the volume model and the solid-angle model, we used a raster resolution of $r = 30$. These values were optimized to the quality of the evaluation results.

5.2 Using Hierarchical Clustering for the Evaluation of Similarity Models

In general, similarity models can be evaluated by computing nearest-neighbor queries (k-nn queries). The major drawback of an evaluation based on k-nn queries is that the quality measure of the similarity model depends on the results of few similarity queries and, therefore, on the choice of the query objects. A model may perfectly reflect the intuitive similarity according to the chosen query objects and would be evaluated as "good" although it produces disastrous results for other query objects. Furthermore, there might be objects where a nearest-neighbor query does not yield any intuitively similar parts. Obviously, we should not discard a similarity model if the chosen query object belongs to noise. As a consequence, the evaluation of similarity models with sample k-nn queries is subjective and error-prone, due to its dependency on the choice of the query objects. In [20] the disadvantages of this approach were demonstrated in full detail.

Therefore, hierarchical clustering was introduced in [20] as an effective way to analyze and compare similarity models. Clustering groups a set of objects into classes where objects within one class are similar and objects of different classes are dissimilar to each other. The result can be used to evaluate which model is best suited for which kind of objects. In addition, using clustering the evaluation of the models is based on the whole data set and not only on a few arbitrary sample objects.

For the evaluation of the various similarity models, the density-based, hierarchical clustering algorithm OPTICS was used. This algorithm is similar to hierarchical Single-Link clustering methods [17] and is described in full details in [3]. The output of OPTICS is a linear ordering of the database objects minimizing a binary relation called *reachability* which is in most cases equal to the minimum distance of each database object to one of its predecessors in the ordering. The reachability values can be plotted for each object of the cluster-ordering computed by OPTICS. Valleys in this plot indicate clusters: objects having a small reachability value are more similar to their predecessor objects than objects having a higher reachability value.

The reachability plot generated by OPTICS can be cut at

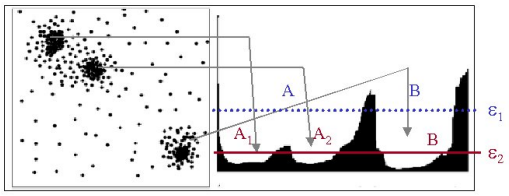


Figure 5: Reachability plot (right) computed by OPTICS for a sample 2-D dataset (left).

any level ε parallel to the abscissa. It represents the density-based clusters according to the density threshold ε: A consecutive subsequence of objects having a smaller reachability value than ε belong to the same cluster. An example is presented in Figure 5: For a cut at the level ε_1 we retrieve two clusters denoted as A and B. Compared to this clustering, a cut at level ε_2 would yield three clusters. The cluster A is split into two smaller clusters denoted as A_1 and A_2 and cluster B has decreased its size. Usually, for evaluation purposes, a good value for ε would yield as many clusters as possible.

5.3 Evaluation of the Effectiveness

The reachability plots generated by OPTICS for all models are depicted in Figure 6, 7, 8 and 9.

Obviously, the volume model performs rather ineffective. The plots computed by OPTICS when applying the model on the Car Dataset and the Aircraft Dataset are depicted in Figure 6(a) and 6(b). Both plots show a minimum of structure indicating that the volume model cannot satisfyingly represent the intuitive notion of similarity.

The solid-angle model performs slightly better. On the Car Dataset, OPTICS found three clusters denoted as A, B, and C in Figure 6(c). We analyzed these clusters by picking out some samples of the objects grouped in each cluster. The result of this evaluation on the Car Dataset is presented in Figure 10(a). As it can be seen, the objects in clusters A and C are intuitively similar but the objects in B are not. Furthermore, there are clusters of intuitively similar objects (e.g. doors), which are not detected. Evaluating the solid-angle model using the Aircraft Dataset we made similar observations. The reachability plot computed by OPTICS (cf. Figure 6(d)) yields a clustering with a large number of hierarchical classes. But the analysis of the objects within each cluster displays that intuitively dissimilar objects are treated as similar. A further observation is the following: objects, that are intuitively similar, are clustered in different groups. This suggests the conclusion that the solid-angle model is also rather unsuitable as a similarity model for our real-world test datasets.

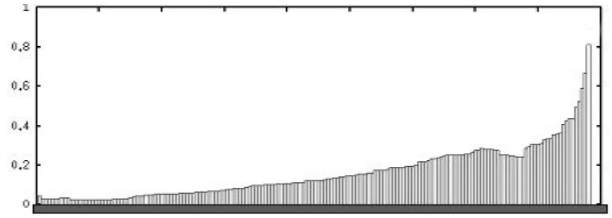
The plots computed by OPTICS for the cover sequence model, the cover sequence model using the minimum Euclidian distance under permutation and the vector set model (cf. Figure 7,8 and 9) look considerably better. We will confirm this observation in the following, evaluating the effectiveness of the different models. We analyzed the cover sequence model without permutations as well as under full permutations i.e. using the Euclidian distance under permutation. Note that the Euclidian distance under permutation is too time consuming for a straightforward calculation, since the runtime complexity increases with the faculty of the number of chosen covers. Therefore, we used the possibility of deriving this distance measure from the matching distance by employing the calculation via the Kuhn-Munkres algorithm as described in section 4.2. Remember that this is achieved by using the squared Euclidian distance for comparing single feature vectors and drawing the square root from the result. The resulting plots (cf. Figure 8) look quite similar to the ones we derived from employing the minimal matching distance based on the normal Euclidian distance, i.e. using the vector set model (cf. Figure 9(c) and 9(d)). A careful investigation of the parts contained in the clusters showed that the cover sequence model using the minimum Euclidian distance under permutation and the vector set model lead to basically equivalent results. Due to this observation and the better possibilities for speeding up k-nn queries, we concentrated on the evaluation of the vector set model. We first compared the vector set model to the cover sequence model without permutations (cf. Figure 7). Furthermore, we used different numbers of covers for the vector set model (cf. Figure 9) in order to show the benefits of a relatively high number of covers for complex CAD objects.

Comparing the vector set model with the cover sequence model on the Car Dataset (cf. Figure 7(a),9(a), and 9(c)) we conclude, that the vector set model is superior. All plots look similar on the first glance. When evaluating the clusters (cf. Figure 10(b) and 10(c)), it turned out that there are clusters which are detected by both approaches and thus appear in both plots, e.g. classes E in Figure 10(b) and 10(c). Nevertheless, we observed the following three shortcomings of the cover sequence model:

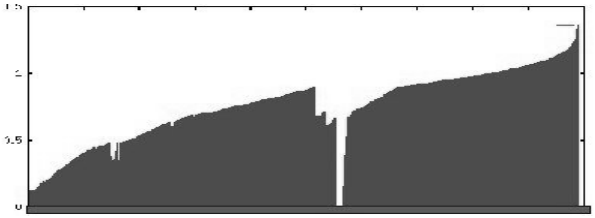
1. Meaningful hierarchies of clusters detected by the vector set model, e.g. G_1 and G_2 in Figure 9(c) which are visualized in Figure 10(c) are lost in the plot of the cover sequence model (Class G in Figure 7(a) evaluated in Figure 10(b)).
2. Some clusters found by the vector set model are not found using the cover sequence model, e.g. cluster F in Figure 10(c).
3. Using the cover sequence model, objects that are not intuitively similar are clustered together in one class (e.g. class X in Figure 7(a) which is evaluated in Figure 10(b)). This is not the case when using the vector set model.

A reason for the superior effectiveness of the vector set model compared to the cover sequence model is the role of permutations of the covers. This is supported by the observations which are depicted in Table 1. In most of all distance calculations carried out during an OPTICS run there was at least one permutation necessary to compute the minimal matching distance.

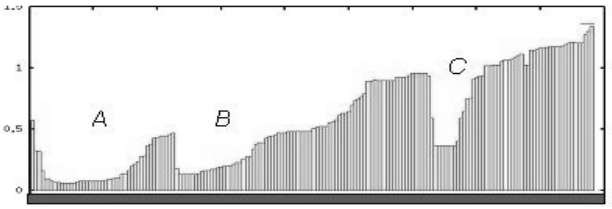
The plots in Figure 9(a) and 9(c) compare the influence of the number of covers used to generate the vector sets on the quality of the similarity model. An evaluation of the clusters



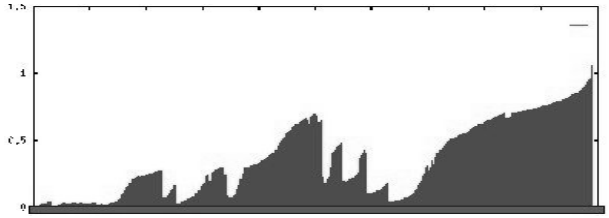
(a) Car Dataset (volume model)



(b) Aircraft Dataset (volume model)

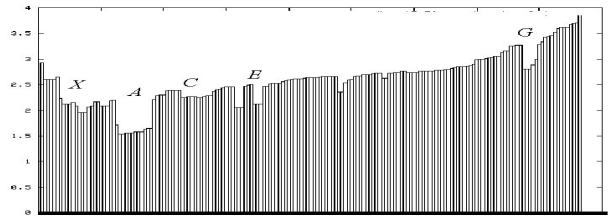


(c) Car Dataset (solid angle model)

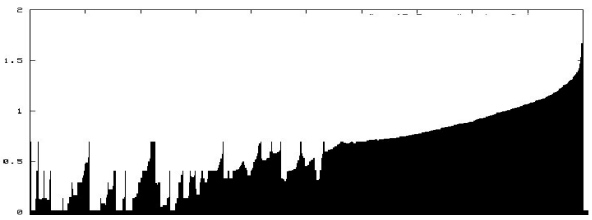


(d) Aircraft Dataset (solid angle model)

Figure 6: Reachability plots computed by OPTICS using the volume (a,b) and solid angle (c,d) model [20].



(a) Car Dataset



(b) Aircraft Dataset

Figure 7: Reachability plots computed by OPTICS using the cover sequence model with 7 covers.

Table 1: Percentage of proper permutations.

No. of covers	Permutations
3	68.2%
5	95.1%
7	99.0%
9	99.4%

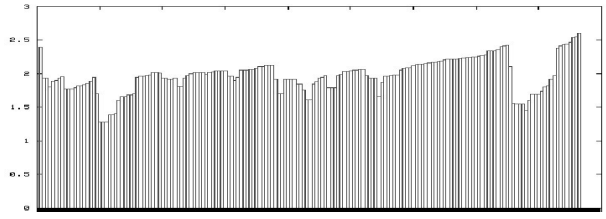
yields the observation, that 7 covers are necessary to model real-world CAD objects accurately. Using only 3 covers we observed basically the same three problems already noticed when applying the cover sequence model.

All the results of the evaluation on the Car Dataset can also be observed evaluating the models on the Aircraft Dataset. As a consequence, the evaluation shows that the vector set model outperforms the other models with respect to effectiveness. Furthermore, we see that we need about 7 covers to model similarity most accurately.

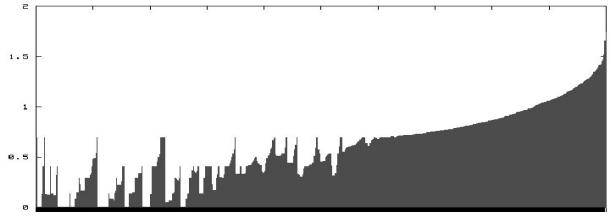
5.4 Evaluation of the Efficiency

The most effective results on our test datasets were generated with $k = 7$ covers, entailing an average permutation rate of 99.0% (cf. Table 1). This leads to the conclusion, that the cover sequence model can only compete with the vector set model with respect to quality, if all permutations are taken into account. Obviously, the vector set model using the minimal matching distance approach is much more efficient than the cover sequence model (one-vector model) using the minimum Euclidian distance under permutation.

To analyze the performance of the filter step, introduced in Section 4, we evaluated k-nn queries, which are the most common query type in similarity search systems. Since the Car Dataset consists of only some 200 objects, it is not suitable for efficiency evaluation. Thus, we ran our efficiency experiments on the Aircraft Dataset only. We took 100 random query objects from the database and examined 10-nn queries. Our test machine was equipped with an INTEL XEON 1.7 GHz processor and 2 GByte main memory. Since data and access structures fitted easily into the main

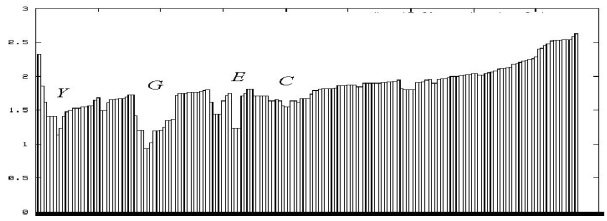


(a) Car Dataset

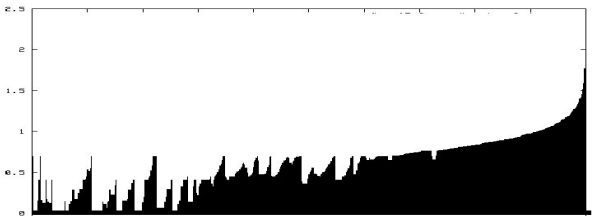


(b) Aircraft Dataset

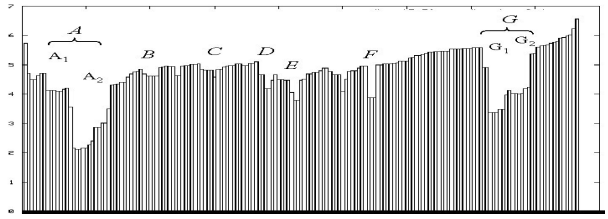
Figure 8: Reachability plots computed by OPTICS using the cover sequence model with the minimum Euclidian distance under permutation with 7 covers.



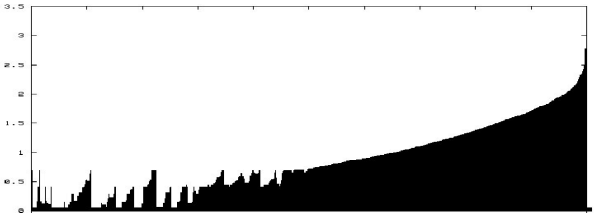
(a) Car Dataset (3 covers)



(b) Aircraft Dataset (3 covers)



(c) Car Dataset (7 covers)



(d) Aircraft Dataset (7 covers)

Figure 9: Reachability plots computed by OPTICS using the vector set model with 3 and 7 covers.

memory, we calculated the I/O cost. One page access was counted as 8 ms and for the costs of reading one byte we counted 200 ns. The results are shown in Table 2.

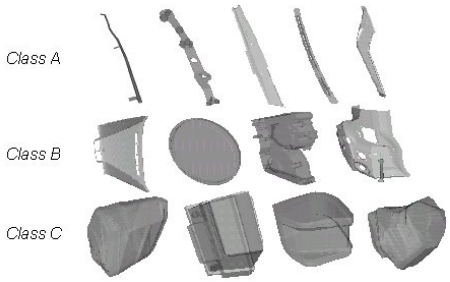
It turns out that the filter step yields a speed-up of factor 10 on the CPU time, but suffers from a higher I/O-time. Nevertheless it provides a speed up factor of about 2 for total time. Furthermore, Table 2 demonstrates, that the run time using the vector set model with filter step is in the same order of magnitude as the one-vector model even without permutation. In our experiments, the vector set approach even outperformed the one-vector model in both CPU time and I/O time. Let us note that in our experiments we based the implementation of the one-vector model on the X-Tree [8], which is penalized by the simulation of I/O time. Since it does not take the idea of page caches into account, an implementation of the one-vector model using the sequential scan exhibited slightly better performance for some combinations of dimensionality and data set size, but the performance was still in the same order of magnitude.

Table 2: Runtimes for sample 10-nn queries in s.

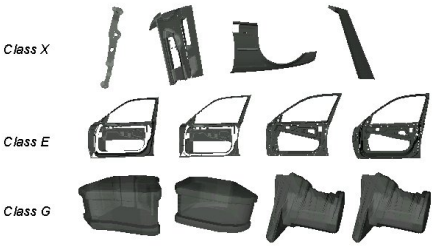
Model	CPU time	I/O time	total time
1-Vect.	142.82	2632.06	2774.88
Vect. Set w. filter	105.88	932.80	1038.68
Vect. Set seq. scan	1025.32	806.40	1831.72

6. CONCLUSIONS

In this paper, we surveyed three feature transformations that are suitable for the use on voxelized CAD data: the volume model, the solid angle model and the cover sequence model. The cover sequence model generates a set of covers of a 3-dimensional object that can be stored in a feature vector. In comparison to the other two models it offers a better notion of similarity. A major problem of the cover sequence model is the order in which the covers are stored within the feature vector. For calculating the similarity of two objects the order realizing minimum distance offers a better similarity measure, but is prohibitive in calculation cost. Our new



(a) Classes found by OPTICS in the Car Dataset using the solid-angle model (cf. Figure 6(c)) [20].



(b) Classes found by OPTICS in the Car Dataset using the cover sequence model (cf. Figure 7(a)).

(c) Classes found by OPTICS in the Car Dataset using the vector set model with 7 covers (cf. Figure 9(c)).

Figure 10: Evaluation of classes found by OPTICS in the Car Dataset.

approach to represent an object as a set of feature vectors avoids this problem. Furthermore, it offers a more general approach for applications working with set-valued objects. In the rest of the paper we described the distance measure on vector sets we used, called minimal matching distance. Minimal matching distance is a metric and computable in $O(k^3)$. To demonstrate how similarity queries can be answered efficiently, we introduced a highly selective filter step that is able to speed up similarity queries by the use of spatial index structures. To evaluate our system we used two CAD data sets. To demonstrate the good notion of similarity provided by the combination of the cover sequence model and the vector set representation, we applied hierarchical clustering as a more objective way to examine similarity measures. Since k-nn queries are the most common operation in similarity search systems, we evaluated the efficiency of the filter step using 100 sample 10-nn queries. It turned out that our new approach yields more meaningful results without sacrificing efficiency.

Since vector set representation provides many advantages for applications working with set-valued objects, we are developing a more general system for managing vector-set-represented objects. With the more general system we plan to examine various other applications for similarity search, such as the retrieval of biomolecular data and images. Another essential goal is the development of fast and flexible algorithms for processing similarity queries on vector set representations.

7. ACKNOWLEDGMENTS

We would like to thank Thomas Seidl, Marco Pötke and especially Riko Jacob for their constructive and helpful comments. Furthermore, we thank Stefan Schönauer for providing us with an efficient implementation of the X-tree.

8. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. "Efficient Similarity Search in Sequence Databases". In *Proc. 4th. Int. Conf. on Foundations of Data Organization and Algorithms (FODO'93), Evanston, ILL*, volume 730 of *Lecture Notes in Computer Science (LNCS)*, pages 69–84. Springer, 1993.

[2] R. Agrawal, K.-I. Lin, H. Sawhney, and K. Shim. "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases". In *Proc. 21th Int. Conf. on Very Large Databases (VLDB'95)*, pages 490–501, 1995.

[3] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. "OPTICS: Ordering Points to Identify the Clustering Structure". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA*, pages 49–60, 1999.

[4] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. "3D Shape Histograms for Similarity Search and Classification in Spatial Databases". In *Proc. 6th Int. Symposium on Large Spatial Databases (SSD'99),*

Hong Kong, China, volume 1651 of *Lecture Notes in Computer Science (LNCS)*, pages 207–226. Springer, 1999.

[5] A. Badel, J. Mornon, and S. Hazout. "Searching for Geometric Molecular Shape Complementarity using Bidimensional Surface Profiles.". *Journal of Molecular Graphics*, 10:205–211, 1992.

[6] S. Berchtold, C. Böhm, H. V. Jagadish, H.-P. Kriegel, and J. Sander. "Independent Quantization: An Index Compression Technique for High-Dimensional Data Spaces". In *Proc. Int. Conf. on Data Engineering (ICDE 2000), San Diego, CA*, pages 577–588, 2000.

[7] S. Berchtold, D. Keim, and H.-P. Kriegel. "Using Extended Feature Objects for Partial Similarity Retrieval". *VLDB Journal*, 6(4):333–348, 1997.

[8] S. Berchtold, D. A. Keim, and H.-P. Kriegel. "The X-Tree: An Index Structure for High-Dimensional Data". In *Proc. 22nd Int. Conf. on Very Large Data Bases (VLDB'96), Bombay, India, pp. 28-39.*, 1996.

[9] S. Berchtold and H.-P. Kriegel. "S3: Similarity Search in CAD Database Systems". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'97), Tucson, AZ*, pages 564–567, 1997.

[10] P. Ciaccia, M. Patella, and P. Zezula. "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces". In *Proc. 23rd Int. Conf. of Very Large Data Bases, Athens, Greece*, pages 426–435, 1997.

[11] M. Connolly. "Shape Complementarity at the Hemoglobin a1b1 Subunit Interface". *Biopolymers*, 25:1229–1247, 1986.

[12] T. Eiter and H. Mannila. "Distance Measures for Point Sets and Their Computation". *Acta Informatica*, 34(2):103–133, 1997.

[13] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, et al. "Efficient and Effective Querying by Image Content". *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[14] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. "Fast Subsequence Matching in Time-Series Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94), Minneapolis, MN*, pages 419–429, 1994.

[15] H. Jagadish. "A Retrieval Technique for Similar Shapes". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'91)*, pages 208–217, 1991.

[16] H. V. Jagadish and A. M. Bruckstein. "On sequential shape descriptions". *Pattern Recocnition*, 1991.

[17] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[18] D. Keim. "Efficient Geometry-based Similarity Search of 3D Spatial Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA*, pages 419–430, 1999.

[19] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. "Fast Nearest Neighbor Search in Medical Image Databases". In *Proc. 22th Int. Conf. on Very Large Databases (VLDB'96)*, pages 215–226, 1996.

[20] H.-P. Kriegel, P. Kröger, Z. Mashael, M. Pfeifle, M. Pötke, and T. Seidl. "Effective Similarity Search on Voxelized CAD Objects". In *Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan*, 2003.

[21] H.-P. Kriegel, T. Schmidt, and T. Seidl. "3D Similarity Search by Shape Approximation". In *Proc. 5th Int. Symposium on Large Spatial Databases (SSD'97), Berlin, Germany*, volume 1262 of *Lecture Notes in Computer Science (LNCS)*, pages 11–28. Springer, 1997.

[22] H. Kuhn. "The Hungarian method for the assignment problem". *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[23] K.-I. Lin, H. V. Jagadish, and C. Faloutsos. "The TV-Tree: An Index Structure for High-Dimensional Data". *VLDB Journal*, 3(4):517–542, 1994.

[24] R. Mehrotra and J. Gary. "Feature-Based Retrieval of Similar Shapes". In *Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria*, pages 108–115, 1993.

[25] J. Munkres. "Algorithms for the assignment and transportation problems". *Journal of the SIAM*, 6:32–38, 1957.

[26] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasmann, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. "The QBIC Project: Querying Images by Content Using Color, Texture, and Shape". In *SPIE 1993 Int. Symposium on Electronic Imaging: Science and Technology Conference 1908, Storage and Retrieval for Image and Video Databases, San Jose, CA*, 1993.

[27] J. Ramon and M. Bruynooghe. "A polynomial time computable metric between point sets", 2001.

[28] R. Schneider, H.-P. Kriegel, B. Seeger, and S. Heep. "Geometry-based Similarity Retrieval of Rotational Parts". In *Proc. Int. Conf. on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg, ML*, pages 150–160, 1989.

[29] T. Seidl and H.-P. Kriegel. "Optimal Multi-Step k-Nearest Neighbor Search". In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 154–165, 1998.