Similarity Assessment Based on Face Alignment Using Attributed Vectors

A. Cardone[1] and S. K. Gupta[2]

[1]University of Maryland, College Park, toniocar@glue.umd.edu
[2]University of Maryland, College Park, skgupta@eng.umd.edu

ABSTRACT

In certain shape similarity assessment applications, type, locations, and orientations of faces in boundary representations plays a key role in determining similarity between two parts. This paper describes an algorithm that assesses similarity between two parts by explicitly aligning their faces and hence accounts for these face attributes. The approach involves extracting attributed applied vectors from the face information of parts and aligning the two sets of attributed applied vectors using rigid body transformations. The algorithm uses a customizable distance function to align attributed applied vectors. The distance between two aligned attributed applied vector sets is used as a measure of similarity between two parts. This paper also presents computational results to illustrate discrimination capability of the algorithm.

Keywords: Shape similarity assessment, alignment of models, and design reuse.

1. INTRODUCTION

Over the last fifteen years 3D CAD systems have become very popular in the industry. These CAD systems are being used to generate 3D models of parts. Nowadays, organizations routinely set up databases of CAD models to enable all participants in the product development process to have access to 3D data to support their functions. CAD databases for even moderate size companies are expected to be large in size. Intuitively, if two products are similar, it is possible to reuse information about one product to derive corresponding information about the other one. There are many possible applications where reuse of information can be of significant value. Representative examples include part-family formation, redesign suggestion generation, supplier selection, cost estimation, tooling design, machine selection, stock selection, and design reuse. The ability to search for similar products in a database by specifying a query product is expected to help companies in significantly reducing the associated time and cost compared with the manual methods of locating the similar products. In [12] an overview on content-based search techniques for parts and assemblies is given.

In certain applications, locations, type, and orientations of faces play a major role in determining similarity between two parts. For example, similarity between two molded parts from the tool maker selection point of view needs to be assessed by analyzing characteristics of part faces. For example, face parameters such as spatial location, type and curvature distribution determine the type and complexity of the tooling needed to manufacture the part. Similarly, the face area determines the size of the tooling. Face features do not always have explicitly defined parameters. Hence we need to identify components of face feature vectors that are significant in determining the similarity between two parts from the tooling point of view.

Shape similarity assessment problem has been studied extensively in literature. We have presented a survey in [2]. Representative techniques include shape histogram-based techniques [9, 13, 18, 21], graph-based techniques [1, 10, 19], spatial function based techniques [5, 6, 11, 16, 17], feature-based techniques [2, 3, 8, 14, 22].

The features can be represented as points or vectors in *n* dimension space where each dimension represents some significant characteristic of the feature. Similarity assessment can then be performed by aligning the two point sets. Many of the formulations of point pattern alignment assume that every point in one set has a close alignment in the other set in terms of Hausdorff distance [7]. A hybrid approach combining both branch-and-bound search of the transformation space with point-to-point alignments was proposed in [20]. Another important class of alignment methods for searching in large object databases is geometric hashing [23].

Existing techniques provides excellent performance for filtering irrelevant parts. However, when locations, type, and orientations of faces play a major role in determining similarity, existing techniques do not seem to have sufficient discrimination capabilities. Hence, we have developed a new shape similarity assessment technique. This paper describes a new algorithm that assesses similarity between two parts by explicitly aligning their faces. The approach involves extracting attributed applied vectors from the face information of parts and aligning the two sets of attributed applied vectors using rigid body transformations. The algorithm uses a customizable distance function to align attributed applied vectors. The distance between two aligned attributed applied vector sets is used as a measure of similarity between two parts. By changing the weights used in the distance function, the method described in this paper can be easily customized to a new application. Our preliminary experiments indicate that the method described in this paper seem to exhibit very good discrimination capabilities.

2. BACKGROUND AND PROBLEM FORMULATION

2.1 Face-based Attribute Applied Vector

In this paper, a face is defined as a part boundary surface region delimited by its natural edges. An edge is a curve belonging to the boundary of the part. The curve is either a segment corresponding to a sharp corner or a set of points corresponding to locally maximum curvature values.

In order to formally define face-based attributed applied vectors (FAAVs), let us recall the definitions of free and applied vectors. A free vector is a vector whose orientation and magnitude are specified. An applied vector is a vector whose orientation, magnitude and point of application are defined. The point of application of a vector represents the position of the vector in the space. FAAVs are mathematically equivalent to attributed applied vectors in 3D Space, where the application points of the vectors represent the position of the face in the space and the vector orientations represent the orientation of the face. Therefore, the problem of aligning two sets of FAAVs is equivalent to the problem of aligning attributed applied vectors. The definition of FAAV location and orientation is given as follows.

FAAV location is a point that represents the position of the corresponding face in the space. It is computed by sampling *n* points from the corresponding face and then computing their center of mass. *FAAV orientation* is a unit vector that represents the orientation of the corresponding face in the space. It is computed by sampling *n* points from the corresponding face and then computing the sum of their corresponding normal unit vectors. The resulting vector is normalized into a unit vector. For some types of faces, such as spherical faces, orientation vector is not defined.

The following FAAV types are considered in this paper: cylindrical, planar, toroidal and spherical. All the rest are defined as general FAAVs. For toroidal and cylindrical FAAVs orientation is defined as the unit vector along the axis of the cylinder or torus. Orientation is not defined for spherical FAAVs. Each of the previously listed FAAV types can be completely characterized by providing the values of certain parameters such as area, curvature distribution and normal vector distribution. In particular, normal vector distribution is characterized by the orientation standard deviation. In order to formally define orientation standard deviation, consider FAAV orientation $\boldsymbol{o}_A$ and the normal vectors $\boldsymbol{o}_i$ sampled from the corresponding face *A*. Consider the discrete function $f_i = \boldsymbol{o}_A \cdot \boldsymbol{o}_i$. The orientation standard deviation is defined as the standard deviation of the discrete function f_i, which is defined as follows.

$$\sigma_f = \sqrt{\sum_{i=1}^{n} \frac{\left(f_i - \mu_f\right)^2}{n}}$$

where $\mu_f = \frac{\sum_{i=1}^{n} f_i}{n}$. A number of techniques can be used to compute the curvature corresponding to each sampled point. We also compute the average mean curvature and the curvature standard deviation for each FAAV by computing these values for the sampled points.

FAAVs for a part are defined for a specific coordinate system. In order to measure the distance between two sets of face-based attributed applied vectors, one set is transformed with respect to the other by using rigid body transformations such that the minimum distance between two sets is obtained. The alignment algorithms rank order the parts in a database based on the degree of similarity with respect to the query part. Figure 1 shows an example of FAAVs of a part.

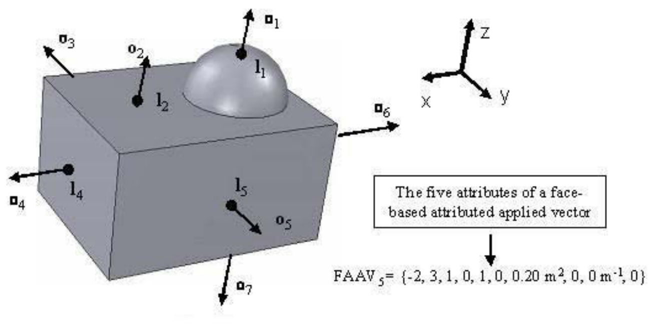
o1
l1
o2
l2
o3
o4
l4
l5
o5
o6
o7
z
x
y
The five attributes of a face-based attributed applied vector
FAAV5= {-2, 3, 1, 0, 1, 0, 0.20 m2, 0, 0 m-1, 0}

Fig. 1. Face Based Attributed Applied Vectors of a 3D object.

2.2 Distance Function for Similarity Assessment

Let $p \in P$ and $q \in Q$ be the two sets of FAAVs corresponding to parts M_P and M_Q. Then, P and Q are compared using the following distance function.

$$\vec{d}(P,Q) = \frac{\sum_{i=1}^{n} \min_{q \in Q} d(p_i, q)}{n} \tag{1}$$

The distance function between two FAAVs $p \in P$ and $q \in Q$ needs to account for relative locations and orientations, the FAAV curvature and the orientation distribution and FAAV type. Each FAAV is represented by using ten components. Specific components are x^p, y^p, z^p, v_x^p, v_y^p, v_z^p, $A(p)$, $\sigma_o(p)$, $\mu_c(p)$, $\sigma_c(p)$. The first three components x^p, y^p and z^p represent the location of the FAAV p, and are transformation-dependent. Similarly the second three components v_x^p, v_y^p and v_z^p represent the orientation of the FAAV p and are also transformation-dependent. The other four components $A(p)$, $\sigma_o(p)$, $\mu_c(p)$ and $\sigma_c(p)$ are transformation-invariant. The seventh component $A(p)$ represents the normalized area of the FAAV. The areas are normalized using the maximum value of the area over all the FAAVs of the parts being compared. The eighth component $\sigma_o(p)$ represents the normalized orientation standard deviation, which is not defined in the case where the FAAV is a sphere or a cylinder. For all the other types, the orientation standard deviation is normalized using the maximum value of the orientation standard deviation over all the FAAVs of the parts being compared. The ninth component $\mu_c(p)$ represents the normalized average mean curvature, which is normalized using the maximum value of the average curvature over all the FAAVs of the parts being compared. The tenth component $\sigma_c(p)$ represents the normalized curvature standard deviation, which is normalized using the maximum value of the curvature standard deviation over all the FAAVs of the parts being compared.

The distance function between FAAVs $p \in P$ and $q \in Q$ is defined as follows.

$$\begin{aligned} d(p,q) &= w_L[(x^p - x^q)^2 + (y^p - y^q)^2 + (z^p - z^q)^2] + \\ & w_O[(v_x^p - v_x^q)^2 + (v_y^p - v_y^q)^2 + (v_z^p - v_z^q)^2] + \big(1 - \delta(p,q)\big) \\ & \begin{bmatrix} w_A(A(p) - A(q))^2 + w_{\sigma o}(\sigma_o(p) - \sigma_o(q))^2 \\ + w_{\mu c}(\mu(p) - \mu(q))^2 + w_{\sigma c}(\sigma_c(p) - \sigma_c(q))^2 \end{bmatrix} + w_T \delta(p,q) \end{aligned} \tag{2}$$

The first three terms account for the difference in position between p and q and relate to FAAV interactions. The second three terms account for the difference in the orientation and relate to the FAAV interactions as well. The last five terms account for the difference in transformation-invariant attributes that are considered. Specifically, the seventh term accounts for the difference in area between the corresponding faces and relates to face size. The eighth, night and tenth terms account for the difference in the orientation standard deviation, the average curvature and the curvature standard deviation between the corresponding faces and relate to face complexity. The eleventh term accounts for the difference in type between the corresponding faces that has been defined in Subsection 2.1. It relates to face complexity as well. The term δ has the following expression.

$$\begin{cases} \delta(p,q)=0 & \text{if type of } p \text{ is equal to type of } q \\ \delta(p,q)=1 & \text{if type of } p \text{ is different from type of } q \end{cases}$$

The quantity $\left(1-\delta(p,q)\right)$ is defined so that when the types of FAAVs p and q do not match most of the terms are not considered. The quantities w_O, w_L, w_A, $w_{\sigma o}$, $w_{\mu c}$, $w_{\sigma c}$, and w_T represent the weights given by the user to all the terms previously defined. The distance function can be customized by: (a) changing the weight associated with each of the terms in the distance function, (b) considering additional transformation-invariant shape parameters as needed.

The distance function defined in Equation (1) is the measure of similarity between parts M_P and M_Q, represented by two sets of FAAVs; smaller the value of the distance given by Equation (1), more similar are the parts M_P and M_Q.

2.3 Problem Statement

Each part has been modeled in its own coordinate system. Therefore, we need to align the parts using rigid body transformations before computing the distance. The parts are represented by using two sets of FAAVs. Hence, as stated previously, the problem of aligning two sets of FAAVs is equivalent to the problem of aligning attributed applied vectors. To align the two sets of attributed applied vectors in 3D, one set has to be moved with respect to the other set. Rigid body transformation of a set of attributed applied vectors in 3D involves six degrees of freedom. The distance function has to be minimized over all the possible configurations of the moving attributed applied vector set with respect to the stationary one. The transformation matrix for the six degrees of freedom transformation is given by $\mathbf{T}=\mathbf{T}(\Delta x,\Delta y,\Delta z,\theta,\varphi,\psi)$ where Δx, Δy, Δz, θ, φ, and ψ are the six degrees of freedom considered. Assuming that P is the moving set, the transformed set P can be written as $\mathbf{T}P$. The distance function defined in Equation (1) can then be written as:

$$\vec{d}(\mathbf{T}P,Q)=\vec{d}(\mathbf{T}P,Q)(\Delta x,\Delta y,\Delta z,\theta,\varphi,\psi) \quad (3)$$

Therefore, assessing the degree of similarity between two parts requires finding the best alignment between two sets of attributed applied vectors by transforming one attributed applied vector set such that the distance function is minimized.

3. OVERVIEW OF APPROACH

Finding the best alignment between two sets of attributed applied vectors in 3D consists of minimizing the distance defined in Equation (1) between them over all the possible transformations applied to one of the two sets. The value of the distance function corresponding to the optimal alignment yields the similarity degree between the two parts corresponding to the attributed applied vector sets. Only database parts whose similarity degree to the query part is very high need to be considered for most manufacturing engineering applications. Hence database parts whose distance from the query part upon optimal alignment is expected to be very high should be pruned without performing the alignment step.

Aligning two sets of attributed applied vectors in 3D is a six degree of freedom problem. In order for two parts to be considered similar, they will need to have least one FAAV of the same type. In fact if the two parts have no common FAAVs, then the distance between the corresponding attributed applied vector sets is going to be very high. Hence the part can be safely pruned as their similarity degree is very low. Thus, five degrees of freedom in this problem can be constrained by considering combinations of FAAVs. Location and orientation of each FAAV of M_P are aligned with every FAAV of M_Q having the same type. The total number of alignments that need to be performed is not expected to be large for most common parts.

Consider a pair of FAAVs $p_i \in P$ and $q_j \in Q$ of the same type equivalent to two attributed applied vectors. Initially, the translation represented by the matrix $\mathbf{T}_{i,j}$ is applied to the two sets P and Q such that the locations of $p_i \in P$ and $q_j \in Q$ are aligned. Then the rotation represented by the matrix $\mathbf{R}_{i,j}$ is applied to the two sets P and Q such that the orientations of $p_i \in P$ and $q_j \in Q$ are also aligned. The center of rotation is the location corresponding to the pair of FAAVs being aligned. Finally the set P is transformed with respect to Q using the one degree of freedom left, which is the rotation θ around the orientation of $p_i \in P$ and $q_j \in Q$ that has been aligned. The algorithm that can solve the alignment problem for the degree of freedom θ consists of three main steps.

The first step involves the partitioning of the theta range $[0,2\pi]$ into theta intervals such that the closest neighbor $q_j \in Q$ to each FAAV $p_i \in P$ is invariant within each interval. This step is described in more detail in Section 4.

Each interval c obtained from the previous step is processed in the following manner. We compute the value of the rotation $\theta(c)$ that minimizes the distance function defined in Equation (1) for interval c. This step is described in more detail in Section 5.

Finally we compute $\theta_{\min}$ such that the distance function defined in Equation (1) reaches the minimum value over all the intervals. This is achieved as follows. For each interval, $\vec{d}(c)$ is the minimum distance corresponding to the value of the rotation $\theta(c)$ obtained in the previous step. Then the minimum distance $\vec{d}_{\min} = \min_{c \in C} \vec{d}(c)$ over all the intervals is selected, where C is the set of all the intervals c of the partitioned theta range $[0, 2\pi]$. The corresponding rotation $\theta_{\min} = \theta(c^*)$, where c^* is the interval in which the minimum distance was found, is the one that minimizes the distance function defined in Equation (1). Hence $\theta_{\min}$ is the aligning transformation sought.

The value of the distance function after the final step is the minimum value of the distance function for a particular FAAV pair alignment. Now, the next alignment is considered and the procedure is repeated. The output is the minimum value of the distance over all the FAAV pair alignments.

The approach for aligning two parts is illustrated using the example shown in Figure 2. Figures 2(a) and 2(b) show the initial positions of two parts M_P and M_Q that are to be compared. Part M_P is obtained by randomly transforming part M_Q. The system, initially, transforms part M_P such that the location and orientation of its FAAV p matches the ones of FAAV q of the same type of part M_Q as shown in Figure 2(c). The system then computes the angle of rotation θ such that the distance function is minimized. The final positions of part M_P after applying rotation θ is shown in Figure 2(d).

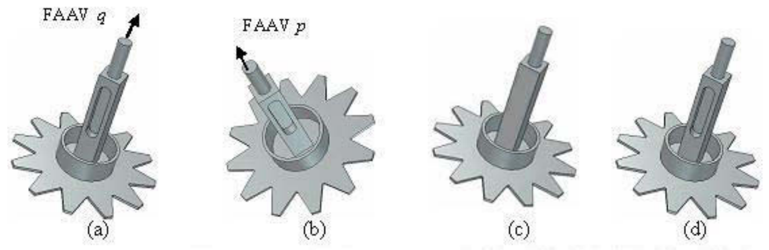

Fig. 2. (a) Initial position of Part M_Q; (b) Initial position of M_P; (c) Position of Part M_P after matching FAAVs p and q of the same type from Parts M_P and M_Q; (d) Final position of the Part M_P After Rotation About The Orientation of Aligned FAAVs p and q.

4. BUILDING THE SET OF THETA INTERVALS FOR THE FAAVS OF SET P

To compute the distance value in Equation (1), the closest neighbor $q_j \in Q$ to each $p_i \in P$ needs to be determined. The closest neighbor $q_j \in Q$ to each $p_i \in P$ changes with the rotation of set P with respect to set Q. Thus, the closest neighbors for each $p_i \in P$ need to be obtained by taking into account the rotation θ around the fixed axis as explained in the previous section. It is necessary to know, for each value of the rotation θ, the closest FAAV $q_j \in Q$ to each FAAV $p_i \in P$. The closest neighbor to each FAAV of P changes only at specific values of θ. Thus, the theta range $[0, 2\pi]$ can be partitioned into a set of theta intervals within which the closest neighbor to each FAAV of P is known and invariant. The main steps of the algorithm that is used for this purpose are listed as follows.

The first step consists of three sub steps that are carried out for each FAAV p_i of P.

The first substep involves, for each possible pair of distinct FAAVs q_k and q_l of Q, partitioning of the theta range $[0, 2\pi]$ into subintervals within which either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$. The partitioning is performed by finding the values of θ such that $d(p_i, q_k) = d(p_i, q_l)$, where d is the distance function defined in Equation (2). This sub step can be carried out analytically and it will be described in more detail below.

The second sub step consists of overlapping the intersecting subintervals obtained in the first substep so that the range $[0, 2\pi]$ is further partitioned into a set of intervals.

For each interval being obtained in the second substep and using the closest neighbors being obtained in the first substep, the FAAV q_j of Q such that $d(p_i, q_j)$ is minimum over all the FAAVs of Q are found in the third substep.

The second step of the algorithm consists of overlapping the set of intersecting intervals being obtained in the first step for each FAAV p_i of P. Within the set of intervals being obtained, the closest neighbor to every FAAV of P from set Q is invariant and known. The algorithm yields the set of theta intervals for the FAAVs of P.

In the first substep, the closest neighbors for each FAAV $p_i \in P$ need to be obtained by using the distance function defined in Equation (2). The distance function accounts for the relevant FAAV attributes. The transformation-invariant attributes need to be considered in obtaining the closest neighbors. The task is carried out analytically as follows. In order to partition the theta range $[0, 2\pi]$ into subintervals within which either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$, it is necessary to find the values of θ such that d(p_i, q_k) = d(p_i, q_l). If there is no such value, it is either d(p_i, q_k) > d(p_i, q_l) or d(p_i, q_k) < d(p_i, q_l) for all the values of θ. The values of θ are obtained solving the following equation:

$$A\cos\theta + B\sin\theta = C \qquad (4)$$

The constant values A, B and C depend on the initial location and orientation of the FAAVs considered, on the center of rotation considered and on the transformation-invariant attributes of the FAAVs considered. The values of the angle θ that are obtained from Equation (4) will partition the theta range $[0, 2\pi]$ into subintervals within which it is easy to verify whether d(p_i, q_k) > d(p_i, q_l) or d(p_i, q_k) < d(p_i, q_l). In some cases, Equation (4) might not have any solution for real values of θ. In this case it is either d(p_i, q_k) > d(p_i, q_l) or d(p_i, q_k) < d(p_i, q_l), which can be verified by substituting any real value for θ in Equation (4).

Observe that the first step of the described algorithm yields the closest neighbors for each FAAV of P separately. A set of theta intervals is built for a particular FAAV $p_i \in P$ such that in each interval the closest FAAV of Q to p_i is known. In Figure 3 the set of theta intervals within the range $[0, 2\pi]$ for the FAAV $p_1 \in P$ is shown. Thus several sets of theta intervals are obtained, one for each FAAV of P. The overlapping of the sets of theta intervals being performed in the second step yields the set of theta intervals for the FAAV s of P. Within each of the intervals the distance given by Equation (1) can be minimized using closed form mathematical formulae. The only independent variable in the formulae is rotationθ.

The sets of theta intervals for each FAAV of P are combined into the set of theta intervals for the FAAVs of P by overlapping so that the resulting range $[0, 2\pi]$ is further partitioned into intervals. Each of the resulting intervals is obtained from the intersection of the intervals of the initial sets. Thus, within any interval of the set of theta intervals for the FAAVs of P, the closest FAAV of Q to each FAAV in P is known. The distance function defined in Equation (1) can now be computed for each interval. The distance function defined in Equation (1) for each interval can be expressed as a function of the location coordinates (x, y, z) and the orientation components (v_x, v_y, v_z) of the FAAVs of P and Q. The location coordinates and the orientation components of P and Q can be expressed as a function of θ, which is the angle of rotation. Thus the distance function defined in Equation (1) is expressed as a function of θ as explained in the next subsection.

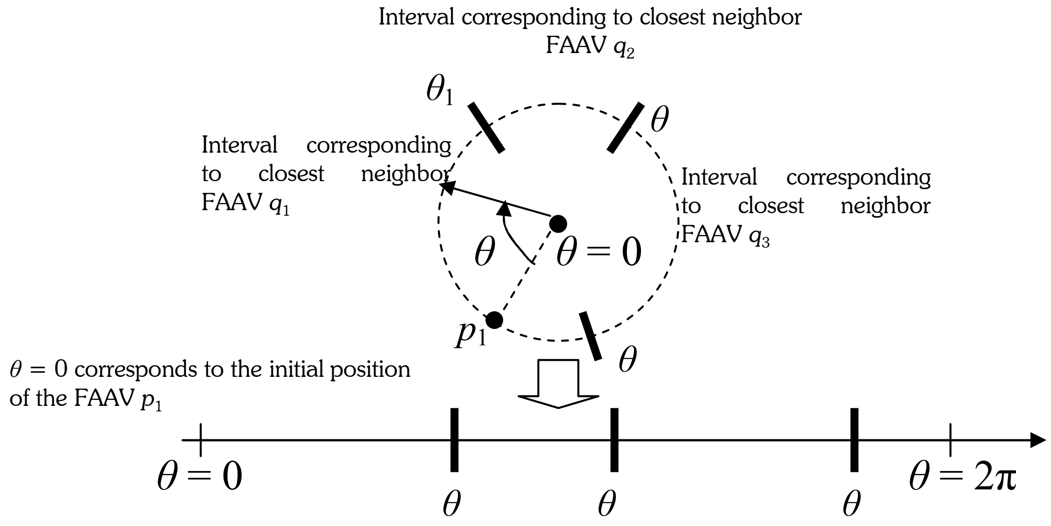
Interval corresponding to closest neighbor FAAV q_2
θ_1
θ
Interval corresponding to closest neighbor FAAV q_1
Interval corresponding to closest neighbor FAAV q_3
θ
$\theta = 0$
p_1
θ
$\theta = 0$ corresponds to the initial position of the FAAV p_1
$\theta = 0$
θ
θ
θ
$\theta = 2\pi$

Fig. 3. Set of Theta Intervals for Face-based Attributed Applied Vector p_1 of P.

5. MINIMIZATION OF THE DISTANCE FUNCTION WITHIN A GIVEN THETA INTERVAL

Consider the location $(x^{p_i}, y^{p_i}, z^{p_i})$ and the orientation $(v_x^{p_i}, v_y^{p_i}, v_z^{p_i})$ of a FAAV p_i in 3D. Let $v_{zo}^{p_i}$ be the initial Z component of the orientation for attributed point $p_i \in P$, while $v_{xyo}^{p_i} = \sqrt{(v_{xo}^{p_i})^2 + (v_{yo}^{p_i})^2}$ is the initial component in the coordinate plane XY before applying the one degree of freedom alignment algorithm described in Section 3. Let (x_B, y_B) be the center of rotation. Define $\theta_o^{p_i}$ as the known initial angle of each FAAV $p_i \in P$ with respect to the center of rotation before applying the one degree of freedom alignment algorithm described above. Similarly let d_{zi} be the Z component and d_{xyi} the XY component of the Euclidean distance between each FAAV $p_i \in P$ and the center of rotation. Also let $\theta_{vo}^{p_i}$ be the known angle of the XY component of the orientation of each FAAV $p_i \in P$ with X axis after the initial alignment between pairs of FAAVs described above.

In the previous subsection the set of theta intervals for all the FAAVs of *P* was built by overlapping the sets of theta intervals of each FAAV. The range [0, 2π] is thus partitioned into a number of intervals. Within each interval the closest FAAV in *Q* to each of the FAAVs in *P* is known. The following definitions, valid within each single interval, will be used.

$$\begin{cases} x^{q_j}(i) = x \text{ coordinate of the position of the closest FAAV } q_j(i) \in Q \text{ to FAAV } p_i \in P \\ y^{q_j}(i) = y \text{ coordinate of the position of the closest FAAV } q_j(i) \in Q \text{ to FAAV } p_i \in P \\ z^{q_j}(i) = z \text{ coordinate of the position of the closest FAAV } q_j(i) \in Q \text{ to FAAV } p_i \in P \\ v_x^{q_j}(i) = x \text{ component of the orientation } v^{q_j}(i) \text{ of the closest FAAV to FAAV } p_i \in P \\ v_y^{q_j}(i) = y \text{ component of the orientation } v^{q_j}(i) \text{ of the closest FAAV to FAAV } p_i \in P \\ v_z^{q_j}(i) = z \text{ component of the orientation } v^{q_j}(i) \text{ of the closest FAAV to FAAV } p_i \in P \end{cases} \tag{5}$$

Consider a single theta interval and a moving FAAV $p_i \in P$. Let θ be the rotation applied to the FAAVs of set *P*. Then the location $(x^{p_i}, y^{p_i}, z^{p_i})$ and the orientation $(v_x^{p_i}, v_y^{p_i}, v_z^{p_i})$ of a FAAV p_i can be written as a function of θ. Hence within a single interval $\vec{d}(P,Q)$ can be written as a function $\vec{d}(\theta)$ of the transformation θ.

Using the notations introduced in Equations (5) $\vec{d}(\theta)$ can be minimized by setting its derivative with respect to θ to zero. By doing this we get the following expression:

$$tg(\theta) = \frac{\sum_{i=1}^{n} \left(\begin{array}{l} w_L[y_B - y^{q_j}(i)]d_{xyi}\cos(\theta_o^{p_i}) - w_L[x_B - x^{q_j}(i)]d_{xyi}\sin(\theta_o^{p_i}) - \\ -w_O v_y^{q_j}(i) v_{xyo}^{p_i}\cos(\theta_o^{v_i}) + w_O v_x^{q_j}(i) v_{xyo}^{p_i}\sin(\theta_o^{v_i}) \end{array} \right)}{\sum_{i=1}^{n} \left(\begin{array}{l} w_L[x_B - x^{q_j}(i)]d_{xyi}\cos(\theta_o^{p_i}) + w_L[y_B - y^{q_j}(i)]d_{xyi}\sin(\theta_o^{p_i}) - \\ -w_O v_x^{q_j}(i) v_{xyo}^{p_i}\cos(\theta_o^{v_i}) - w_O v_y^{q_j}(i) v_{xyo}^{p_i}\sin(\theta_o^{v_i}) \end{array} \right)} \tag{6}$$

Observe that the distance function $\vec{d}(\theta)$ is a continuous function, and it is also bounded. The values of θ resulting from Equation (6) can identify local minima or local maxima of the distance function, depending on the sign of the second derivative. Hence it is necessary to check the sign of the second derivative of $\vec{d}(\theta)$ corresponding to the solution of Equation (6). The values of θ that yield a positive value for the second derivative are local minima. Among them the θ value corresponding to the global minimum will be chosen.

Equation (6) yields the transformation θ, applied to the set of FAAVs *P*, which minimizes the distance between the sets of FAAVs *P* and *Q*. This value of the transformation is valid only within a single interval of the set of theta intervals for all the FAAVs of *P*. In general the value of θ that is found is not guaranteed to lie in the interval where the distance function is defined. Values of θ that lie outside the corresponding interval have no physical meaning and can be discarded. In fact, a theorem described in [4] guarantees that none of them will be the θ value corresponding to the global minimum.

Equation (6) has been obtained by differentiating the distance function with respect to θ, which is a standard minimization technique. Thus, the transformation value obtained for an interval *c* of the set of theta intervals for all the FAAVs of *P* yields the best possible alignment between the two FAAV sets for all permissible transformations within the interval *c*.

6. RESULTS

A software system has been implemented based on the algorithms presented in this paper in C++ programming language using Microsoft Foundation Classes (MFC) and OpenGL on a Windows 2000 platform. We use ACIS 7.0 as the geometric kernel. The input to the system is a query part and the directory in which all the previously designed parts are stored. We first prune parts based on the techniques described in [15]. Then, the system performs the alignment using the algorithms described in this paper. The output models are rank ordered based on this distance function starting with the one having the smallest distance value. FAAV parameters are computed from the boundary representation of the parts.

The database used for all the experiments consists of 150 parts. The weights w_L, w_O, w_A, $w_{\sigma o}$, $w_{\mu c}$, $w_{\sigma c}$, and w_T are selected by the user. The weights can be modified by the user to increase/decrease the influence of FAAV attributes on the distance function.

We compared our algorithms with Geometric Software Systems Limited (GSSL) system. This system is based on the Fourier transformation based technique described in [5, 6]. GSSL system exhibits a very good performance and directly works on boundary representation. Hence it serves as a very good bench mark for assessing discrimination power of our new technique. During these experiments, the weights w_L, w_O, w_A, $w_{\sigma o}$, $w_{\mu c}$, $w_{\sigma c}$, and w_T are set to 1. We conducted many tests and found that in many cases both techniques produced almost identical results. However, we found that there existed cases when the two techniques produced significantly different results. For the sake of brevity, in this paper we only report two representative cases where two techniques produced significantly different results. Figure 4 shows a query part and the top three matches produced by our method. Figure 5 shows the top three results produced by the GSSL system. The top three results reported by our method were not present in the similar parts returned by the GSSL system. Figure 6 shows another query part and the top three matched produced by our techniques. Figure 7 shows the top three results produced by the GSSL system. Once again the top three results reported by our techniques were not present in the similar parts returned by GSSL system. Based on purely qualitative analysis, we believe that our system has a much higher discrimination capability than the GSSL system.

7. CONCLUSION

This paper describes new algorithms for identifying those parts in a database that are similar to a given query part in and hence can be potentially used as a basis for locating potential tool makers for the query part. We have developed a distance function based on FAAVs. We have also developed a new algorithm that performs FAAV alignment to minimize this distance function. We have implemented the algorithm to demonstrate the proof of the concept. We have tested the algorithm on several examples in order to assess its discrimination capability. Our results indicate that the algorithm described in this paper has better discrimination capability than the Fourier transform based technique in an application where type, locations, and orientations of faces in boundary representations plays a key role in determining similarity between two parts. Finding similar parts from the tool maker selection point of view is an example of such an application. Our results show that it is feasible to improve discrimination capability by explicitly aligning part faces. In our opinion, directly utilizing part faces will have the following additional major advantages. First, this method can very easily account for non-geometric attributes (e.g., tolerances) attached to the faces. These additional attributes can be easily added to the distance function. Second, the distance function can be easily customized by adjusting attribute weights and hence this approach can be tailored to new applications. Finally, the use of directional distance functions in our method allows us to easily adopt this technique to support partial matching between parts. In our future work, we plan to present experimental results to illustrate these benefits.

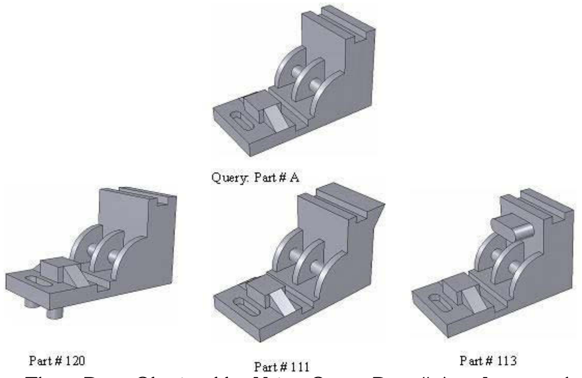

Fig. 4. Top Three Parts Obtained by Using Query Part # A as Input to the System.

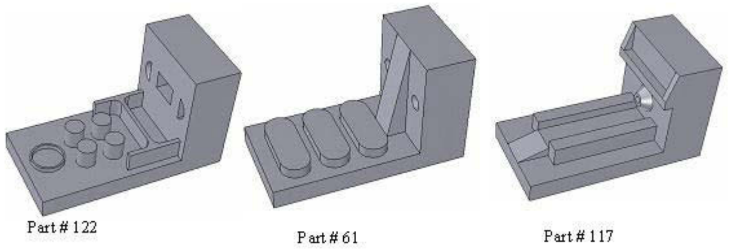

Fig. 5. Top Three Parts Obtained by Using Query Part # A as Input to the GSSL System.

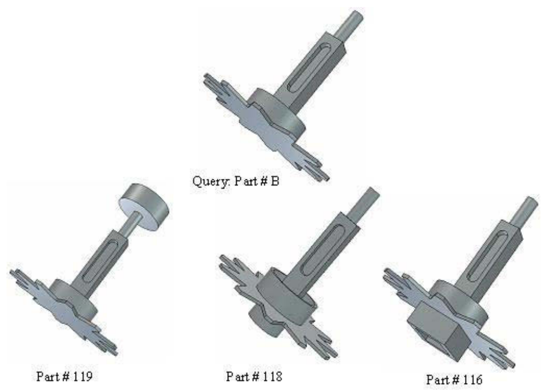

Fig. 6. Top Three Parts Obtained by Using Query Part # B as Input to the System.

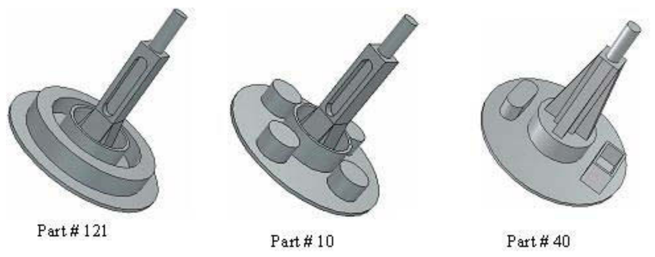

Fig. 7. Top Three Parts Obtained by Using Query Part # B as Input to the GSSL System.

8. REFERENCES

[1] Bespalov, D., Regli, W. C. and Shokoufandeh, A., Reeb Graph Based Shape Retrieval For CAD, In Proceedings Of *23rd ASME DETC Computers And Information In Engineering (CIE) Conference*, Chicago, IL, 2003.

[2] Cardone, A., Gupta, S. K. and Karnik, M., A Survey Of Shape Similarity Assessment Algorithms For Product Design And Manufacturing Applications, *Journal Of Computing And Information Science In Engineering*, Vol. 3, No. 2, 2003, pp 109-118.

[3] Cardone, A., Gupta, S. K. and Karnik, M., Identifying Similar Parts For Assisting Cost estimation Of Prismatic Machined Parts. In Proceedings of *ASME Design For Manufacturing Conference*, Salt Lake City, UT, September 2004.

[4] Cardone, A., A Feature-Based Shape Similarity Assessment Framework, Ph.D. Thesis, Mechanical Engineering Department, University of Maryland, College Park, August 2005.

[5] Chakraborty, T., Venkataraman, S. and Sohoni, M., A fast 3D Shape SearchTechnique For 3D Cax/PDM Repositories, *Technical Paper, Society Of Manufacturing Engineers*, August 16th 2005.

[6] Chakraborty, T., Shape-Based Clustering Of Enterprise CAD Databases, *Computer Aided Design and Applications*, Vol. 2, Nos. 1-4, 2005, pp 145-154.

[7] Chew, L. P., Dor, D., Efrat, A. and Kedem, K., Geometric Pattern Matching In D Dimensional Space, *Discrete And Computational Geometry*, Vol. 21, 1999, pp 257-274.

[8] Cicirello, V. and Regli, W. C., Managing Digital Libraries For Computer-Aided Design, *Computer Aided Design*, Vol. 32, No. 2, 2000, pp 119-132.

[9] Corney, J., Rea, H., Clark, D., Pritchard, J., MacLeod, R. and Breaks, M., Coarse Filters for Shape Matching, *IEEE Computer Graphics and Applications*, Vol. 22, No. 3, 2003, pp 65-74.

[10] El-Mehalawi, M. and Miller, R. A., A Database System Of Mechanical Components Based On Geometric And Topological Similarity, Part II: Indexing, Retrieval, Matching, And Similarity Assessment, *Computer-Aided Design*, Vol. 35, No. 1, 2003, pp 95-105.

[11] Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D. and Jacobs**,** D., A Search Engine for 3D Models, *ACM Transactions on Graphics*, Vol. 22, No. 1, 2003, pp 83-105.

[12] Gupta, S. K., Cardone, A. and Deshmukh, A., Content-Based Search Techniques for Searching CAD Databases, *Computer-Aided Design & Applications*, Vol. 3, No. 6, 2006, pp 811-819.

[13] Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y. and Ramani, K., A Multi-Scale Hierarchical 3D Shape Representation For Similar Shape Retrieval, In Proceedings Of *Tools and Methods for Competitive Engineering Conference*, Lausanne, Switzerland, Aril 2004.

[14] Karnik, M. V., Gupta, S. K. and Magrab, E. B., Geometric Algorithms for Containment Analysis of Rotational Parts, *Computer Aided Design*, Vol. 37, No. 2, 2005, pp 213-230.

[15] Karnik, M. V., Anand, D. K., Eick, E., Gupta, S. K. and Kavetsky, R., Integrated visual and geometric search tools for locating desired parts in a part database, *Computer-Aided Design & Applications*, Vol. 2, No. 6, 2005, pp 727-736.

[16] Ko, K. H., Maekawa, T. and Patrikalakis, N. M., An Algorithm For Optimal Free-Form Object Matching, *Computer Aided Design*, Vol. 35, No. 10, 2003, pp 913-923.

[17] Ko, K. H., Maekawa, T. and Patrikalakis, N. M., Algorithms For Optimal Partial Matching Of Free-Form Objects With Scaling Effects, *Graphical Models*, Vol. 67, No. 2, 2005, pp 120-148.

[18] Lou, K., Prabhakar, S. and Ramani, K., Content Based Three Dimensional Engineering Shape Search, In Proceedings Of *20th International Conference On Data Engineering*, 2004, pp 754-765, Boston, MA.

[19] McWherter, D., Peabody, M., Shokoufandeh, A. and Regli, W.C., Solid Model Databases: Techniques and Empirical Results, *Journal Of Computer And Information Science In Engineering*, Vol. 1, No. 4, 2001, pp 300-310.

[20] Mount, D. M., Netanyahu, N. S. and Le Moigne, J., Efficient Algorithms For Robust Point Pattern Matching, *Pattern Recognition*, Vol. 32, 1999, pp 17-38.

[21] Osada, R., Funkhouser, T., Chazelle, B. and Dobkin, D., Shape Distributions, *ACM Transactions on Graphics*, Vol. 21, No. 4, 2002, pp 807-832.

[22] Ramesh, M., Yip-Hoi, D. and Dutta, D., Feature-Based Shape Similarity Measurement For Retrieval Of Mechanical Parts, *Journal Of Computing And Information Science In Engineering*, Vol. 1, No. 3, 2001, pp 245-256.

[23] Wolfson, H. J. and Rigoutsos, I., Geometric Hashing: An Overview. *IEEE Computational Science and Engineering*, Vol. 4, 1997, pp 10-21.