Pattern Recognition of 3D CAD Objects : Towards an Electronic Yellow Pages of Mechanical Parts*

George Cybenko, Aditya Bhasin and Kurt D. Cohen
Thayer School of Engineering
Dartmouth College, Hanover, NH
3d@lost-ark.Dartmouth.EDU
http://comp-engg-www.dartmouth.edu/~3d

January 25, 1996

Abstract

Industry estimates show that aggressive reuse of existing inventory could reduce the delivered cost of large, complex manufactured systems by as much as 20%. In most cases these savings are not captured, principally because the current technology to locate reusable designs and inventory, which uses taxonomies and other artificial indices, is too cumbersome. Furthermore, current systems do not address the design phase - where opportunities for reuse first arise. This paper describes a computer-based technology that will be an important step towards solving this problem.

For mechanical parts, the modern design phase starts with computer aided design (CAD) packages. Given a prototype design of a solid object, the design engineer should be able to determine whether the part under consideration is already designed or in manufacture. Our system does this by using physical shape as a direct index to existing designs and manufactured components, eliminating time-consuming and error-prone searches of the taxonomy. Other applications of this technology include identification of warehoused parts according to scanned shape and efficient management of 3-dimensional objects in computer animation and virtual reality systems.

Our system takes a standard digital representation of a solid object, such as in IGES form, and produces a surface triangular mesh representing the boundary of the object. The surface mesh allows a voxel approximation representation of the solid which is computed by flood filling. Zeroth, first and second order geometrical moments are used to normalize the orientation of the solid. Then a variety of volumetric invariants are computed and used as features. These features determine a hash function which maps similar shapes to closely related feature vectors. Nearby feature vectors identify a small subset of objects which are compared using symmetric differencing on a voxel by voxel basis. This voxel symmetric difference gives a rank ordering of similarity between 3-dimensional shapes in the database and the object under consideration. Using this small subset of like-shaped objects, the design engineer can browse a reasonable subset of parts from the complete database.

The described research has significant applications in industries which seek to reuse existing designs and inventory thereby reducing manufacturing costs. Applications in aerospace, automobile and machine tool industries are most promising and urgent. We expect that this research will lead to the commercial development of software that will enhance existing CAD and database systems.

1 Introduction

With the advent of powerful Computer-Aided Design (CAD) tools in the last two decades, many manufacturing enterprises use computer software to design and model mechanical parts before building them. Although

[1]This work was supported in part by Air Force Office of Scientific Research Grants F49620-93-1-0266 and F49620-95-1-0305; Navy Office of Naval Research Grant N0014-95-1-1204 and NH Space Grant Consortium, NASA Grant 92-05. Thanks to Markem Corp., Keene, NH, for providing the test dataset.

these tools have improved the design process, they have not addressed the problem of reusing previous designs and existing inventory. To determine whether mechanical parts have already been designed, a designer must resort to legacy taxonomies kept in microfiche and paper records – a time consuming and inefficient process. To avoid such searches, designers typically redesign the part, wasting time and money. For instance, stories of simple fasteners being redesigned in the automobile industry are common. Studies show that while 60 to 80 percent of a product are committed at the design stage, the cost of design engineering is unlikely to exceed 5 percent of the total budget of the project [Har92]. The use of a product that could facilitate reuse in this expensive design stage could significantly reduce design times and costs.

The primary reason for the lack of an appropriate electronic database is the inability to search on the basis of *shape*. If a designer needs a new cam, for example, a database search on the word *cam* will undoubtedly yield far more records than a designer desires. The geometry of cams has no standard description either within or across companies. The same can be said of other mechanical parts.

We have developed a new search technology based on direct three-dimensional representation of an object, which allows search on the basis of *shape*. This search engine will be of particular use to heavily design-dependent industries such as machine tool, automobile and aircraft manufacturers, and would considerably reduce design time and cost. Moreover, potentially even larger savings could be realized in reduced inventory and manufacturing costs.

Such a system can be extended to an electronic “yellow pages” of mechanical parts. Manufacturers of mechanical parts could make networked databases of their products available to designers, allowing them to search for parts all over the world. Some manufacturing and CAD companies (John Deere and Autodesk) are already offering CD-ROM versions of mechanical parts but searches are presently restricted to text and part numbers. We offer the additional ability to search such databases on the basis of shape and to make these searches possible in a distributed way over computer networks. In this paper, we describe the ingredients of such a solid object search engine and its current status.

We describe a solid object search engine which can be used in the following applications when customized appropriately:

- solid object searching by design intensive companies (aerospace, automobile, machine tool, etc.) outside of any particular CAD system;
- CAD database searching within existing CAD systems (Aries, AutoCAD, etc.);
- multimedia database searching of solid objects (Illustra, see `http://www.illustra.com/`);
- inventory and warehouse management;
- management of solid objects for virtual environments (OpenGL, see `http://www.sgi.com/Technology/openGL/`).

We believe that manufacturing CAD systems and networked multimedia technologies are about to combine to allow new efficiencies in manufacturing. Exciting new opportunities exist in applying the techniques we describe here to databases of solid objects descriptions. This area is relatively technically unexplored but the problem itself is well defined and solvable.

2 Background

Our work builds on state-of-the-art ideas in computer representation of solid CAD objects, pattern analysis and distributed information technology. In this section, we must describe the main conceptual steps of our approach.

We begin with the international standard Initial Graphical Exchange Specification (IGES) representation of a mechanical CAD object. Virtually all mechanical CAD systems currently support the export and import of solid objects defined in IGES. An IGES file is essentially a computer program that describes how a solid object can be rendered. Two IGES files describing the same or similar objects will in general be very different, their structure depending on the sequence of design steps taken to define the object. These design steps will be different if done by different designers or even the same designer at different times. Therefore, attempting to compare two IGES files directly is essentially equivalent to asking whether two C programs perform the same computation. If the two files are identical then they describe the same object. However, if the files are different in any substantive way, it is not possible to make a conclusive statement without further analysis.

Our approach to this analysis involves actually rendering the object represented by an IGES CAD file and computing translationally and rotationally invariant features of the resulting object. These steps are quite different from those arising in solid object recognition for image understanding problems. In that application, one or more 2-dimensional images of an object are the starting point or key. In our case, we already begin with a nonunique solid object representation. Object recognition from 2-D image data is notoriously hard but we are solving a simpler problem in some respects. In our application, a major difficulty arises from the fact that we are forced to deal with a rigid standardized language (IGES) creating other types of difficulties.

Once the object is rendered into voxels (3-dimensional cells), we compute various moments which can be used to normalize the orientation of the object. Classical second order moments are among the features we use as keys. Additional invariant features are volumetric moments described below yielding about a dozen invariants altogether. These invariants have the property that small changes in the object result in correspondingly small changes in the features so that these features are robust.

These features are numerical and form the basis for a multidimensional search. The feature vectors define points. Two points are *close* if the objects they represent are geometrically similar at least according to the features we use. Conversely, similar objects map to nearby points. We use Euclidean space distance to measure nearness of the feature vectors.

Given an object's feature vector, our algorithm's first stage returns objects with nearby feature vectors. In the second stage, those objects are rendered into a voxel representation as well and a voxel by voxel comparison is made which essentially computes the symmetric difference between the two objects in a normalized orientation. The symmetric difference is normalized by volume in turn to give a final *relevance* ranking between 0 and 1. A score near 1 means high similarity with smaller values representing proportionally more dissimilarity.

The voxel representation uses much memory and the voxel-by-voxel comparison is extremely time consuming. This is the reason we use relatively small feature vectors in the first stage. Those feature vectors can be computed once for each object's IGES representation. These feature vectors require little additional memory to store and are quickly compared. Only the subset of objects with similar feature vectors are subjected to the more intensive voxel comparison. This is an important step in making our approach tractable.

Our approach is based on geometric moments and not on geometric "features" that a purely rule-based approach might take. That is, one could try to decompose solid objects into cannonical solids such as spheres, cylinders, boxes, and so on, with some spatial relationships between those subobjects. Such approaches have been tried with varying success for two dimensional object matching and could be extended to three dimensions but we are not aware of any such attempts yet. If such a method is tried, it would be appropriate to compare performance with the method described here.

We are not aware of any development efforts along the lines of the work described here. A new company, Illustra (see `http://www.illustra.com/`) is developing multimedia database systems but we are not aware of any solid object capabilities yet.

An outline of the system is shown in Figure 1 below.

3 Algorithmic Approach

3.1 Translation From CAD Format to Solid Object

The first problem we address is that of translating a CAD description of an object to a format which can be used to standardize the object and determine it's geometric features for use in the pattern matching. We use triangular surface meshes to represent the object, a format from which recreating the solid object is practical.

Due to the proprietary nature of various CAD file formats, we investigated CAD interchange formats to perform this first translation step, and evaluated both the IGES and DXF formats.

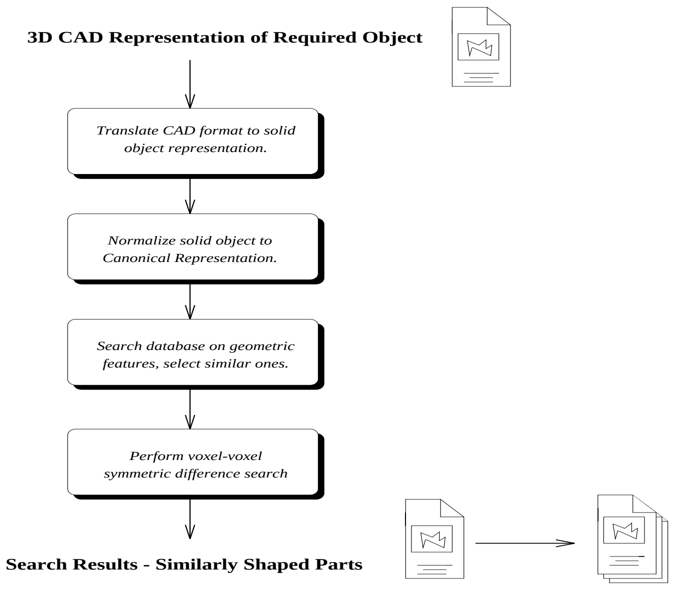
3D CAD Representation of Required Object
Translate CAD format to solid object representation.
Normalize solid object to Canonical Representation.
Search database on geometric features, select similar ones.
Perform voxel-voxel symmetric difference search
Search Results - Similarly Shaped Parts

Figure 1: System Overview

3.1.1 Format Translation

AutoCAD ASCII Drawing Interchange Format (DXF) This format supports three dimensional entities by using a series of 3DLINE, 3DFACE and 3DPOLYMESH primitives, representing three dimensional lines, faces, and the polygons bounding a three dimensional surface respectively. In trying to implement some kind of translator from DXF to three finite element surface mesh mentioned above, we encountered some problems with the format itself. First, there is no standard for how a particular object or face should be represented using combinations of these primitives. For example, a solid sphere created using a particular software package saved the sphere as a series of 3DLINEs, whereas another used arc primitives to generate the same wireframe effect. Secondly, the primitives themselves are somewhat loosely defined. The 3DFACE primitive, for example, may save a face as a set of 3-dimensional points, or four, and there is no requirement that the points should be coplanar. Last, even though the surface primitives are available, the software packages we tested tended not to use them for the simple models we tried. The ambiguities in the "standard" led us to look elsewhere for a more complete and hopefully more rigid format.

Initial Graphics Exchange Specification (IGES) File Format The IGES file format, designed by National Bureau of Standards [US 88] is a very complete and widely used format in the industry. Three dimensional surfaces can be defined in a large number of formats, such as trimmed (parametric) surfaces, spline surfaces, non-uniform rational B-splines (NURBS) surfaces, etc. Again, we found that there were many ways of representing the same surface, and only the NURBS entities lent themselves to triangular mesh generation. However, there are still problems with this approach. Among them, the IGES specification does not insist on NURBS surfaces sharing boundaries completely, and so there could be holes in the resulting mesh. This would not allow a complete reconstruction of the object from the mesh, which depends on the mesh forming a completely closed region.

Until a suitable interchange format can be found, we are using more time-intensive methods to generate the triangular meshes we need from solid object representations in the IGES format, since it is a more complete and widely used format. These approaches include obtaining surface meshes from Finite Element (FE) analysis systems and Stereolithography (SL) modules. Currently, our system supports FE and SL file formats from the MSC/ARIES CAD system, and the T-GRID FE format from the Fluent/GEOMESH CAD system. Work is currently underway to automate this process using new file formats such as STEP and SGI Inventor, the software we use to render and view the images.

Using these finite element or stereolithography translation modules, our system produces a surface triangular representation of an object. The surface triangles are used to create a voxel representation for the discretized object. We use recursive flood filling to fill the exterior of the object and then complement that to get the discretized object in a voxel representation. This voxel representation is the basis for moment computations and subsequent voxel-by-voxel intersection operations which are explained below. It is important to point out that we have done theoretical analyses of this discretization step and can quantify feature computation errors that arise here. Those errors figure into determining acceptable variances for features.

3.2 Normalization of the Object to a Canonical Representation

Once the objects have been appropriately represented, they need to be normalized to a canonical orientation to allow accurate voxel-voxel comparison.

Normalization is a process which transforms an object to a standard orientation. This canonical representation retains all of the geometric information and meets a set of normalization criteria imposed on the moments of the objects [GC93]. The object orientation which satisfies the normalization criterion is not necessarily unique however [AMP85]. This is discussed at greater length in the section on degeneracy. For an object defined by the density function $\rho(x,y,z)$, the three dimensional moments of order lmn are defined as:

$$m_{lmn} = \int\int\limits_{-\infty}^{\infty}\int x^l y^m z^n \rho(x,y,z)\,dx\,dy\,dz \qquad l,m,n = 1,2,3,\ldots$$

The normalization criteria achieve standard translation, volume and orientation.

3.2.1 Canonical Position

Position is normalized by aligning the centroid of the object with the origin of the coordinate system. The centroid is defined as

$$(\hat{x},\hat{y},\hat{z}) = (\frac{m_{100}}{m_{000}}, \frac{m_{010}}{m_{000}}, \frac{m_{001}}{m_{000}}).$$

The position-normalized density function is

$$\hat{\rho}(x,y,z) = \rho(x-\hat{x}, y-\hat{y}, z-\hat{z}).$$

Moments of this density function are central moments μ_{lmn}.

3.2.2 Canonical Size

Normalization of size is achieved by scaling the object to a specified volume, C. A scaling factor α is applied so that

$$m_{000} = \int\int\limits_{-\infty}^{\infty}\int \hat{\rho}(\alpha x, \alpha y, \alpha z) = C.$$

3.2.3 Canonical Orientation

The criteria for normalization of orientation are imposed on the central moments of the object, μ_{lmn}. The first condition is met by applying a rotation to the object so that the matrix of second-order central moments, $\mathbf{M}$, becomes diagonal. Using the notation $\mu_{xy} = \mu_{110}$ and so on, the moment matrix is defined as

$$\mathbf{M} = \begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{xy} & \mu_{yy} & \mu_{yz} \\ \mu_{xz} & \mu_{yz} & \mu_{zz} \end{pmatrix}$$

When properly oriented, this matrix is diagonal $\mu_{xy} = \mu_{xz} = \mu_{yz}$. This rotation, which is computed from the eigenvectors of the original moment matrix, aligns the principal axes of the object along the coordinate axes. These are at least eight symmetries which satisfy this. To remove this ambiguity, further conditions are imposed on the orientation. We further require that $\mu_{xx} \geq \mu_{yy} \geq \mu_{zz}$ and that the maximum extent of the object is greater in each positive half-space.

3.2.4 Degeneracy

When two or more of the diagonal central moments, or extents in the positive and negative directions are equal, this method will not produce a unique normalized orientation. This is a problem in practice when these quantities are equal to within the variances/tolerances expected given the discretization procedure. Objects which are symmetric about one axis can be recognized [Mar89]. Two dimensional rotationally symmetric objects can be normalized as well [PL92, TC91]. Some of the methods used to normalize 2-D objects can be extended to 3-D objects, but the computation of the necessary descriptors is complicated and may be inaccurate due to numerical computations. When degeneracy is a problem, various rotations and discrete symmetries can be compared against the template, or other features can be used. Canonical orientation is important only for voxel-by-voxel comparisons and do not affect the rotationally invariant features that we use in the first stage of the procedure.

3.3 Feature Extraction and Database Searching

Once we have the normalized, digital representation of the object, we extract features from the objects which make searching the database efficient.

3.3.1 Rotation-Scale-Translation Invariants

RST invariants are features which are invariant to *rotation*, *shifting*, and *translation*. These features can be used to compare all objects including discrete density functions sampled at a different rate, and objects which are rotationally symmetric.

3.3.2 Second-Order 3D Moment Invariants

Second-order 3D moment invariants combine second order moments of the density function to produce an *RST* invariant. Three of these features are extracted. These features are made invariant to translation by using central moments. The central moments are made invariant to scaling by normalizing with respect to the volume, μ_{000}.

$$\kappa_{lmn} = \frac{\int \int_{-\infty}^{\infty} \int x^l y^m z^n \rho(x,y,z)\, dx\, dy\, dz}{\mu_{000}^{\frac{5}{3}}} \qquad l+m+n=2$$

Since the characteristic function is invariant under rotation, the characteristic equation, $P(\lambda) = \lambda^3 + a\lambda^2 + b\lambda + c$, of the matrix yields translation and scale invariant second-order features. Namely, the coefficients a, b, and c are the features we use [SH80, LD89].

3.3.3 Spherical-Kernel Moment Invariants

Like the second-order moment invariants, the *spherical-kernel moment invariants* use central moments to achieve invariants to translation. The centroid of the object is the center of a spheres S_r used as the kernel of the moment. Sphere features are generated by the equation:

$$(Sf)_r = \int \int_{-\infty}^{\infty} \int S_r \rho(x,y,z)\, dx\, dy\, dz.$$

In addition to being invariant to translation, these features are also invariant to rotation since the sphere is invariant to rotation. Scale invariance is achieved by normalizing these features with respect to a volume and by using the density function of an object which has been scaled to a standard volume. Two sets of *spherical-kernel moment invariants* have been tested. The first set is generated by a set of sphere with radii that increase linearly and are normalized with respect to the maximum value the sphere feature, Sf_1, could attain.

$$S1_r = \frac{Sf_r}{\int_0^{2\pi} \int_0^{\pi} \int_0^{i} d\phi\, d\theta\, dr}$$

These features weight all radii equally paying no particular attention to any region of the object. The second set of *spherical-kernel moment invariants* pays more attention to the outer edge of objects where similar objects may differ. These features are generated with a set of spheres with radii increasing as r^3 and are normalized with respect to the volume of the object:

$$S2_i = \frac{Sf_i}{m_{000}}.$$

3.3.4 Other Features

Various heuristic features are calculated while calculating the above features. Such features include the dimensions of the smallest rectangular box aligned with the coordinate axes which contains the object. The centroid of the object is another of these features, and the surface area of the object is the final feature calculated.

3.4 Searching the database

After normalizing the object, the system places the normalized form in a database. The object is completely described within this database by three kinds of files

1. A list of triangular surface mesh nodes and connectivities that define the object
2. the RST invariant features calculated for the object, and,
3. the digital voxel representation of the object.

The database is first searched by performing a cosine query on relevant features in the feature file.

The features extracted from the objects are normalized to be of order one. These feature vectors are compared based on the angle between them. The comparison metric we use is:

$$C_{\mathbf{xy}} = 1 - 2\frac{\arccos\left(\frac{\mathbf{x}\cdot\mathbf{y}}{|\mathbf{x}||\mathbf{y}|}\right)}{\pi}$$

This procedure is very fast, and efficient for searching through a large database of objects not only because of its speed, but also because of its computationally inexpensive nature. A more intensive (but less efficient) search can be performed between two objects by comparing them voxel-for-voxel.

3.4.1 Voxel-Voxel Comparison

Template matching is an approach to comparing two 3D objects. Mismatch between two normalized objects with density functions f and g can be computed as

$$\int\int_{-\infty}^{\infty}\int (f-g)^2\,dv = \int\int_{-\infty}^{\infty}\int f^2\,dv + \int\int_{-\infty}^{\infty}\int g^2\,dv - 2\int\int_{-\infty}^{\infty}\int fg\,dv.$$

For given objects, $\int\int_{-\infty}^{\infty}\int f^2\,dv$ and $\int\int_{-\infty}^{\infty}\int g^2\,dv$ are fixed. The mismatch is small only if the match $\int\int_{-\infty}^{\infty}\int fg\,dv$ is large [RK76]. For digital representations of the objects, normalized *voxel-voxel* match M_{fg} is measured in an analogous way.

$$M_{fg} = \frac{\sum_i\sum_j\sum_k f(i,j,k)g(i,j,k)}{Max(\sum_i\sum_j\sum_k f(i,j,k), \sum_i\sum_j\sum_k g(i,j,k))}$$

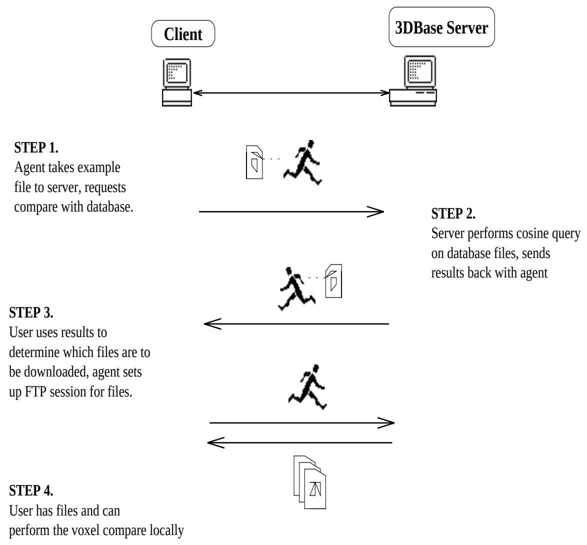
Client
3DBase Server
STEP 1.
Agent takes example file to server, requests compare with database.
STEP 2.
Server performs cosine query on database files, sends results back with agent
STEP 3.
User uses results to determine which files are to be downloaded, agent sets up FTP session for files.
STEP 4.
User has files and can perform the voxel compare locally

Figure 2: Schematic of the Agent-based Network Implementation

This compares f and g voxel-for-voxel, so it measures both the similarity in shape as well as differences in orientation. If both objects are manipulated to a canonical position, size and orientation, the *voxel-voxel* match measures geometric similarity.

In the *voxel-voxel* comparison, the density functions are directly compared against each other. It can be more efficient to extract features form the objects and compare based on those features. The features should contain much of the information found in the density function, but in a more compact representation. These features may also overcome the problems of comparing rotationally symmetric objects if they are invariant to rotation.

3.5 Agent-Based Networked Implementation

All the steps above have been put together to create a system that may be used as a networked yellow pages of mechanical parts. We use an agent-based architecture to implement such a network so that queries may be made over the entire network. An intelligent agent system is being concurrently developed at Dartmouth for use in this application (see `http://www.cs.dartmouth.edu/~agent`) [Gra95]. The structure of the system is shown in figure 2.

In order to perform a search, a user at a client machine generates a rough query in their CAD system, and exports it to the 3dbase system in one of the available Finite Element or Stereolithography formats. 3dbase then generates the canonical representation, calculates the features, and sends the features of the object to the remote 3dbase server on which a query is to be carried out, using an agent (step 1). The server then performs the fast feature comparison, and return the results, via the same agent, back to the querying machine(step 2), where the user can determine which files they want to study more carefully. The local agent then automatically sets up a File Transfer Protocol (FTP) session to download the files from the remote machine that have been tagged by the user (step 3). The more exhaustive voxel-by-voxel searches can be then performed by communicating entire object files back to the local machine so as to minimize resource utilization on the remote machine (step 4).

3.6 Results

We obtained a small collection of experimental parts from a local hardware manufacturer to test the database. The CAD descriptions of these parts were supplied to us in IGES format. Each file was then read into the MSC/ARIES CAD package from which we obtained a triangular mesh of the object using the stereolithography package. Once all the files were meshed, they were read into the 3dbase system, where they were

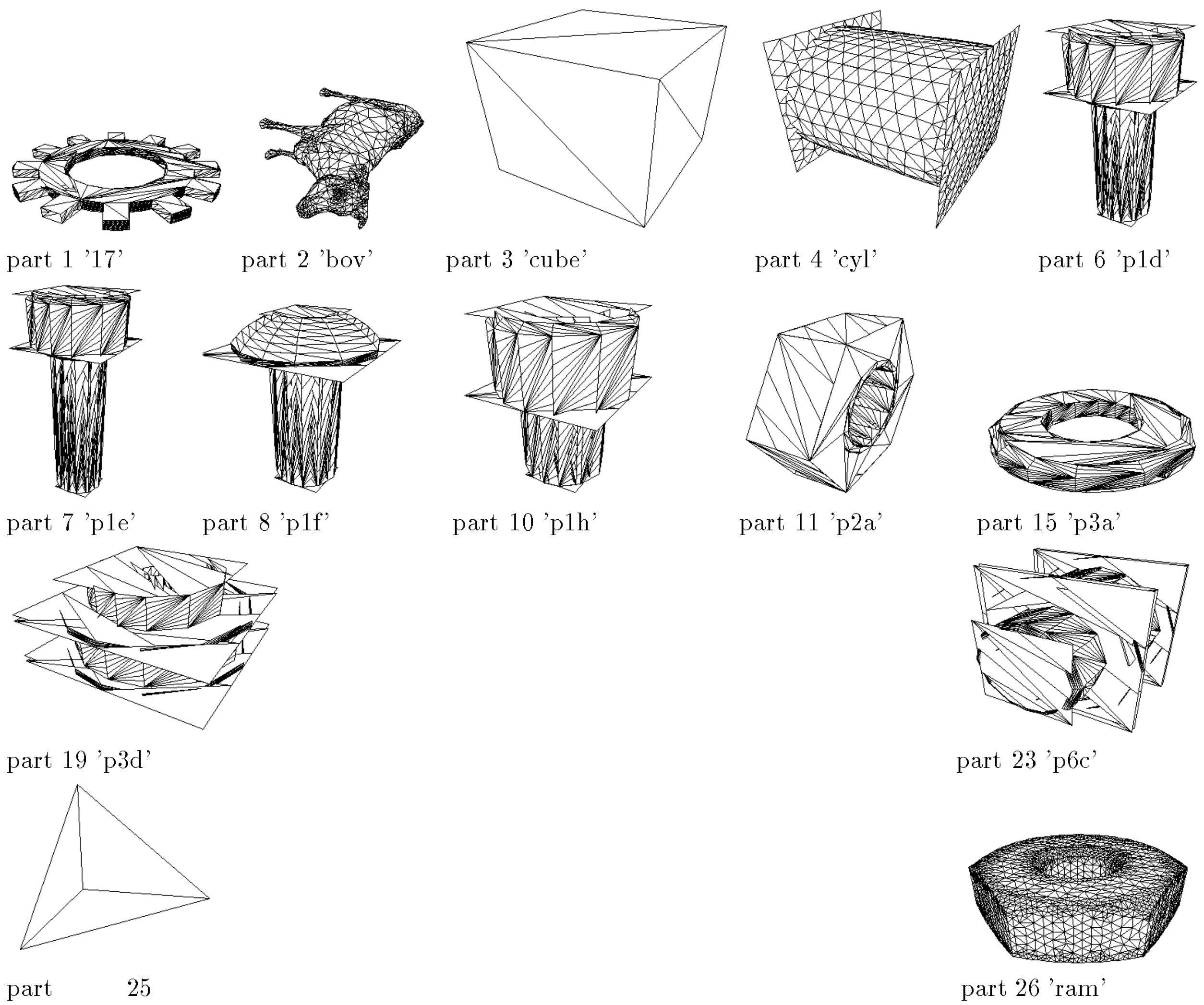

Figure 3: Selected parts from the experimental database

digitized and normalized as explained above, at three separate resolutions (10, 20 and 100 thousand voxels). Samples from the database, in their canonical forms, are shown in figure 3.

The database consists of different classes of objects, screws, washers, nuts, gears, and some standard objects: a cylinder, a cube and a pyramid. We performed both the feature and voxel-by-voxel compares on this database with a view to determining :

- if the system was able to recognize similar objects as a human expert would
- how the feature-based algorithm compared with the "brute-force" voxel-by-voxel algorithm as a measure of similarity

Figure 4 shows the results of the feature compare of the database against itself. Each small square in the plot represents the similarity coefficient of the two objects compared. A white square is a perfect match (similarity=1.0) and a black square indicates no similarity at all. One noticeable feature in the plot is the presence of light square regions along the main diagonal. This can be attributed to the fact that like objects in the database are numbered sequentially (e.g. parts 5-9 (Fig 3) are all screws). The light squares thus show that the feature compare identifies similar looking parts correctly. Figure 5 shows the results of a feature

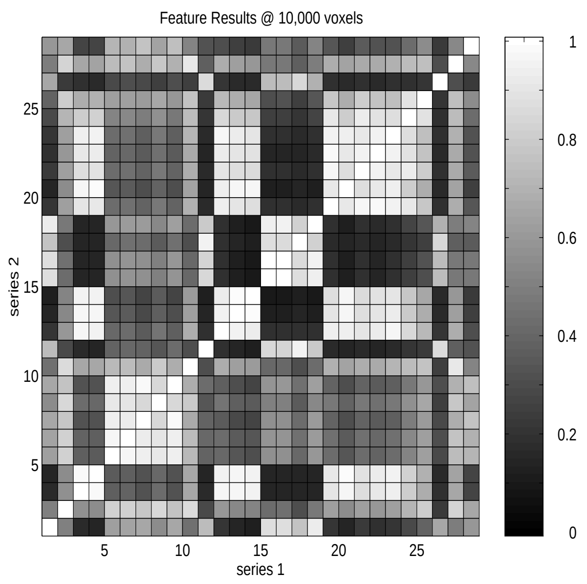

Figure 4: Database feature-compared to itself, objects at 10 Kvoxels

compare of all the screws at all three resolutions, which emphasizes this point, and shows additionally that there is no significant effect of resolution on the similarity coefficients generated. This plot indicates that the last screw is somewhat different from the rest of the screws. Figure 3 confirms this, the last screw (part 10), has a shorter, wider stem, and a longer, wider head than the rest of the screws.

Figure 6 shows the database compared to itself using the voxel-voxel symmetric difference compare. Once again, we can see the lighter square regions near the diagonal indicating the high similarity coefficients of like objects. Both the feature and the voxel plots also exhibit distinct dissimilarities between objects 15-19 and the rest of the database. This is because these objects are all washers (Figure 3), which have holes in the middle of them, whereas most of the objects are filled solids. It is important to note that this phenomenon is more obvious in the voxel compare than in the feature compare. The voxel compare yields a very low similarity coefficient because of the volume normalization of the symmetric difference, whereas the feature compare yields a higher one because of the radial symmetries of the objects. This also indicates an additional shortcoming in the voxel compare in that it is highly sensitive to the orientation of the object if the object is rotationally symmetric.

On the whole, the feature and voxel compares are not extremely different. Figure 7 shows the difference in similarity coefficients of the two compares. Most of the squares are very dark (indicating almost no difference) except for parts 11 and 26, both of which are rotationally symmetric objects for which the voxel compare yields very low similarity coefficients.

On this small database of mechanical parts, the 3dbase system performs very well in determining the similarities of various objects in the database.

4 Summary

We have described a working system for CAD solid object indexing and retrieval. The system begins with IGES file formats and extracts various geometric moments that are rotationally, translationally and scale invariant. Those moments are used as features to index the objects and compare them for similarity. Finer resolution comparisons are done using voxel by voxel symmetric differencing. Our implementation and experiments indicate that the method is very promising for more complete automation. Our work addresses shape matching and recognition but does not at present deal with text that is embedded in the CAD files. Virtually all mechanical design CAD systems allow annotation of the file in various ways. This text can be extracted from an IGES representation and used in a free text search to complement the purely

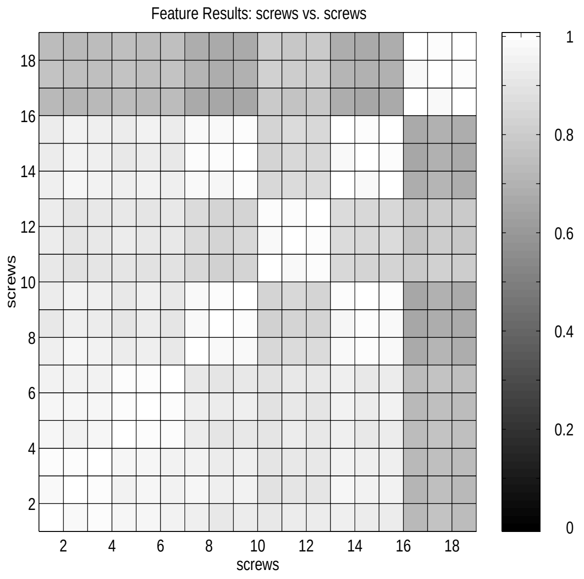

Figure 5: All screws feature compare, objects at all resolutions

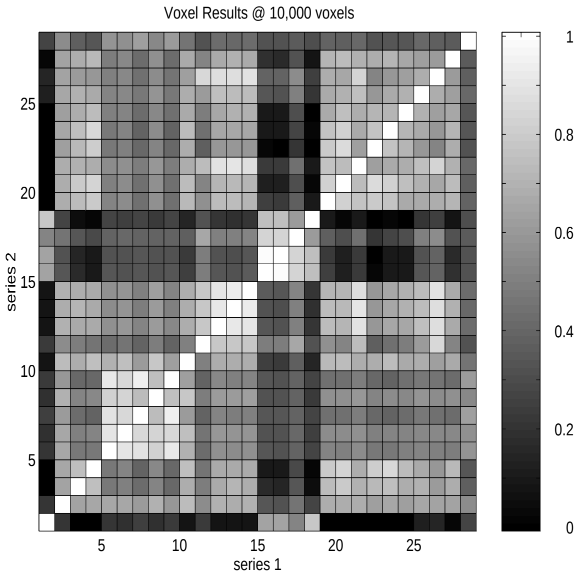

Figure 6: Database voxel-compared to itself, objects at 10 Kvoxels

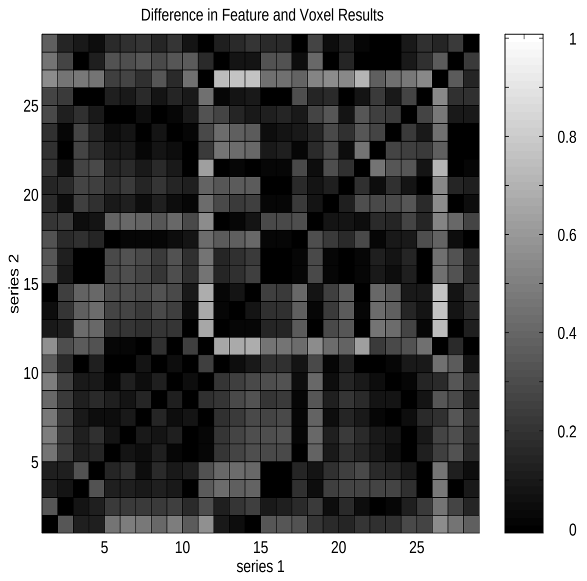

Figure 7: Difference between feature and voxel similarity coefficients

geometric search we currently perform. This text often contains taxonomic references, dimensions, material composition, machining information and end application of the part. Software systems for free text searching are relatively mature and widely available. They typically return a numerical “relevance” ranking which can be combined with the geometric ranking we now provide. The combination can be in the form of a linear combination specified by the user. Future work on this project includes scalability studies to see if the approach scales to thousands of objects and development of text search capabilities.

References

[AMP85] Yaser S. Abu-Mostafa and Demetri Psaltis. Image normalization by complex moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):46–55, January 1985.

[GC93] J.M. Galvez and M. Canton. Normalization and shape recognition of three-dimensional objects by 3d moments. *Pattern Recognition*, 26(5):667–680, 1993.

[Gra95] Robert S. Gray. Agent tcl: A transportable agent system. In *Proccedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95)*, December 1995.

[Har92] J.R. Hartley. *Concurrent Engineering : Shortening Lead Times, Raising Quality, and Loweing Costs.* Productivity Press, 1992.

[LD89] Chong-Huah Lo and Hon-Son Don. 3-d moment forms: Their construction and application to object identification and positioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1053–1064, October 1989.

[Mar89] Giovanni Marola. On the detection of the axes of symmetry of symmetric and almost symmetric planar images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):104–108, January 1989.

[PL92] Soo-Chang Pei and Chao-nan Lin. Normalization of rotationally symmetric shapes for pattern recognition. *Pattern Recognition*, 25(9):913–920, 1992.

[RK76] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing.* Academic Press, 1976.

[SH80] Firooz A. Sadjadi and Ernest L. Hall. Three-dimensional moment invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(2):127–136, March 1980.

[TC91] Wen-Hsiang Tsai and Sheng-Lin Chou. Detection of generalized principal axes in rotationally symmetric shapes. *Pattern Recognition*, 24(2):127–136, 1991.

[US 88] US Department of Commerce. *Initial Graphics Exchange Specification 4.0*, June 1988.