Shape Distributions

Robert Osada, Thomas Funkhouser,
Bernard Chazelle, and David Dobkin
Princeton University, Princeton, NJ 08540, USA

February 12, 2002

Abstract

Measuring the similarity between 3D shapes is a fundamental problem, with applications in computer graphics, computer vision, molecular biology, and a variety of other fields. A challenging aspect of this problem is to find a suitable shape signature that can be constructed and compared quickly, while still discriminating between similar and dissimilar shapes.

In this paper, we propose and analyze a method for computing shape signatures for arbitrary (possibly degenerate) 3D polygonal models. The key idea is to represent the signature of an object as a *shape distribution* sampled from a *shape function* measuring global geometric properties of an object. The primary motivation for this approach is to reduce the shape matching problem to the comparison of probability distributions, which is simpler than traditional shape matching methods that require pose registration, feature correspondence, or model fitting.

We find that the dissimilarities between sampled distributions of simple shape functions (e.g., the distance between two random points on a surface) provide a robust method for discriminating between classes of objects (e.g., cars versus airplanes) in a moderately sized database, despite the presence of arbitrary translations, rotations, scales, mirrors, tessellations, simplifications, and model degeneracies. They can be evaluated quickly, and thus the proposed method could be applied as a pre-classifier in a complete shape-based retrieval or analysis system concerned with finding similar whole objects. The paper describes our early experiences using shape distributions for object classification and for interactive web-based retrieval of 3D models.

1 Introduction

Determining the similarity between 3D shapes is a fundamental task in shape-based recognition, retrieval, clustering, and classification. Its main applications have traditionally been in computer vision, mechanical engineering, and molecular biology. However, due to three recent developments, we believe that 3D model databases will become ubiquitous, and the applications of 3D shape analysis and matching will expand into a wide variety of other fields. First, improved modeling tools and scanning devices are making acquisition of 3D models easier and less expensive, creating a large supply of publically available 3D data sets (e.g., the Protein Data Bank [34]). Second, the World Wide Web is enabling access to 3D models constructed by people all over the world, providing a mechanism for wide-spread distribution of high quality 3D models (e.g., avalon.viewpoint.com). Finally, 3D graphics hardware and CPUs have become fast enough and cheap enough that 3D data can be processed and displayed quickly on desktop computers, leading to a high demand for 3D models from a wide range of sources.

Unfortunately, since most 3D file formats (VRML, 3D Studio, etc.) have been designed for visualization, they contain only geometric and appearance attributes, and usually lack semantic information that would facilitate automatic matching. Although it is possible to include meaningful structure and semantic tags in some 3D file formats (the "layer" field associated with entities in AutoCad models is a simple example), the vast majority of 3D objects available via the World Wide Web will not have them, and there are few standards regarding their use. In general, 3D models will be acquired with scanning devices, or output from geometric manipulation tools (file format conversion programs), and thus they will have only geometric and appearance information, usually completely void of structure or semantic information. Automatic shape-based matching algorithms will be useful for recognition, retrieval, clustering, and classification of 3D models in such databases.

Databases of 3D models have several new and interesting characteristics that significantly affect shape-based matching algorithms. Unlike images and range scans, 3D models do not depend on the configuration of cameras, light sources, or surrounding objects (e.g., mirrors). As a result, they do not contain reflections, shadows, occlusions, projections, or partial objects, which greatly simplifies finding matches between objects of the same type. For example, it is plausible to expect that the 3D model of a horse contains exactly four legs of roughly equal size. In contrast, any 2D image of the same horse may contain fewer than four legs (if some of the legs are occluded by tall grass), or it may contain "extra legs" appearing as the result of shadows on the barn and/or reflections in a nearby pond, or some of the legs may appear smaller than others due to perspective distortions. These problems are vexing for traditional computer vision applications, but generally absent from 3D model matching.

In other respects, representing and processing 3D models is more complicated than for sampled multimedia data. The main difficulty is that 3D surfaces rarely have simple parameterizations. Since 3D surfaces can have arbitrary topologies, many useful methods for analyzing other media (e.g., Fourier analysis) have no obvious analogs for 3D surface models. Moreover, the dimensionality is higher, which makes searches for pose registration, feature correspondences, and model parameters more difficult, while the likelihood of model degeneracies is higher. In particular, most 3D models in large databases, such as the World Wide Web, are represented by "polygon soups" – unorganized and degenerate sets of polygons. They seldom have any topology or solid modeling information; they rarely are manifold; and most are not even self-consistent. We conjecture that *almost every 3D computer graphics model available today contains missing, wrongly oriented, intersecting, disjoint, and/or overlapping polygons*. As a few classic examples, the Utah teapot is missing its bottom and rim, and the Stanford Bunny [43] has several holes along its base. The problem with these degenerate representations is that most interesting geometric features and shape signatures are difficult to compute, and many others are ill-defined (e.g., what is the volume of a teapot with no bottom?). Meanwhile, fixing the degeneracies in such 3D models to form a consistent solid region and manifold surface is a difficult problem [12, 32, 48], often requiring human intervention to resolve ambiguities.

In this paper, we describe and analyze a method for computing 3D shape signatures and dissimilarity measures for arbitrary objects described by possibly degenerate 3D polygonal models. The key idea is to represent the signature of an object as a *shape distribution* sampled from a *shape function* measuring global geometric properties of the object. The primary motivation for this approach is that the shape matching problem is reduced to the comparison of two probability distributions, which is a relatively simple problem when compared to the more difficult problems encountered by traditional shape matching methods, such as pose registration, parameterization, feature correspondence, and model fitting. The challenges of this approach are to select discriminating shape functions, to develop efficient methods for sampling them, and to compute the dissimilarity of probability distributions robustly. This paper presents our initial steps to address these issues. For each issue, we describe several options and present experimental evaluation of their relative performance. Overall, we find that the proposed method is not only fast and simple to implement, but it also provides useful discrimination of 3D shapes and thus is suitable as a pre-classifier for a recognition or similarity retrieval system.

The remainder of the paper is organized as follows. The next section contains a summary of related work. An overview of the proposed approach appears in Section 3, while detailed descriptions of issues and proposed solutions for implementing our approach appear in Section 4. Section 5 presents results of experiments aimed at evaluating the robustness and discrimination of shape distributions. Section 6 describes our initial efforts in building a 3D search engine based on shape distributions. Finally, Section 7 contains a summary of our experiences and proposes topics for future work.

2 Related Work

The problem of determining the similarity of two shapes has been well-studied in several fields. For a broad introduction to shape matching methods, please refer to any of several survey papers [2, 8, 15, 46, 51, 70]. To briefly review, prior matching methods can be classified according to their representations of shape: 2D contours, 3D surfaces, 3D volumes, structural models, or statistics.

The vast majority of work in shape matching has focused on characterizing similarity between objects in 2D images (e.g., [21, 30, 39, 49]). Unfortunately, most 2D methods do not extend directly to 3D model matching. The main problem is boundary parameterization. Although the 1D boundary contours of 2D shapes have a natural arc length parameterization, 3D surfaces of arbitrary genus do not. As a result, common representations of 2D contours for shape matching, such as Fourier descriptors [6], turning functions [7], bending energy functions [73], arch height functions [45], size functions [68, 69], and order structures [20] have no direct analogs for 3D models. Similarly, computationally efficient methods based on dynamic programming (e.g., [63] and [67]) cannot be applied to 3D objects. Another problem is that the dimensionality of 3D data is higher, which makes registration, finding feature correspondences, and fitting model parameters more expensive. As a result, methods that match shapes using deformations [3, 40, 50, 57, 65] are far more difficult in 3D.

Shape matching has also been well-studied for 3D objects. For instance, representations for registering and matching 3D surfaces include Extended Gaussian Images [35], Spherical Attribute Images [24, 25], Harmonic Shape Images [74], and Spin Images [41]. Unfortunately, these previous methods usually assume that a topologically valid surface mesh or an explicit volume is available for every object. In addition, volumetric dissimilarity measures based on wavelets [31] or Earth Mover's Distance [56] usually rely upon a priori registration of objects' coordinate systems, which is difficult to achieve automatically and robustly. Geometric hashing [44] is a potential solution, but it requires a large amount of storage for complex models.

Another popular approach to shape analysis and matching is based on comparing high-level representations of shape. For instance, model-based approaches first decompose a 3D object into a set of features (or parts), and then compute a dissimilarity measure between objects based on the differences between their features and/or their spatial relationships. Example representations of this type include generalized cylinders [18], superquadrics [61], geons [72], deformable regions [13], shock graphs [59], medial axes [11], and skeletons [19, 62]. These methods work best when 3D models can be segmented into a canonical set of features naturally and correspondences can be found between features robustly. Unfortunately, these tasks are difficult and not always well-defined for arbitrary 3D polygonal models (e.g., what is the canonical skeleton for an unconnected set of polygons?). Moreover, feature detection and segmentation algorithms tend to be sensitive to small perturbations to the model, placing undue burden on subsequent feature correspondence and dissimilarity computation steps. Finally, the combinatorial complexity of finding correspondences in large discrete models usually leads to long computation times and/or large storage requirements.

Finally, shapes have been compared on the basis of their statistical properties. The simplest approach of this type is to evaluate distances between feature vectors [26] in a multidimensional space where the axes encode global geometric properties, such as circularity, eccentricity, or algebraic moments [52, 64]. Other methods have compared discrete histograms of geometric statistics. For example, Thacker et al [1, 5, 9, 10, 28, 29, 55, 66], Huet et al. [36], and Ikeuchi et al. [38] have all represented shapes in 2D images by histograms of angles and distances between pairs of 2D line segments. Belongie et al. [14] and Mori et al. [47] have represented 2D shapes by the distributions of points on their boundaries with respect to multiple reference frames. For 3D shapes, Ankerst et al. [4] has used shape histograms decomposing shells and sectors around a model's centroid. Besl [16] has considered histograms of the crease angle for all edges in a 3D triangular mesh. Besl's method is the most similar to our approach. However, it works only for manifold meshes, it is sensitive to cracks in the models and small perturbations to the vertices, and it is not invariant under changes to mesh tessellation. Moreover, the histogram of crease angles does not always match our intuitive notion of rigid shape. For example, adding any extra arm to a human body results in the same change to a crease angle histogram, no matter whether the new arm extends from the body's shoulder or the top of its head.

To summarize, many previous approaches have difficulty with 3D polygon soups because they invariably require a solution to at least one of the following difficult problems: reconstruction, parameterization, registration, or correspondence. The motivation behind our work is to develop a fast, simple, and robust method for matching 3D polygonal models without solving these problems.

3 Overview of Approach

Our approach is to represent the shape signature for a 3D model as a probability distribution sampled from a *shape function* measuring geometric properties of the 3D model. We call this generalization of geometric histograms a *shape distribution*. For instance, one example shape distribution, which we call *D2*, represents the distribution of Euclidean distances between pairs of randomly selected points on the surface of a 3D model. Samples from this distribution can be computed quickly and easily, while our hypothesis is that the distribution describes the overall shape of the represented object. Once we have computed the shape distributions for two objects, the dissimilarity between the objects can be evaluated using any metric that measures distance between distributions (e.g., L_N norm), possibly with a normalization step for matching scales.

The key idea is to transform an arbitrary 3D model into a parameterized function that can be compared with others easily (see Figure 1). In our case, the domain of the shape function provides the parameterization (e.g., the *D2* shape distribution is a 1D function parameterized by distance), and random sampling provides the transformation.

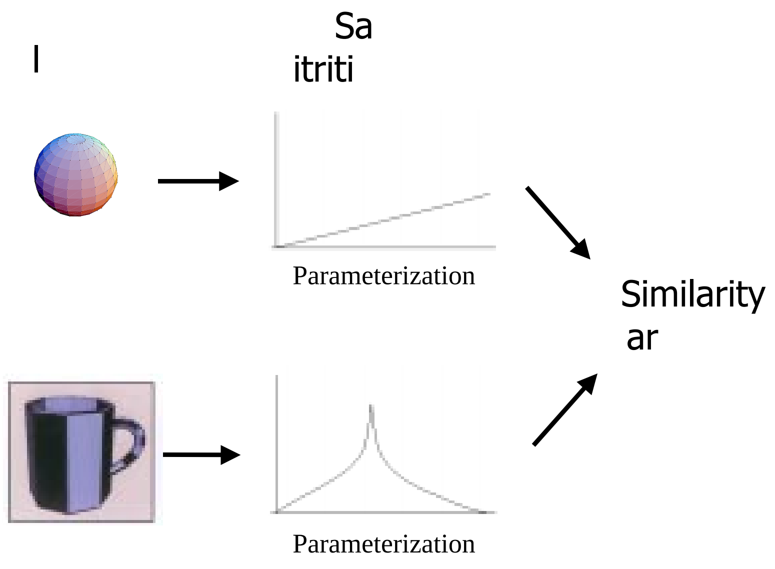


Figure 1: Shape distributions facilitate shape matching because they represent 3D models as functions with a common parameterization.

The primary advantage of this approach is its simplicity. The shape matching problem is reduced to sampling, normalization, and comparison of probability distributions, which are relatively simple tasks when compared to prior methods that require reconstructing a solid object or manifold surface from degenerate 3D data, registering pose transformations, finding feature correspondences, or fitting high-level models. Our approach works directly on the original polygons of a 3D model, making few assumptions about their organization, and thus it can be used for similarity queries in a wide variety of databases, including ones containing degenerate 3D models, such as those currently available on the World Wide Web.

In spite of its simplicity, we expect that our approach is useful for discriminating whole objects with different gross shapes (results of experiments testing this hypothesis are presented in Section 5). The motivations for this hypothesis is that it satisfies several properties desirable for a shape matching method:

- **Invariance:** shape distributions have all the transformation invariance properties of the sampled shape function. For instance, the *D2* shape function yields invariance under rigid motions and mirror imaging. In this case, invariance under scaling can be added by normalization of shape distributions before comparing them and/or by factoring out scale during the comparison. Other shape functions that measure angles or ratios between lengths are invariant to all similarity transformations.

- **Robustness:** random sampling ensures that shape distributions are insensitive to small perturbations. Intuitively, since every point in a 3D model contributes equally to the shape distribution, the magnitude of changes to the shape distribution are related to the magnitude of the changes to the 3D model. For example, if a small percentage of a 3D model is perturbed (e.g., by adding random noise, by adding a small bump onto a surface, or by adding small objects arbitrarily throughout space), then a distribution of random samples from the model must also change by a small percentage. This property provides insensitivity to noise, blur, cracks, and dust in the input 3D models. We conjecture that the distributions for most global shape functions based on distances and/or angles also vary continuously and monotonically for local shape changes.

- **Metric:** the dissimilarity measure produced by our approach adopts the properties of the norm we use to compare shape distributions. In particular, if the norm is a metric, so is our dissimilarity measure. This property holds for most common norms, including L_N norms, Earth Mover's Distance, etc.

- **Efficiency:** construction of the shape distributions for a database of 3D models is generally fast and efficient. For instance, the complexity of taking S samples of the *D2* shape function from a 3D model with N triangles is $Slog(N)$. The resulting shape distributions can be approximated concisely by functions with constant complexity storage and comparison costs.

- **Generality:** shape distributions are independent of the representation, topology, or application domain of the sampled 3D models. As a result, our shape similarity method can be applied equally well to databases with 3D models stored as polygon soup, meshes, constructive solid geometry, voxels, or any other geometric representation as long as a suitable shape function can be computed from each representation. Moreover, a single database (such as the World Wide Web) can contain 3D models in a variety of different representations and file formats. Finally, shape distributions can be used in many different application domains for comparison of natural, deformable shapes (e.g., animals) and/or man-made objects (e.g., machined parts).

The interesting issues to be addressed in implementing the proposed shape matching approach are: 1) to select discriminating shape functions, 2) to construct shape functions for each 3D model efficiently, and 3) to compute a dissimilarity measure for pairs of distributions.

The following sections describe our initial efforts to address these issues. The goal of our study is to investigate whether shape distributions can be used to produce a dissimilarity measure with the desirable properties listed above, while providing enough discrimination between similar and dissimilar shapes to be useful for a particular application. We consider this work to be a preliminary investigation, and thus we consider only simple and general-purpose methods to address each issue (in Section 4). We evaluate them experimentally for a database of 3D polygonal models downloaded from the World Wide Web (in Section 5), and we describe a prototype search engine for retrieval of 3D models based on similarity to interactively sketched queries (in Section 6).

4 Method

In this section, we provide a detailed description of the methods we use to build shape distributions from 3D polygonal models and compute a measure of their dissimilarities.

4.1 Selecting a Shape Function

The first and most interesting issue is to select a function whose distribution provides a good signature for the shape of a 3D polygonal model. Ideally, the distribution should be invariant under similarity transformations and tessellations, and it should be insensitive to noise, cracks, tessellation, and insertion/removal of small polygons.

In general, any function could be sampled to form a shape distribution, including ones that incorporate domain-specific knowledge, visibility information (e.g., the distance between random but mutually visible points), and/or surface attributes (e.g., color, texture coordinates, normals and curvature). However, for the sake of clarity, we focus on a small set of shape functions based on geometric measurements (e.g., angles, distances, areas, and volumes). Specifically, in our initial investigation, we have experimented with the following shape functions (see Figure 2):

- ***A3:*** Measures the angle between three random points on the surface of a 3D model.
- ***D1:*** Measures the distance between a fixed point and one random point on the surface. We use the centroid of the boundary of the model as the fixed point.
- ***D2:*** Measures the distance between two random points on the surface.
- ***D3:*** Measures the square root of the area of the triangle between three random points on the surface.
- ***D4:*** Measures the cube root of the volume of the tetrahedron between four random points on the surface.

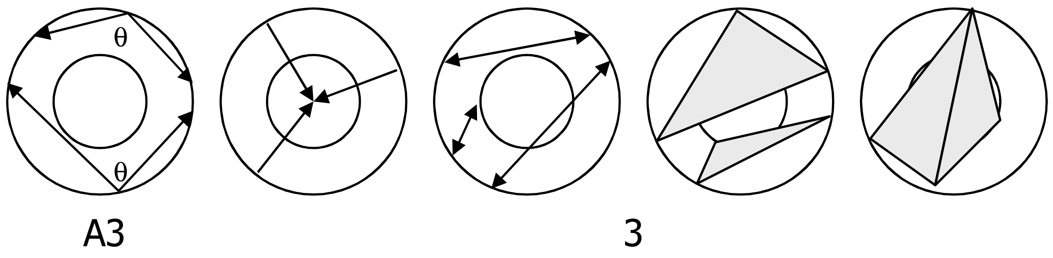


Figure 2: Five simple shape functions based on angles (A3), lengths (D1 and D2), areas (D3), and volumes (D4).

These five shape functions were chosen mostly for their simplicity and invariances. In particular, they are quick to compute, easy to understand, and produce distributions that are invariant to rigid motions (translations and rotations). They are invariant to tessellation of the 3D polygonal model, since points are selected randomly from the surface. They are insensitive to small perturbations due to noise, cracks, and insertion/removal of polygons, since sampling is area weighted. In addition, the *A3* shape function is invariant to scale, while the others have to be normalized to enable comparisons. Finally, the *D2, D3,* and *D4* shape functions provide a nice comparison of 1D, 2D, and 3D geometric measurements.

In spite of their simplicity, we find these general-purpose shape functions to be fairly distinguishing as signatures for 3D shape, as significant changes to the rigid structures in the 3D model affect the geometric relationships between points on their surfaces. For instance, consider the *D2* shape function, whose distributions are shown for a few canonical shapes in Figure 3(a-f). *Note how each distribution is distinctive.* Note also how continuous changes to the 3D model affect the *D2* distributions. For instance, Figure 3(g) shows the distance distributions for ellipsoids of different semi-axes lengths (a, b, c) overlaid on the same plot. The left-most curve represents the *D2* distribution for a line segment – i.e., ellipsoid (0, 0, 1); the right-most curve represents the *D2* distribution for a sphere – i.e., ellipsoid (1, 1, 1); and, the remaining curves shows the *D2* distribution for ellipsoids in between – i.e., ellipsoid (r, r, 1) with $0 < r < 1$. Note how the change from sphere to line segment is continuous. Similarly, Figure 3(h-i) shows the *D2* distributions of two unit spheres as they move 0, 1, 2, 3, and 4 units apart. In each distribution, the first hump resembles the linear distribution of a sphere, while the second hump is the cross-term of distances between the two spheres. As the spheres move farther apart, the *D2* distribution changes continuously.

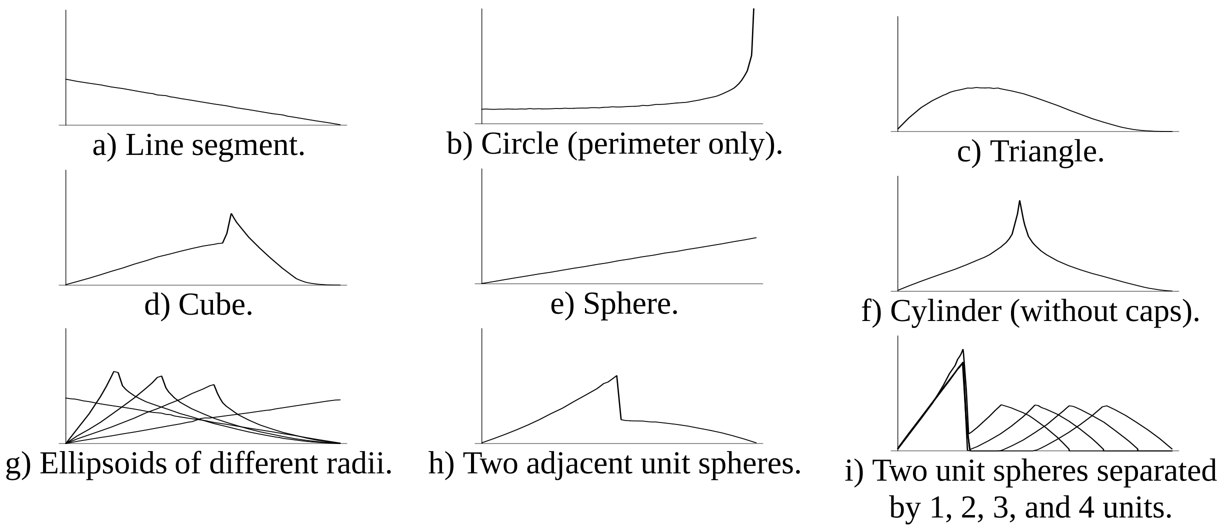


Figure 3: Example *D2* shape distributions. In each plot, the horizontal axis represents distance, and the vertical axis represents the probability of that distance between two points on the surface.

4.2 Constructing Shape Distributions

Having chosen a shape function, the next issue is to compute and store a representation of its distribution. Analytic calculation of the distribution is feasible only for certain combinations of shape functions and models (e.g., the *D2* function for a sphere or line). So, in general, we employ stochastic methods. Specifically, we evaluate N samples from the shape distribution and construct a histogram by counting how many samples fall into each of B fixed sized bins. From the histogram, we reconstruct a piecewise linear function with V ($\leq B$) equally spaced vertices, which forms our representation for the shape distribution. We compute the shape distribution once for each model and store it as a sequence of V integers.

One issue we must be concerned with is sampling density. The more samples we take, the more accurately and precisely we can reconstruct the shape distribution. On the other hand, the time to sample a shape distribution is linearly proportional to the number of samples, so there is an accuracy/time tradeoff in the choice of N. Similarly, more vertices yields higher resolution distributions, while increasing the storage and comparison costs of the shape signature. In our experiments, we have chosen to err on the side of robustness, taking a large number of samples for each histogram bin. Empirically, we have found that $N = 1024^2$ samples, $B = 1024$ bins, and $V = 64$ vertices yields shape distributions with low enough variance and high enough resolution to be useful for our initial experiments. Adaptive sampling methods could be used in future work to make robust construction of shape distributions more efficient.

A second issue is sample generation. Although it would be simplest to sample vertices of the 3D model directly, the resulting shape distributions would be biased and sensitive to changes in tessellation. Instead, our shape functions are sampled from random points on the surface of a 3D model. Our method for generating unbiased random points with respect to the surface area of a polygonal model proceeds as follows. First, we iterate through all polygons, splitting them into triangles as necessary. Then, for each triangle, we compute its area and store it in an array along with the cumulative area of triangles visited so far. Next, we select a triangle with probability proportional to its area by generating a random number between 0 and the total cumulative area and performing a binary search on the array of cumulative areas. For each selected triangle with vertices (A, B, C), we construct a point on its surface by generating two random numbers, r_1 and r_2, between 0 and 1, and evaluating the following equation:

$$P = (1 - \sqrt{r_1})A + \sqrt{r_1}(1 - r_2)B + \sqrt{r_1}r_2C \tag{1}$$

Intuitively, $\sqrt{r_1}$ sets the percentage from vertex A to the opposing edge, while r_2 represents the percentage along this edge (see Figure 4). Taking the square-root of r_1 gives a uniform random point with respect to surface area.

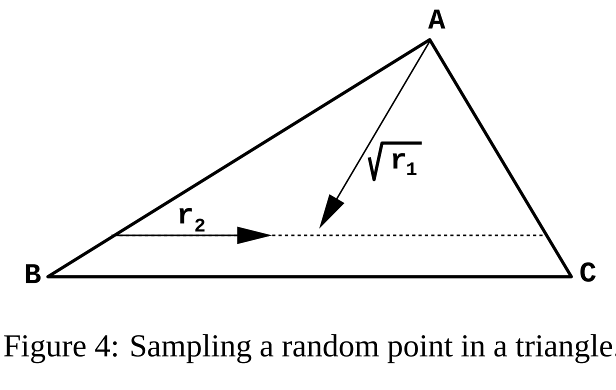


Figure 4: Sampling a random point in a triangle.

4.3 Comparing Shape Distributions

Having constructed the shape distributions for two 3D models, we are left with the task of comparing them to produce a dissimilarity measure. There are many standard ways of comparing two functions f and g representing probability distributions [53]. Examples include the Minkowski L_N norms, Kolmogorov-Smirnov distance, Kullback-Leibler divergence distances [42], Match distances [58, 71], Earth Mover's distance [56], and Bhattacharyya distance [17]. Other methods, perhaps based on 2D curve matching [2, 70], could also be used.

In our implementation, we have experimented with eight simple dissimilarity measures. The first two are the χ^2 statistic and the Bhattacharyya distance, which are commonly used for comparing binned data. The remaining six are Minkowski L_N norms of the probability density functions (pdfs) and cumulative distribution functions (cdfs) for $N = 1, 2, \infty$ (see Figure 5). [1] The dissimilarity measures are computed as follows, assuming f and g represent pdfs for two models, while $\hat{f}$ and $\hat{g}$ represent the corresponding cdfs – i.e., $\hat{f}(x) = \int_{-\infty}^{x} f$:

- **χ^2:** $D(f, g) = \int \frac{(f-g)^2}{f+g}$.
- **Bhattacharyya:** $D(f, g) = 1 - \int \sqrt{fg}$.
- **PDF L_N:** Minkowski L_N norm of the pdf: $D(f, g) = (\int |f - g|^N)^{1/N}$.
- **CDF L_N:** Minkowski L_N norm of the cdf: $D(f, g) = (\int |\hat{f} - \hat{g}|^N)^{1/N}$.

Since each shape distribution is represented as a piecewise linear function, analytic computation of these norms can be done efficiently in time proportional to the number of vertices used to store the distributions.

For certain shape functions, we must add a normalization step to the comparison process to account for differences in scale. So far, we have investigated three methods for normalization (see Figure 6): 1) align the maximum sample values, 2) align the mean sample values (or similarly the

[1]The cdf L_1 and the cdf L_∞ norms are the L_1 Earth Mover's [56] and the Kolmogorov-Smirnov distances, respectively.

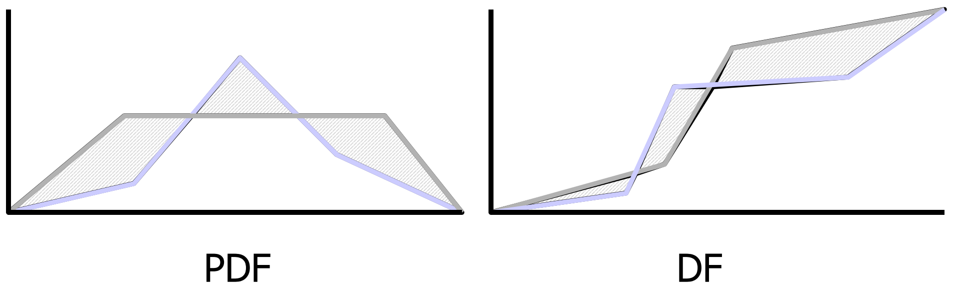


Figure 5: Comparison of shape distributions can be performed by computing the L_N norm of the PDF (left) or CDF (right) representations. In this example, areas of the gray regions in each plot represent the corresponding L_1 norm.

median), and 3) search for the scale that produces the minimal dissimilarity measure during each comparison. The first two of these can be evaluated analytically, and thus are very fast and they can be applied on a per model basis prior to comparison. However, they may not produce the minimal dissimilarity measures due to mismatching scales. The third method performs an optimization procedure during each pair-wise distribution comparison. Specifically, if f and g represent shape distributions for two models, the minimal dissimilarity measure with normalization is defined as:

$$\min_s D(f(x), sg(sx)) \tag{2}$$

We have implemented a simple method to perform the search over s . First, we scale our distributions so that the mean sample in each distribution has value 1. Then we evaluate $D(f(x), sg(sx))$ for values of $\log s$ from -10 to 10, in 100 equally spaced intervals. We return the minimum among the results as the dissimilarity measure for the normalized shape distributions.

5 Experimental Results

The methods described in the preceding sections have been implemented in C++ and incorporated into a shape matching system that runs on Silicon Graphics and PC/Windows computers. We have integrated them into a shape-based classifier (Section 5.3) and an interactive search engine (Section 6).

In order to test the effectiveness of the proposed methods, we executed a series of shape matching experiments with a database of 3D models downloaded from a variety of sites on the World

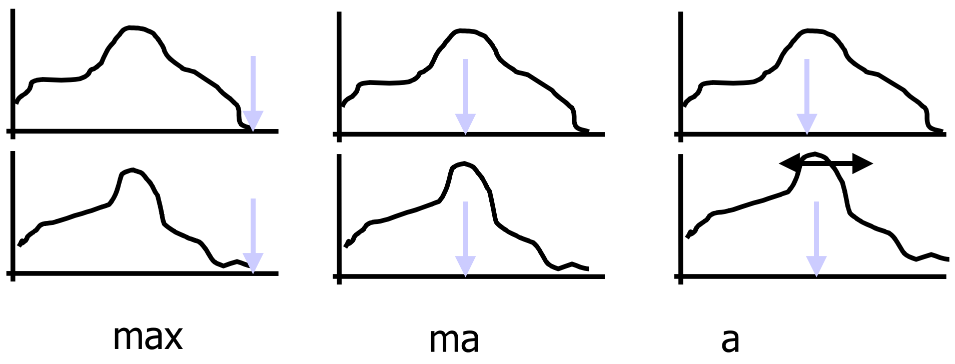


Figure 6: Shape distributions can be normalized for scale using the maximum sample value (left), the mean (middle), or by searching for the best scale factor during comparison (right).

Wide Web. The models comprised sets of independent polygons, without structure, adjacency information, or registered coordinate systems. The models contained anywhere from 20 to 186,000 polygons, with the average model containing around 7,000 polygons. Very few of the models formed a single manifold surface or even a well-defined solid region. Instead, they almost all contained cracks, self-intersections, missing polygons, one-sided surfaces, and/or double surfaces – none of which caused significant artifacts during rendering with a z-buffer, but all of which are problematic for most 3D shape matching algorithms. The experiments were run on a PC with a 400MHz Pentium II processor and 256MB of memory. In our initial experiments, we used the *D2* shape function, the *MEAN* normalization method, and the PDF L_1 norm for computing our dissimilarity measure.

5.1 Robustness Results

In our first experiment, we tested the robustness of our dissimilarity measure to transformations and perturbations of the 3D models. Specifically, we chose ten representative 3D models (shown in Figure 7), and applied eight transformations to each of them. The resulting database had nine versions of each model (the original and eight transformed variants), making 90 models in all. The transformations were as follows:

- **Scale:** Grow by a factor of 10 in every dimension.
- **Anisotropic scale:** Grow by 5% in the Y dimension and 10% in the Z dimension.

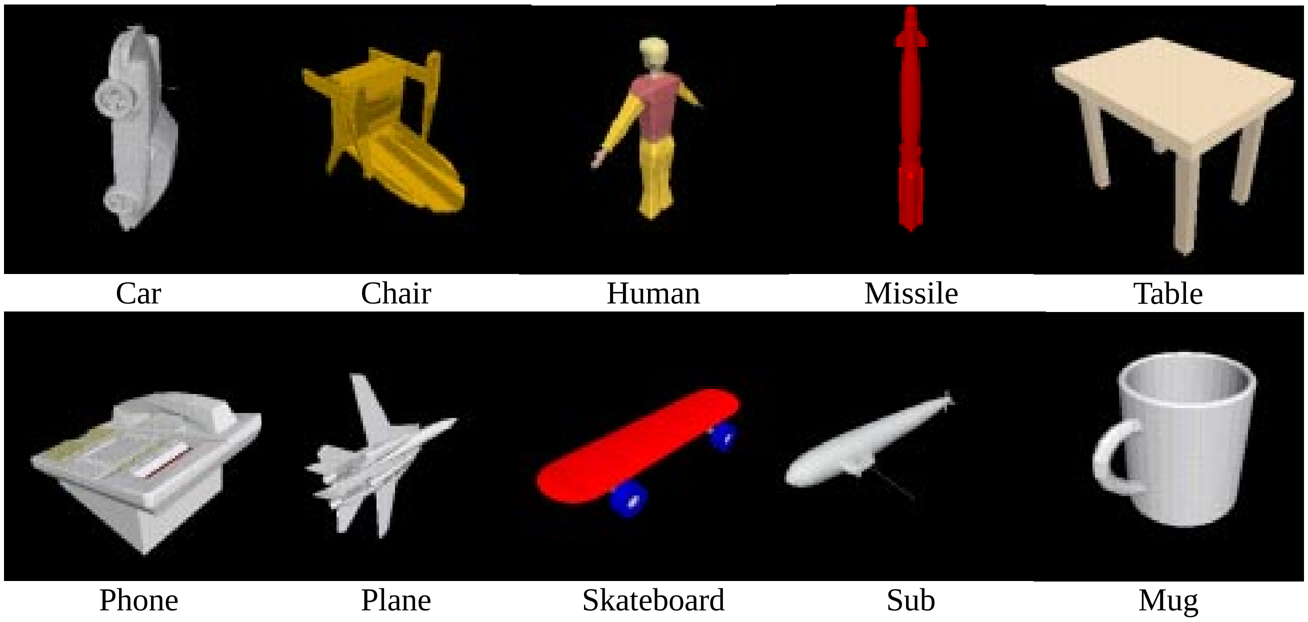
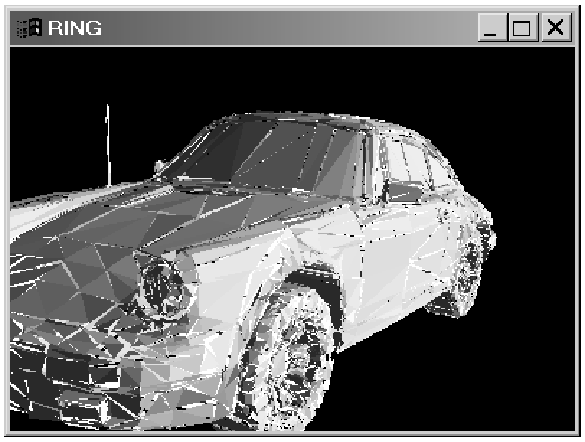


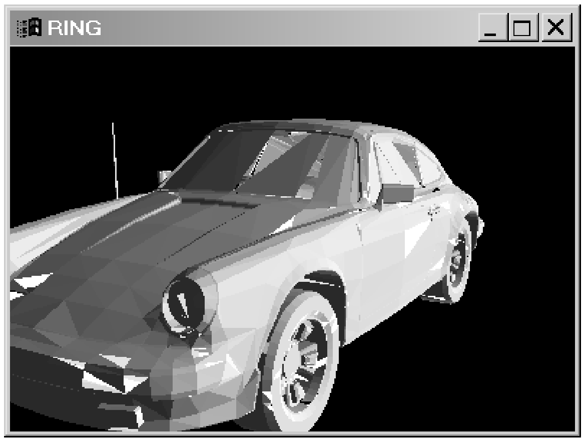
Figure 7: Images of the ten 3D models used in our initial robustness experiments.

- **Rotate:** Rotate by 45 degrees around the X axis, then the Y axis, then the Z axis.
- **Mirror:** Mirror over the YZ plane, then over the XZ plane, then over the XY plane.
- **Shear:** Grow the Y and Z dimensions by 5% and 10% of the X dimension, respectively.
- **Noise:** Perturb each vertex randomly by 1% of the longest length of the model's bounding box. As vertices were not shared by adjacent polygons, this transformation introduced thin cracks (see Figure 8(a)).
- **Delete:** Randomly remove 5% of the polygons (note the holes in the bumper and windshield in Figure 8(b)).
- **Insert:** Randomly insert copies of 5% of the polygons.

We tested the robustness of our sampling method by generating the *D2* shape distribution for each model. The resulting shape distributions for all 90 models are plotted in Figure 9 (scaled to align their mean values). Note that only ten distinct curves are apparent in the plot. This is because each "thick" curve appears as the result of seven nearly overlapping shape distributions computed for different variants of the same model. For instance, the curve containing the tall spike in the middle is drawn seven times, once for each variant of the mug model. The results



(a) 1% Noise.



(b) 5% Deletion.

Figure 8: Car model after (a) perturbing vertices by 1%, or (b) deleting 5% of polygons.

of this experiment demonstrate our sampling method's repeatability and confirm its robustness to similarity transformations, noise, and small cracks and holes.

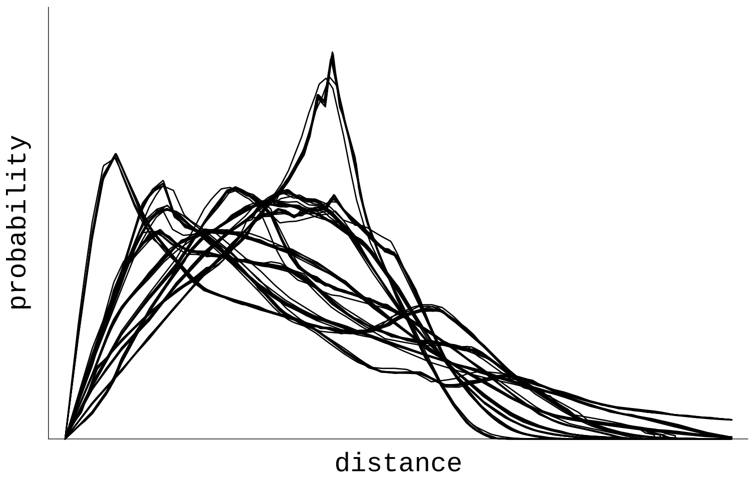


Figure 9: *D2* shape distributions for seven variants of ten models.

We further investigated the robustness of our method by testing it with different polygon tessellations of two 3D shapes. We used the Simplification Envelopes software provided by Cohen et al. [23] to produce 8 versions of the Stanford Bunny [43] ranging from 70,000 down to 600 triangles, and 6 versions of a sphere ranging from 200 down to 28 triangles. Then, we constructed *D2* shape distributions for each of these versions. The resulting 14 curves are shown in Figure 10. Note that the shape distributions vary slightly from original models to the simplified versions, but not significantly when compared to differences between the original models. This experiment corroborates our expectation that shape distributions are insensitive to changing tessellation and, more specifically, stable under model simplification.

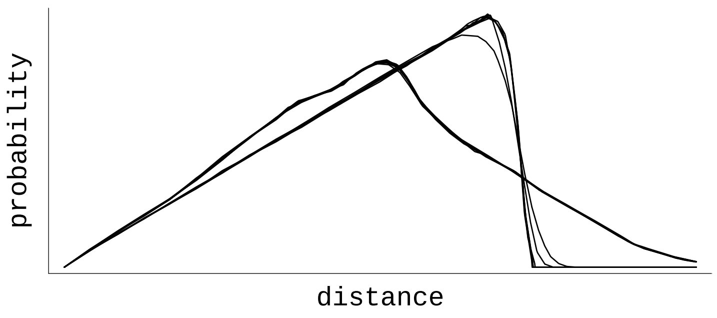


Figure 10: *D2* shape distributions for tessellations of two models.

As a final step in this experiment, we investigated the ability of our shape matching method to discriminate among classes of shapes in this simple database. Specifically, we computed our dissimilarity measure for all pairs of the 90 shape distributions, using the the mean values for normalization and the pdf L_1 norm for comparison. The resulting dissimilarity measures are shown as a matrix in Figure 11. In this visualization of the matrix, the lightness of each element (i, j) is proportional to the magnitude of the computed dissimilarity between models i and j. That is, each row and column represent the dissimilarity measures for a single model when compared to all other models in the database (the matrix is symmetric). Darker elements represent better matches, while lighter elements indicate worse matches. The ordering of the classes is alphabetical so one should not expect any particular darkness pattern except the clearly visible 9x9 blocks of matrix elements with indistinguishable colors. This pattern demonstrates the robustness of our distribution comparison method, as all variants of the same model produce almost the same dissimilarity measure when compared to all variants of every other model. Moreover, note the darker blocks of 9x9 matrix elements along the main diagonal. This pattern results from the fact that all 9 variants of every shape match each other better than they match any other shape. Accordingly, for this simple database, our method could be used to assign all 90 models perfectly to one of the 10 classes using a nearest neighbor classifier.

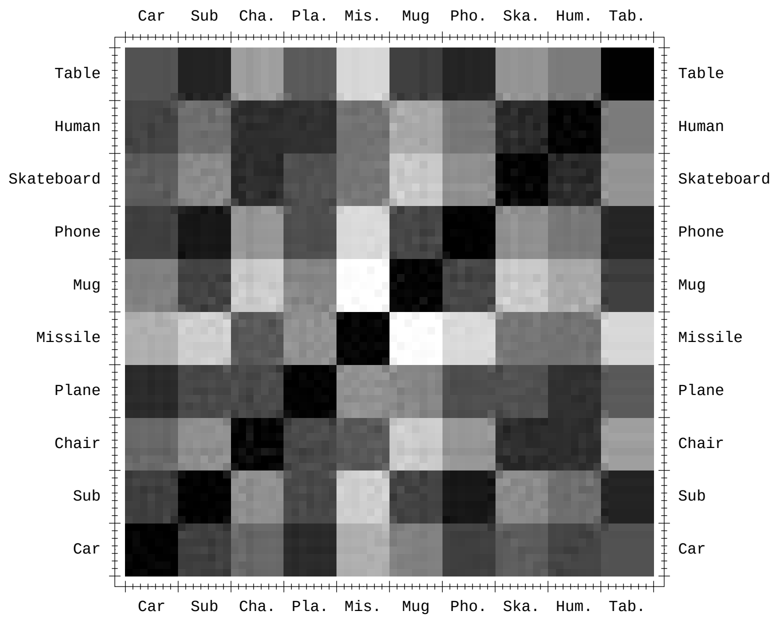


Figure 11: Similarity matrix for nine variants of ten 3D models. Lightness indicates the dissimilarity between models.

5.2 Similarity Results

These initial results are encouraging, but a more interesting and challenging test is to determine how well our method can match classes of shapes in a larger and more diverse database. To investigate this question, we executed a series of tests on a database of 133 models retrieved from the World Wide Web and grouped qualitatively (by function more than by shape) into 25 classes by a third party. Figure 13 summarizes the types and sizes of these classes.

There are several noteworthy characteristics of this test database. First, note that each class contains an arbitrary number of objects, usually determined by how many models were found in a quick search of the Web (e.g., the plane class has significantly more models than the others). Second, note that the *similarity between classes* varied greatly. For instance, some classes were very similar to one another (e.g., pens and missiles look alike), while some were quite distinct (belts). Third, note that the *similarity of objects within each class* also varied. Some classes (such as ball, mug, openbook, pen, and sub) contained 3D models with shapes greatly resembling each other, while others (such as animal, boat, car, and plane) contained models with a wide variety

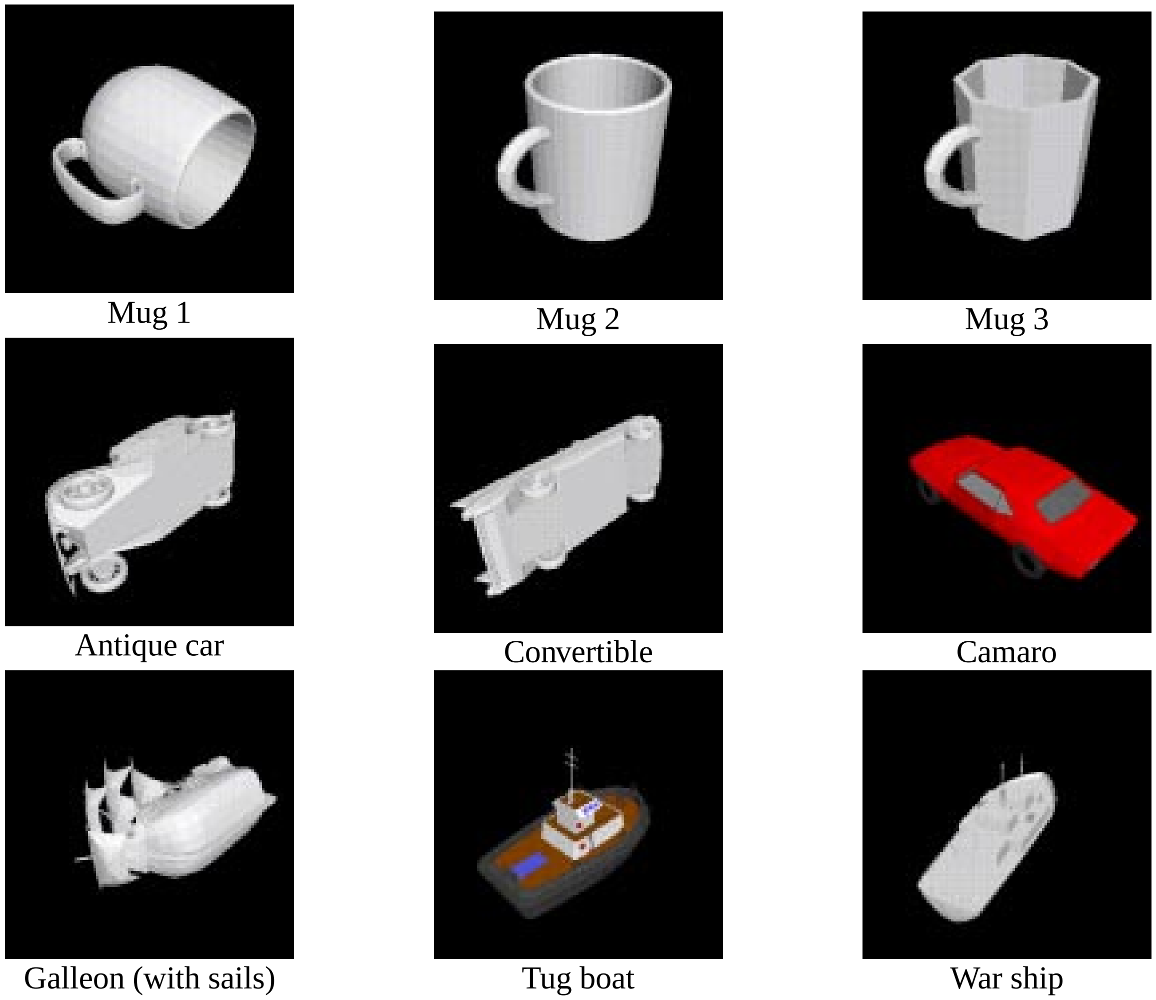


Figure 12: Example classes of shapes in our database: mugs (top row), cars (middle row), and boats (bottom row).

of shapes. This diversity within classes is shown in Figure 12, which contains pictures of three models from the mug, car, and boat classes. Note that the mugs are visually quite similar, while the cars and boats are significantly different. Other classes are even more diverse. For instance, the planes class contains biplanes, fighter jets, propeller planes, and commercial jets, all similar in function, but quite different in shape.

To investigate the ability of our shape matching methods to match classes of objects, we ran an experiment by computing the *D2* shape distributions for all 133 models. They are shown in Figure 13 – each plot shows the *D2* distribution for all 3D models of one class, normalized by their means. Examining these distributions qualitatively, we find that the shape distributions for most

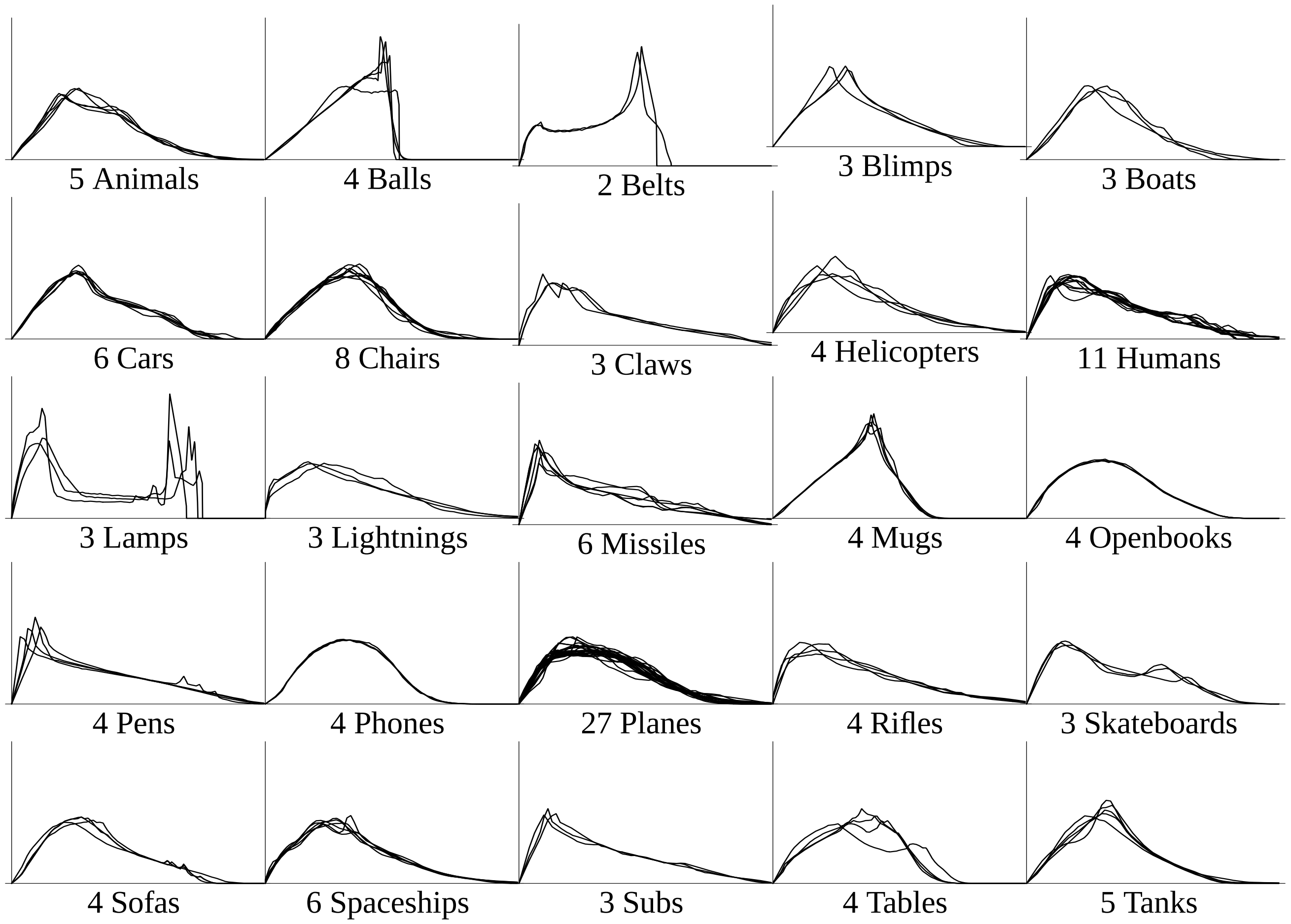


Figure 13: D2 shape distributions for 133 models grouped into 25 classes. Each plot represents a probability distribution of distance.

objects within a single class are highly correlated, as multiple curves appear almost on top of one another. Moreover, many of the classes have a distinctive shape distribution that could be used for classification. For instance, balls have a nearly linear distribution with a sharp falloff on the right, mugs have one sharp peak in the middle, belts have a peak on the right, and lamps have two large peaks with a valley in between.

To a limited degree, it is possible to infer the gross shape of some objects from their *D2* shape distributions. For instance, referring to Figure 14, balls have a distribution resembling a sphere, belts resemble a circle, mugs resemble a cylinder, and lamps resemble two spheres separated by some distance.

Although several classes (e.g., cars) have similar-looking unimodal shape distributions, the

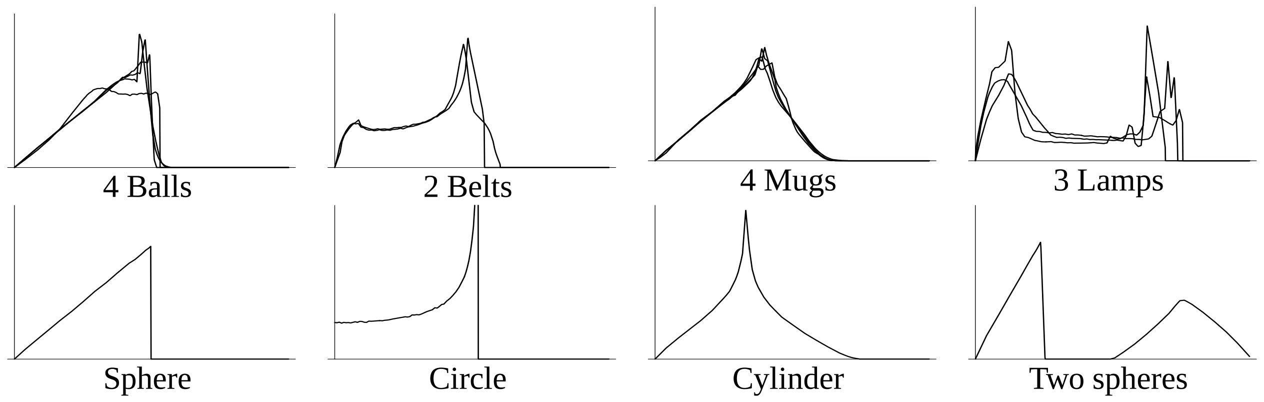


Figure 14: *D2* shape distributions indicate the global shape of an object.

"humps" in the distributions of different classes are usually distinguishable by their locations, heights, and shapes. For instance, consider the Figure 15, which shows the shape distributions for 5 tanks (gray) and 6 cars (black). Although the gross characteristics of these two classes of shape distributions are similar (i.e., a single hump), it seems that a computer program should be able to discriminate between them, generally producing a dissimilarity measure that is lower for pairs of models within the same class.

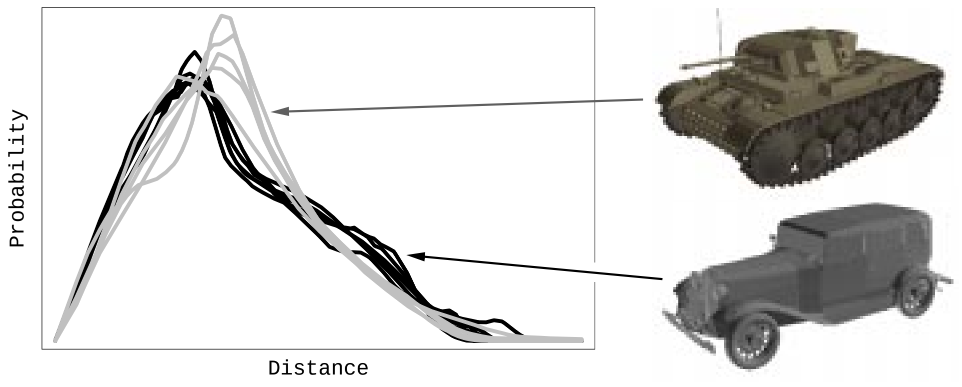


Figure 15: Shape distributions of 5 tanks (gray) and 6 cars (black).

To test this hypothesis, we used the *D2* shape function, *MEAN* normalization method, and L_1 norm to compute the dissimilarity for all pairs of 133 models in our test database. Figure 16 shows the similarity matrix for this test. As in Figure 11, the lightness of each element (i, j) is proportional to the magnitude of the computed dissimilarity between models i and j (i.e., darker elements represent better matches). Thus, if the similarity metric were ideal (i.e., if it were able to read the mind of the human that formed the classes), the dissimilarity measures for models in the same class would be less (appear darker) than for ones in different classes. That is, we hope to see a sequence of darker blocks along the main diagonal, with sizes corresponding to the numbers of models in each class, with mostly lighter colors in the off-diagonal matrix elements. Given the ambiguity and diversity of the database, we believe that it would be optimistic to expect this result.

Examining the matrix in Figure 16, we see that the dissimilarity values computed with our method are fairly discriminating in this test. There are many dark blocks readily apparent along the main diagonal corresponding to groups of objects within the same class that produce good matches (e.g., mugs, phones, chairs, planes, spaceships, etc.). Meanwhile, most elements off-diagonal are lighter shades, indicating relatively few false positives. Off-diagonal blocks of dark elements often represent matches between classes (e.g., spaceship versus plane). Surprisingly, several of the classes with very diverse models (e.g., cars and planes) can be distinguished very clearly as dark blocks in this plot, indicating that our method is useful for discriminating them from other models in the database in spite of their diversity. On the other hand, there are other classes whose models did not match well in this test (e.g., boats, helicopters, etc.). Some of these failures are due to the inherent difficulties of shape matching without real world knowledge (e.g., the boats were more similar in function than shape), while others are probably due to the limitations of our implementation (e.g., L_N norms produce large dissimilarities for lamps, even though their *D2* distributions are very distinctive).

5.3 Classification Results

To test how well our shape matching method works for object classification and to determine which combinations of shape functions, normalization methods, and comparison norms yield the best results, we ran a series of “leave one out” classification tests. In every test, we compared the shape distribution of each model in the database (the query model) against all others and classified each model according to its closest matches. The test was repeated 90 times for all combinations of the 5 shape functions, 3 normalization methods, and 6 comparison norms described in Section 4.

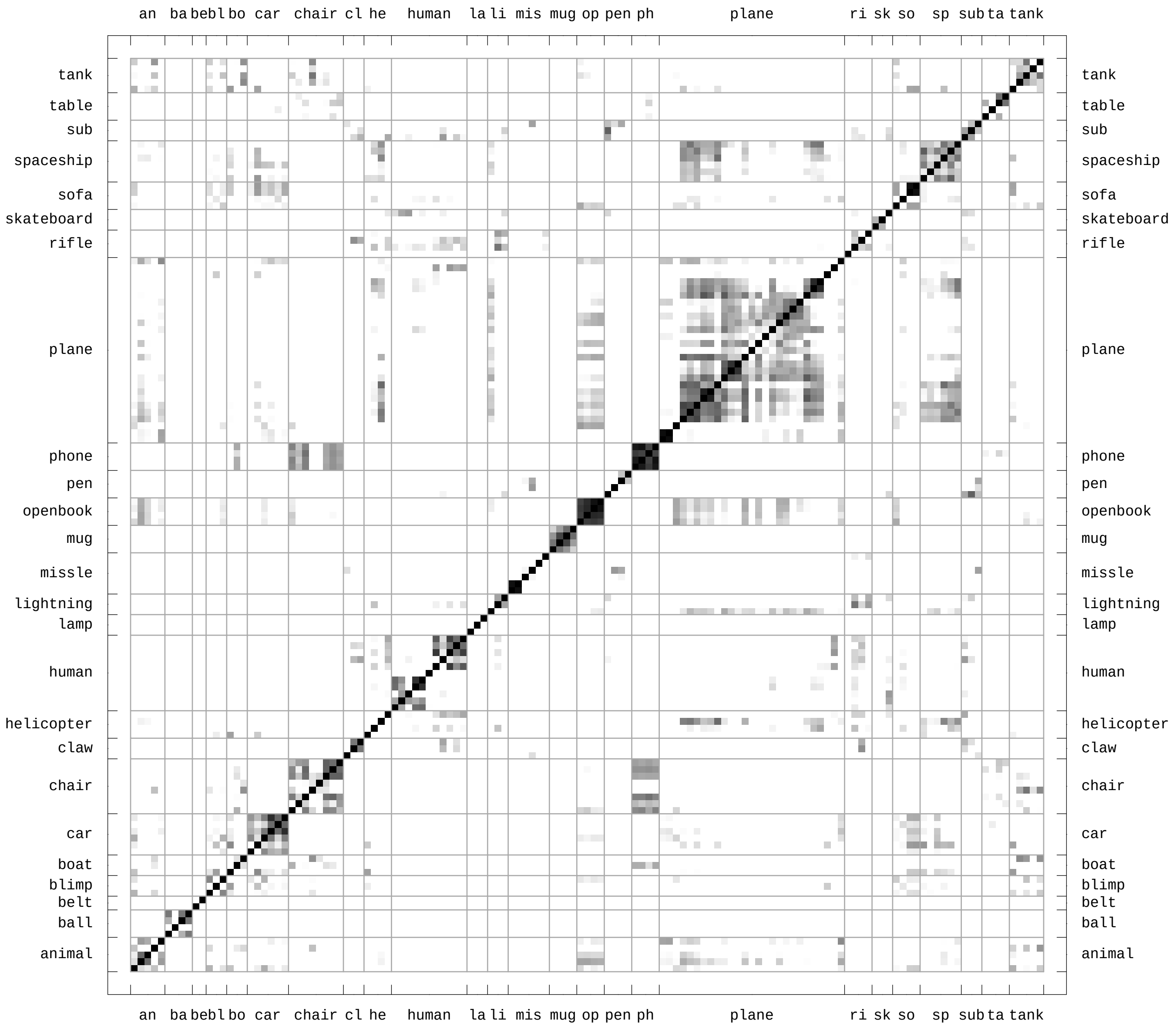


Figure 16: Similarity matrix for our database of 133 3D models.

Tables 1-3 contain three cross-sections of the results measured in these tests. In each table, the first column indicates the shape function, normalization method, or comparison norm used (unless otherwise specified, the *D2* shape function, the *MEAN* normalization method, and the PDF L_1 norm was used). The "First Tier" column lists the percentage of top $k-1$ matches (excluding the query) from the query's class, where k is the size of the class. This criteria is stringent, since each model in the class has only one chance to be in the first tier. An ideal matching would give no false positives and return a score of 100%. The "Second Tier" column lists the same type of result, but for the top $2(k-1)$ matches. The "Nearest Neighbor" column lists the percentage of test in which the top match was from the query's class. Finally, the right-most column contains the computation times for sampling and comparison of shape distributions. Note that sampling times are in seconds, while comparison times are in milliseconds.

From the results in these tables, we make the following observations. First, the *D2* shape function classified objects better than the other four shape functions. There are several plausible explanations for why the others did not perform as well (see Figure 17). First, the *D1* distributions tend to be sensitive to bumps on the surface of an object (note the spikes in the *D1* distributions for missiles in Figure 17 caused by fins protuding from the missiles' bodies). Second, *D1* distributions seem to be sensitive to shifts in an object's center of mass (note the offsets in peaks of the *D1* distributions for mugs in Figure 17 caused by different sized handles). Third, the *A3*, *D3* and *D4* distributions produce distributions that are similar in shape for diverse object classes, probably due to averaging (note the smoothness of the *D3* and *D4* distributions in Figure 17). Although the intra-class diversity for *D3* and *D4* is smaller than for the other shape functions, the inter-class diversity is also smaller, which results in a less discriminating classifier. Overall, in our tests, the *D2* shape function produces distributions with the best combination of distinctiveness and stability, which leads to the best object classifier. Interestingly, we found that combining all five shape functions into a voting-based multiclassifier system [26] provides only 2-3% improvement in classification rates.

Second, in our tests, the *MAX* scaling method is not as good as *MEAN* or *SEARCH* as it suf-

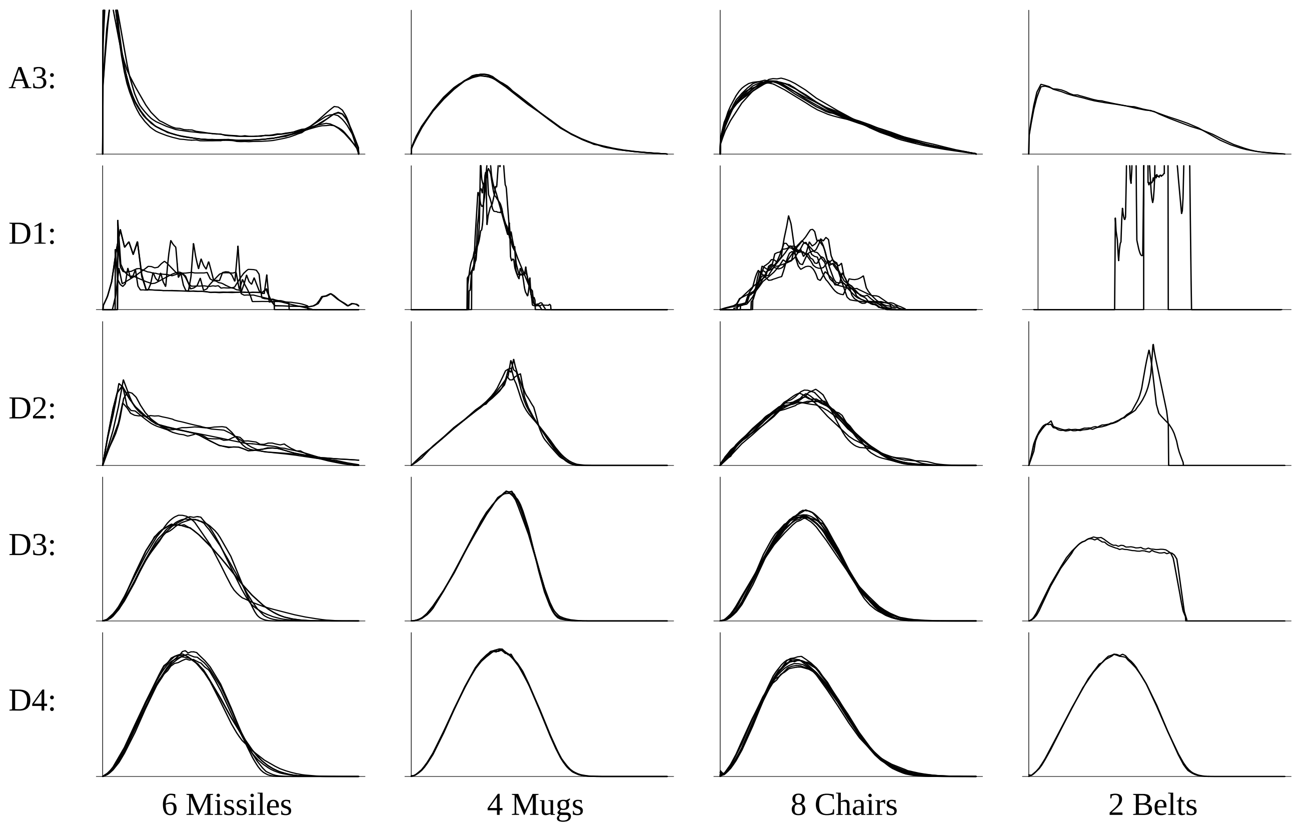


Figure 17: A3, D1, D2, D3, and D4 shape distributions for several object classes. In each plot, the vertical axis represents probability.

Shape Function	First Tier	Second Tier	Nearest Neighbor	Sample Time (s)
A3	38%	54%	55%	12.6
D1	35%	48%	56%	8.6
D2	49%	66%	66%	8.6
D3	42%	58%	58%	13.5
D4	32%	42%	47%	15.8

Table 1: Evaluation of shape functions (using *MEAN* and PDF L_1).

Scale Method	First Tier	Second Tier	Nearest Neighbor	Compare Time (ms)
MAX	41%	56%	63%	0.1
MEAN	49%	66%	66%	0.1
SEARCH	49%	66%	68%	9.0

Table 2: Evaluation of normalization methods (using *D2* and PDF L_1).

Norm Method	First Tier	Second Tier	Nearest Neighbor	Compare Time (ms)
χ^2	48%	63%	64%	0.1
Bhattacharyya	45%	60%	63%	0.1
PDF L_1	49%	66%	66%	0.1
PDF L_2	47%	64%	62%	0.1
PDF L_∞	42%	59%	61%	0.1
CDF L_1	46%	63%	59%	0.2
CDF L_2	44%	63%	59%	0.1
CDF L_∞	43%	59%	57%	0.1

Table 3: Evaluation of comparison methods (using *D2* and *MEAN*).

fers from instability due to high variation in the maximum sample found. Scaling the entire shape distribution based on the maximum sample affects the signature for the entire object. This result is intuitive, as the mean is a more stable statistic than the maximum. For the other normalization methods (*MEAN* and *SEARCH*) the classification results were approximately the same. We found that searching helps minimize the difference between shape distributions, but this did not significantly improve the discriminability of the method on this database. As a result, we conclude it is better to normalize shape distributions according to the means, which can be done once per model and can be incorporated into an indexed shape-based retrieval system.

Third, the PDF L_1 norm performed the best for comparing shape distributions in our test. While the differences are small, it outperformed even the χ^2 statistic and Bhattacharyya distance. In general, the pdfs did better than the cdfs, possibly because peaks and valleys of pdf curves are easier to discriminate using L_N norms than the steep areas and plateaus of cdf curves. Meanwhile, the L_2 and L_∞ norms performed worse than the L_1 norms, in general. For higher N, the L_N norms become less forgiving of large differences, and thus perhaps our comparisons became more sensitive to outliers or normalization errors.

Overall, using the *D2* shape distribution, mean normalization, and L_1 norm, our system produced a top-match within the same class for 66% of the models. This result is shown visually in Figure 18. Each row i represents a shape matching query with the i^{th} object. Black matrix elements in each row indicate the two top matches (the query object and its nearest neighbor); red elements indicate other objects matching in the first tier; and, blue elements represent other matches in the second tier. Looking at the matrix, we conclude that the top matches are mostly from the same class as the query object and that shape distributions provide a useful signature for shape-based retrieval of 3D models.

5.4 Comparison to Moments

As a final test, we compare our shape distribution method against a classifier that represents 3D models with a sequence of discrete surface moments:

$$m_{pqr} = \int_{boundary} x^p y^q z^r \, dx \, dy \, dz \tag{3}$$

In our implementation of the moments-based classifier, the first two moments are used to register the models in a common coordinate system (as in [27]):

1. **Translation:** translate so first moments vanish.
2. **Rotation:** calculate second moments and assemble the associated covariance matrix. Factor the covariance matrix using Singular Value Decomposition (SVD), and orient the model by applying the unitary matrix of this decomposition.

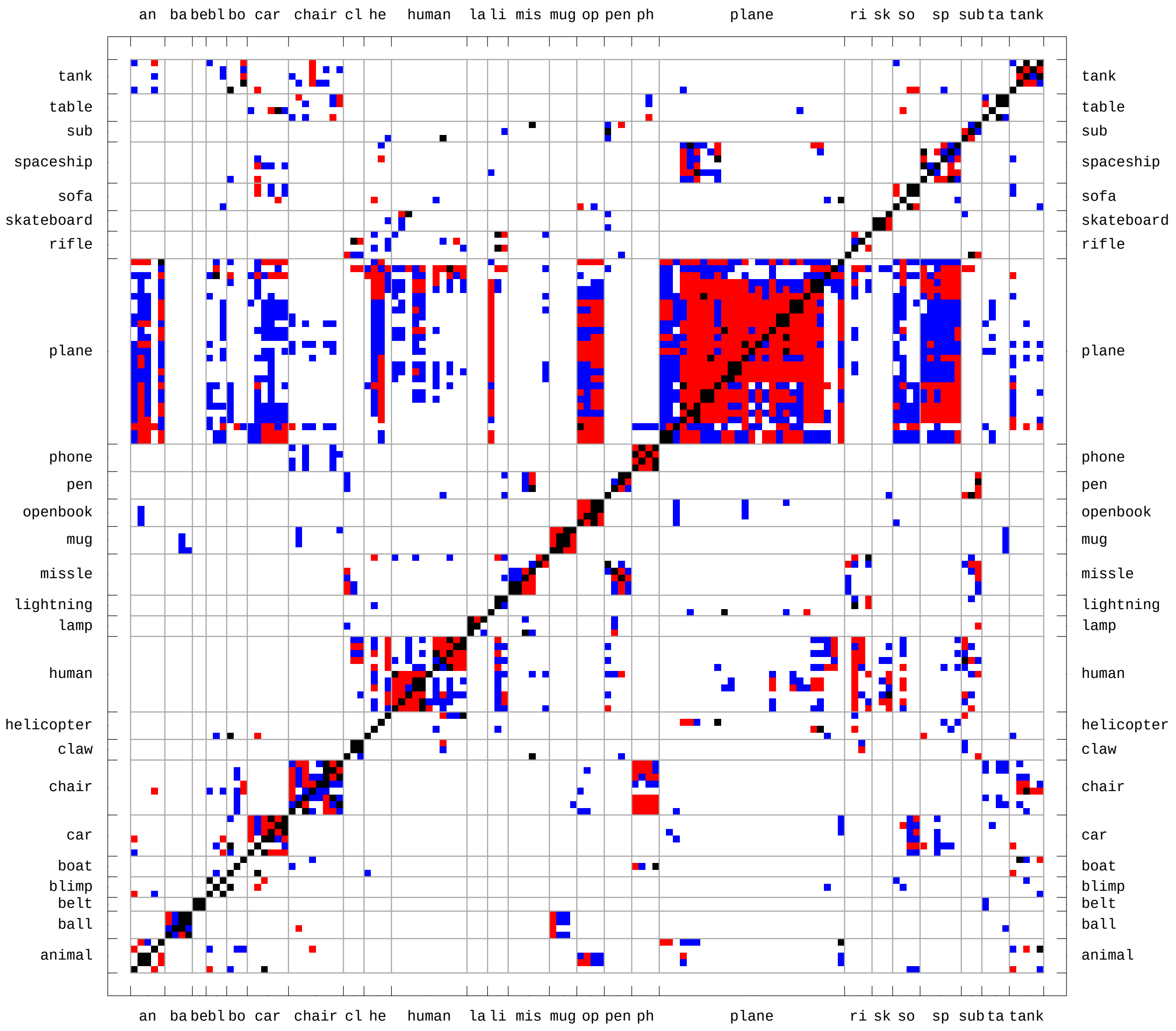


Figure 18: Matrix representing results of classification queries (rows) for each of 133 3D models. Black matrix elements indicate the query object and its nearest neighbor. Red elements indicate other top matches in the first tier. Finally, blue elements represent the remaining matches in the second tier.

3. **Scale:** scale so that the maximum eigenvalue value of the SVD is 1.

After registration, moments up to a user-specified order are calculated and stored as a shape signature. In our notation, *M3* specifies that 3rd order moments and lower were used as shape descriptors (i.e., $p + q + r \leq 3$), and similarly for *M4* through *M7*. The shape signatures are compared using a component-by-component L2 norm.

Table 4 and Figure 19 compare the results achieved with this moment-based classifier versus the method proposed in this paper. Specifically, Table 4 reports "First Tier," "Second Tier," and "Nearest Neighbor" statistics, while Figure 19 shows plots of precision-recall curves averaged over all objects in the test database. We find that *D2* shape distributions outperform moments in these for classification of models in our tests. The differences are more significant for more stringent classification criteria (i.e., First-Tier) and for higher-order moments, which are known to be sensitive to noise [52]. Further studies are required to test whether *D2* shape distributions perform better than moments for larger databases or for other shape matching applications.

Method	First Tier	Second Tier	Nearest Neighbor
D2	49%	66%	66%
M3	35%	46%	63%
M4	41%	52%	64%
M5	28%	38%	55%
M6	34%	44%	54%
M7	27%	33%	51%

Table 4: Comparison of classification results achieved with shape distributions versus moments of different orders.

6 3D Search Engine

As an initial study of applications for shape distributions, we have developed a web-based search engine for 3D polygonal models. The motivation is to provide a tool with which users can retrieve models from a 3D database based on their shape attributes.

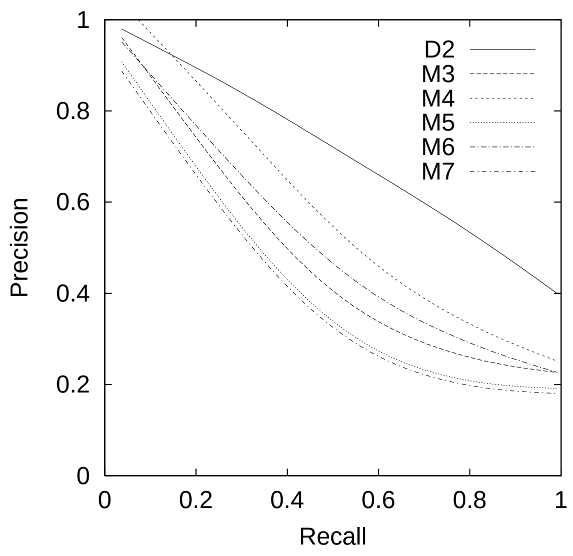


Figure 19: Precison-recall plots comparing retrieval results achieved with shape distributions versus moments of different orders.

In the early prototype system [22], the user draws a 3D *query model* using a Java applet (Teddy [37]) running in any web browser. The resulting set of polygons is sent to a web server, which computes its *D2* shape distribution and a dissimilarity measure for all models in a 3D database using the methods described in this paper. Links to images and 3D data for the K most similar models are returned in a web page for the user to peruse.

Sample results obtained with this 3D search engine are shown in Table 5 (these queries represent typical results – they are not a sampling of our best results). The images in the leftmost column show the query 3D models drawn by the user, while the columns on the right show the closest matches among the 3D models in our test database of 133 models (dissimilarity values are shown below each match). For instance, a query with a crudely drawn chair (top row) returned four chairs and a table in the top five matches; a query with a cartoonish four-legged animal (second row) returned a triceratops, a cow, two tanks, and a blimp; and, so on. In all examples, the 3D query models were drawn in a couple of minutes or less by an undergraduate student, Joyce Chen, and the results were returned within ten seconds or so. In these experiments, almost all of the time was spent building the shape distribution for the query model, while comparing the shape distributions was almost instantaneous. For larger databases, more sophisticated indexing methods would be required for interactive performance.

Even though this simple 3D search engine is rather crude, it seems to provide a useful tool for retrieving models based on their gross shapes. Further investigation is required to determine whether this tool is useful in practice and which query interfaces are most useful for shape-based retrieval of models from large databases.

7 Discussion and Conclusion

The main contribution of this paper is the idea of using random sampling to produce a continuous probability distribution to be used as a signature for 3D shape. The key feature of this approach is that it provides a framework within which arbitrary and possibly degenerate 3D models can be transformed into functions with natural parameterizations, allowing simple function comparison methods to produce robust dissimilarity metrics.

Our initial experiences verify many of the expected features of this approach. First, it is simple to implement – e.g., our whole system requires around 2000 lines of C++ code. Second, it is fast – e.g., the system takes around ten seconds to construct a shape distribution for typical 3D models containing thousands of polygons, and it computes the dissimilarity measure for any pair of shape distributions in less than a millisecond. Third, invariance and robustness properties can be ensured by choosing shape functions and norms with the desired properties – e.g., the *D2* shape function is invariant to rigid body and mirror transformations, and it is insensitive to noise, blur, cracks, tessellation, and dust in the input 3D models. Normalization of shape distributions provides invariance to scale, and using the L_N norm for comparison of distributions ensures that our dissimilarity measure is a metric.

Our experimental results demonstrate that shape distributions can be fairly effective at discriminating between groups of 3D models. Overall, we achieved 66% accuracy in our classification experiments with a diverse database of degenerate 3D models assigned to functional groups. The *D2* shape distribution was more effective than moments during our classification tests. Unfortunately, it is difficult to evaluate the quality of this result as compared to other methods, as it

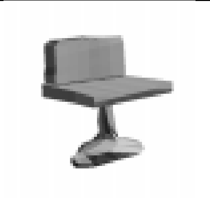
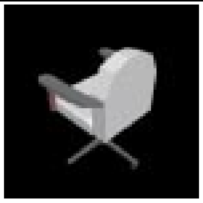
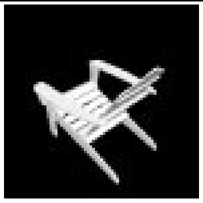
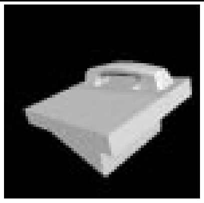
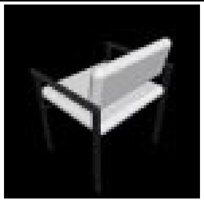
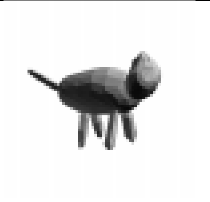
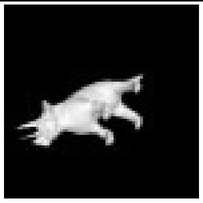
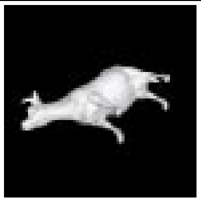
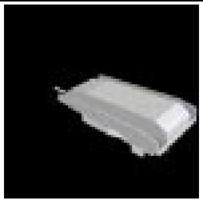
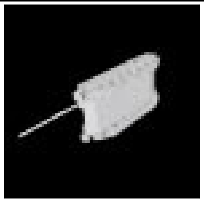
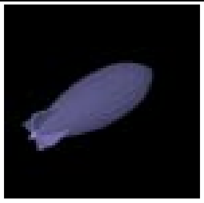
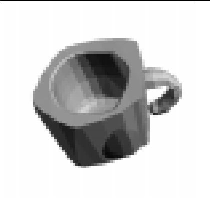
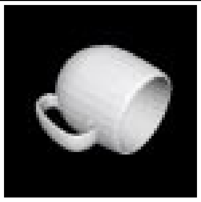
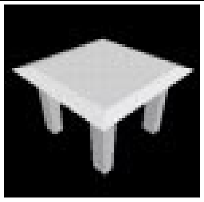
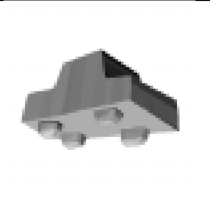
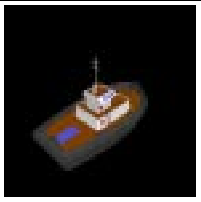
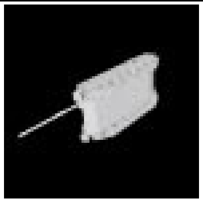
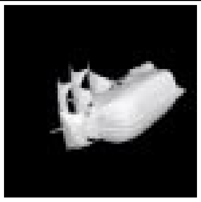
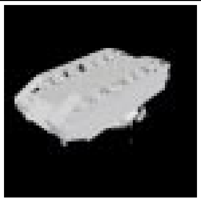
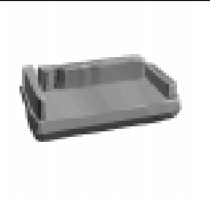
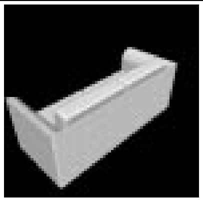
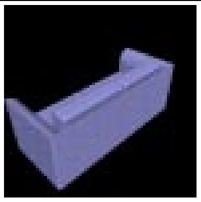
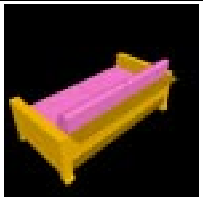
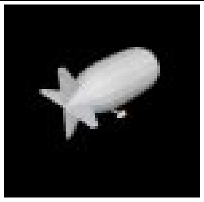
Query	Five Top Matches				
					
	0.060	0.066	0.078	0.079	0.086
					
	0.051	0.052	0.073	0.086	0.101
					
	0.050	0.088	0.096	0.108	0.132
					
	0.073	0.073	0.079	0.087	0.087
					
	0.064	0.065	0.071	0.088	0.090

Table 5: 3D query models (left column) and the five top matches for each query returned by web-based search engine. *D2* dissimilarity measures are shown below each match.

depends largely on the details of our test database. However, we believe that it demonstrates that our method is useful for the discrimination of 3D shapes, at least for preclassification prior to more exact similarity comparisons with more expensive methods.

An important issue for further research in 3D shape matching is development of benchmark databases containing degenerate 3D polygonal models so that different shape analysis methods can be compared. Of course, there are also many improvements that could be made to our initial prototype system in future work. First, one could investigate more sophisticated shape functions, possibly based on domain specific information or local geometric properties suitable for articulated figures. Second, it would be possible to develop more efficient shape distribution sampling and reconstruction methods, possibly based on adaptive strategies. Third, one could investigate using shape distributions with more sophisticated classifiers, possibly based on neural networks. Fourth, combining shape distributions with other attributes (e.g., surface colors, moments, etc.) for improved discriminability is an interesting topic for further investigation. Finally, we are currently investigating a variety of user interfaces for specifying 3D shape-based queries in interactive retrieval systems.

Another important topic for future work is to study the theoretical properties of shape distributions. For instance, it would be nice to develop a theory concerning which shape functions and norms will be good classifiers of shape. We are investigating provable properties for the *D2* shape function. Uniqueness properties for homometric *discrete* point sets (ones with the same distance distribution) have been proven by Skiena et al. [60]. They developed upper and lower bounds on the number of non-congruent homometric discrete point sets in arbitrary dimensions. Properties have also been proven for the Radon transform for a convex region C in the plane [54, 33]. This transform maps any oriented line to the length of its intersection with C. It completely specifies the region C, and it can be inverted fairly efficiently. The Radon transform has found many uses in X-ray tomography. Proving uniqueness properties for other continuous shape functions may have similar implications for reconstruction and manipulation of 3D models represented by shape distributions.

Finally, it would be interesting to investigate whether the proposed shape matching method is useful in other application domains, such as mechanical CAD, medicine, or molecular biology.

Acknowledgements

We would like to thank Patrick Min, David Jacobs, Adam Finkelstein, and Adam's dad for their valuable discussions. Patrick also helped by assembling a bibliography on 3D shape matching, and Mao Chen retrieved and labeled the 3D models from the World Wide Web for our experiments. Joyce Chen and Patrick Min implemented the web-based search engine. Thanks also go to Alfred P. Sloan Foundation and the National Science Foundation for partial funding of this work.

References

[1] F.J. Aherne, N.A. Thacker, and P.I. Rockett. Optimal pairwise geometric histograms. *BMVC*, pages 480–490, 1997.

[2] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation: A survey. Technical Report B 96-11, EVL-1996-142, Institute of Computer Science, Freie Universität Berlin, 1996.

[3] Y. Amit, U. Grenander, and M. Piccioni. Structural image restoration through deformable templates. *J. Am. Statistical Assn.*, 86(440):376–387, 1991.

[4] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. Nearest neighbor classification in 3d protein databases. *ISMB*, 1999.

[5] A.P.Ashbrook, N.A.Thacker, P.I.Rockett, and C.I.Brown. Robust recognition of scaled shapes using pairwise geometric histograms. *BMVC*, pages 503–512, July 1995.

[6] Klaus Arbter, Wesley E. Snyder, Hans Burkhardt, and Gerd Hirzinger. Application of affine-invariant fourier descriptors to recognition of 3-d objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):640–647, July 1990.

[7] Esther M. Arkin, L. Paul Chew, Daniel P. Huttenlocher, Klara Kedem, and Joseph S.B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, March 1991.

[8] F. Arman and J.K. Aggarwal. Model-based object recognition in dense-range images - a review. *ACM Computing Surveys*, 25(1):5–43, 1993.

[9] A.P. Ashbrook, P.I. Rockett, and N.A. Thacker. Multiple shape recognition using pairwise geometric histogram based algortihms. *IEEE Image Processing*, July 1995.

[10] A.P. Ashbrook, N.A. Thacker, and P.I. Rockett. Pairwise geometric histograms: A scaleable solution for recognition of 2d rigid shapes. *9th SCIA*, 1:271–278, 1995.

[11] Eric Bardinet, Sara Fernández Vidal, Sergio Damas Arroyo, Grégoire Malandain, and Nicolas Pérez de la Blanca Capilla. Structural object matching. Technical Report DECSAI-000303, Dept. of Computer Science and AI, University of Granada, Spain, February 2000.

[12] G. Barequet and S. Kumar. Repairing cad models. *IEEE Visualization '97*, pages 363–370, 1997.

[13] Ronen Basri, Luiz Costa, Davi Geiger, and David Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:2365–2385, 1998.

[14] Serge Belongie, Jitendra Malik, and Jan Puzicha. Matching shapes. *ICCV*, 2001.

[15] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *Computing Surveys*, 17(1):75–145, March 1985.

[16] P.J. Besl. Triangles as a primary representation. *Object Recognition in Computer Vision*, LNCS 994:191–206, 1995.

[17] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematics Society*, 35:99–110, 1943.

[18] T.O. Binford. Visual perception by computer. *IEEE Conference on Systems Science and Cybernetics*, 1971.

[19] Jules Bloomenthal and Chek Lim. Skeletal methods of shape manipulation. *Shape Modeling and Applications*, pages 44–47, 1999.

[20] Stefan Carlsson. Order structure, correspondence, and shape based categories. *Shape, Contour and Grouping in Computer Vision*, pages 58–71, 1999.

[21] S.F. Chang and J.R. Smith. Extracting multi-dimensional signal features for content-based visual query. *SPIE Symposium on Visual Communications and Signal Processing*, 2501(2):995–1006, May 1995.

[22] Joyce Chen. *The Search for 3D Models*. Junior Independent Work, Computer Science Department, Princeton University, June 2001.

[23] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Brooks, Jr., and William Wright. Simplification envelopes. *SIGGRAPH*, pages 119–128, August 1996.

[24] H. Delingette, M. Hebert, and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. *Image and vision computing*, 10(3):132–144, April 1992.

[25] H. Delingette, M. Hebert, and K. Ikeuchi. A spherical representation for the recognition of curved objects. *ICCV*, pages 103–112, 1993.

[26] R.O. Duda, P.E. Hart, and David Stork. *Pattern Classification, Second Edition*. John Wiley & Sons, New York, 2001.

[27] Michael Elad, Ayellet Tal, and Sigal Ar. Similarity between three-dimensional objects - an iterative and interactive approach. *submitted for publication*, 2001.

[28] A.C. Evans, N.A. Thacker, and J.E.W. Mayhew. Pairwise representation of shape. *11th ICPR*, 1, 1992. 133-136.

[29] A.C. Evans, N.A. Thacker, and J.E.W. Mayhew. The use of geometric histograms for model-based object recognition. *4th BMVC*, pages 429–438, September 1993.

[30] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, and Q. Huang. Query by image and video content: the qbic system. *IEEE Computer*, 28(9):23–32, 1995.

[31] James Gain and James Scott. Fast polygon mesh querying by example. SIGGRAPH Technical Sketches, 1999.

[32] A. Gueziec, G. Taubin, F. Lazarus, and W. Horn. Converting sets of polygons to manifold surfaces by cutting and stitching. *IEEE Visualization '98*, pages 383–390, 1998.

[33] S. Helgason. The radon transform. *Progress in Mathematics, Springer, 2nd ed.*, 5, 1999.

[34] L. Holm and C. Sander. Touring protein fold space with dali/fssp. *Nucleic Acids Research*, 26:316–319, 1998.

[35] B.K.P. Horn. Extended gaussian images. *Proc. of the IEEE*, 72(12):1671–1686, December 1984.

[36] B. Huet and E.R. Hancock. Structural indexing of infra-red images using statistical histogram comparison. *IWISP*, pages 653–656, November 1996.

[37] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. *SIGGRAPH*, pages 409–416, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[38] Katsushi Ikeuchi, Takeshi Shakunaga, M.D. Wheeler, and Taku Yamazaki. Invariant histograms and deformable template matching for sar target recognition. *Computer Vision and Pattern Recognition*, 1996.

[39] C. Jacobs, A. Finkelstein, and D. Salesin. Fast multiresolution image querying. *SIGGRAPH*, pages 277–286, 1995.

[40] A. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(3):267–278, 1996.

[41] Andrew E. Johnson and Martial Hebert. Using spin-images for efficient multiple model recognition in cluttered 3-d scenes. *IEEE PAMI*, 21(5):433–449, 1999.

[42] S. Kullback. *Information Theory and Statistics*. Dover, 1968.

[43] Stanford University Computer Graphics Laboratory. http://graphics.stanford.edu/data, 1996.

[44] Y. Lamdam and H.J. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. *ICCV*, December 1988.

[45] Y. Lin, J. Dou, and H. Wang. Contour shape description based on an arch height function. *Pattern Recognition*, 25:17–23, 1992.

[46] Sven Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[47] Greg Mori, Serge Belongie, and Jitendra Malik. Shape contexts enable efficient retrieval of similar shapes. *CVPR*, 2001.

[48] T.M. Murali and Thomas Funkhouser. Consistent solid and boundary representations from arbitrary polygonal data. *Computer Graphics (1997 SIGGRAPH Symposium on Interactive 3D Graphics)*, pages 155–162, March 1997.

[49] V. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, 1995.

[50] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(7):715–729, 1991.

[51] Arthur R. Pope. Model-based object recognition: A survey of recent research. Technical Report TR-94-04, University of British Columbia, January 1994.

[52] Richard J. Prokop and Anthony P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphics Models and Image Processsing*, 54(5):438–460, 1992.

[53] Jan Puzicha, Yossi Rubner, Carlo Tomasi, and Joachim M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *IEEE International Conference on Computer Vision*, pages 1165–1173, 1999.

[54] A. G. Rann and A. I. Katsevich. The radon transform and local tomography. *CRC Press*, 1996.

[55] P.A. Riocreux, N.A. Thacker, and R.B. Yates. An analysis of pairwise geometric histograms for view-based object recognition. *BMVC*, September 1994.

[56] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. *6th ICCV*, pages 59–66, January 1998.

[57] S. Sclaroff and A. Pentland. Modal matching for correspondence and recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(6):545–561, June 1995.

[58] H. C. Shen and A. K. C. Wong. Generalized texture representation and metric. *Computer, Vision, Graphics, and Image Processing*, 23:187–206, 1983.

[59] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. *Computer Vision*, pages 222–229, 1998.

[60] S. Skiena, W. Smith, and P. Lemke. Reconstructing sets from interpoint distances. *Proceedings of the Sixth Annual Symposium on Computational Geometry*, pages 332–339, 1990.

[61] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *Pattern Analysis and Machine Intelligence*, 12(2):131–147, February 1990.

[62] Duane Storti, George Turkiyyah, Mark Ganter, Chek Lim, and Derek Stal. Skeleton-based modeling operations on solids. *Solid Modeling*, pages 141–154, 1997.

[63] C. Tappert. Cursive script recognition by elastic matching. *IBM Journal of Res. Develop.*, 26(6):765–771, 1982.

[64] G. Taubin and D.B. Cooper. *Geometric Invariance in Computer Vision*, chapter Object recognition based on moment (of algebraic) invariants. MIT Press, 1992.

[65] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.

[66] N.A. Thacker, P.A. Riocreux, and R.B. Yates. Assessing the completeness properties of pairwise geometric histograms. *Image and Vision Computing*, 13(5):423–429, 1995.

[67] W. Tsai and S. Yu. Attributive string matching with merging for shape recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7:453–462, 1985.

[68] C. Uras and A. Verri. On the recognition of the alphabet of the sign language through size functions. *IAPR*, pages 334–338, 1994.

[69] C. Uras and A. Verri. Computing size functions from edge maps. *International Journal of Computer Vision*, 23(2):169–183, 1997.

[70] Remco C. Veltkamp and Michiel Hagedoorn. State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.

[71] M. Werman, S. Peleg, and A. Rosenfeld. A distance metric for multi-dimensional histograms. *Computer, Vision, Graphics, and Image Processing*, 32:328–336, 1985.

[72] K. Wu and M.D. Levine. Recovering parametrics geons from multiview range data. *CVPR*, pages 159–166, June 1994.

[73] I. Young, J. Walker, and J. Bowie. An analysis technique for biological shape. *Computer Graphics and Image Processing*, 25:357–370, 1974.

[74] Dongmei Zhang and Martial Hebert. Harmonic maps and their applications in surface matching. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, 1999.