A Geometric Approach to 3D Object Comparison

Marcin Novotni and Reinhard Klein

University of Bonn, Computer Graphics Group
Römerstr. 164, Bonn, Germany
{marcin,rk}@cs.uni-bonn.de

Abstract

Along with the development of 3D acquisition devices and methods and increasing number of available 3D objects, new tools are necessary to automatically analyze, search and interpret these models. In this paper we describe a novel geometric approach to 3D object comparison and analysis. To compare two objects geometrically we first properly position and align the objects. After solving this pose estimation problem, we generate specific distance histograms that define a measure of geometric similarity of the inspected objects.

The geometric approach is very well suited for structures with moderate variance, e.g. bones, fruits, etc. The strength of the approach is proven through results of several tests we performed on different data sets.

1 Introduction

Recent advances in the area of computer vision and 3D-acquisition techniques have led to rapidly growing amount of 3D-data. Furthermore, the World Wide Web is enabling access to 3D models acquired and constructed by people all over the world. Currently, several web sites allow free downloads of thousands of publically available 3D models.

These developments are changing the way we deal with the problem of 3D data generation which was also pointed out by T. Funkhouser in a recent course on geometric modelling [7]. Up to now, the field of 3D modelling concentrated on how to model artificial representations of 3D entities from scratch or from a few already available objects. In the future, the question how to search for 3D models, how to automatically analyze them and how to adapt them to our own purposes will be increasingly relevant.

An important issue in this general context is how to search for 3D objects in a similar way as we already search for text, images, audio, and video. Up to now there are only few references to the specific problem of content based retrieval of 3D models, however, an extensive amount of literature can be found on the related fields of computer vision, computational geometry and geometric modelling. The success in elaborating a solution to content based retrieval of 3D models depends heavily on existence of an efficient and intelligent comparison of 3D objects. Therefore, as a starting point we aimed at analysis of and experimentation with *geometric similarity*, as opposed to other existing approaches relying on invariant features [5, 18, 1, 2]. We believe that the human notion of similarity is based on what the objects "look like" which in turn depends in great extent (but not exclusively) on the overall geometry. Motivated by this fact, we developed a novel method of *geometric comparison* of 3D-objects which is described in this paper.

The problem of geometric similarity measurement consists essentially of two subproblems: the pose estimation, where two objects are properly positioned and aligned [4, 12, 5, 15, 14], and the comparison or similarity measurement of objects.

In a similar way to this approach, Saupe [17] in his object retrieval algorithm attempts to align two objects based on PCA (Principal Component Analysis). His object recognition algorithm is based on a very simple feature extraction method. The drawback of his pose estimation is that he uses surface properties instead of volumetric properties to construct the covariance matrix used for orientation calculation. On the other hand, his feature vectors are adequate only for a very rough classification or search of objects. As an effect of these, the applicability and robustness of his approach is very limited.

There are also some initial research activities aiming towards the above goals at Princeton University, US. A nice overview of associated problems and possible start-

ing points may be found at [7].

In our new algorithm, as a first step, we solve the pose estimation problem. Then, instead of a feature extraction step, we calculate a volumetric error between one object and a sequence of offset hulls of the other object and vice versa. This provides us with two histograms that are used for the similarity measurement.

The outline of the paper is as follows. In section 2 we describe our settings and state the problem. Section 3 describes our approach for solving the pose estimation problem. In the following section 4 we describe the notion of distance fields and briefly summarize our distance computation method. After that, we introduce the histogram computation and evaluation in section 5. Results are presented in section 6. Finally, we conclude with a short summary and draft the future directions.

2 Problem statement and solution outline

Our ultimate aim is to be able to give a quantitative measure on how two objects resemble each other. For example, one is easily able to differentiate between the geometric shape of a banana and an apple – we attempt to construct a method that will be able to accomplish this automatically.

Figure 1. In case of similar objects only a small fraction of their volumes are outside of the other objects boundary.

Note the following: if one takes two nearly identical datasets and align them so that the surfaces fit one onto another as accurately as possible, every point inside one of the objects will either be contained by the other one or be in a small distance from the other's boundary and vice versa, cf. Figure 1. In contrast to that, in the case of two different objects, there will be volume portions of each object that are in relatively large distance from the surface outside of the other object.

In our algorithm, after the pose estimation, we determine for each object how big portions of its volume are outside of the other object – as an efficient technique to accomplish this task, we use discrete *3D distance fields*. Based on the information already explicitly contained in the distance fields, for each discrete element of the volumes we look up its distance from the boundary of the other object and construct a specific histogram containing these values. We eventually compute a quantitative measure of resemblance based on these histograms.

As it can be seen in Figure 2, a crucial prerequisite of gaining a correct measure of geometric difference is the appropriate pose estimation which consists of computing the scaling, translation and rotation of the objects, so that their surfaces lay one onto another. Mathematically, this task may be reformulated as an optimization problem. In our setting, the input models are restricted to closed polygonal meshes. As we will see in later sections, we need a "watertight" model to correctly compute the alignment and the signed distance fields.

Figure 2. The bananas are markedly similar but in order to be able to apply our method relying on geometrical similarity, they have to be properly scaled, translated and rotated.

3 Pose estimation

There are a number of publications in the field of Image Processing and Computer Vision considering the problem of pose estimation. The goal is always to apply an affine transformation to the inspected objects to bring them into a standard pose. Our algorithm relies on geometric moments up to the second order, however, there are theoretical results by Canterakis and Schulz-Mirbach based on invariant theory [18, 3, 1, 2] claiming to have completely solved the pose estimation problem by incorporating higher order moments. There are two major reasons why we did not use these methods in our current system:

- **Computation of higher order moments:** The central problem in computation of higher order moments is the evaluation of integrals defining these moments, cf. Equation 1. Lien and Kajiya [13] give a method to integrate over non-convex polyhedra, their algorithm proceeds by projecting the triangles of the boundary to the barycenter, and evaluates the integrals for the tetrahedra generated this

way. This method seems to be suitable, however we suspect that long, thin tetrahedra would cause numerical instability. Hatamian [8] presented an algorithm for efficient computation of arbitrarily high order moments in 2D discrete images which may easily be modified to handle 3D volumetric data. In this latter case an inaccuracy would be introduced by sampled nature of the data – we suppose that this effect dramatically increases with the order of the moments.

- **Implementation of the methods:** The results based on invariant theory mentioned above seem to have firm mathematical basis, however, their implementation would presume methods of high numerical sophistication in order to achieve a reasonable efficiency. Moreover, we are not aware of a successful implementation of these algorithms.

For these reasons we did not consider the implementation of the above approaches at present.

In the remaining part of this section, we describe our algorithm for pose estimation, the arising issues and our solutions to them.

3.1 Basics

In our method, we chose to apply mass properties[1] to position and align the objects. We first compute the central moments of the objects up to the second order, a three-dimensional moment μ_{ijk} is described by:

$$\mu_{ijk} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y,z)\, x^i y^j z^k \, dx\, dy\, dz, \tag{1}$$

where $f(x,y,z)$ describes the object. Using this notion, the mass m, center of mass $\mathbf{r}$ and inertia tensor $\mathbf{I}$ are given by formulas:

$$m = \mu_{000}, \tag{2}$$

$$\mathbf{r} = 1/m \begin{pmatrix} \mu_{100} \\ \mu_{010} \\ \mu_{001} \end{pmatrix}, \tag{3}$$

$$\mathbf{I} = \begin{pmatrix} \mu_{020}+\mu_{002} & -\mu_{110} & -\mu_{101} \\ -\mu_{110} & \mu_{200}+\mu_{002} & -\mu_{011} \\ -\mu_{101} & -\mu_{011} & \mu_{200}+\mu_{020} \end{pmatrix} \tag{4}$$

[1] We use the concept *mass properties* according to the traditional terminology of physics, however, since we deal with homogenous density, it might be intuitively more correct to call them *volume properties*.

We first determine the mass or volume of the object, then compute the center of mass and translate the object so that the center of mass is in the origin i. e. $\mathbf{r} = \mathbf{0}$. The function $f(x,y,z)$ describes the body in this new frame, consequently, what we eventually get is a central inertia tensor. After this, we compute the principal axes which define an orthogonal coordinate system where the inertia tensor matrix $\mathbf{I}$ is diagonal. The principal axes are identified by the eigenvectors of the inertia tensor, in order to get a correct alignment, we match the eigenvectors corresponding to each principal axis for the two objects, and make the necessary corrections to get right-handed coordinate systems. Finally, we scale the models by normalizing the volumes.

We chose the above procedure because of its relative simplicity and numerical robustness. To compute mass properties, we use an algorithm by Mirtich [16], which computes the mass, center of mass and the central inertia tensor of polyhedral models.

3.2 Problems

As it can be seen, our algorithm for pose estimation is fairly simple and efficient, however, it has to be emphasised that in general, the application of the above procedure without further processing does not result in a correct estimation. This is due to the following problems:

1. As already mentioned above, the principal axes are identified by the eigenvectors of the inertia tensor. The principal axes of two inspected objects are matched according to eigenvalues which eventually results in a rotational transformation. However, due to the lack of information about the direction of each axis, a two way ambiguity is present for every axis, cf. Figure 3. This means that we have a total of 4 configurations resulting in exactly the same set of principal axes.

Figure 3. Both depicted configurations have the same principal axes and eigenvalues – further computations are needed to align the objects properly.

2. If the eigenvectors of the inertia tensor are very different, it is easy to find the correct correspondences. However, if the eigenvectors are similar, a further ambiguity is introduced since the corresponding principal axes may be switched without affecting the eigenvalues. Furthermore, the similar eigenvalues may imply a completely symmetrical mass distribution around an axis (e.g. a cylinder) or around the center of mass (e.g. a sphere). In this case no unique set of principal axes can be extracted, cf. Figure 4.

Figure 4. The depicted object has a completely symmetrical mass distribution around the z axis rectangular to the image plane. Therefore an infinite number of valid principal axes exist – an extended approach is called for.

3.3 Solutions

The central idea of solutions to the problems described in the previous subsection is the minimization of symmetric difference ΔV of the objects A and B

$$\Delta V = A \nabla B$$

The symmetric difference is an equivalent of XOR operation between the volumes occupied by the objects, therefore for volumetric (i.e. voxelized) objects where every voxel represents a unit of volume, it is simply and efficiently computable. Hereinafter, although not explicitly indicated, we always transform the object A while object B remains in its original pose.

1. For the first problem mentioned above, we consider all the possible rotations $\mathbf{R}_i$ that do not change the principal axes and compute the symmetric difference ΔV for each of them. We finally choose the correct alignment $\mathbf{R}$ corresponding to the minimal ΔV:

$$\mathbf{R} = \min_{\mathbf{R}_i}(\Delta V(\mathbf{R}_i)|0 \leq i \leq 3).$$

Considering the notion of symmetric difference, a similar approach was used by Lensch, Heidrich and Seidel [11] in their recently published method for aligning a 2D image to a 3D model.

2. In case of similar eigenvalues we proceed in a manner that is analogous to the previous solution. We consider all the possible rotations around the axis or center point of the symmetry and choose the one corresponding to the minimal ΔV. For a cylindrical symmetry this can be formulated by

$$\mathbf{R} = \min_{\mathbf{R}(\alpha)}(\Delta V(\mathbf{R}(\alpha))|0 \leq \alpha \leq 2\pi)$$

where α denotes the angle of rotation around the axis of symmetry and $\mathbf{R}(\alpha)$ denotes the according rotation. In case of spherical symmetry the formula for the correct alignment:

$$\begin{aligned}\mathbf{R} \quad = \quad & \min_{\mathbf{R}(\phi,\theta,\psi)}(\Delta V(\mathbf{R}(\phi,\theta,\psi))| \\ & \phi \in [0,2\pi], \theta \in [0,\pi], \psi \in [0,2\pi])\end{aligned}$$

where the angles (ϕ, θ, ψ) denote the Euler angles for parameterization of three-dimensional rotations.

As mentioned above, to handle the second problem, we consider all the possible rotations – in this case we discretize the angle intervals. For a cylindrical symmetry (the mass distribution is symmetrical around one of the principal axes), this results in an efficient approach, but in case of spherical symmetry we believe that this simple approach would be impractical due to the large amount of configurations. However, according to our experiments with different models, objects with spherical symmetry are relatively rare.

In both cases we need to revoxelize one of the objects for each configuration. This might seem to be computationally very expensive, however, we accomplish this by simply computing the distance field of the transformed object which due to hardware acceleration makes the procedure efficient (cf. next section).

4 3D discrete distance fields

Since the core idea of our comparison method is based on distance fields, in this section we review this concept and briefly describe the method of computation. The concrete approach we applied in our system was first published by Klein, Schilling and Strasser [10], it was originally used for reconstruction and simplification of surfaces from contours.

4.1 Distance field

A distance field is essentially a space where every point has a corresponding value indicating its Euclidean distance to the nearest feature of the objects present in this space. In our method, we compute 2D distance fields for parallel slices of the inspected object and then convert this data into a regular 3D raster.

Let $z_0, ..., z_n$ be the coordinates of parallel slices, Ω be the inspected object and

$$\Omega_i = \{(x, y) \mid (x, y, z_i) \in \Omega\} \tag{5}$$

the corresponding cross sections or slices. The 2D distance fields of slices at $z_0, ..., z_n$ are

$$D_i(x, y) = \begin{cases} -dist((x, y), \partial\Omega_i) & \text{if } (x, y) \in \Omega_i \\ dist((x, y), \partial\Omega_i) & \text{otherwise} \end{cases} \tag{6}$$

where $\partial\Omega_i$ denotes the boundary of Ω_i which is described in this case by the cross section polygons in the slice plane, $dist$ denotes the Euclidean distance of a point from a feature in $\partial\Omega_i$.

4.2 Computation of distance fields

The implementation of 2D distance field computation makes use of the possibilities of modern 3D graphics hardware, our major source was [10] published by Klein, Schilling and Strasser, but a similar idea was published by Hoff et Al in [9]. The result of the process is a 2D raster image which contains at each pixel its distance value from the original contour.

Let $p_1, \ldots, p_n$ be one of the polygons describing the contour. Note that the p_i values are floating point numbers and they do not necessarily lie on pixel centers of our image. For the line-segments of the polygon two quadrilaterals are defined in such a way, that the z-value on the plane quadrilaterals indicate the Euclidean distance to the line-segment. The corresponding surface for vertices, where again the z-value represents the Euclidean distance from the vertex is a cone with its apex in the vertex itself. We approximate this cone by several triangles. Note, that depending on the angle of successive line-segments only a small part of the cone, approximated by one or two triangles, is sufficient. Finally, all objects defined this way are rendered into the z-buffer. After rendering, the z-buffer contains the correct distance values and is read out.

Having computed the 2D distance fields within successive slices of an object, this data has to be extended to form a 3D discrete distance field. In the computations, a slight modification of 3D distance is introduced which

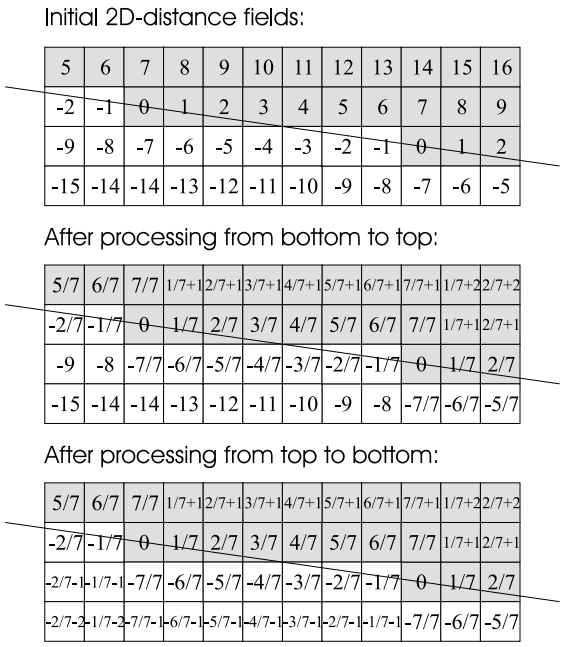

Figure 5. Calculating the 3D-distance field. The distance between neighboring slices is assumed to be 1 Pixel. The upper two pixels in the first column show the special situation described in the text. As a sign change occurs, the two pixel values are adjusted in such a way, that the location of the interpolated zero remains on the contour (drawn from the upper left to the lower right). This is done applying the equation in the text, with $a = 5$ and $b = -2$.

makes the algorithm straightforward and efficient. Instead of using the 3D Euclidean norm (L_2-norm, $||\cdot||_2$), the method incorporates a new distance composed of the Euclidean distance in the 2D slices and the distance in z-direction. This distance is a conservative estimate for the Euclidean 3D distance. For the points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ this modified distance is defined as follows

$$D_{3D}(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \mid z_{p_1 p_2} \mid . \tag{7}$$

The 2D distances are already present, therefore only the $z_{p_1 p_2}$ component has to be computed in a simple post processing step. The algorithm itself traverses the slices two times, the first time in ascending order and the second time in descending order. The distance values of each slice are propagated to the next slice by adding (or

subtracting) the distance s between two slices, measured in units of pixels. If the propagated distance is lower than the already present value, this value is overwritten with the propagated value.

A special situation occurs if the sign between the pixels of the two slices changes, since in this case the surface of the object crosses between the two pixels. If this is the case, we calculate the new distance values for both voxels as follows: Let a be the positive distance value of one of the voxels and $-b$ the negative distance of the other voxel. Then we replace a with $min(a, \frac{a}{a+b} * s)$ and $-b$ with $-min(b, \frac{b}{a+b} * s)$, see Figure 5. The use of the simple distance in z-direction ensures that the distance value of each pixel has to be propagated to only one pixel in each neighboring slice, which makes the algorithm simple and fast. For a more detailed description of this algorithm, please refer to [10].

5 Computation of geometric similarity

As a result of successful pose estimation, the volumes of the two inspected objects overlap as much as possible. If two objects are similar, only a small part of their respective volume is outside of the boundary of the other object, and the average distance of these volumes from the boundary is small. In order to be able to measure such similarity, we compute offset hulls of each object based on the distance fields, cf. Figure 6. We first define a step size ε and consider for both objects A and B the offset hulls $H_A(i\varepsilon)$ and $H_B(i\varepsilon), i \in [0, N]$ defined by isosurfaces at an $i\varepsilon$ distance from the boundary. This may be formulated as follows:

$$H_A(i\varepsilon) = A \oplus S(i\varepsilon)$$

and

$$H_B(i\varepsilon) = B \oplus S(i\varepsilon),$$

where $\oplus$ denotes the Minkowski sum operator and $S(i\varepsilon)$ denotes a sphere with radius of $i\varepsilon$ and center at the origin. Note that the step size ε has to be uniform in order for the results for different objects to be comparable – we accomplish this by making it proportional to the volumes of objects (see the algorithm for extraction of histograms presented below). Finally, for each object we compute the *distance histograms* $h_A(i)$ and $h_B(i)$ indicating how much of the volume of the object B is inside the $i\varepsilon$ offset hull of object A

$$h_A(i) = \frac{V(H_A(i\varepsilon) \cap B)}{V(B)}$$

and vice versa

$$h_B(i) = \frac{V(H_B(i\varepsilon) \cap A)}{V(A)}$$

where $V(.)$ denotes the volume.

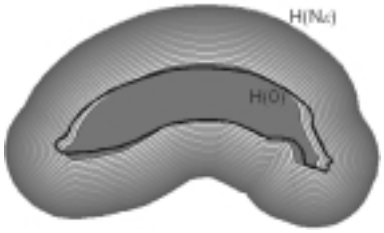

Figure 6. The offset hulls are already implicitly available from the distance field, the volume of banana with black contour is checked against these hulls.

5.1 Computation of the distance histograms

As a first step, we determine a maximal distance value d_{max} defining the range of our histograms: $d_{max} = \sqrt[3]{1}$. Remember that all the objects' volumes are normalized, therefore this range guarantees a uniform similarity measure for different kinds of objects.

Each voxel v has an associated value $val(v)$ according to the distance of the voxel from the boundary of the object according to Equation 6. Let $DF_A(p)$ and $DF_B(q)$ be the *p*'th and *q*'th voxel of the distance fields corresponding to objects A and B, respectively, $val(DF_A(p))$ and $val(DF_B(q))$ denote the corresponding distance values where $0 \leq p, q < N$, N is the total number of voxels. We proceed as follows:

> We traverse all the voxels of the distance field of object A and determine whether a given voxel is outside or inside of object B. If the latter is the case, we determine into which smallest offset hull of object A the voxel belongs and increment the corresponding counter of voxels with such properties. We also perform the same procedure vice versa. A resulting histogram is shown in Figure 7.

Although the computation of $h_A(i)$ and $h_B(i)$ appears to be computationally very expensive, we can benefit from the already generated distance fields. Therefore, we only have to perform a simple counting to obtain the desired values, our algorithm for extraction of histograms is as follows

1. split the interval $[0, d_{max}]$ into n subintervals $I(j)$ of equal length where the index $j \in [0, n-1]$

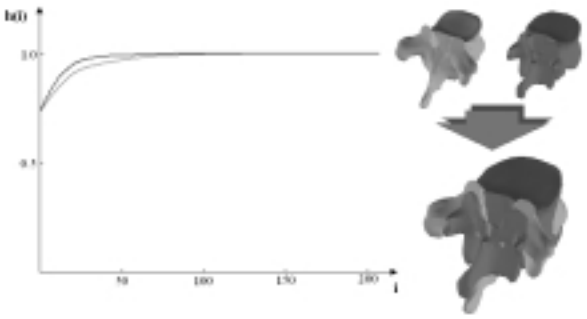

Figure 7. The compared spine bones and the result of pose estimation is depicted on the right. The histograms shown on the left consist of 200 baskets corresponding to offset hulls (we used the notation of $h(i)$ without a lower index because both histograms $h_A(i)$ and $h_B(i)$ are depicted). The hull corresponding to the 200'th basket has an offset of $\sqrt[3]{V}$ where V is the (equal and unit) volume of the objects. As it can be seen, the majority of the volume of each object is inside the other object and already the 30'th offset hull contains almost all of the volumes in both cases.

2. for each voxel index $p \in [0, N-1]$:

   if $val(DF_B(p)) < 0$ then

   if $val(DF_A(p)) > 0$ then

   determine interval index for which $val(DF_A(p)) \in I(j)$

   $h_B(j) = h_B(j) + 1$

   endif

   else $h_B(0) = h_B(0) + 1$

   endif

   if $val(DF_A(p)) < 0$ then

   if $val(DF_B(p)) > 0$ then

   determine interval index for which $val(DF_B(p)) \in I(j)$

   $h_A(j) = h_A(j) + 1$

   endif

   else $h_A(0) = h_A(0) + 1$

   endif

3. integrate and normalize h_A and h_B

Note that only one pass through both of the distance fields is necessary, therefore, the resulting method is remarkably efficient.

5.2 Interpretation of histograms

After obtaining the histograms, in order to extract the desired similarity measure, we need an automatic method for its interpretation. As a first approach, we accumulate the volumes remaining outside of each offset hull for both objects which is in fact the sum of the areas between the y-axis of the histogram and the both curves; we call this value a *normalized volume error*. This gives a straightforward and efficiently computable similarity measure which proved to be gainful for a wide range of applications, cf. Section 6.

As it can be seen in Figure 7, the histograms contain the same value at $i = 0$ – this is the consequence of the fact that $h_A(0) = h_B(0)$ is exactly the amount of overlapping volumes which is obviously equal for both objects. A further observation is that the histograms are generally not equal – this means that according to our measure, object A differs from object B in a distinct way than vice versa. Mathematically, this implies that our similarity measure d is unsymmetric: $d(A, B) \neq d(B, A)$. Furthermore, note that in geometric sense, the sequence of offset hulls generated during the distance histogram computation converge to a common asymptote which is a sphere for every bounded object.

6 Results

As a first experimental application, we applied our system to search in the bone data sets depicted in the Figure 8. The results gained by these experiments allowed us to verify and analyse our algorithms. The samples of results presented in Table 1 prove that our method recognizes the 3D objects most similar to the reference models.

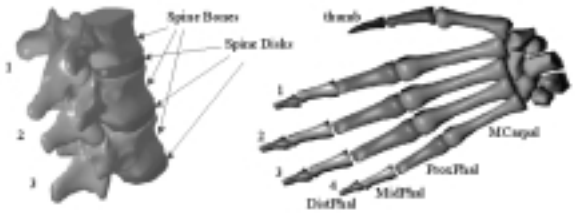

Figure 8. The bone data sets used in our experiments. The grouping indicated by the different colors was achieved automatically based on the similarity measure calculated with our algorithm.

	DisPhal5
DisPhal3	0.042339
DisPhal2	0.414619
BananaClosed	13.732788
Mcarpal5	28.932654
DisPhalhm	29.788163
SpineDisk2	33.392127
SpineBone1	56.317485

	SpineBone1
SpineBone2	6.782950
SpineBone3	8.854751
SpineDisk2	30.292984
BananaClosed	38.090223
ProxPhal3	67.535064
Mcarpal2	114.933660
DisPhal4	343.346326

Table 1. A sample of normalized volumetric errors – the orderings like the two presented here deliver the most similar matches. In the upper table the results of search for objects similar to the bone DisPhal5 can be seen (cf. Figure 8). The results of the search of objects similar to SpineBone1 are in the lower table. The objects are ordered according to their normalized volume error. Note the excellent matching with the human notion of shape similarity.

Moreover, we gain an ordering between the matches which allows even for classifying the models. As it can be seen in Table 3, our algorithm is able to automatically distinguish between different kinds of bones. Note that in certain cases the threshold values of normalized volumetric errors defining the classes of bones may need to be a priori set. Consider for example the hand bones ProxPhal4 and DisPhal2: the normalized volumetric error between them is 6.464720 and we want these bones to be grouped into two different classes. However, for the bones SpineBone1 and SpineBone3 we have an error of 8.854751 and we want them to be classified into the same group. This means that for the spine bones which are considerably more complex structures than the spine disks or hand bones, one should set a higher threshold value to get a correct classification. On the other hand, though, it is relatively easy to distinguish between spine bones and other kinds of bones, cf. Table 2 and 3.

Spine data set		
Object #1	*Object #2*	*Normalized volumetric error*
SpineDisk2	SpineDisk3	0.700581
SpineBone1	SpineBone2	6.782950
SpineBone1	SpineBone3	8.854751
SpineBone2	SpineBone3	1.707098
SpineDisk1	SpineDisk2	1.437496
SpineDisk1	SpineDisk3	0.956824
SpineBone1	SpineDisk2	30.292984
SpineBone1	ProxPhal3	67.535064

Table 2. Excerpts from results of classification of spine bones. The objects in the upper part of the table are classified as similar, the ones in the lower part are classified as different.

Hand data set		
Object #1	*Object #2*	*Normalized volumetric error*
DisPhal2	DisPhal5	0.414619
DisPhal4	DisPhal5	0.000031
MidPhal3	MidPhal5	0.066761
MidPhal2	MidPhal4	0.496065
ProxPhal2	ProxPhal4	0.244528
ProxPhal2	ProxPhalth	3.848611
ProxPhal4	DisPhal2	6.464720
DisPhal5	DisPhalhm	29.788163
DisPhal5	Mcarpal5	28.932654

Table 3. Excerpts from results of classification of hand bones.

7 Conclusions and future work

In this paper we presented a set of methods aiming towards automatic comparison of 3D objects. As a prerequisite of further computations we solved the pose estimation problem for objects represented by closed, "water-tight" meshes, our aligning algorithm delivers a transformation that brings the inspected objects into a pose where the volumes of the models overlap maximally. It should be emphasised that the pose estimation problem is a major issue in a number of applications, and that our solution to this problem is not limited to the specific field of similarity measurement.

We introduced the notion of *distance histograms* as a basic tool for the determination of geometric similar-

ity of stiff objects. The interpretation of the distance histograms using the normalized volume error delivers a simple but efficient sorting criteria. Furthermore, we described an efficient algorithm for the computation of these histograms. In addition, we showed how the distance histograms can be interpreted and finally, we presented some results proving the strength and efficiency of our algorithm.

The results provide an introspection into the properties of our similarity measurement algorithm – the main conclusion is that our system delivers a measure on the *overall* similarity of inspected objects. This feature allows for searching for a reference object in 3D databases and setting up an ordering among the matches which eventually delivers the objects of interest. Furthermore, as shown in the previous section, the algorithm may easily be adapted even for geometric similarity based classification of 3D models.

As a short-term goal, we plan to consider other ways of evaluating our histograms which might result in even better final similarity measure. There also remain a number of problems waiting to be taken care of concerning handling of unclosed meshes, importing the various 3D file-formats, etc. We are aware of new results in the area of distance fields [6] – since the distance fields are a main tool in our method, the study of these results and their possible incorporation into our system is also of great interest. Our current method is provenly applicable for measurement of geometric similarity of 3D objects with relatively stiff structure, however, we believe that an extended version of the presented approach could handle articulated bodies as well – the investigation of this topic is also among our short term goals. Furthermore, we intend to examine whether the introduction of further invariants improve our current pose estimation method. As for longer term goals, we plan to incorporate texture information and other features into the comparison method. We also intend to apply the experiences and results gained in this specific area to elaborate new methods of 3D object synthesis.

References

[1] N. Canterakis. Complete moment invariants and pose determination for orthogonal transformations of 3d objects. Technical report, Technische Universitt Hamburg-Harburg, 1996.

[2] N. Canterakis. 3d zernike moments and zernike affine invariants for 3d image analysis and recognition. In *11th Scandinavian Conf. on Image Analysis*, 1999.

[3] N. Canterakis and H. Schulz-Mirbach. Algorithms for the construction of invariant features. Technical report, Technische Universitt Hamburg-Harburg, 1994.

[4] T. Faber and E. Stokely. Orientation of 3-d structures in medical images. *Transactions on Pattern Analysis and Machine Intelligence*, 10(5):626–633, September 1988.

[5] D. Forsyth, J. L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3-d object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:971–991, 1991.

[6] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *ACM SIGGRAPH*, 2000.

[7] T. Funkhouser. Geometric modeling for computer graphics, http://www.cs.princeton.edu/courses/archive/spring00/cs598b/, 2000.

[8] M. Hatamian. A real-time two-dimensional moment generating algorithm and its single chip implementation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1986.

[9] K. E. I. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. In *ACM SIGGRAPH*, 1999.

[10] R. Klein, A. Schilling, and W. Strasser. Reconstruction and simplification of surfaces from contours. In *Pacific Graphics*, 1999.

[11] H. P. A. Lensch, W. Heidrich, and H. P. Seidel. Automated texture registration and stitching for real world models. In *Pacific Graphics*, 2000.

[12] S. X. Liao and M. Pawlak. On image analysis by moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:254–266, 1996.

[13] S. L. Lien and J. T. Kajiya. A symbolic method for calculating integral properties of arbitrary nonconvex polyhedra. *IEEE Computer Graphics and Applications*, 1984.

[14] C. Lo and H. 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.

[15] C. Lo and H. Don. Pattern recognition using 3-d moments. In *International Conference on Pattern Recognition*, volume 1, pages 540–544, 1990.

[16] B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1996.

[17] D. Saupe. Content-based 3d model retrieval, http://www.informatik.uni-leipzig.de/cgip/, 2000.

[18] H. Schulz-Mirbach. On the existence of complete invariant feature spaces in pattern recognition. In *IEEE International Conference on Pattern Recognition*, volume 2, pages 178–182, 1992.