The Visual Computer manuscript No.
(will be inserted by the editor)

G. Passalis · T. Theoharis · I. A. Kakadiaris

PTK: A Novel Depth Buffer-Based Shape Descriptor for Three-Dimensional Object Retrieval

Abstract The increase in availability and use of digital three-dimensional (3D) synthetic or scanned objects, makes necessary the availability of basic database operations, such as retrieval. Retrieval methods are based on the extraction of a compact shape descriptor; the challenge is to design a shape descriptor which describes the original object in sufficient detail to make accurate 3D object retrieval possible. Building on previous work, this paper proposes a novel depth buffer-based shape descriptor (called PTK) which encompasses symmetry, eigenvalue-related weighting and an object thickness related measure to provide accuracy that surpasses previous state-of-the-art methods. Evaluation of the novel method's parameters as well as a direct comparison to other approaches is performed using publicly available and widely used databases.

Keywords Depth buffer · Object Retrieval · Symmetry

1 Introduction

Databases of three-dimensional (3D) objects are becoming increasingly available and include real-life objects digitized with 3D scanners and synthetic objects produced by modeling software. The proliferation of such databases on the internet is also increasing. As is the case in relational, text and two-dimensional (2D) image databases there are a number of basic operations that must be available on 3D object databases; such operations include comparison, classification, and retrieval. However, 3D objects encode more information than previous objects, making these tasks more challenging.

3D object retrieval can be applied on a large number of important tasks, making it one of the most widely researched topics. From web-based search engines (e.g., given a query object to find similar objects), to security applications (e.g., given a human face dataset to find its match in a database) there is a need for accurate and efficient object retrieval. All object retrieval methods share the same basic idea: a 3D object is processed once in order to extract a compact and accurate description of itself (called a *descriptor*); a database of descriptors is thus created. When a query object is given, its descriptor is extracted and compared against the database of descriptors (using a distance metric such as L^1 or L^2) to retrieve the most similar objects from the database.

The descriptor is thus an alternative representation of the object that can be used in place of the original (e.g. polygonal) representation. The main challenge is to create a descriptor which results in accurate 3D object retrieval.

1.1 Related Work

Three dimensional object representation techniques can be categorized [8] into a) histogram-based, b) topology based, c) view-based, and d) shape-based.

Histogram-based methods such as the ones proposed by Osada *et al.* [13] and Ankerst *et al.* [2], even though they have several advantages (e.g., noise robustness, rotational invariance), in general are considered less descriptive compared to shape-based methods. Topological methods include Sundar *et al.* [19], who proposed the use of the skeleton of an object for object retrieval, and Hilaga *et al.* [7]. A drawback of topological methods is that relatively small anomalies on the object's surface can have a significant impact on the topological properties of the object. Additionally different classes of objects that share the same topology are difficult to discern using such methods.

G. Passalis, T. Theoharis
{passalis,theotheo}@di.uoa.gr
Computer Graphics Group
Department of Informatics and Telecommunications
University of Athens, Ilisia 15784, GREECE

I. A. Kakadiaris
ikakadia@central.uh.edu
Computational Biomedicine Lab
Department of Computer Science
University of Houston, Texas 77204, USA

View-based methods [11,3] extract 2D views of the objects and use them as features for retrieval purposes (e.g., silhouettes). There is an unavoidable loss of descriptiveness when projecting 3D geometry onto a 2D view. In our previous work on object retrieval, we extended this idea to 3D views and presented a 3D rigid object retrieval method which employs depth buffers for representing and comparing the objects [20]. Specifically, multiple depth buffers per object (computed from different points of view) are compared for surface and volume similarity. In this paper, we also propose a depth buffer-based descriptor, which integrates part of our previous work but it is improved significantly (see Sec. 1.2).

Frome *et al.* [5] introduced two shape descriptors, 3D shape contexts and harmonic shape contexts, and used them for recognizing objects in noisy range images. Kazhdan *et al.* [9,10] used spherical harmonics as shape descriptors for 3D shape retrieval. Spherical harmonics is an attractive mathematical tool for this task due to its inherent rotationally invariant nature.

Laga *et al.* [12] proposed the use of geometry images for matching 3D objects. They apply the work of Gu *et al.* [6] on geometry images to transform the 3D polygonal data to geometry images. Before they convert an object to its geometry image representation, they align it using Principal Component Analysis. Unfortunately, they evaluate their method using a limited dataset (120 models) without comparing it to other state-of-the-art methods.

Vranic [22] proposed several shape descriptors, which were depth buffer-based, silhouette-based, ray-based and hybrid (based on the combination of the above descriptors). Each shape descriptor produces a feature vector, and vectors from different objects are compared using L^1 or L^2 distances to determine similarity. In Sec. 3.3 we used Vranic's 3D search engine website [21] to evaluate the performance of our method and compare it against previously proposed descriptors.

1.2 Shape Descriptors

Shape descriptors of the geometry of an object are an alternative representation that encompasses information about its shape that is independent of the original object's representation. A shape descriptor produces for each object a set of coefficients that define a *feature vector*: $FV = \{C_1, C_2, ..., C_n\}$. In order to be able to directly compare two feature vectors from different objects, n must be constant and each coefficient C_i must describe the same feature in both objects. The distance of these two feature vectors using a specific metric (e.g., L^1, L^2) is correlated with the difference of the shape of these two objects. According to [22] the desirable properties of a 3D shape descriptor are the following:

p. 1 non-restrictiveness to closed or orientable polygonal meshes,
p. 2 invariance with respect to translation, rotation, scaling and reflection of a 3D-object,
p. 3 robustness with respect to mesh anomalies (e.g., floating vertices, degenerate triangles),
p. 4 robustness with respect to levels-of-detail (e.g., different tessellation),
p. 5 robustness with respect to surface noise and outliers,
p. 6 multi-resolution feature representation,
p. 7 efficient feature extraction,
p. 8 compact representation,
p. 9 efficient search procedure and
p. 10 shape discrimination.

In this paper, we utilize depth buffers in order to compute an estimate of the thickness of an object along a given direction, and use this information to represent the object and perform comparisons in the spectral domain. Even though the depth buffer has already been used as a shape descriptor, we propose a number of important modifications such as the use of symmetry for alignment and the use of eigenvalue-related weighting (see Sec. 2). These modifications allow our novel method to outperform previous state-of-the-art in terms of performance, as detailed in Sec. 3.

In our previous work [20], the depth buffer was not used directly as a shape descriptor. It was produced at the comparison step of the method, and for each pair of objects that were compared, different depth buffers were used. Therefore, they could not describe the object in a unique way. In contrast, in methods where the depth buffer is used as a shape descriptor (e.g., the proposed method, Vranic's method [22]), the depth buffers are computed only once during an initialization step and can then be used as an alternative to the polygonal representation of the object.

The rest of the paper is organized as follows: Sec. 2 describes our method in detail, Sec. 3 presents our evaluation results using publicly available databases, and Sec. 4 provides a summary and proposes further work.

2 Method

The proposed method consists of two steps, a preprocessing step that needs to be applied once on each object in the database to produce a feature vector, and the retrieval step, where feature vectors from different objects are compared to determine similarity.

- **Preprocessing.** For each object:
  - Normalize the scale of the object and translate its center of mass to $(0, 0, 0)$.
  - Align the object using symmetry information and Principal Component Analysis.
  - Acquire the depth buffers.
  - Transform the buffers from the spatial to the spectral domain.

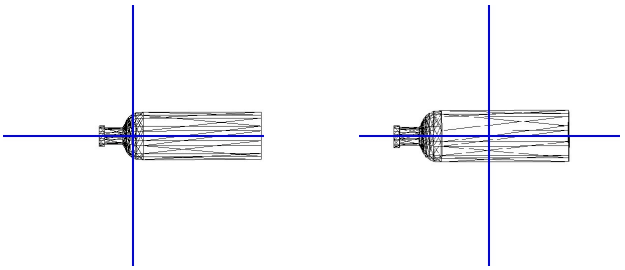


Fig. 1 Computing the center of mass for non-regularly sampled objects: (a) Un-weighted approach; (b) Weighted approach.

  - Apply weighting to spectral coefficients produce final feature vector.
- **Retrieval.** Compare two feature vectors by computing their difference using an L^1 or L^2 metric.

We will show that our method has all the desired properties described in Sec. 1.2. None of the steps of the proposed method has a requirement for closed or orientable objects (p. 1). The normalization and alignment of the objects (Sec. 2.1, 2.2) offer invariance with respect to translation, rotation and scaling (p. 2).

The depth buffer (see Sec. 2.3), at high resolutions can offer accurate shape discrimination (p. 10) and is insensitive to the object's tesselation since it regularly resamples the object (p. 4). Additionally, mesh anomalies have minimal impact on depth buffer computation (p. 3) while the presence of surface noise can be addressed with the application of a simple 2D denoising filter on the depth images (p. 5). The depth buffers can be acquired using specialized graphics hardware resulting in efficient feature extraction (p. 7).

The spectral transform (see Sec. 2.4) offers an inherently compact and multi-resolution representation (p. 6,8), while it allows the direct comparison of the coefficients of the spectral domain resulting in efficient retrieval (p. 9). Additionally it is invariant to reflection (p. 2) which was not addressed in the alignment step.

2.1 Normalization

Since 3D objects found in databases can have arbitrary translation and scaling parameters, the normalization step scales them to a common reference volume and translates them so that their center of mass is at $(0, 0, 0)$ in R^3.

The straightforward method to compute the center of mass is to sum all vertices of the polygonal mesh and divide the sum by their number. This computation method violates property 4 (Sec. 1.2), since areas of the object with more detailed tessellation will have a higher contribution. This is shown in Fig. 1(a) where the bottle's center of mass is not computed correctly.

To tackle this problem we employ a *weighted* approach. Instead of summing the vertices, we sum the barycenter of each triangle multiplied by the area of the triangle:

$$\overline{S} = \sum_f \frac{A_f}{A_T} \frac{\overline{V_f^a} + \overline{V_f^b} + \overline{V_f^c}}{3}$$

where A_f and $\overline{V_f^a}, \overline{V_f^b}, \overline{V_f^c}$ are the area and vertices of triangle f respectively while A_T is the total area of the object. With this approach, all areas of the object will contribute to the center of mass independently of their tessellation, since higher tessellation means more triangles but less weight per triangle and vice versa. This is shown in Fig. 1(b). Note that differences in tessellation may still introduce discrepancies in the center of mass computation. However, our method overcomes these problems since the absolute position of the center of mass does not affect the difference nor the sum of the depth buffers (see Sec. 2.3).

After the center of mass is translated to the origin, we need to uniformly scale the object so that it fits in a given space, such as the unit cube. This can be accomplished by finding the vertex with the maximum distance from the center of mass, and dividing the coordinates of all vertices by this distance. This method, however, is very sensitive to the presence of outliers, thus violating property 5 (Sec. 1.2).

Instead of setting the maximum distance of any vertex equal to 1 we set the mean distance equal to 1. Again we use a weighted approach to compute the mean distance (m):

$$m = \sum_f \frac{A_f}{A_T} \left| \frac{\overline{V_f^a} + \overline{V_f^b} + \overline{V_f^c}}{3} \right|$$

and then divide all vertices with this value. Note that this approach does not guarantee that the maximum distance is within a certain volume. The majority of the vertices of any object will fit within a cube of a certain size; we empirically set the edge of the cube equal to 5, and disregard any vertex that lies outside this limit. The size of the cube's edge must be the same for all objects. The selection of the size of the cube involves a trade-off: larger cubes increase the percentage of vertices that lie inside them, but decrease the actual volume that the object occupies within the cube, making the method less descriptive.

2.2 Alignment

All shape descriptors that do not have an inherent rotation invariant representation (such as the one proposed here) require an alignment step in order to position all

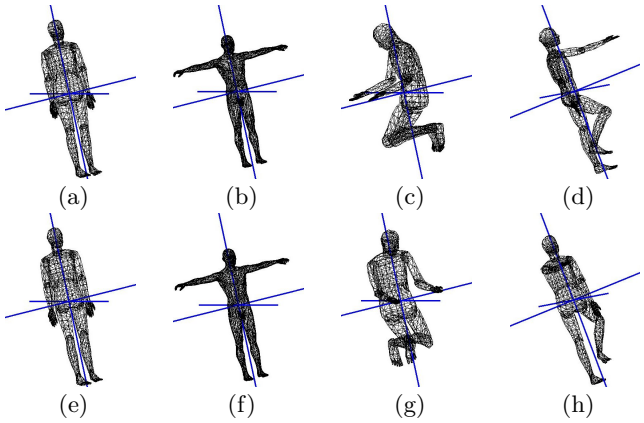


Fig. 2 Four objects from human class aligned with different methods: (a,b,c,d) PCA; (e,f,g,h) Symmetry and PCA.

objects in a standard orientation. These descriptors, compared to rotation invariant methods (e.g., spherical harmonics [10]), have the drawback that they depend heavily on the alignment step. Should this step fail to perform as expected on a specific object, it is likely that this object will not be classified correctly. This drawback is counterbalanced by the higher descriptiveness offered by such shape descriptors (e.g., depth buffer-based descriptor) [22].

The most commonly used technique for alignment is Principal Component Analysis (PCA) [18]. Each object is analyzed in three principal axes (or eigenvectors), and according to their eigenvalues these vectors are mapped to the X,Y,Z axes. As in the normalization step, we use the barycenters of the triangles, multiplied by their corresponding area, instead of the vertices themselves in order to be independent of the object's tessellation. Note that only the computation of the covariance matrix is changed, the rest of the method remains unaffected.

PCA is generally considered to work well for a variety of classes. There are cases, however, where it fails, and this failure cannot be compensated for in later steps of the method. The problem is caused when two objects that belong on the same class, have different principal axes with respect to their distinctive shape features. There are a number of reasons behind this, such as deformations, different aspect ratio of object parts, movement of limbs, etc. In Fig. 2(a, b, c, d) this is shown for objects belonging to the human class. The most important eigenvector in all four cases is the one from head to toes, but the second and third most important eigenvectors are switched in cases (c,d) compared to (a,b).

To overcome some of the limitations of PCA we propose a method of alignment which incorporates symmetry. This novel method is based on the fact that most real life objects are symmetrical with respect to a plane. Additionally, this plane of symmetry is expected to be similar for objects belonging to the same class, and it generally remains unaffected by small scale deformations. Even if these deformations introduce a certain amount of asymmetry, usually it is not sufficient to change the orientation of the plane radically, allowing the application of this approach in databases containing both symmetric and asymmetric objects (such as the ones utilized in this paper). In cases of object classes that have more than one plane of symmetry we select the plane with the maximum symmetry, in some framework of measurement.

A single plane is not enough to solve the alignment problem as we need three orthocanonical vectors to completely define an orientation. The perpendicular vector to the plane of symmetry is used as the first one and for the remaining two we use the eigenvectors acquired by projecting all vertices onto the plane of symmetry and performing a 2D PCA. Note that this approach does not determine whether the perpendicular vector should be mapped to +X or -X axis (the same for the two eigenvectors and +Y,-Y,+Z,-Z axes). However, the spectral transform that we employ in Sec. 2.4 is invariant to these issues. In Fig. 2(e, f, g, h) the misalignment caused by the PCA-only approach is corrected by our novel symmetry approach, since the human's plane of symmetry is hardly affected by the movement of limbs. Our approach performed well even in the fourth case, where the model is asymmetric.

2.2.1 Determining Symmetry Planes

Given a plane in R^3 we measure the object's symmetry along this plane by computing the difference between the front and back depth buffers that were acquired using a projection parallel to this plane. For example, in Fig. 3, the symmetry along X direction is defined as the difference between the x_1 and x_2 depth buffers. Our goal is to find the plane that minimizes this difference.

To this end, we employ a general optimization technique known as Enhanced Simulated Annealing (ESA) [17]. ESA has the advantage that it globally minimizes functions that have an arbitrary number of continuous variables, without requiring knowledge of the function's derivatives. Its non-deterministic nature does not guarantee optimal results, but in general, ESA is unlikely to be trapped in local minima. To completely eliminate the local minima problem, the process can be repeated a few times, each time starting with different initial conditions and keeping only the best solution. The time ESA needs to converge varies, since it depends on the object's resolution. On average, symmetry computation takes approximately 1 sec for each object of the databases.

In our case, the objective function computes the difference between two depth buffers and uses four variables in order to describe arbitrary orientations with *quaternions* [1]. A quaternion is defined as $q = (s, \overrightarrow{v})$, where s is a scalar, and $\overrightarrow{v}$ is a vector ($\overrightarrow{v} = (x, y, z)$). A rotation about a unit vector $\overrightarrow{n}$ by an angle θ can be computed using the quaternion $q = (cos\frac{\theta}{2}, sin\frac{\theta}{2}\overrightarrow{n})$. Quaternions were preferred over Euler angles because they do not

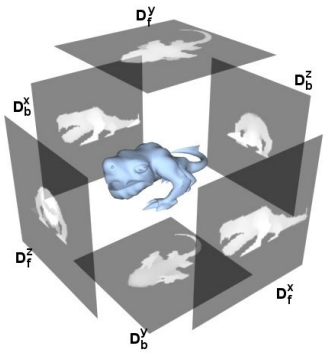


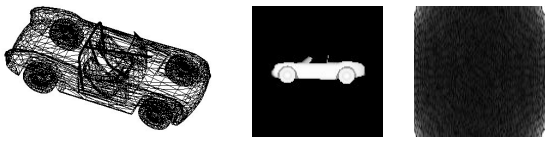
Fig. 3 3D object with its depth buffers along X,Y,Z directions.

suffer from problems like *gimbal lock* (where the rotation around one axis is cancelled by the rotation around the remaining two). In our previous work [14], the combination of ESA with depth buffers was also employed successfully in matching complementary 3D rigid objects.

2.3 Depth Buffer Acquisition

After the object is normalized and aligned, we acquire the depth buffers. The depth buffer is a regular grid of scalar values that holds the distance between the near clipping plane and the object's surface. It is acquired by rendering the object using an orthographic projection which uses as clipping planes the faces of the cube described in Sec. 2.1. As shown in [15], in order to describe most objects accurately we need front and back depth buffers from three orthocanonical directions (Fig. 3). Note that we already used depth buffers for determining the symmetry plane, but these depth buffers were of extremely low resolution, and therefore useless as shape descriptors.

Previous depth buffer-based approaches directly use these depth buffers [22,20]. We propose the following novel modification: instead of storing the front and back depth buffers along each direction, we store their difference and their sum. For example, for the X direction, we store $D^x_{dif} = D^x_f - D^x_b$ and $D^x_{sum} = D^x_f + D^x_b$. The two new depth buffers (D^x_{dif}, D^x_{sum}) require the same storage space as the original two (D^x_f, D^x_b) and there is no information gain or loss since we can produce the one pair from the other. The difference lies in the fact that the D_{dif} holds the thickness of the object at each pixel, not the distance of the object's surface from the near clipping plane. This value is insensitive to errors in the translation of the object along the direction of projection, making it more suitable for comparison purposes. In contrast, D_{sum} has no such property and no physical meaning, it is used only as a complement to D_{dif} and therefore is considered less important and it is assigned lower weights as shown in Sec. 2.5.



(a) (b) (c)

Fig. 4 (a) Object from car class; (b) Depth buffer along Y direction after alignment; (c) FFT of the depth buffer.

2.4 Spectral Transform

For a given object, the depth buffers acquired in the previous step are highly descriptive but not suitable for direct comparison with the buffers from another object. The spectral domain is more suitable and robust for such purposes; we therefore apply a spectral transform on these buffers. The discrete 2D fourier transform (F_d) of the depth buffer D of size S is given by:

$$F_d(m,n) = \sum_{x=0}^{S-1}\sum_{y=0}^{S-1} D(x,y)e^{-j2\pi\frac{mx+ny}{S}}$$

In practice we use the FFT algorithm (denoted with F_f)instead of DFT due to its lower computational cost. The only restriction that it imposes is that the buffers must have power-of-two dimensions.

As shown in Fig. 4(c) most information is concentrated on the four corners of the image. For a SxS depth buffer we keep KxK coefficients as follows:

$$\begin{aligned} F(m,n) = {} & |F_f(m,n)| + |F_f(m,S-n-1)| + \\ & |F_f(S-m-1,n)| + \\ & |F_f(S-m-1,S-n-1)| \end{aligned}$$

for $m,n = 0..K-1$. We sum the norm of the coefficients (since FFT produces complex numbers) from all four corners in order to have a reflection invariant representation.

Additionally, we apply individual weights on the FFT coefficients depending on their position in the buffer:

$$F_w(m,n) = \frac{2K-i-j}{2K}F(m,n)$$

where F_w is the weighted version of F, both of dimension KxK. This innovation is based on the idea that lower frequency coefficients hold more important information and should be assigned a higher weight. In contrast, higher frequency coefficients are suppressed since they are more sensitive to noise.

2.5 Weighting Scheme

The feature vector is the concatenation of the coefficients from the spectral transforms of the six buffers. For each direction (X,Y,Z) we produce spectral coefficients from the sum and difference of the depth buffers. On previous depth buffer-based descriptors the coefficients from each

direction were treated equally, based on the assumption that they encode the same amount of information on average.

We propose a novel weighting scheme that is contrary to this assumption: each direction is given an eigenvalue-related weight, which is applied to all of its coefficients. The idea is that if a direction encodes more information, then it should be assigned a higher weight. For an object aligned using PCA, there are three eigenvectors E_1, E_2, E_3 with increasing eigenvalues. The X direction is coaxial with E_1, meaning that E_2 and E_3 are "visible" in the depth buffer, encoding the maximum amount of information. The Y direction is coaxial with E_2 and Z with E_3. The final weights were empirically set to $(3, 2, 1)$ for the X,Y and Z directions. If our symmetry modification is used instead of the standard PCA, the second weight is assigned to the vector perpendicular to the plane of symmetry, while the remaining two weights are assigned to the two eigenvectors obtained from the 2D PCA. This redistribution of weights was selected empirically, based on how much information we expect to be encoded along the direction perpendicular to the plane of symmetry.

The coefficients produced from the buffer that holds the depth difference are more important than the ones produced from the buffer that holds the depth sum (see Sec. 2.3). We therefore decrease the weights of the buffers that hold the sums (relative to the buffers that hold the differences) by approximately 70%.

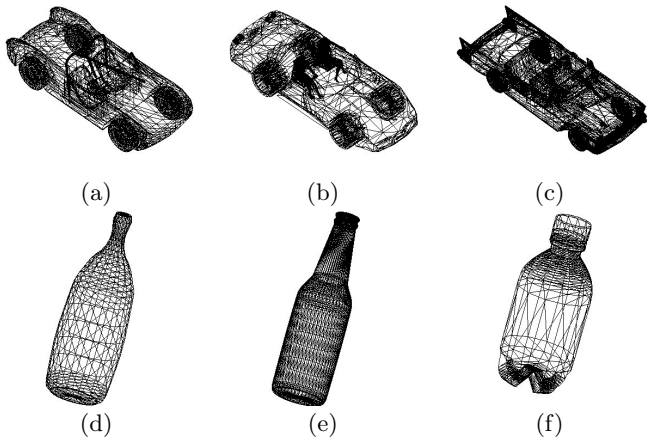


Fig. 5 Two object classes from the KN database: (a,b,c) examples from the car class; (d,e,f) examples from the bottle class.

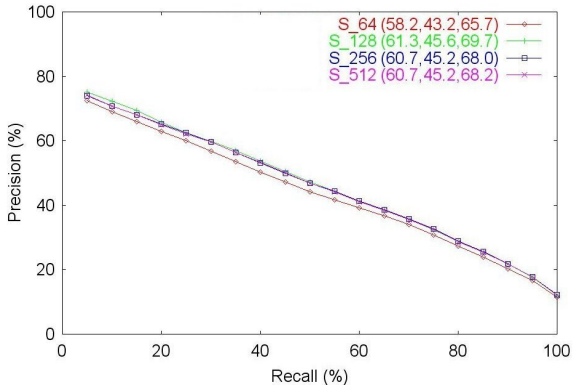


Fig. 6 Variable depth buffer resolutions. Performance reported on KN database.

3 Results

We first analyze the proposed method's behavior for different settings and subsequently evaluate it by comparing it with other state-of-the-art methods. Additionally, we show the performance gain for each of the innovations introduced in Sec. 2. The results are compared in precision-recall curves. For each shape descriptor three numbers are provided, in addition to its curve. The first is the average precision for recall 5% to 50%, the second is the average precision for recall 5% to 100% and the third is the percentage of correctly classified nearest neighbors. The third number represents the percentage of the queries where the best match that is reported belongs in the same class with the query object. In all figures, for our method, we use a depth buffer resolution of 128×128, $K = 48$, a L^1 distance metric, and all modifications enabled, unless stated otherwise.

3.1 Databases

We utilized two publicly available and well known databases in order to evaluate our method by comparing it againt other state-of-the-art shape descriptors on the same databases. The first database is provided by the Princeton Shape Benchmark [16], includes a total of 1814 objects, and it is divided into two subsets of 907 objects each (test and training), which will be referred to as *PR1* and *PR2* respectively. The second database is provided by Vranic's 3D Search Engine [21], includes 1841 objects, and will be referred to as *KN*. The objects in these databases are given in a 3D polygonal representation, have arbitrary orientation and scaling, and they are not necessarily closed. Along with each database, a classification is provided, which groups objects that belong to the same class. Examples of these classes are given in Fig. 5.

3.2 Efficiency and Robustness Evaluation

In this section the behavior of our method is analyzed for different settings. The effect of depth buffer resolutions is depicted in Fig. 6, which shows the precision-recall curves for resolutions from 64×64 to 512×512. Performance is not significantly affected by depth buffer resolution, proving the robustness of our method with respect to this parameter. Additionally, for resolutions higher

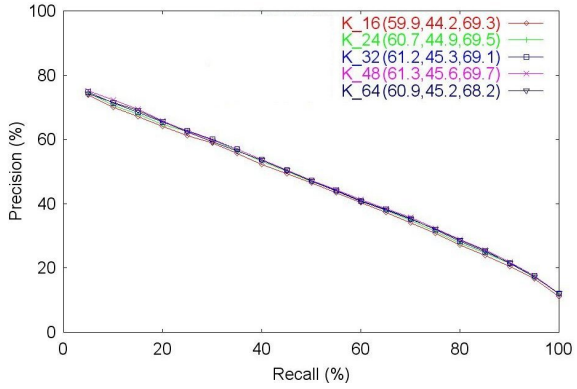


Fig. 7 Different K parameter settings (see Sec. 2.4). Performance reported on KN database.

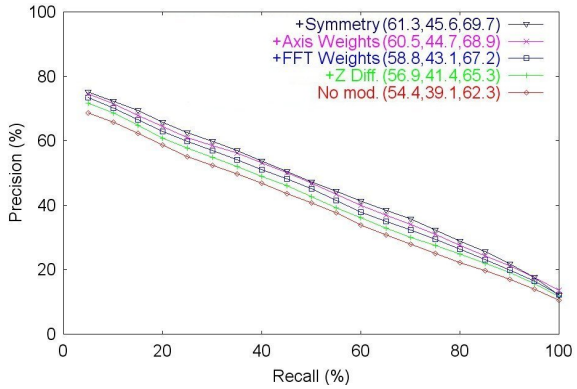


Fig. 8 Effect of proposed innovations over original depth buffer shape descriptor performance on the KN database. From bottom up, each modification is added, increasing the cumulative performance.

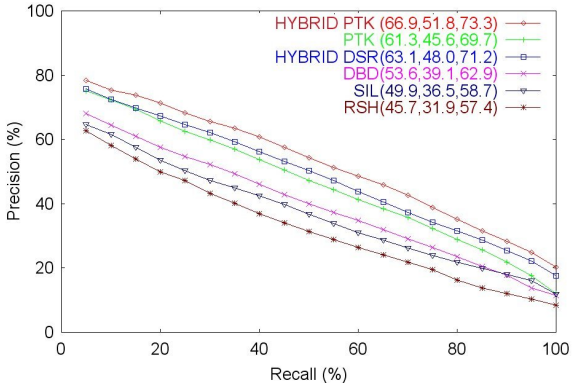


Fig. 9 Our method versus state-of-the-art using the KN database.

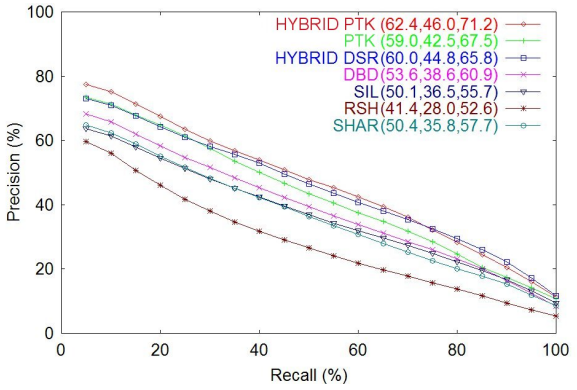


Fig. 10 Our method versus state-of-the-art using the PR1 database.

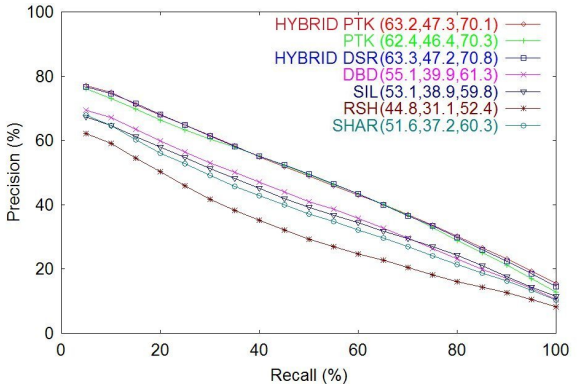


Fig. 11 Our method versus state-of-the-art using the PR2 database.

than 128×128 there is no performance gain, while there is an increased computational cost. Fig. 7 depicts the performance of our method with respect to the K parameter (see Sec. 2.4). Again, the results consistency show that we can increase the efficiency of our method (in terms of storage space and retrieval time) without a significant performance penalty. For example, a K value equal to 16 produces a feature vector of size $16 \cdot 16 \cdot 6 = 1536$. This is only 11% of the size of the feature vector for which we get the peak performance ($48 \cdot 48 \cdot 6 = 13824$). Note that for extremely low K values (lower than 10) our performance starts to deteriorate as the descriptive power is significantly reduced.

Fig. 8 shows the effect of the innovations proposed in this paper individually, in terms of performance gain. Using our implementation of the original depth buffer-based shape descriptor [22] we get 62.3% correctly classified nearest neighbors (Fig. 8, **No mod**). By using the sum and difference of the depth buffers instead of the buffers themselves, as described in Sec. 2.3, this number is increased to 65.3% (Fig. 8, **Z Diff**). A further increase to 67.2% results from using the weights described in Sec. 2.4 (Fig. 8, **FFT Weights**). If the weighting scheme of Sec. 2.5 is also used, we get 68.9% (Fig. 8, **Axis Weights**), and by introducing the final modification (symmetry-based alignment, Sec. 2.2), the peak performance of 69.7% is achieved (Fig. 8, **Symmetry**).

3.3 Performance Evaluation

In this section, we compare our method against state-of-the-art shape descriptors. In Figs. 9, 10 and 11, *DBD*

refers to the original depth buffer based descriptor, *SIL* refers to a silhouette-based descriptor, *RSH* refers to a ray-based feature vector and *HYBRID DSR* refers to a hybrid descriptor that combines coefficients from the previous three. These descriptors were computed using the publicly available software found in [21] and they are described in detail in [22]. In addition, in Figs. 10 and 11 we also report the performance of the spherical harmonics method (*SHAR*) presented by Kazhdan *et al.* [9]. Our method is referred to as *PTK*. We also combined our method with the hybrid descriptor by replacing the coefficients from the original depth buffer-based descriptor with the coefficients from our method. This is referred to as *HYBRID PTK* and shows the overall performance benefit that can be achieved by the hybrid descriptor if the previous method is replaced with the one proposed here.

Figs. 9, 10 and 11 report the performance on KN, PR1 and PR2 databases respectively. All descriptors perform similarly across all three databases, indicating that the databases are of similar difficulty. In all three cases our method (*PTK*) clearly outperforms the original depth buffer-based descriptor (*DBD*). The average percentage of correctly classified nearest neighbors is 69.1% for our method against 61.5% for *DBD*. In fact, our method's performance alone is close to the previously proposed hybrid method in all cases (69.1% against 69.2% on average). Additionally, with the exception of PR2, the new hybrid descriptor (*HYBRID PTK*) outperforms the original one (*HYBRID DSR*), with the average percentage of correctly classified nearest neighbors percentage equal to 71.5% against 69.2%.

Note that the new hybrid descriptor has not benefited greatly from the proposed method (based on the performance gap between the old and new depth buffer descriptors). The reason behind this is that the original hybrid descriptor was optimized (weights, number of coefficients) for the *DBD* descriptor. We did not optimize the hybrid descriptor for our method; we simply replaced the *DBD* coefficients with the *PTK* coefficients, as the main objective of this paper is to compare the proposed method against the previously proposed depth buffer descriptor and other basic state-of-the-art descriptors, rather than to propose a novel hybrid descriptor.

To measure computational time, we used the KN database (containing 1841 objects) and compared our method against the original depth buffer based approach proposed by Vranic. In order to process the whole database, our method without the symmetry-based alignment needed 2 minutes, with the symmetry-based alignment it needed 39 minutes, while the original depth buffer approach needed 11 minutes (results were measured on a 3Ghz Pentium IV with 1GB of RAM). The symmetry-based alignment adds a significant computational cost but needs to be performed only once for each new object that is inserted into the database. Therefore, it does not hinder performance during the retrieval phase. Moreover, our method without the symmetry step was able to outperform the original depth buffer-based approach by a large margin.

4 Conclusion/Future Work

A novel depth buffer-based shape descriptor was presented. The proposed method takes advantage of the object's symmetry to perform better alignment, combines the depth buffers from each direction in a novel way (by computing their difference and sum) and incorporates a weighting scheme for the spectral coefficients. Each one of these modifications offers a performance gain. Their combined use results in an improved descriptor that outperforms current state-of-the-art non-hybrid shape descriptors. When combined with other descriptors, an improved hybrid descriptor can be created which outperforms the best published hybrid descriptor.

As explained in Sec. 3.3 the improved hybrid descriptor is not optimized. Further research could result in an optimum hybrid descriptor that combines the ideal number of coefficients from the proposed depth buffer descriptor with coefficients from other descriptors, in order to achieve the highest possible performance.

Another possible improvement to the performance of our method may result from the use of a more accurate technique for the computation of the object's thickness in each axis. In Sec. 2.3 the object's thickness is only approximated, since only two outer depth buffers are used. To accurately compute the thickness we need to employ an arbitrary number of layered depth buffers using a technique known as depth peeling [4]. This modification however, may introduce errors when used with non-closed objects.

5 Acknowledgments

G. Passalis would like to acknowledge financial support from the Greek General Secretariat for Research and Technology (GSRT) under the PENED program.

References

1. http://mathworld.wolfram.com/
2. Ankerst, M., Kastenmüller, G., Kriegel, H.P., , Seidl, T.: 3D shape histograms for similarity search and classification in spatial databases. In: Proc. of 6th International Symposium on Spatial Databases, pp. 207–226. Hong-Kong, China (1999)
3. Chen, D., Tian, X., Shen, Y., Ouhyoung, M.: On visual similarity based 3D model retrieval. Eurographics **22**(3), 223–232 (2003)
4. Everitt, C.: Interactive order-independent transparency. Tech. rep., NVIDIA Corporation (2001)
5. Frome, A., Huber, D., Kolluri, R., Bulow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: Proc. of the European Conference on Computer Vision (2004)

6. Gu, X., Gortler, S., Hoppe, H.: Geometry images. In: Proc. of SIGGRAPH, pp. 355–361. San Antonio, TX (2002)
7. Hilaga, M., Shinagawa, Y., Kohmura, T., Kunii, T.: Topology matching for fully automatic similarity estimation of 3D shapes. In: Proc. of SIGGRAPH, pp. 203–212. Los Angeles, CA (2001)
8. Hlavaty, T., Skala, V.: A survey of methods for 3D model feature extraction. Seminar on Geometry and Graphics in Teaching Contemporary Engineer (2003)
9. Kazhdan, M.: Shape representations and algorithms for 3D model retrieval. Ph.D. thesis, Princeton University (2004)
10. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3D shape descriptors. In: Proc. of Eurographics Symposium on Geometry Processing, pp. 167–175. Aachen, Germany (2003)
11. Krüger, T., Wickel, J., Alvarado, P., Kraiss, K.: Feature extraction from VRML models for view-based object recognition. In: Proc. of the 4th European Workshop on Image Analysis for Multimedia Interactive Services, pp. 391–394. London (2003)
12. Laga, H., Takahashi, H., Nakajima, M.: Geometry image matching for similarity estimation of 3D shapes. In: IEEE Proc. of Computer Graphics International, pp. 490–496. Crete, Greece (2004)
13. Osada, R., Funkhauser, T., Chazalle, B., Dobkins, D.: Matching 3D models with shape distribution. ACM Transactions on Graphics **21**(4), 807–832 (2002)
14. Papaioannou, G., Karabassi, E., Theoharis, T.: Reconstruction of three-dimensional objects through matching of their parts. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(1), 114–124 (2002)
15. Passalis, G., Kakadiaris, I., Theoharis, T.: Efficient hardware voxelization. In: IEEE Proc. of Computer Graphics International, pp. 374–377. Crete, Greece (2004)
16. Shilane, P., Min, P., Kazhdan, M., Funkhouser, T.: The princeton shape benchmark. In: Proc. of Shape Modeling International. Genova, Italy (2004)
17. Siarry, P., Berthiau, G., Durbin, F., Haussy, J.: Enhanced simulated annealing for globally minimizing functions of many-continuous variables. ACM Transactions on Mathematical Software **23**(2), 209–228 (1997)
18. Strang, G.: Linear Algebra and its Applications, 3rd edn. Harcourt Brace Jovanovich (1988)
19. Sundar, H., Silver, D., Gagvani, N., Dickinson, S.: Skeleton-based shape matching and retrieval. In: Proc. of Shape Modelling and Applications Conference, pp. 130–142. Seoul, S. Korea (2003)
20. Vajramushti, N., Kakadiaris, I.A., Theoharis, T., Papaioannou, G.: Efficient 3D object retrieval using depth images. In: Proc. of the 6th ACM SIGMM International Workshop on Multimedia information, pp. 189–196. New York, USA (2004)
21. Vranic, D.: Content-based classification of 3D-models by capturing spatial characteristics. http://merkur01.inf.uni-konstanz.de/CCCC/
22. Vranic, D.: 3D model retrieval. Ph.D. thesis, Universitat Leipzig (2004)

List of Figures