3D Shape Retrieval from a 2D Image as Query

Masaki Aono* and Hiroki Iwabuchi*
* Toyohashi University of Technology, Aichi, Japan
E-mail:aono@tut.jp Tel: +81-532-44-6764

***Abstract*—3D shape retrieval has gained popularity in recent years. Yet we still have difficulty in preparing a 3D shape by ourselves for query input. Therefore an easy way of doing 3D shape search is much awaited in terms of query input. In this paper, we propose a new method for defining a feature vector for 3D shape retrieval from a single 2D photo image. Our feature vector is defined as a combination of Zernike moments and HOG (Histogram of Oriented Gradients), where these features can be extracted from both a 2D image and a 3D shape model. Comparative experiments demonstrate that our approach exhibits effectiveness as an initial clue to searching for more relevant 3D shape models we have in mind.**

I. Introduction

In recent years, use of 3D shape models has been expanding rapidly in a wide range of fields, including 3D movies, 3D TVs, design of industrial products, and 3D medical imaging. At the same time, in order to efficiently manage a 3D shape model database, an innovative technology has become necessary to search and retrieve, based on the contents of the shape of a 3D model. However, in the research on 3D shape retrieval in the past, most efforts have been devoted to providing a given 3D model as a query. If there is no 3D model at hand, it is very difficult to search an appropriate 3D shape model with high accuracy.

In this paper, we propose a method for 3D shape retrieval from a photograph, taken by a digital camera, for instance, as a query. As far as we know, there has been little research on 3D model retrieval from 2D photo images. Ansary et al [1] indicated one of such approaches to 3D shape model retrieval from 2D photo images, using Zernike moments generated from multiple viewpoints of silhouette images from 3D model. However, their approach has two drawbacks. First, position and scale of a given object greatly affects the search results. Second, searching for an object that contains surface convexities and concavities (e.g. human face, car body shape) might mislead the search results, because such a surface detail might be hidden in the silhouette.

In our proposed method, we attempt to circumvent these two problems. For the first problem, in order to align the position and the scale of a given 2D image as a query, with the images that are generated from 3D shape models through rendering and kept in 3D shape database, we apply normalization to both 2D images and 3D shape models before extracting features. For the second problem, we propose a composite feature vector for representing both a 3D shape model and a 2D query photo image. Specifically, we propose a composite feature vector as a combination of Zernike moments which correspond to silhouette images, and the HOG (Histogram of Oriented Gradients) features which correspond to shaded depth-buffer images. Naturally, by attempting to integrate multiple features, the dimensions of combined feature vectors tend to become large. We also have noticed that HOG feature vectors are suitable for representing objects that have surface convexities and concavities, but are not robust against rotation. Moreover, we have reduced the number of dimensions, by applying Principal Components Analysis (PCA) for each block of HOG features, which we call PCA-HOG. As for the sensitivity to rotation, we have generated multiple images from a single 2D image, by rotating it little by little with respect to the center of the image multiple times in advance. In addition, we have generated mirror images of each rotated image with respect to the vertical axis assuming the origin at the center of the image.

We have conducted experiments to confirm the effectiveness of our method against Ansary et al's method. To evaluate our method, we have used the Princeton Shape Benchmark [2], a standard 3D shape benchmark widely used. As a result, we have achieved significantly better search accuracy.

In the following sections, we will first describe normalization of both 3D (pose normalization) and 2D (size normalization). We will then describe feature alignment between a 2D image and 3D shape models, followed by the similarity computation from aligned features. Finally, we will demonstrate our method through experiments in Experiment and Evaluation section, and summarize our results in the Conclusion.

II. Related Work

3D shape retrieval has gained popularity ever since a well-known benchmark data called Princeton Shape Benchmark (PSB) data [2] became available. Most of the previous research on 3D shape retrieval has focused on encoding of 3D shape features (or descriptors) [3], in terms of robustness, size, position, and orientation.

Among many shape features, a group of researchers defined theirs by means of multiple 2D projected images. The earliest such approach was proposed by Chen et al [4], [5]. They are also one of the precursors of using Zernike moments, while Novotni et al [6] used Zernike moments around the same time, for 3D shape retrieval. Defining 3D shape features from multiple 2D images leads to the idea of searching 3D shape models from 2D images. The early such approaches included those of Mahmoudi et al [7] and Ohbuchi et al [8].

In their FOX-MIIRE 3D-Model Search Engine [9], [10], Ansary et al [1] proposed an approach to 3D shape retrieval from 2D images, where they used 49 dimensions of Zernike

moment coefficients generated from silhouette images, each of which in turn was produced from a 3D model, by looking at the model from different multiple viewpoints. In addition, they introduced a probabilistic Bayesian method for indexing 3D models. With their search engine, it is possible to play with 3D models, sketches, and photos as a query.

Since we will compare our approach with that of Ansary et al, we will estimate their approach in more detail. Ansary et al's method first computes a unit sphere, and generates a set of 320 uniformly spread points on the sphere, assuming these points are the characteristic viewpoints to generate silhouette images. Then, for each generated silhouette image, they compute Zernike moments. Finally they cluster 320 views into the best characteristic view set, using X-Means algorithm [11] and Bayesian Information Criteria (BIC). However, their approach has two drawbacks. First, the posture, position, and scale of a given object as well as camera parameters for 2D photos greatly affect the search results. Second, searching for an object that contains surface convexities and concavities (e.g. human face, car body shape) might end up in unintended search results, because such surface detail might be hidden in the silhouette.

As another example of 3D retrieval from 2D silhouettes, Napoléon el al [12] conducted similar research with multiple silhouette images. Their query includes not only 2D silhouettes, but hand-drawn sketches. They introduced several interesting ideas including silhouette/contour alignment using dynamic programming in a coarse-to-fine way for search efficiency. Their approach does not account for 3D shape retrieval from a 2D photo as a shaded query image.

III. Normalization

Before alignment of a set of 3D shapes and a 2D query image, it is necessary to perform normalization of both 3D shapes and a 2D image. The normalization can be regarded as a preprocessing for feature alignment between 3D and 2D features.

A. 3D Pose Normalization

One of the problems of the previous approach by Ansary et al [1] is that the search result is susceptible to posture, location, and scale variations of input query. To alleviate this problem, we have applied normalizations to both 2D input query image as well as 3D shape models. For 3D pose normalization, we follow our previous work [13].

Most of the pose normalization methods proposed so far, based on Principal Component Analysis (PCA), rely on computing the principal axes from the collection of points on the surface of a given 3D shape [14], [15]. We herewith call the pose normalization method using the collection of points on the surface *PointSVD*.

With *PointSVD*, we first randomly generate m points by means of Osada's method [16]. Specifically, to generate a uniformly random point $\mathbf{p}$ on an arbitrary triangle composed of vertices $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, we employ the following formula:

$$\mathbf{p} = (1-\sqrt{r_1})\mathbf{a} + \sqrt{r_1}(1-r_2)\mathbf{b} + \sqrt{r_1}r_2\mathbf{c}$$

In our implementation, two random variables r_1 and r_2 in the above equation are computed by a Niederreiter [17] pseudo-random number generator. We then compute the centroid $\bar{\mathbf{p}}$ and the orientation matrix Q. The centroid $\bar{\mathbf{p}}$ of a 3D triangle is computed by taking the average of randomly generated points on the triangle as follows:

$$\bar{\mathbf{p}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{p}^{(i)}$$

The generated points are translated in space so that the centroid becomes the origin of the space, which is represented by point transformation matrix P.

$$P = \begin{pmatrix} p_x^{(1)} - \bar{p}_x & p_x^{(2)} - \bar{p}_x & \dots & p_x^{(m)} - \bar{p}_x \\ p_y^{(1)} - \bar{p}_y & p_y^{(2)} - \bar{p}_y & \dots & p_y^{(m)} - \bar{p}_y \\ p_z^{(1)} - \bar{p}_z & p_z^{(2)} - \bar{p}_z & \dots & p_z^{(m)} - \bar{p}_z \end{pmatrix},$$

where P is decomposed by Singular Value Decomposition or SVD as below:

$$P = U\Sigma W^T,$$

where U and W are 3×3 orthogonal matrices, and Σ is a 3×3 diagonal matrix, having its singular values in descending order. The rotation matrix Q is computed by taking the transpose of left singular vectors as represented by U.

$$Q = \hat{U}^T$$

Finally, we compute the reflection matrix F from rotated point sets P', with $P' = QP$ as follows:

$$F = \begin{pmatrix} sign(f_x) & 0 & 0 \\ 0 & sign(f_y) & 0 \\ 0 & 0 & sign(f_z) \end{pmatrix}$$

where

$$f_x = \sum_{i=0}^{m} sign(p'^{(i)}_x)(p'^{(i)}_x)^2 \qquad .$$

f_y and f_z are similarly defined. The size and location invariance is achieved by transforming the matrix V into V' with k number of points, where V is a matrix already having rotational and reflective invariance as per below:

$$V = \begin{pmatrix} v_x^{(1)} - \bar{p}_x & v_x^{(2)} - \bar{p}_x & \dots & v_x^{(k)} - \bar{p}_x \\ v_y^{(1)} - \bar{p}_y & v_y^{(2)} - \bar{p}_y & \dots & v_y^{(k)} - \bar{p}_y \\ v_z^{(1)} - \bar{p}_z & v_z^{(2)} - \bar{p}_z & \dots & v_z^{(k)} - \bar{p}_z \end{pmatrix}$$

$$V' = FQV.$$

It should be noted that the size invariance is achieved by calculating the distance between the centroid and an arbitrary point on the surface divided by the largest distance between the centroid and the farthest point on the surface if we put the 3D object in a unit sphere centered at the centroid. Our *PointSVD* is different from Ohbuchi et al's method [14] in the way we resolve the reflective invariance. Specifically, in Ohbuchi et al's method, the distance computation between feature vectors is carried out by considering the reflective ambiguity, whereas the method is heavily dependent on their own feature vectors,

which make it difficult to apply to other feature vectors. On the other hand, as one of earlier pose normalization methods, "Continuous PCA" by Vranic et al [15] determines the principal axes with an ordinary Principal Component Analysis (PCA) accompanied by the approximation of numerous points on the surface. Continuous PCA resolves the reflective ambiguity by using these approximate points on the surface. Because the method relies on approximation and is sensitive to surface tessellation, it often happens that almost similar objects end up with significantly different pose normalizations.

Our *PointSVD* first generates random points on the surface as usual, then analyzes the principal axes and resolves reflective ambiguity. The reflective ambiguity is resolved similar to Ohbuchi et al's method, yet our method does not depend on feature vectors similar to Vranic et al's method [15]. In a sense, *PointSVD* combines the advantages of the two methods.

B. 2D Image Normalization

In addition to the pose normalization of 3D shape models, 2D query image has to be normalized to make it independent of size, location, and orientation of the object in the 2D image. We first enclose the object in the 2D image by a bounding box to trim the background. We then make the resolution of the remaining image be size 256 by 256.

IV. Feature Alignment between 2D Image and 3D Shape Models

Our proposed 3D shape retrieval system is illustrated in Fig. 1. Unlike one of the previous works by Ansary et al [1], we have a composite features extracted and kept in the database (DB) as shown in Fig. 1.

One of the problems with the previous approach by Ansary et al [1] is that the surface convexities and concavities are lost during the feature alignment between 2D images and 3D shape models. This is especially true of shapes such as car bodies and humans, where complex surface bumps are abundantly observed. Technically, this is due to the fact that only silhouette projections have been used, attempting to align features between 2D images and 3D shape models.

For better feature alignment in order to provide the search system with better clues to similarities between 2D images and 3D shape models, we have taken the approach with a composite feature that accounts for surface convexities and concavities faithfully to some extent.

It should be noted that in order to align 2D and 3D features, they have to be identical in terms of dimensions and semantic meaning. As we will describe later, our features are two-fold. One of the features comes from Zernike moments [18], similar to the approach of Ansary et al. On the other hand, we introduce HOG (Histogram of Oriented Gradients) [19], as another feature, attempting to account for surface convexities and concavities. These features will be elaborated in the following sections.

A. Feature Computation from 3D Shape Models

Unlike ordinary feature definitions for 3D shape search, it is extremely difficult to preserve compatibility between 3D and 2D features. Our main idea is to maintain a composite of two different features, common to both 3D and 2D, which makes it possible to compute the similarity between 3D shapes preprocessed and kept in our system database, and a given 2D image as a query.

Specifically, we first introduce a collection of shaded *depth-buffer* images rendered from 3D shape models, by projecting each 3D shape to a view-plane after pose normalization. Because depth-buffer images have "depth" information as gray scale pixels, it is theoretically possible to cope with the surface convexities and concavities. However, to align features obtained by depth-buffer images with a 2D image having unknown orientation, it is natural to think of maintaining a collection of depth-buffer images; otherwise, predicting a 2D query image becomes a formidable task. The question that remains is how many viewpoints (i.e. projected view planes) are sufficient to deal with a given 2D query image of unknown orientation. This problem is by no means easy to resolve.

After trial and error, we have decided to apply depth-shaded rendering to obtain depth-buffer images from 26 distinct viewpoints as illustrated in Fig. 2. This process has to be done in advance to construct "Feature set 1" in Fig. 1. An example of 26 depth-buffer images is shown in Fig. 3. From these depth-buffer images, we will compute the HOG features [19] discussed in the next section.

On the other hand, another common feature we have adopted is a collection of Zernike moments [18]. These moments have a salient feature of rotational invariance, which consists of an orthogonal set of moments with polar representation. In our approach, Zernike moments are extracted from binary silhouette images, in the way similar to Ansary et al [1] and Chen [4]. Actually we adopt the same number of dimensions (49 dimensions) for Zernike moments with Ansary et al's approach, because in the Experiment section we will compare our approach with that of Ansary et al. However, since there is a symmetry of views for binary silhouette images, we reduce to 14 viewpoints as illustrated in Fig. 4, instead of 26 viewpoints for shaded depth-buffer images as shown in Fig. 3 generated by the 26 distinct viewpoints illustrated in Fig. 2. Zernike moments computed from silhouette images thus far are kept in our database as "Feature set 2" in Fig. 1. The overall 3D composite feature vector generation is depicted in Fig. 5.

It is by now straightforward to make two separate features from a given 2D image as illustrated below in Fig. 1. The basic idea here is that we regard a given 2D query image as a projected image from a virtual 3D shape object similar to those kept in our 3D shape database. The only difference is that a 2D query image is assumed to be just a single snapshot, instead of taken from either 26 or 14 different viewpoints. Still, it is possible to extract both the HOG features and the Zernike moments, so that we can compare the similarity between the 2D query image with those kept in our 3D shape database.

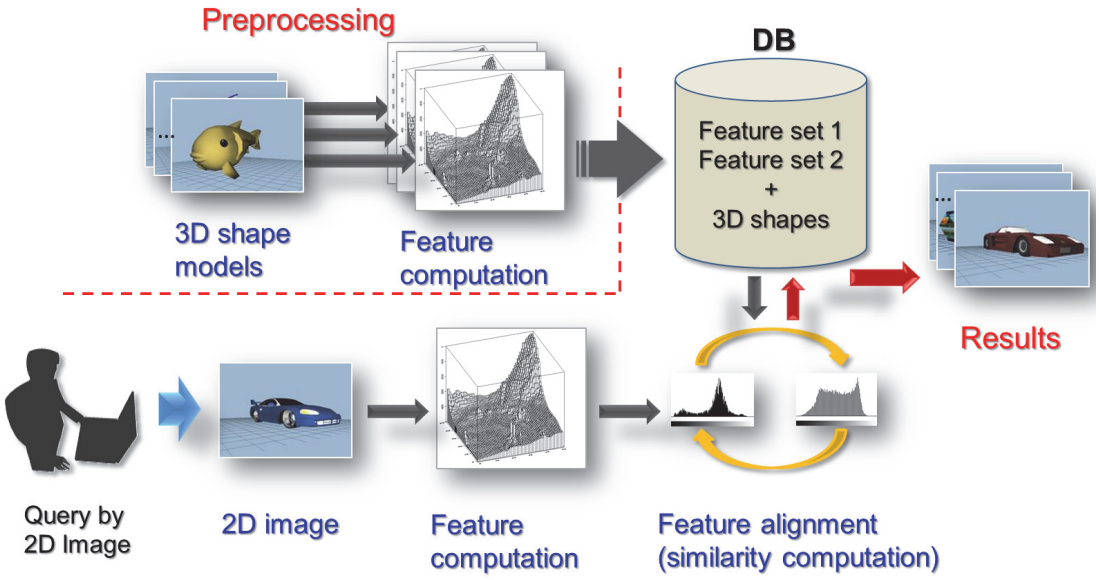

Fig. 1. System concept: 3D shape retrieval from 2D image

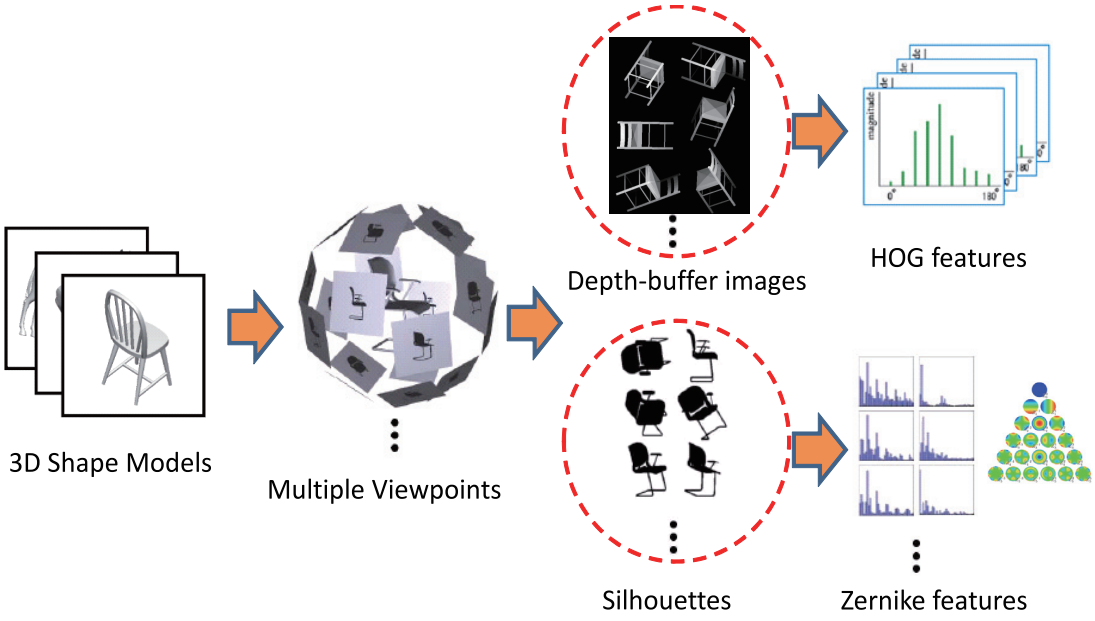

Fig. 5. Flow of computing proposed composite feature vectors from a 3D model

V. Similarity Computation from Aligned Features

In this section, we describe how we compute the similarity between a given 2D query image and 3D shape models, where both of them have two separate features. Before showing how it is going to be done, we would like to briefly overview what the actual features are expressed mathematically and how they are computed.

A. Zernike moments as a feature from silhouette images

The basic procedure for computing Zernike moments as a feature is summarized as follows:

1) Determine ρ (a parameter in the radial direction) and θ (a parameter in the angular direction) from polar expression of Zernike polynomial in a unit circle
2) Compute real and imaginary parts of Zernike moments from ρ and θ from Zernike polynomial
3) From N by N binary (silhouette) image, compute Zernike moments of size n by m, assuming that the degree be denoted by n and the repetition number be denoted by m.

In the subsequent description, we use notation from Amayeh et at [20]. Zernike polynomial denoted by $V_{n,m}(x, y)$ is a

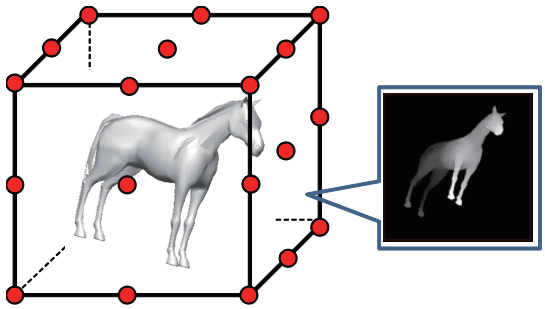

Fig. 2. 26 viewpoints selected after pose normalization

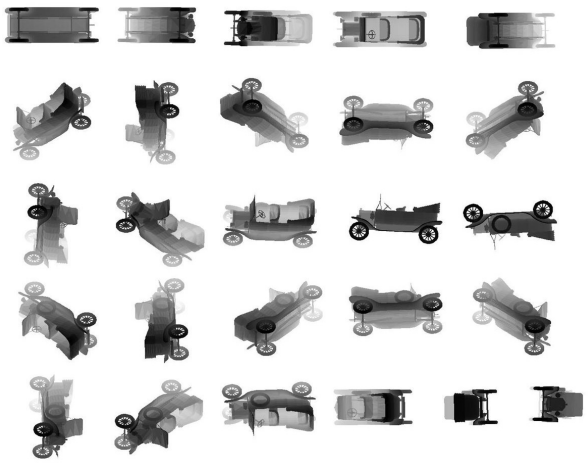

Fig. 3. Depth-buffer images obtained by rendering a collection of 3D shape models projected onto viewplanes from 26 different viewpoints

complex polynomial with an orthogonal basis in a unit circle. Mathematically, it is represented by the following formula:

$$V_{n,m}(x,y) = r_{n,m}(\rho)e^{jm\theta}(0 \le \rho \le 1, 0 \le \theta < 2\pi),$$

where n represents the degree of polynomial, m is a repetition number having the property that $n-|m|$ is even and $|m| \le n$. $\rho = \sqrt{x^2+y^2}$ is a distance from the center to (x,y), and $\theta = \tan^{-1}(y/x)$ is an angle between x axis and a vector

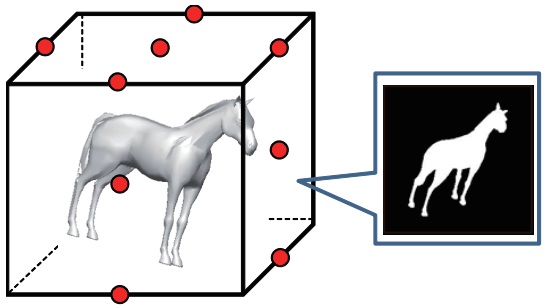

Fig. 4. 14 viewpoints for silhouette images

(x,y). $r_{n,m}(\rho)$ is expressed per below:

$$r_{n,m}(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s(n-s)!\rho^{n-2s}}{s!((n+|m|)/2-s)!((n-|m|)/2-s)!}$$

Zernike moment $A_{n,m}$ with degree n and repetition number m is expressed by the following formula:

$$A_{n,m} = C_n \int\int_E f(x,y)V^*_{n,m}(\rho,\theta)dxdy,$$

where $C_n = (n+1)/\pi$ and the pixel value at (x,y) is denoted by $f(x,y)$. By digitizing this continuous form, we obtain the following:

$$\overline{A_{n,m}} = C_n \sum_{(x,y)\in E} f(x,y)V^*_{n,m}(\rho,\theta)$$

$$E = \{(x,y)|x^2+y^2 \le 1\}$$

In our approach, we compute Zernike moments for each circular area of radius R, expressed by $x^2+y^2 \le R^2$, where (x,y) denotes a pixel coordinate. Since the unit circular area is represented by $x^2+y^2 \le 1$, we transform pixel (x,y) by $x \leftarrow x/R$ and $y \leftarrow y/R$. It is also noted that if we assume the origin to be at $(0,0)$, the central pixel value may not be considered. To avoid this situation, we pick up a subpixel coordinate $(1/2, 1/2)$ as the origin. From these considerations, two parameters ρ and θ are defined as follows:

$$\rho = \sqrt{(x-R-1/2)^2+(y-R-1/2)^2}$$

$$\theta = \tan^{-1}\frac{y-R-1/2}{x-R-1/2}$$

The actual Zernike moments are computed by the next equation with real and imaginary part 90 degree phased:

$$\overline{A_{n,m}} = C_n[Re(x,y) - jIm(x,y)]$$

$$\begin{aligned} Re(x,y) &= \sum_{(x,y)\in E} f(x,y)r_{n,m}(\rho)\cos m\theta \\ Im(x,y) &= \sum_{(x,y)\in E} f(x,y)r_{n,m}(\rho)\sin m\theta \end{aligned}$$

B. HOG as a feature from shaded depth-buffer images

For a shaded image projected on a plane, it is well-known that there are a variety of ways to extract shape features for shape matching. Examples include Scale Invariant Feature Transform (SIFT) [21], [22] and Speeded-Up Robust Features (SURF) [23]. These are classified into *local* shape features, and often referred to as "bag of keypoints" (BOK) [24] or "bag of visual words" (BOVW) [25]. In particular, SIFT has a salient feature of robustness to most image transforms including rotation. It should also be noted that local features tend to be verbose, which makes dimensionality reduction such as PCA fit very well. Examples include PCA-SIFT [26].

On the other hand, in the area of general shape recognition such as human shape recognition, HOG (Histogram of Oriented Gradients) reportedly behaves better than PCA-SIFT

and wavelets such as Haar-like features [19]. Our 3D shape database includes quite a few human shapes and animals as illustrated in the Experiment section, and we have adopted the HOG as another feature that corresponds to shaded depth-buffer images for 3D shape as well as a 2D (shaded) snapshot given as a query. HOG is known to be robust to geometric and photometric transformations, except object rotation. Unlike other local features, HOG is computed on a dense grid of uniformly spaced cells and uses an overlapping set of cells called a *block*. A block is a unit of our normalization of HOG features.

Our HOG feature is computed by the following steps:

1) Compute gradient magnitude m and gradient orientation θ
2) Compute a histogram by taking gradient orientation as horizontal axis
3) Normalize features for each *block*

Gradient magnitude m and gradient orientation θ are defined by the following formula:

$$m(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2}$$

$$\theta(x,y) = \tan^{-1}\frac{f_y(x,y)}{f_x(x,y)},$$

where

$$\begin{aligned} f_x(x,y) &= L(x+1,y) - L(x-1,y) \\ f_y(x,y) &= L(x,y+1) - L(x,y-1) \end{aligned}$$

and $L(x,y)$ denotes the image value at pixel (x,y).

In HOG, as suggested above, we usually define a *cell*, a collection of pixels. In our case, we define a cell as the 16×16 square pixel region shown in Fig. 6. Fig. 7 illustrates how original $L(x,y)$ (shaded depth-buffer image) is associated with a cell. As suggested by Dalal [19], we divide the gradient orientation into 9 bins for 0-180° (i.e. by 20° for each bin). This makes it possible to represent a HOG feature by a 9 dimensional vector per cell, or $\mathbf{f}(\text{cell}[i,j]) = [f_1(i,j), f_2(i,j), ..., f_9(i,j)]$. A feature vector for a cell as well as the derived histogram is illustrated in Fig. 6.

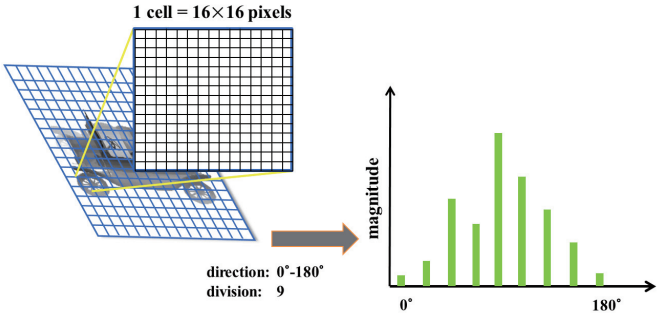

Fig. 6. Definition of a histogram and a cell composed of 16 by 16 pixels

In the third step, we normalize the histogram illustrated in the rightmost graph in Fig. 6. Gradient magnitudes vary over a wide range owing to local variations in illumination and foreground-background contrast. Therefore, effective local contrast normalization turns out to be essential for good performance. However, instead of normalizing a cell, we normalize a *block*, consisting of $q \times q$ cells. Through trial and error, we let $q = 3$. Thus, a block consists of 3×3 cells. It should be also noted that the normalization works very well if we permit overlapping blocks as illustrated in Fig. 8.

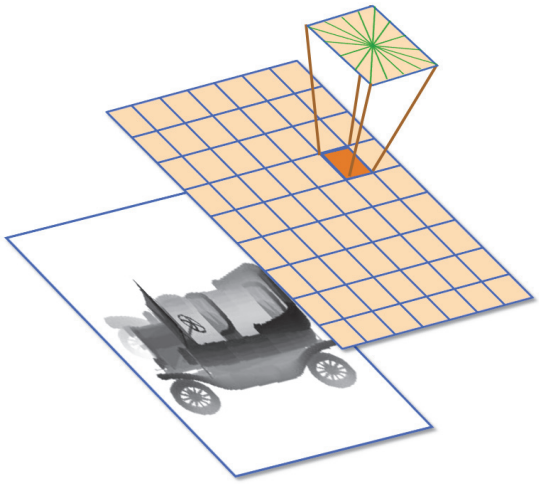

Fig. 7. For each cell, we can define a HOG feature vector with nine dimensions

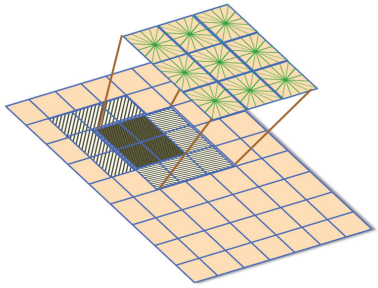

Fig. 8. When normalization has taken place, we allow block overlaps

Let $\mathbf{v}$ be the unnormalized block vector. The normalized block vector $\tilde{\mathbf{v}}$ with L_2-norm is computed per below:

$$\tilde{\mathbf{v}} = \frac{\mathbf{v}}{\sqrt{(\sum_{k=1}^{q\times q\times bin} v(k)^2) + \epsilon^2}},$$

where ϵ denotes a small constant for smoothing, and bin denotes the number of bins for gradient orientation. In our implementation, $bin = 9$. Since we adopt a shaded depth-buffer image resolution of 256×256 pixels, a cell of 16×16 pixels, a block of 3×3 cells, and a block overlapping nearby blocks, both horizontal and vertical directions need to shift 14 (= 16 - 2 outmost cells) times to sum up a HOG feature vector, which results in 14×14 blocks. In total, a HOG feature vector can be expressed by the following formula:

$$\mathbf{f}(\text{cell}[i,j])_k = [f_{1,k}(i,j), f_{2,k}(i,j), ..., f_{9,k}(i,j)],$$

where k is a block index $1, ..., 14 \times 14$, and i and j are the cell indices taking values 1, 2, and 3, respectively. Thus, the dimension of a HOG feature vector amounts to 15,876 (= $14 \times 14 \times 3$).

C. Enhancement to HOG

We have noticed that there are two major problems with a naïve way of producing a HOG feature vector. It may be summarized per below:

- A naïve HOG generates a large feature vector. This might have redundant information and might suffer from the curse of dimensionality.
- A HOG is known to have rotational sensitivity.

To cope with these problems, we have taken the following countermeasures.

1) Avoiding large dimensions of HOG: Redundant information as well as the curse of dimensionality that might occur with a high-dimensional HOG feature vector can be circumvented by dimensionality reduction. We have observed that most of the redundancy occurs since we allow block overlaps. It is therefore natural to apply dimensionality reduction each time block is generated. Although there are many sophisticated nonlinear dimensionality reductions [27] known to date, we have applied the simplest linear dimensionality reduction, Principal Component Analysis (PCA), to each block, inspired by Kee's PCA-SIFT [26]. To hold important information, we use the largest k components so that cumulative proportion becomes over 0.90 (90%) as a rule of thumb.

2) Rotational sensitivity: A HOG turns out to be one of the good local features for capturing surface convexities and concavities by means of edge gradients. However, we must solve the problem of rotational sensitivity to some extent. The key to alleviating this problem is to prepare a priori images rotated by 45 degrees as well as their mirror images as shown in Fig. 9. Note that this treatment is only applied to a 2D query image.

D. Similarity Computation

Once two feature vectors are set, we are ready to perform similarity computation between a given 2D query image and 3D shape models kept in the database. Instead of doing direct similarity computation, it is much easier to perform dissimilarity computation by defining distance between a pair of objects represented by a composite of feature vectors. In the following, we describe dissimilarity computation for each feature, and then define our proposed dissimilarity computation as a whole.

1) Dissimilarity computation for Zernike moments: Assuming $\mathbf{x}$ and $\mathbf{y}$ to be Zernike moment features from a 2D query image and a silhouette image projected from a 3D shape model, dissimilarity between $\mathbf{x}$ and $\mathbf{y}$ is represented by the following equation:

$$d_Z(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{N} |x_k - y_k|$$

Normalized dissimilarity is thus represented with L1-norm by

$$\tilde{d}_Z(\mathbf{x}, \mathbf{y}) = \frac{d_Z(\mathbf{x}, \mathbf{y})}{\|d_Z(\mathbf{x}, \mathbf{y})\|_1 + \epsilon}.$$

2) Dissimilarity computation for PCA-HOG: Unlike Zernike moments, feature vectors for HOG (or PCA-HOG) are defined to each block of cells, consisting of a collection of histograms based on gradient magnitude and gradient orientation. We therefore need to know the proportion of overlaps at each element of feature vector, and also need to skip from further computation for the sake of efficiency, if either of the elements is zero. It is therefore reasonable to first introduce a measure of similarity, instead of dissimilarity, which takes care of overlaps and zero-element skips with Bhattacharya's coefficients [28] as follows:

$$sim_H(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{N} \sqrt{x_k y_k}$$

Normalized similarity can be represented with L1-norm by

$$\tilde{sim}_H(\mathbf{x}, \mathbf{y}) = \frac{sim_H(\mathbf{x}, \mathbf{y})}{\|sim_H(\mathbf{x}, \mathbf{y})\|_1 + \epsilon}.$$

It is now straightforward to define the dissimilarity per below:

$$\tilde{d}_H(\mathbf{x}, \mathbf{y}) = 1 - \tilde{sim}_H(\mathbf{x}, \mathbf{y})$$

3) Dissimilarity computation as a whole: Given dissimilarities for Zernike moments and PCA-HOG, all that remains is how we combine them to make a dissimilarity measure as a whole. We propose to define a linear combination of the two dissimilarities as expressed by the following formula:

$$d(\mathbf{x}, \mathbf{y}) = \alpha \tilde{d}_Z(\mathbf{x}, \mathbf{y}) + \beta \tilde{d}_H(\mathbf{x}, \mathbf{y}),$$

where $\alpha + \beta = 1$. To determine these weights, we adopt the so-called *purity selection* method [29]. The purity selection method tries to measure the "coherence" of the retrieved objects in the first positions of the ranking, at the cost of giving multiple queries in advance as a learning phase to generate similarity rankings for the queries.

VI. Experiments and Evaluations

In this section, we will describe comparative experiments with the previous approach by Ansary et al [1]. The experiments have been conducted by a PC with Intel Core i7 920 CPU, 16GB memory, and Debian Linux operating system with Java 1.6. Princeton Shape Benchmark (PSB) data have been used [2]. Query 2D photo images as shown in Fig. 10 were collected from the Internet. We selected those 2D photo images which have clear backgrounds, trying to avoid extra preprocessing to identify what objects sitting in the background must be removed.

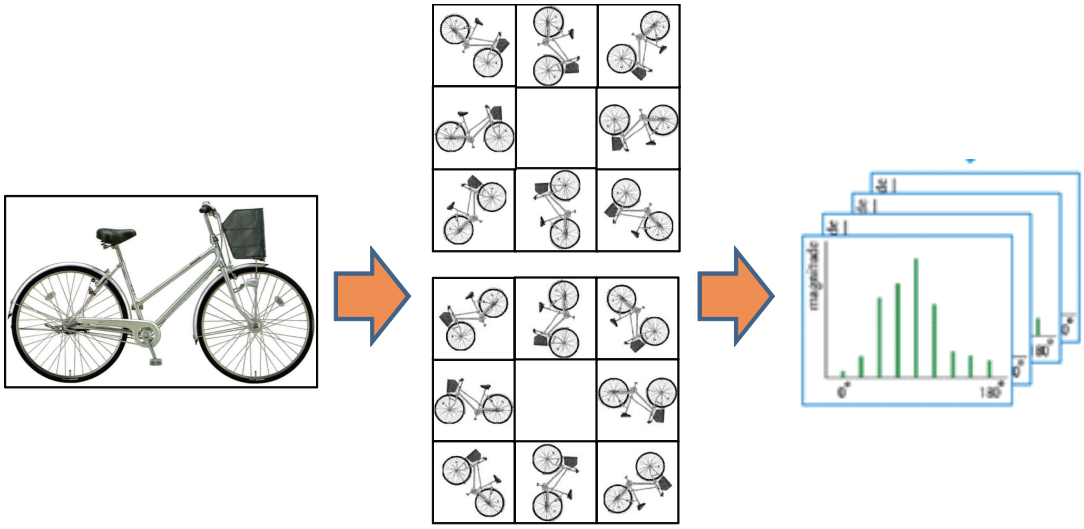

Fig. 9. Rotational sensitivity of HOG can be alleviated by having rotated images in advance as well as their mirror images

A. Evaluation Measures

The evaluation measures we selected include Recall, Precision, First Tier (1-Tier), Second Tier (2-Tier), and Nearest Neighbor (NN) [2], [30]. Let $rel(x)$ be the number of objects that are *relevant* among the top x rankings, let K be the number of closest matches returned, and let C be the number of objects in the category belonging to a query. Then, the evaluation measures are given by the following formula:

$$\text{Recall} = \frac{rel(K)}{C}$$

$$\text{Precision} = \frac{rel(K)}{K}$$

$$\text{First Tier} = \frac{rel(C-1)}{C-1}$$

$$\text{Second Tier} = \frac{rel(2(C-1))}{C-1}$$

$$\text{Nearest Neighbor} = rel(1)$$

Micro-averaged values [31] are calculated by constructing a global contingency table (where table has category (row) by category (column) size) and then calculating evaluation measures using these sums. We avoided macro-average, since macro-averaged scores are calculated by first computing evaluation measures for each category and then using their average. Some of the categories in PSB have only a few shape models, which makes macro-average have a large variance across different categories.

B. Overall Results

The averaged comparison result of our proposed method against the "baseline" method of Ansary et al [1] is shown in Fig. 11. Their ranking results can be verified by FOX-MIIRE 3D-Model Search Engine [32].

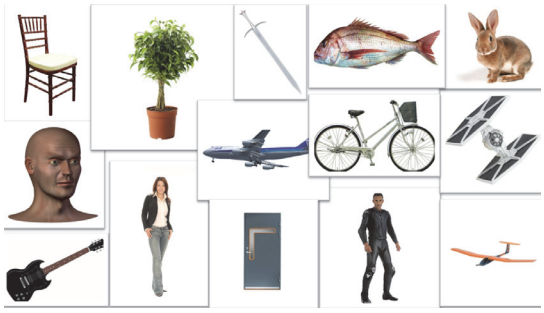

Fig. 10. Sample 2D input query images crawled from the Internet

Table I summarizes the average behavior of other evaluation measures. Our proposed method outperforms the baseline method by 11.2% in First Tier (1-Tier), 9.1% in Second Tier (2-Tier), and 26.2% in Nearest Neighbor (NN). This means that our idea of the composite features with Zernike moments and HOG (or PCA-HOG) are effective with respect to PSB data set, compared with Ansary et al's method. Table II shows the experimental results of the dimensionality reduction of HOG by PCA (PCA-HOG), against a plain HOG. PCA-HOG turns out to be better in 1-Tier and NN. The dimension of the original HOG was 15,925 (15,876 for HOG itself plus 49 for Zernike), while the dimension of PCA-HOG was 3,536. The reduction ratio is approximately 78%. As mentioned previously, the dimension of PCA-HOG was determined by the cumulative proportion of principal components becoming over 90%, corresponding to the largest k eigenvalues of HOG feature vectors.

C. Feature Statistics

Since our proposed features consist of HOG and Zernike moments, it is only natural to elucidate which one is more effective if we separately apply the two features to 3D shape

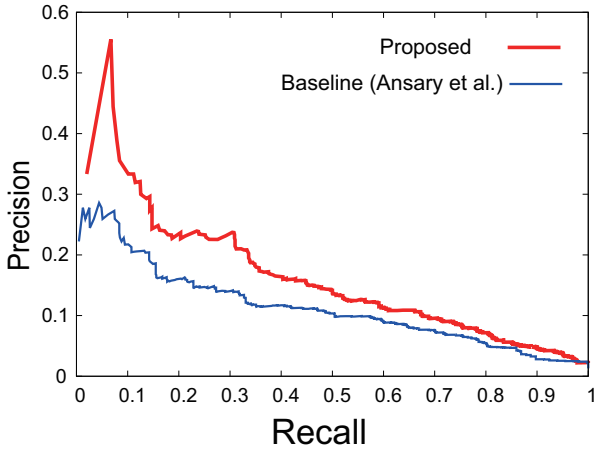

Fig. 11. Average recall-precision graph of our proposed method and baseline method of Ansary et al [1]

TABLE I
COMPARISON OF OUR PROPOSED METHOD AND BASELINE METHOD IN TERMS OF 1-TIER, 2-TIER, AND NN

Method	1-Tier (%)	2-Tier (%)	NN (%)
Baseline	14.3%	24.2%	16.7%
Proposed	**25.5**%	**33.1**%	**42.9**%

retrieval. Fig. 12 shows the recall-precision graph of three results; proposed composite approach, HOG-only approach, and Zernike-only approach. Obviously, HOG is more effective than Zernike moments. However, it should be noted that this is accomplished at the cost of consuming much more dimensions than Zernike moments.

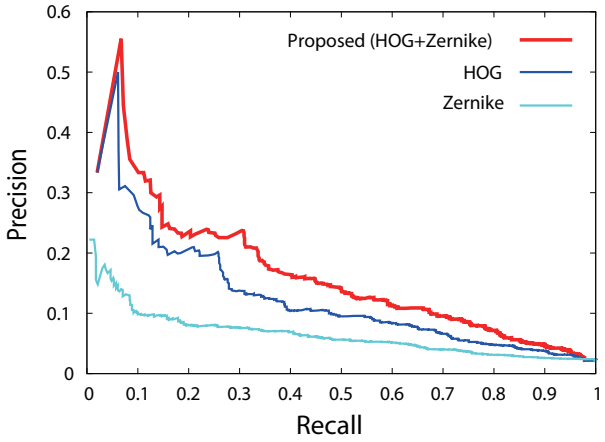

Fig. 12. Recall-precision graph comparing our proposed, HOG-only, and Zernike-only approaches

TABLE II
COMPARISON OF HOG AND PCA-HOG IN TERMS OF 1-TIER, 2-TIER, NN, AND DIMENSIONS

Method	1-Tier (%)	2-Tier (%)	NN (%)	Dim.
HOG	23.1%	**37.5**%	41.3%	15,925
PCA-HOG	**25.5**%	33.1%	**42.9**%	**3,536**

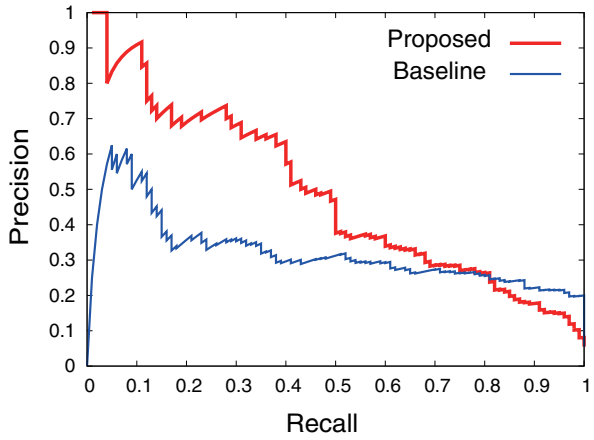

Fig. 13. Recall-precision graph of *human* category with respect to our proposed method and baseline method

D. Selected Category Comparison

Fig. 13 shows an averaged recall-precision graph for *human* category in PSB, given a collection of 2D human photo images such as those included in Fig. 10. As shown in Fig. 14, the top 5 ranking results reveal that the baseline method lists three completely different objects, while our proposed method lists human shapes in the first four out of five. We investigated why this behavior occurred in the baseline method, and conjectured that something as depicted in Fig. 15 might have occurred. Specifically, the baseline method relied on the similarity of two objects from silhouette images, while our proposed method took advantage of both silhouette and shaded depth-buffer images, which we consider makes for the remarkable difference.

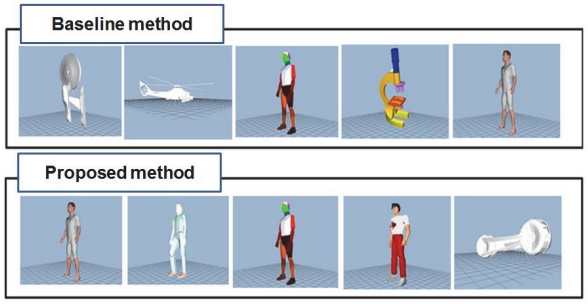

Fig. 14. Top 5 ranked search results of baseline and our proposed method

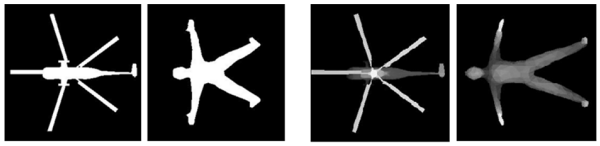

Fig. 15. One reason why baseline method fails to have good top 5 results. The helicopter silhouette looks similar to the silhouette of a man with his hands wide open, while our proposed method use shaded depth-buffer images, which make a difference between the two features.

VII. Conclusion

We proposed a new composite feature definition for 3D shape retrieval from a single 2D query photo image. Our feature was composed of a weighted combination of Zernike moments from silhouette images and HOG (Histogram of Oriented Gradients) features from shaded depth-buffer images. Given the 3D shape database, in advance we computed these two features separately using multiple projections and incorporated them into our system. Then, given a 2D snapshot photo, we assumed it as a single projection of a 3D shape and computed its Zernike moments and HOG features. Finally, we proposed the method of combining the two features into a composite feature vector and computed the similarity (or dissimilarity) between the feature and those kept in our 3D shape database. Through our comparative experiments, we demonstrated that our proposed method outperformed the previous method by Ansary et al.

In the future, we plan to investigate more about the optimal features for 3D shape retrieval from a collection of 2D query images.

Acknowledgment

This research was conducted by the Strategic Information and Communication R&D Promotion Programme (SCOPE 112306001) of the Ministry of Internal Affairs and Communications (MIC) Japan.

References

[1] T. F. Ansary, J.-P. Vanderborre, and M. Daoudi, "3D-Model Search Engine from Photos," *ACM International Conference on Image and Video Retrieval (CIVR)*, pp. 89–92, 2007.

[2] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," in *SMI '04 Proceedings of the Shape Modeling International*, pp. 167–178, 2004.

[3] B. Bustos, D. Keim, D. Saupe, and T. Schreck, "Content-Based 3D Object Retrieval," *IEEE Computer Graphics and Applications*, vol. 27, no. 4, pp. 22–27, 2007.

[4] D.-Y. Chen, *Three-Dimensional Model Shape Description and Retrieval Basde on LightField Descriptors*. PhD thesis, National Taiwan University, June 2003.

[5] D.-Y. Chen, X.-P. Tian, Y.-T. Chen, and M. Ouhyoung, "On Visual Similarity Based 3D Model Retrieval," *Computer Graphics Forum*, vol. 22, no. 3, pp. 223–232, 2003.

[6] M. Novotni and R. Klein, "3D Zernike Descriptors for Content Based Shape Retrieval," in *Proc. SM '03 Proceedings of the eighth ACM symposium on Solid modeling and applications*, pp. 216–225, 2003.

[7] S. Mahmoudi and M. Daoudi, "3D Models Retrieval by Using Characteristic Views," in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR '02)*, pp. 457–460, 2002.

[8] R. Ohbuchi, M. Nakazawa, and T. Takei, "Retrieving 3D Shapes based on their Appearance," in *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR '03)*, pp. 39–45, 2003.

[9] U. Lille, "FOX-MIIRE 3D Retrieval System." http://www-rech.enic.fr/3dretrieval/.

[10] T. F. Ansary, M. Daoudi, and A. Vandeborre, "A Bayesian 3D Search Engine using Adaptive Views Clustering," *IEEE Transactions on Multimedia*, vol. 9, no. 1, pp. 78–88, 2007.

[11] D. Pelleg and A. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 727–734, 2000.

[12] T. Napoléon and I. Sahbi, "From 2D Silhouettes to 3D Object Retrieval: Contributions and Benchmarking," *EURASIP Journal of Image and Video Processing*, vol. 2010, p. 17, 2010.

[13] A. Tatsuma and M. Aono, "Multi-Fourier spectra descriptor and augmentation with spectral clustering for 3D shape retrieval," *Visual Computer*, vol. 25, no. 8, pp. 785–804, 2009.

[14] R. Ohbuchi, T. Otagiri, M. Ibato, and T. Yakei, "Shape-Similarity Search of Three-Dimensional Models Using Parameterized Statistics," in *Proc. Pacific Graphics 2002*, pp. 265–274, 2002.

[15] J. R. D. V. Vranić, D. Saupe, "Tools for 3D-object retrieval: Karhunen-Loeve transform and spherical harmonics," in *Proc. IEEE 4th Workshop on Multimedia Signal Processing*, pp. 293–298, 2001.

[16] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape Distributions," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 807–832, 2002.

[17] P. Bratley, B. L. Fox, and H. Niederreiter, "Algorithm 738: Programs to Generate Niederreiter's Low-discrepancy Sequences," *ACM Transactions on Mathematical Software*, vol. 20, no. 4, pp. 494–495, 1994.

[18] F. Zernike, "Beugungstheorie des Schneidenverfahrens und seiner verbesserten Form, der Phasenkontrastmethode," *Physica*, vol. 1, pp. 689–704, 1934.

[19] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPRf05)*, pp. 886–893, 2005.

[20] G. Amayeh, A. Erol, G. Bebis, and M. Nicolesacu, "Accurate and efficient computation of high order zernike moments," in *ISVC'05 Proceedings of the First international conference on Advances in Visual Computing (Lecture Notes in Computer Science 3804)*, pp. 462–469, 2005.

[21] D. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proc. of the International Conference on Computer Vision*, pp. 1150–1157, 1999.

[22] D. Lowe, "Distinctive Image Features from Scale-Invariant Key Points," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[24] C. Dance, J. Willamowski, L. Fan, C. Bray, and C. Csurka, "Visual categorization with bags of keypoints," in *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[25] Y. Yang and S. Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," in *ACM GIS f10*, Nov. 2010.

[26] Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors (PDF)," in *IEEE CVPR '04*, pp. 506–513, 2004.

[27] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Springer, 2007.

[28] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer, 2009.

[29] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranić, "Automatic Selection and Combination of Descriptors for Effective 3D Similarity Search," in *Proc. IEEE MCBAR'04*, pp. 514–521, 2004.

[30] S. Y. Tang, "An Evaluation of Local Shape Descriptors for 3D Shape Retrieval," 2012. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909219.

[31] P. Min, *A 3D Model Search Engine*. PhD thesis, Princeton University, Jan. 2004.

[32] T. Lille1, "FOX-MIIRE 3D-Model Search Engine." http://www-rech.telecom-lille1.eu:8080/3Dretrieval/.