CCIT Report #475 March 2004

Mesh Retrieval by Components

Ayellet Tal Emanuel Zuckerberger
Department of Electrical Engineering
Technion
Email: ayellet@ee.technion.ac.il

***Abstract*— We describe an approach for retrieving three-dimensional objects similar to a given one from a database. The key idea of our technique is to decompose each object into its meaningful components, and fit each component to a basic shape. This decomposition is represented as an attributed graph, which is considered the *signature* of the object. Our signature leverages human vision theories such as Marr's and Biederman's. We show that this signature gives rise to a retrieval algorithm which is invariant to non-rigid transformations. Finally, a system which realizes our technique was built and tested on a database of about 400 objects. The paper presents the retrieval results and conclusions are being drawn.**

I. Introduction

Given a database of objects and a specific object, our goal is to retrieve from the database objects similar in shape to the specific one. We assume that the objects are given as polygonal meshes, the most common representation in computer graphics applications. Though the problem has been extensively investigated in the context of images [27], [1] and polygonal curves [2], [13], [3] it is a relatively new research topic for meshes [11], [23], [12], [24], [29].

A common practice is to represent each object by a few properties – a *signature* – and base the retrieval on the similarity of the signatures. Various signatures have been proposed in the literature. Some signatures consist of local properties of the shapes, but not their global structures. For instance, in [24], histograms of properties such as colors and normals are considered while probability shape distributions are discussed in [23]. Another alternative is to voxelize the given mesh and use a spherical harmonic representation [15].

Other papers consider global properties, such as a shape moments signature [11] or a *sphere projection* signature which computes the amount of "energy" required to deform an object into a sphere[18]. In these cases the objects need to be normalized ahead of time.

Our goal is to compare the global structures of the meshes. In [12], it is proposed to use a multiresolutional Reeb graph (MRG) as a signature. In general, the Reeb graph is a skeleton determined using a scalar function. In particular – geodesic distances are used in [12]. We too, propose to represent an object by a graph. However, our graph follows the footsteps of human visual perception theories such as Marr's [21] and Biederman's [6]. Practically, these approaches lead to very small graphs which are advantageous both computationally and storage-wise.

Marr [21] claims that the human brain constructs a three-dimensional viewpoint-independent model of the image seen. This model consists of objects and spatial inter-relations between them. Every three-dimensional object is segmented into primitives, which can be well approximated by a few simple shapes. Biederman's Recognition-By-Components (RBC) theory [4], [5] claims that the human visual system tends to segment complex objects at regions of deep concavities into simple basic shapes, *geons*. The simple attributed shapes along with the relations between them form a stable three-dimensional mental representation of an objects.

Our approach attempts to succeed these theories. The key idea is to decompose each object into its meaningful component and to match each component to a basic shape. After determining the relations between these components, an attributed graph representing the decomposition is constructed and considered the object's *signature*. Given a database of signatures and one specific signature, this signature is compared to other signatures in the database, and the most similar objects are retrieved.

Computationally, constructing this signature for a mesh in three dimensions should be easier than doing so for its projection into an image. After all, the whole object can be "seen", and problems like occlusion, self-occlusion, lighting effects and reflections, are avoided. Thus both segmentation and basic shape matching are facilitated.

Another important benefit of the proposed signature is its invariance to non-rigid-transformations. For instance, given a human object, we expect its signature to be similar to signatures of other human objects whether they bend, fold their legs or point forward. Figure 1 illustrates this as well as the results of our experiments. In this figure, the most similar objects to the human test object at the upper left corner were retrieved. All the 19 humans in a database consisting of 388 objects, were ranked among the top 21 objects, and 17 among the top 17. Invariance to non-rigid-transformations is hard to achieve when only the geometry of an object is considered.

An additional advantage of the proposed signature is being compact. Thus, signatures can be easily stored even for large databases and transfered between databases.

The remaining of the paper is structured as follows. Section II outlines our approach. Sections III– IV address the main issues involved in the construction of a signatures. In particular, Section III discusses mesh decomposition into meaningful components while Section IV describes the determination of basic shapes. Section V presents our experimental results. Section VI concludes and discusses future research directions.

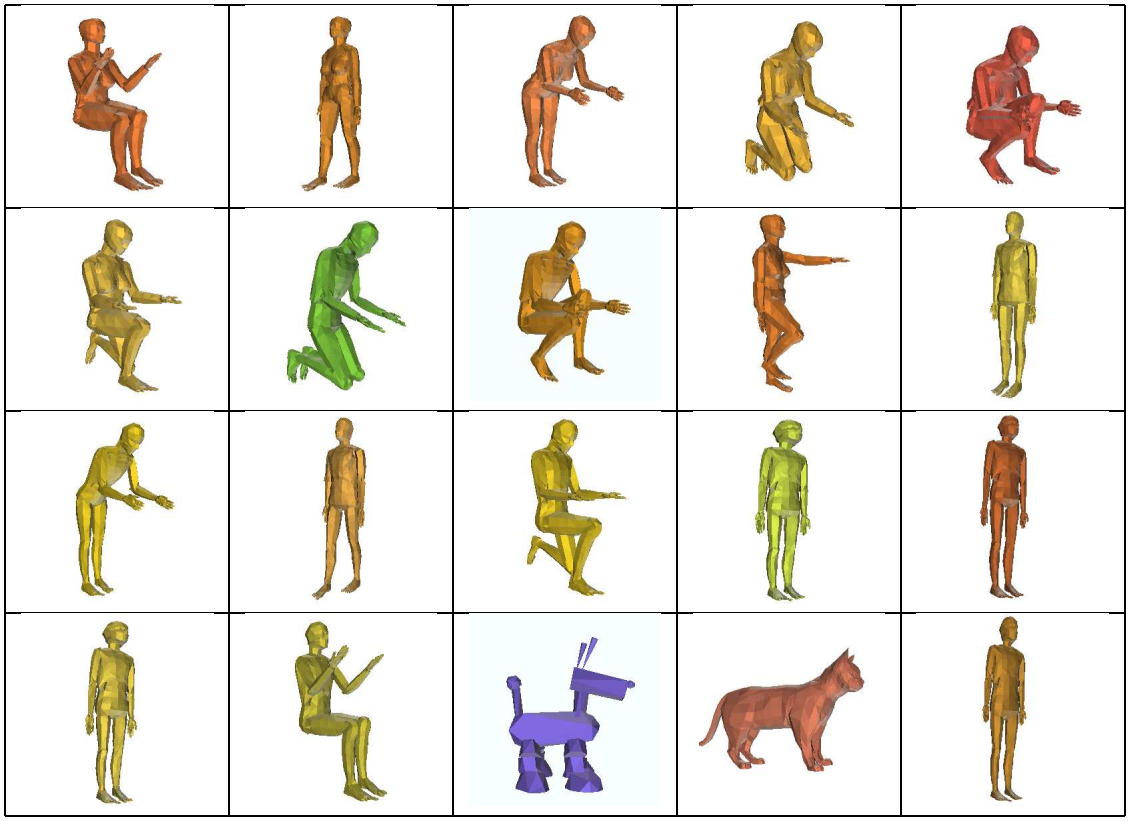


Fig. 1. Retrieval of top 20 objects similar to to the top left-most human figure

II. System Overview

Given a database of meshes in a standard representation consisting of vertices and faces (e.g., VRML), and one specific object O, the goal is to retrieve from the database objects similar to O. This section outlines our technique.

We start with a couple of definitions. Let S be an orientable mesh. It need neither be triangulated nor closed or a 2-manifold.

Definition 2.1: **Decomposition:** $S_1, S_2, \ldots S_k$ is a decomposition of S iff (i) $\forall i, 1 \leq i \leq k$, $S_i \subseteq S$, (ii) $\forall i$, S_i is connected, (iii) $\forall i \neq j, 1 \leq i, j \leq k$, S_i and S_j are face-wise disjoint and (iv) $\cup_{i=1}^{k} S_i = S$.

Definition 2.2: **Decomposition graph:** Given a decomposition $S_1, S_2, \cdots S_k$ of a mesh S, a graph $G(V, E)$ is its corresponding decomposition graph iff each component S_i is represented by a node $v_i \in V$ and there is an arc between two nodes in the graph iff the two corresponding components share an edge in S.

In an off-line stage, *signatures* are computed for each object in the database and stored. This is done in three steps. First, the object is decomposed into a small number of meaningful components. Next, each component is classified as a basic shape: a spherical surface, a cylindrical surface, a cone surface or a planar surface. Finally, a *signature* – an attributed decomposition graph – is constructed, such that the attributes were chosen according to [4], [5]. Each node is associated with an attribute – the basic shape of the corresponding component. Moreover, each arc is attributed by the relative surface area of its endpoint components (i.e., greater, smaller, equal). We elaborate on mesh decomposition and shape determination in the next sections.

In an online interactive step, the user specifies an object (which might or might not belong to the database), and the system retrieved from the database the most similar objects to this object. This step requires the comparison of graphs.

Graph matching and subgraph isomorphism has been applied to many problems in computer vision and pattern recognition e.g., [19], [26], [32], [17], [25], [9], [33]. In our work we use the work presented in [22] which uses error-correcting subgraph isomorphism. However, any other existing system for sub-graph isomorphism can be used as well.

The key idea is to define a graph edit operation for each possible error type. Possible operations are deletion, insertion and substitution of nodes and arcs (i.e., changing attributes). Application–dependent cost functions are associated with each edit operation. Given a couple graphs, a sequence of edit operations with minimal cost is sought, such that applying the sequence to one of the graphs results in a subgraph isomorphism with the other.

Formally, we are given two graphs, $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$, where V (V') is the set of nodes of G (G'), E (E') is its set of arcs, μ (μ') is a function which assigns attributes to nodes and ν (ν') is a function which assigns attributes to arcs. We are also given a set of graph edit

operations and their corresponding cost functions. The goal is to find the optimal error-correcting subgraph isomorphism (Δ, g), where Δ is a sequence of edit operations and g is an isomorphism, such that there is a subgraph isomorphism g from $\Delta(G)$ to G' and the cost $C(\Delta)$ of Δ is minimal.

The algorithm that finds (Δ, g) maintains a search tree. The root of the search tree contains an empty mapping and is associated with cost 0. At the next level of the search tree, the first node of G is mapped onto nodes in G'. Each such mapping, along with its corresponding cost of the relevant edit operation, is a node in the search tree. The generation of the next nodes is then guided by the cost of the edit operations. The node representing the mapping with the lowest cost in the current search tree is explored by mapping a new node of G onto every node of G' that has not yet been used in the path and the corresponding costs are calculated.

When the first mapping γ' describing a complete subgraph isomorphism from G to G' is found, a threshold parameter is set to the cost $C(\gamma')$ of γ'. A node having a cost greater than the threshold is never explored. Other nodes are explored until a mapping with the minimal cost is found that represents a subgraph isomorphism from the complete graph G to G'.

This procedure is applied to the graph representing the query object against each graph in the database. It returns a corresponding error value for each pair. The lower the error, the less edit operations are required (or the "cheaper" these operations are), and thus the more similar the objects are. The objects are therefore retrieved in a ascending order of their error values.

III. Mesh Decomposition

The first issue is how to decompose a mesh into its meaningful components. One option is to decompose it into solids, as common in computational geometry [8]. There are a few drawbacks to this approach. First, it is applicable only to two-manifolds, whereas most of the existing models found on the Internet are not. Second, programming solid decomposition algorithms is still considered challenging. Finally, solid decomposition might suffer a quadratic blowup, which makes it impractical for large models and in particular to our application.

The other option is to decompose only the surfaces of the objects. This is the approach we pursue in this paper.

Since "*the human visual system tends to segment complex objects at regions of deep concavities into simple basic shape*" [4], [5], convexity should obviously be taken into account.

In [14] it is proposed to consider convexity as well as geodesic distances for determining a decomposition. Though the algorithm indeed finds the meaningful components, it is too expensive for our application which requires the decomposition of large databases. Since a rough decomposition suffices in our cases, we can trade-off the quality of the decomposition for the speed of producing it.

In [7] the problem of decomposing a mesh into convex sub-meshes is discussed, where a sub-mesh is called *convex* if it lies entirely on the boundary of its convex hull. It is proved that the optimization problem is NP-complete. Nevertheless, linear greedy flooding heuristics are used for generating convex decompositions. These heuristics work on the dual graph H of mesh S, where nodes represents facets and arcs join nodes associated with adjacent facets. The class of greedy flooding heuristics refers to the strategy of starting from some node in H and traversing H, collecting nodes along the way as long as the associated facets form a convex sub-mesh. When no adjacent nodes can be added to the current component, a new component is started and the traversal resumes.

Another linear decomposition algorithm proposed in the literature is the *Watershed decomposition algorithm* [28], [20] which decomposes a mesh into *catchment basins*, or $watersheds$. Let h, $h : V \rightarrow R$ be a discrete height function defined over V, the set of vertices of S. A $watershed$ is a subset of V, consisting of vertices whose path of steepest descent terminates in the same local minimum of h.

The key idea of the Watershed decomposition algorithm is to let vertices descend until a labeled region is encountered, where all the minima are labeled as a first step. A post-processing merging step is needed in order to avoid over segmentation.

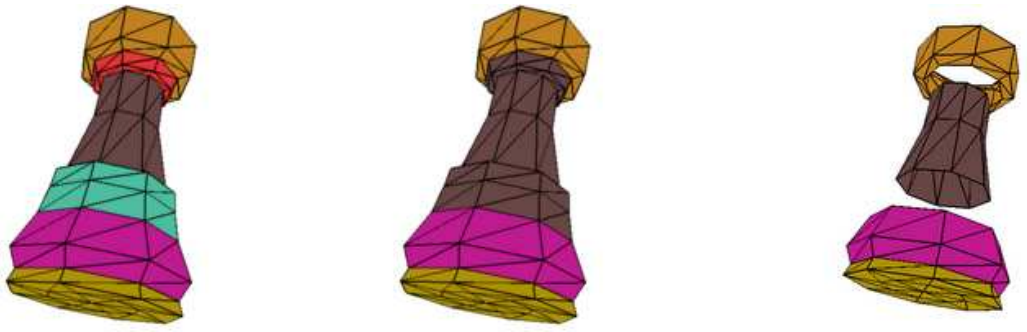
As noted in [34], one problem of the watershed algorithm regards the type of components generated by it [20]. The algorithm might produce different decompositions for input models having similar geometries but different triangulations. This is highly undesirable for retrieval applications. Moreover, even planar regions might be decomposed, which is often unacceptable. These problems arise from the definition of the height function as an approximation of the curvature defined on the vertices (or edges) of the mesh, which causes the value of the curvature to depend on the exact triangulation.

To solve this problem it is proposed to define a new height function which does not depend on a curvature estimation, but rather on the edges of the mesh: $h(edge) = 1 - \cos\theta$ where θ is the dihedral angle of the edge. Since this height function does not depend on the degree of the vertices, planar sub-meshes are not decomposed.

Yet, we still face the major problem with both the watershed decomposition and the convex decomposition – over-segmentation (i.e., obtaining a large number of components). The reason for over-segmentation is the fact that most large objects have many small concavities, giving rise to many components. The goal of our application, however, is to obtain only a handful of components.

To solve over-segmentation, it is proposed in [20] to merge regions whose watershed depth is below a certain threshold. Other, more general solutions, are studied in [34] and we follow them here.

The first solution is to merge components based on their surface areas. The intuition is that small components are less vital to recognition [4]. Let $Area(S)$ be the surface area of mesh S. If the surface area of a component is less than $qArea(S)$ and the sum of the surface areas of the remaining components is greater than $pArea(S)$ (where p and q are



(a) Convex Decomposition 7 components

(b) Convex Decomposition, merging 4 components

(c) Convex Decomposition , deleting 4 components

(d) Watershed 12 components

(e) Watershed, merging 5 components

(f) Watershed, deleting 5 components

Fig. 2. Decompositions of a rook

parameters), this component is merged with a neighboring component having the largest surface area. This process is done in ascending order of surface areas and continues until all the small components become sufficiently large.

The drawback of merging is that it might result with undesirable shapes. In other words, even though the initial components were determined by convexity, after the components have been merged, their shape might turn out to be complex and thus will not fit any basic shape.

To overcome this drawback, the other solution, which our experiments have indicated to be the best, is to ignore the small components altogether. Only the original large components are taken into account both in the construction of the decomposition graph and in determining the components' basic shapes. The small components are used only to determine the neighboring relations between the large components. Our experiments, which will be discussed in Section V, support Biederman's observation that "*recognition can be fast and accurate*" even if "*only two or three geons of a complex object are visible*" [6].

Figure 2 presents the results obtained by four variants of the general scheme, when applied to a *rook* model. Obviously, the number of components decreases when small components are merged or deleted. Moreover, as can be seen in Figures 2(c),2(f), even when the small components are deleted, there is still sufficient information to visually recognize the rook. Finally, Figures 2(e) demonstrates the drawback of merging – the red component does not resemble any basic shape.

In summary, the first step in constructing a signature of an object is to decompose it into a handful meaningful components. A decomposition algorithm should be efficient, since it should run on large databases which contain large objects. In this section we have addressed these issues by augmenting linear algorithms – the *watershed decomposition* and a *greedy convex decomposition* – with a post-processing step which either eliminates small components or merges them with their neighbors. We will show in the sequel how the various algorithms compare in the context of retrieval.

IV. Basic Shape Determination

The second issue in the construction of a signature is basic shape determination. Given a sub-mesh, which of the four basic shapes – a spherical surface, a cylindrical surface, a cone surface or a planar surface – better fits this component?

Our problem is related to the problem of fitting implicit polynomials to data and using polynomial invariants to recognize three-dimensional objects. In [31], a method based on minimizing the mean square distance of the data points to the surface is described. A first-order approximation of the real distance is used. In [16], a fourth-degree polynomial $f(x, y, z)$ is sought, such that the zero set of $f(x, y, z)$ is stably bounded and approximates the object's boundary. A probabilistic framework with an asymptotic Bayesian approximation is used in [30].

In order to fit a basic shape to a component, we first sample the given component. A non-linear least-squares optimization problem, which fits each basic shape to the set of sample points, is then solved. The approximate mean square distance from the sample points to each of the basic surfaces is minimized with respect to a few parameters specific for each basic shape. The basic shape with the minimal fitting error represents the shape attributes of the component. The algorithm for fitting the points to a surface is based on [31]. We formalize it below.

Let $f : R^n \rightarrow R^k$ be a smooth map, having continuous first and second derivatives at every point. The set of zeros of f, $Z(f) = \{Y | f(Y) = 0\}$, $Y \in R^n$ is defined by the implicit equations $f_1(Y) = 0, \cdots, f_k(Y) = 0$ where $f_i(Y)$ is the i-th component of f, $1 \leq i \leq k$.

The goal is to find the approximate distance from a point $X \in R^n$ to the set of zeros $Z(f)$ of f. In the linear case, the Jacobian matrix $Jf(X)$ of f with respect to X is a constant $Jf(X) = \mathbf{C}$, and $f(Y) = f(X) + \mathbf{C}(Y - X)$. The unique point $\hat{Y}$ that minimizes the distance $\|Y - X\|$, constrained by $f(Y) = 0$, is given by $\hat{Y} = X - \mathbf{C}^{\dagger} f(X)$, where $\mathbf{C}^{\dagger} = \mathbf{C^T}(\mathbf{C}\mathbf{C^T})^{-\mathbf{1}}$ is the pseudo-inverse [10]. If C is invertible then $\mathbf{C}^{\dagger} = \mathbf{C^{-1}}$. Finally, the square of the distance from X to $Z(f)$ is given by

$$dist(X, Z(f))^2 = \|\hat{Y} - X\|^2 = f(X)^T (\mathbf{C}\mathbf{C}^T)^{-1} f(X). \quad (1)$$

For the nonlinear case, Taubin [31] proposes to approximate the distance from X to $Z(f)$ with the distance from X to the set of zeros of a linear model of f at X, $\tilde{f} : R^n \rightarrow R^k$, where $\tilde{f}$ is defined by the truncated Taylor series expansion of f, $\tilde{f}(Y) = f(X) + Jf(X)(Y - X)$. But, $\tilde{f}(X) = f(X)$, $J\tilde{f}(X) = Jf(X)$, and the square of the approximated distance from a point $X \in R^n$ to the set of zeros $Z(f)$ of f is given by

$$dist(X, Z(f))^2 \approx f(X)^T (Jf(X) Jf(X)^T)^{-1} f(X). \quad (2)$$

Specifically, for the basic shapes we are considering, $n = 3$, $k = 1$, and the set of zeros $Z(f)$ of f is a surface in three-dimensions. The Jacobian $Jf(X)$ has only one row and $Jf(X) = (\nabla f(X))^T$, where $\nabla f(X)$ is the gradient of $f(X)$.

In this case, the approximated distance becomes

$$dist(X, Z(f))^2 \approx f(X)^2 / \|\nabla f(X)\|^2. \quad (3)$$

Moreover, we are interested in maps described by a finite number of parameters $(\alpha_1, \cdots, \alpha_r)$. Let $\phi : R^{n+r} \rightarrow R^k$ be a smooth function, and consider maps $f : R^n \rightarrow R^k$, which can be written as $f(X) \equiv \phi(\alpha, X)$, where $\alpha = (\alpha_1, \cdots, \alpha_r)^T$, $X = (X_1, , \cdots, X_n)$ and $\alpha_1, \cdots, \alpha_r$ are the parameters.

The approximated distance from X to $Z(\phi(\alpha, X))$ is then

$$\begin{aligned} &dist(X, Z(\phi(\alpha, X)))^2 = \delta_\phi(\alpha, X)^2 \quad (4) \\ &\approx \phi(\alpha, X)^T (J\phi(\alpha, X) J\phi(\alpha, X)^T)^{-1} \phi(\alpha, X). \end{aligned}$$

In particular, for three-dimensional space

$$\delta_\phi(\alpha, X)^2 \approx \phi(\alpha, X)^2 / \|\nabla \phi(\alpha, X)\|^2. \quad (5)$$

We can now formalize the fitting problem. Let $P = \{p_1, \cdots, p_m\}$ be a set of n-dimensional data points and $Z(\phi(\alpha, X))$ the set of zeros of the smooth function $\phi(\alpha, X)$. In order to fit P to $Z(\phi(\alpha, X))$ we need to minimize the approximated mean square distance $\Delta_P^2(\alpha)$ from P to $Z(\phi(\alpha, X))$:

$$\Delta_P^2(\alpha) = \frac{1}{m} \sum_{i=1}^{m} \delta_\phi(\alpha, p_i)^2 \quad (6)$$

with respect to the unknown parameters $\alpha = (\alpha_1, \cdots, \alpha_r)^T$.

This is equivalent to minimizing the length of the residual vector $Q = (Q_1, \cdots, Q_m)^T$

$$\|Q(\alpha)\|^2 = \sum_{i=1}^{m} Q_i(\alpha)^2 = m \Delta_P^2(\alpha) \quad (7)$$

where $Q_i(\alpha) = \delta_\phi(\alpha, p_i)$, $i = 1, \cdots, m$.

The Levenberg-Marquardt algorithm can be used to solve this nonlinear least squares problem. This algorithm iterates the following step

$$\alpha^{n+1} = \alpha^n - (JQ(\alpha^n) JQ(\alpha^n)^T + \mu_n I_m)^{-1} JQ(\alpha^n)^T Q(\alpha^n),$$

where $JQ(\alpha)$ is the Jacobian of Q with respect to α: $J_{ij} Q(\alpha) = \frac{\partial Q_i}{\partial \alpha_j}(\alpha)$, for $i = 1, \cdots, m$, and $j = 1, \cdots, r$, and μ_n is a small nonnegative constant which makes the matrix $JQ(\alpha^n) JQ(\alpha^n)^T + \mu_n I_m$ positive defined.

At each iteration, the algorithm reduces the length of the residual vector, converging to a local minimum.

THE DISTANCE OF A 3D POINT TO THE BASIC SHAPES:
We can now explicitly define the square of the distance $\delta_\phi(\alpha, X)$ from a three-dimensional point X to the set of zeros $Z(\phi(\alpha, X))$ of $\phi(\alpha, X)$ for our basic shapes, three of which are quadrics (i.e., sphere, cylinder, cone) and the fourth is linear (i.e., plane).

A quadric, in homogeneous coordinates, is given by $X^T M X = 0$ in the global coordinate system, where M is a 4×4 matrix and X is a vector in R^4. In its local coordinate system, it is given by $X'^T M' X' = 0$, where $X = T_r R_x R_y R_z S_c X'$, T_r is a translation matrix R_x, R_y, R_z are rotation matrices and S_c is a scale matrix.

If M' is known, M can be calculated and the equation of the quadric in the global coordinate system can be obtained.

$$\phi(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_x, s_y, s_z, X) = X^T M X = 0,$$

where the parameters are the translation, rotation and scale.

Then, for each basic quadric, the square of the approximated distance $\delta_\phi(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_x, s_y, s_z, X_p)$ from a three-dimensional point X_p to the quadric can be determined by

$$\begin{aligned} &\delta_\phi(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_x, s_y, s_z, X_p)^2 \approx \\ &\approx \frac{\phi(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_x, s_y, s_z, X_p)^2}{\|\nabla \phi(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_x, s_y, s_z, X_p)\|^2} = \\ &= \frac{\phi(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_x, s_y, s_z, X_p)^2}{(\frac{\partial \phi}{\partial x})^2 + (\frac{\partial \phi}{\partial y})^2 + (\frac{\partial \phi}{\partial z})^2} \end{aligned} \quad (8)$$

Hereafter we use the above equation to calculate δ_ϕ for each quadric basic shape, which are all special cases of the above.

For a spherical surface with radius $r_0 = 1$, defined in its local coordinate system centered at the center of the sphere, we have

$$M' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

$\phi(t_x, t_y, t_z, r, x, y, z) = (x-t_x)^2+(y-t_y)^2+(z-t_z)^2-r^2 = 0.$

For a cylindrical surface with radius $r_0 = 1$, defined in its local coordinate system, where the z axis is the axis of the cylinder,

$$M' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

The implicit equation in the global coordinate system is

$$\begin{aligned} &\phi(t_x, t_y, t_z, \theta_x, \theta_y, r, x, y, z) = \\ &= D_1(x-t_x)^2 + D_2(y-t_y)^2 + D_3(z-t_z)^2 + \\ &+2C_1(x-t_x)(y-t_y) + 2C_2(x-t_x)(z-t_z) + \\ &+2C_3(y-t_y)(z-t_z) - r^2 = 0 \end{aligned} \tag{9}$$

where

$$\begin{aligned} D_1 &= \cos^2\theta_y, \\ D_2 &= \cos^2\theta_x + \sin^2\theta_x \sin^2\theta_y, \\ D_3 &= \sin^2\theta_x + \cos^2\theta_x \sin^2\theta_y, \\ C_1 &= \sin\theta_x \sin\theta_y \cos\theta_y, \\ C_2 &= -\cos\theta_x \sin\theta_y \cos\theta_y, \\ C_3 &= \sin\theta_x \cos\theta_x \cos^2\theta_y, \\ B_1 &= -t_x D_1 - t_y C_1 - t_z C_2, \\ B_2 &= -t_x C_1 - t_y D_2 - t_z C_3, \\ B_3 &= -t_x C_2 - t_y C_3 - t_z D_3. \end{aligned}$$

Note that (t_x, t_y, t_z) can be any point on the cylinder axis, thus the cylinder is over parameterized. This can be solved by setting one of these three parameters to zero.

For of a cone surface with $g_0 = r_0/h_0 = 1$, where r_0 is the radius and h_0 is the height, defined in its local coordinate system, where the z axis is the axis of the cone and the origin of the coordinate system is the apex of the cone,

$$M' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The implicit equation in the global coordinate system is

$$\begin{aligned} &\phi(t_x, t_y, t_z, \theta_x, \theta_y, g, x, y, z) = \\ &= D_1(x-t_x)^2 + D_2(y-t_y)^2 + D_3(z-t_z)^2 + \\ &+2C_1(x-t_x)(y-t_y) + 2C_2(x-t_x)(z-t_z) + \\ &+2C_3(y-t_y)(z-t_z) = 0 \end{aligned} \tag{10}$$

where

$$\begin{aligned} D_1 &= \cos^2\theta_y - g^2\sin^2\theta_y, \\ D_2 &= \cos^2\theta_x + \sin^2\theta_x \sin^2\theta_y - g^2\sin^2\theta_x \cos^2\theta_y, \\ D_3 &= \sin^2\theta_x + \cos^2\theta_x \sin^2\theta_y - g^2\cos^2\theta_x \cos^2\theta_y, \\ C_1 &= (1+g^2)\sin\theta_x \sin\theta_y \cos\theta_y, \\ C_2 &= -(1+g^2)\cos\theta_x \sin\theta_y \cos\theta_y, \\ C_3 &= (1+g^2)\sin\theta_x \cos\theta_x \cos^2\theta_y, \\ B_1 &= -t_x D_1 - t_y C_1 - t_z C_2, \\ B_2 &= -t_x C_1 - t_y D_2 - t_z C_3, \\ B_3 &= -t_x C_2 - t_y C_3 - t_z D_3. \end{aligned}$$

Finally, a plane is defined by the equation $ax+by+cz+d = 0$. The square of the distance from a point $p = (x_p, y_p, z_p)$ to the plane is simply

$$\delta_\phi(a, b, c, d, x_p, y_p, z_p)^2 = \frac{(ax_p + by_p + cz_p + d)^2}{a^2 + b^2 + c^2}.$$

V. Experimental Results

We tested our retrieval algorithm on a database consisting of 388 objects which were downloaded from the World-Wide Web. Among the 388 objects we identified six classes: 19 models of human figures, 18 models of four-legged animals, 9 models of knives, 8 models of airplanes, 7 models of missiles and 7 models of bottles. The other models were not classified.

Four different decomposition heuristics were used in our experiments:

1) Greedy convex decomposition, where small patches are deleted;
2) Greedy convex decomposition, where small patches are merged with their neighbors;
3) Watershed decomposition, where small patches are deleted;
4) Watershed decomposition, where small patches are merged with their neighbors.

Based on these four decomposition heuristics, four signature databases were built. Identical retrieval experiments were applied to each database. In each experiment a test object was chosen and the system was queried to retrieve the most similar objects to this test object in ascending order. At least one member from each of the six classes was considered as a test object.

Figures 3– 6 demonstrate some of our results. In each figure the test object is the left-most, top object, and the objects retrieved are ranked from left to right. In particular, Figure 3 presents the most similar objects to *Detpl* (at the top-left), as retrieved by our algorithm. All the eight airplanes of the class were retrieved among the top eleven. Figure 4 presents the results of retrieving objects similar to *Cat2*. Sixteen out the eighteen members of the 4-legged animal class were retrieved among the top twenty. Figure 5 presents the retrieved most similar objects to *Knifech*. Eight out of the nine knifes of the class were retrieved among the top ten. Figure 6 demonstrates

Fig. 3. The most similar objects to *Detpl* (top left) as retrieved by the {Convex decomposition, delete small patches} sub-method

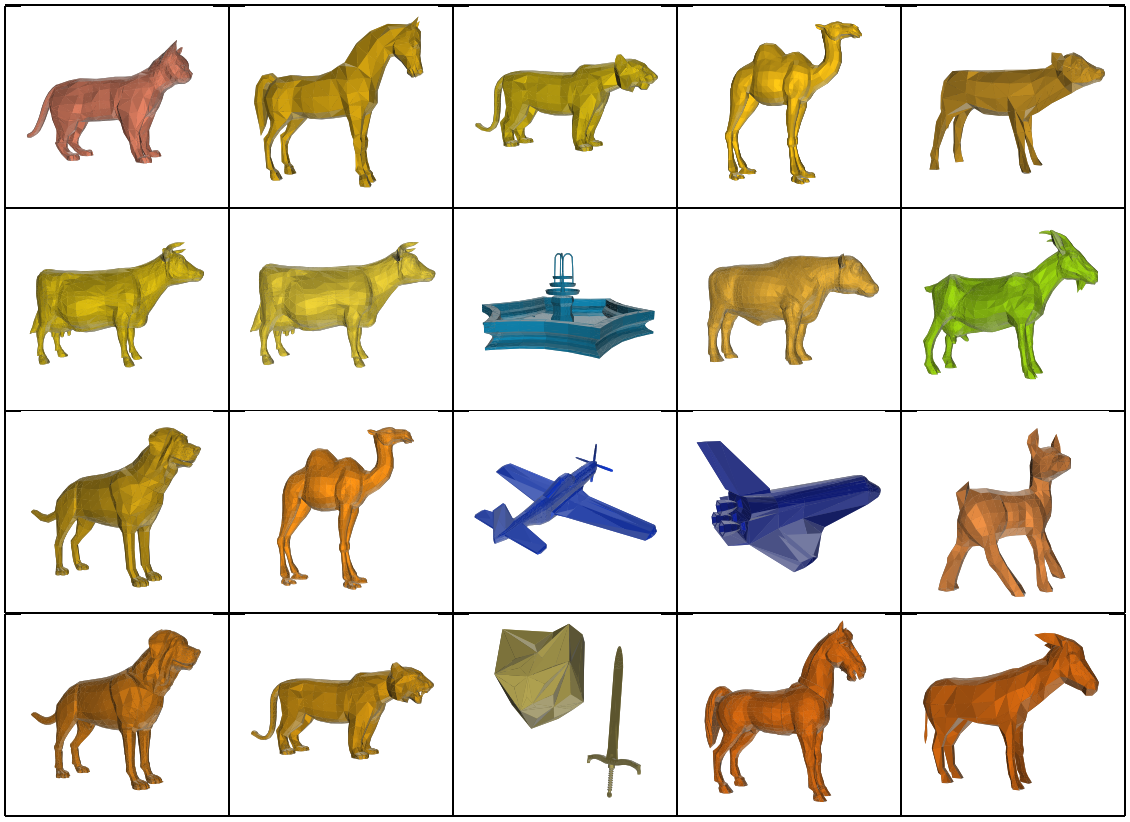


Fig. 4. The most similar objects to *Cat2* as retrieved by the {Watershed decomposition, merge small patches} sub-method

the most similar objects to the missile at the top left as retrieved by our algorithm. Six out of the the seven class members were retrieved among the top nine. Note that in all the above cases the members of each class differ geometrically. Yet, their decomposition graphs are similar and therefore they were found to be similar.

There is one class, however, where our algorithm does not perform as well, the class of bottles. This class contains seven members (see Figure 7). Though the objects seem similar geometrically, their connectivity differs. The *Beer*, *Ketchup* and *Tabasco* bottles consist each of 4-8 disconnected components while *Bottle3*, *Champagne*, *Whiskey* and *Plastbtl*

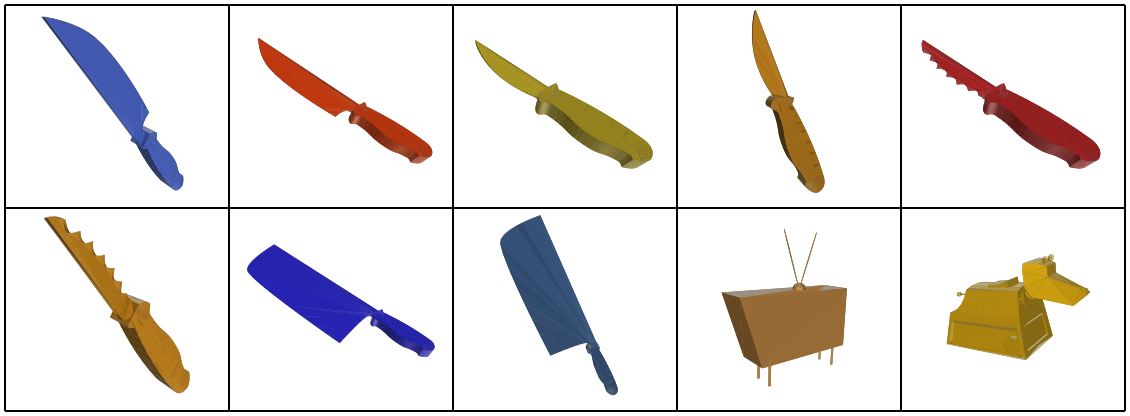


Fig. 5. The most similar objects to *Knifech* as retrieved by the {Watershed decomposition, delete small patches} sub-method

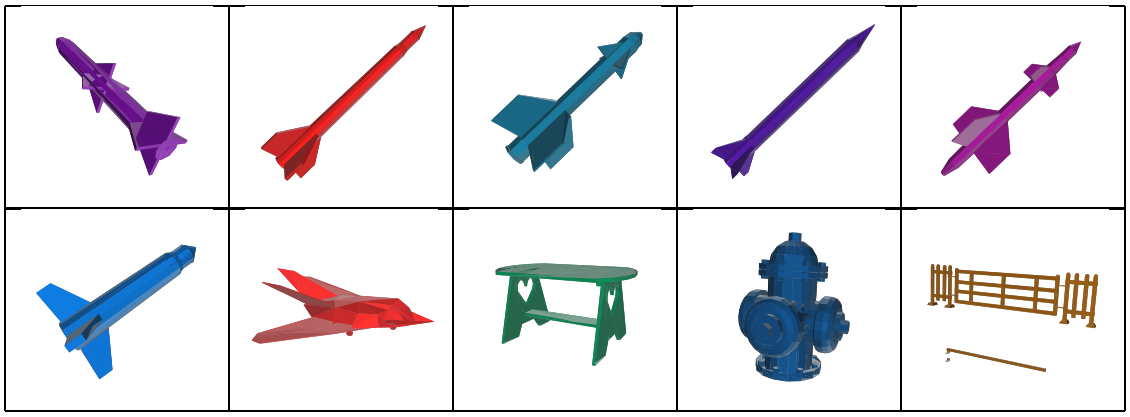


Fig. 6. The most similar objects to *Aram* as retrieved by the {Convex decomposition, merge small patches} sub-method

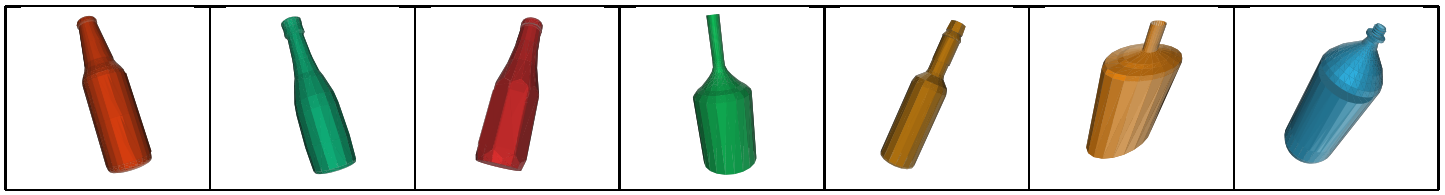


Fig. 7. The bottle class

consist each of only one or two components. Because the geometric similarity is not reflected in the structure of the objects, the results of the retrieval experiments are not as good, as can be seen in Table I.

Table I summarizes the results of our experiments. The first column shows the classes and the test objects. For each class, the number of members of the class N is shown. The next four columns of the table summarize the results obtained for each test object and for each sub-method. Each result (n/m) represents the number of the members of the same class n retrieved among the top m objects. The best result for each test object is emphasized. The best method for each test object and for each class is presented in the last column.

A couple of general conclusions can be drawn. First, the Watershed decomposition outperform convex decomposition. This fact might be surprising since convexity is the main factor in human segmentation. However, since the problem is hard and heuristics need to be applied, the resulting convex segmentation is usually not optimal. Moreover, the height function used in the Watershed algorithm considers convexity as well.

Second, considering only the original large components and deleting the small ones performs better than merging small components with their neighbors. This can be explained by the fact that merging results in complex shapes which might cause the failure of our basic shape determination procedure.

VI. CONCLUSION

We presented in this paper a new signature for shape-based retrieval of meshes. This signature is an attributed graph, where each node represents a meaningful component of the object, and there are arcs between nodes whose corresponding components are adjacent in the model. Every node is attributed with the basic shape found to best match the component while

Class(N)/ Object	Convex delete	Convex merge	Watershed delete	Watershed merge	best method
Airplanes(8)					**Watershed, delete**
Detplane	**5/6**	4/6	**5/6**	4/6	Watershed, delete
	6/9	4/9	**7/9**	5/9	Watershed, delete
	8/16	6/16	**8/16**	7/16	Watershed, delete
Worldw	3/6	1/6	**6/6**	5/6	Watershed, delete
	5/9	2/9	**6/9**	**6/9**	Watershed, delet/merge
	7/16	5/16	**8/16**	**8/16**	Watershed, delete/merge
747	4/6	3/6	**5/6**	4/6	Watershed, delete
	4/9	3/9	5/9	**6/9**	Watershed, merge
	6/16	4/16	**7/16**	**7/16**	Watershed, delete/merge
Animals(18)					**Watershed, delete/merge**
Cat2	6/8	**7/8**	6/8	**7/8**	Convex/Watershed, merge
	8/14	10/14	**11/14**	**11/14**	Watershed, delete/merge
	11/20	13/20	14/20	**16/20**	Watershed, merge
Tiger3	6/8	7/8	7/8	**8/8**	Watershed, merge
	9/14	10/14	**11/14**	10/14	Watershed, delete
	10/20	11/20	**13/20**	12/20	Watershed, delete
Deer	2/8	2/8	**8/8**	7/8	Watershed, delete
	5/14	6/14	**11/14**	10/14	Watershed, delete
	9/20	8/20	15/20	**15/20**	Watershed, merge
Humans(19)					**Watershed, delete**
Woman2	8/10	**10/10**	**10/10**	9/10	Watershed, delete & Convex, merge
	10/17	14/17	**17/17**	16/17	Watershed, delete
	16/24	18/24	**19/24**	18/24	Watershed, delete
Child3y	**10/10**	9/10	**10/10**	**10/10**	Watershed, delete/merge & Convex, delete
	15/17	15/17	15/17	**16/17**	Watershed, merge
	16/24	17/24	**19/24**	**19/24**	Watershed, delete/merge
Knives(9)					**Watershed, delete**
Knifech	3/6	2/6	**6/6**	**6/6**	Watershed, delete/merge
	3/8	4/8	**8/8**	7/8	Watershed, delete
	5/15	4/15	8/15	**9/15**	Watershed, merge
Knifest	5/6	4/6	**6/6**	5/6	Watershed, delete
	6/8	5/8	**6/8**	5/8	Watershed, delete
	8/15	6/15	**8/15**	**8/15**	Watershed, delete
Missiles(7)					**Convex, delete/merge**
Aram	**6/6**	**6/6**	3/6	3/6	Convex, delete/merge
	6/10	**6/10**	5/10	5/10	Convex, delete/merge
Bottles(7)					**Watershed, merge**
Beer	2/3	2/3	1/3	**3/3**	Watershed, merge
	3/6	**3/6**	1/6	**3/6**	Convex, delete/merge & Watershed, merge

TABLE I

SUMMARY OF THE EXPERIMENTAL RESULTS

each arc is attributed with the relative surface area of its adjacent nodes.

To construct this signature, two issues were addressed. First, we described a few alternatives for efficient decomposition algorithms. We used well-known algorithms and added a simple post-processing step in order to end up with only a handful of components. Second, we presented a technique for finding the best match between a given sub-mesh and pre-defined basic shapes.

Finally, we built a system that realizes our approach. A database consisting of 388 objects was collected from the WWW and six classes were classified within it. In our experiments, at least one object from each class was used as a test object.

The results, which are presented in the paper, are generally good. Not only does the algorithm retrieve objects that belong to the same class, but also it does so in the presence of non-rigid transformations. The paper also present a comparison of various sub-methods and draws conclusions.

A major benefit of our signature is being invariant to non-rigid transformations. Moreover, unlike some other signatures, normalization is not needed as a pre-processing step. In addition, the algorithm for generating signatures is simple and efficient. The signature is very compact and thus can be stored for large databases.

Our technique has a couple of drawbacks. First, the signa-

ture depends on the connectivity of the given objects, which might cause geometrically-similar objects to be considered different. Second, the graph matching algorithm we use is relatively slow.

In the future we intend to look at faster graph matching algorithms that people have implemented in other domains and adopt them. Moreover, we intend to add more attributes to the signature, such as textures and colors, which might be important for some retrieval applications.

REFERENCES

[1] IEEE Computer (1995). Special issue on content based image retrieval, 28, 9.

[2] H. Alt, B. Behrends and J. Blömer. *Approximate matching of polygonal shapes*, Proc. 7th Annu. ACM Sympos. Comput. Geom, 1991, 186–193

[3] H. Alt and M. Godau. *Measuring the resemblance of polygonal curves*, 8th ACM Symposium on Computational Geometry (1992), 102-109.

[4] I. Biederman. Recognition-by-components: A theory of human image understanding, Psychological Review 94, 115-147, 1987.

[5] I. Biederman. Aspects and extensions of a theory of human image understanding, In Pylyshyn Z. editor. *Computational Processes in Human Vision: An Interdisciplinary Perspective*, 370-428, New York: Ablex, 1988.

[6] I. Biederman. *Visual object recognition*, In S. Kosslyn, D. Osherson, editors. An Invitation to Cognitive Science, Vol. 2: *Visual Cognition*. MIT Press, 121-165, 1995.

[7] B. Chazelle, D.P. Dobkin, N. Shouraboura and A. Tal. Strategies for polyhedral surface decomposition: An experimental study, Computational Geometry: Theory and Applications, 7(4-5), 327-342, 1997.

[8] B. Chazelle and L. Palios. *Decomposition algorithms in geometry*, in Algebraic Geometry and its Applications, C. Bajaj, Ed., Chap.27, Springer-Verlag, pp. 419–447, 1994.

[9] W.J. Christmas, J. Kittler and M. Petrou. *Structural matching in computer vision using probabilistic relaxation*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, no. 8, 749-764, 1995.

[10] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*, section 9.2.1, 328-331, first edition, New-York, Wiley, 1973.

[11] M. Elad, A. Tal, S. Ar. *Content Based Retrieval of VRML Objects - An Iterative and Interactive Approach*, EG Multimedia, Sep. 2001, 97-108.

[12] M. Hilaga, Y. Shinagawa, T. Kohmura, and TL Kunii. *Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes*, SIGGRAPH, 203-212, 2001.

[13] D.P. Huttenlocher, K. Kedem and M. Sharir. *The upper envelope of Voronoi surfaces and its applications*, Discrete Computational Geometry, 9, (1993), 267-291.

[14] S. Katz and A. Tal. *Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts*, SIGGRAPH 2003, Volume 22 , Issue 3, July 2003, 954-961.

[15] M. Kazhdan, T. Funkhouser and S. Rusinkiewicz. *Rotation invariant spherical harmonic representation of 3D shape descriptors*, Symposium on Geometry Processing, Aachen, Germany, June, 2003.

[16] D. Keren, D. Cooper and J. Subrahmonia. *Describing complicated objects by implicit polynomials*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 1, 38-53, 1994.

[17] S.W. Lee, J.H. Kim and F.C.A. Groen. *Translation-, rotation-, and scale-invariant recognition of hand-drawn symbols in schematic diagrams*, Int. J. Pattern Recognition and Artificial Intelligence, vol. 4, no. 1, 1-15, 1990.

[18] G. Leifman, S. Katz, A. Tal and R. Meir. *Signatures of 3D Models for Retrieval*, The 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics, February 2003, 159–163 .

[19] S.W. Lu, Y. Ren and C.Y Suen. *Hierarchical attributed graph representation and recognition of handwritten chinese characters*, Pattern Recognition, vol. 24, 617-632, 1991.

[20] A.P. Mangan and R.T. Whitaker. *Partitioning 3D surface meshes using watershed segmentation*, IEEE Transactions on Visualization and Computer Graphics, Vol. 5 No. 4, 308-321, 1999.

[21] D. Marr. *Vision - A computational investigation into the human representation and processing of visual information*, first edition, W.H. Freeman, San Francisco, 1982.

[22] B.T. Messmer. *GMT - Graph Matching Toolkit*, PhD Thesis, University of Bern, 1995.

[23] R. Osada, T. Funkhouser, B. Chazelle, D. Dobkin, *Matching 3D Models with Shape Distributions*, International Conference on Shape Modeling and Applications, 154-166, 2001.

[24] E. Paquet, A. Murching, T. Naveen, A. Tabatabai and M. Rioux, *Description of Shape Information for 2-D and 3-D Objects*, Signal Processing: Image Communication: 103-122, 2000.

[25] A. Pearce, T. Caelli and W.F. Bischof. *Rulegraphs for graph matching in pattern recognition*, Pattern Recognition, vol. 27, no. 9, 1231-1246, 1994.

[26] J. Rocha and T. Pavlidis. *A shape analysis model with applications to a character recognition system*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 16, 393-404, 1994.

[27] Rui Y. and Huang T. S. and Chang S-F. Image Retrieval: Current Techniques, Promising Directions, and Open Issues. Journal of Visual Communication and Image Representation, 10, (1), 1999, pp. 39-62.

[28] J.C. Serra. *Image Analysis and Mathematical Morphology*, first edition, London: Academic Press, 1982.

[29] H.Y. Shum, M. Helbert, K. Ikeuchi, *On 3D Shape Similarity*, Proceedings of IEEE Comuter Vision and Pattern Recognition, 526-531, 1996.

[30] J. Subrahmonia, D.B. Cooper and D. Keren. *Practical reliable Bayesian recognition of 2D and 3D objects using implicit polynomials and algebraic invariants*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 5, 505-519, 1996.

[31] G. Taubin. *Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 11, 1115-1138, 1991.

[32] Y.-K. Wang, K.-C. Fan and J.-T Horng. *Genetic-based search for error-correcting graph isomorphism*, IEEE Trans. Systems, Man, and Cybernetics, vol. 27, 588-597, 1997.

[33] E.K. Wong. *Model matching in robot vision by subgraph isomorphism*, Pattern Recognition, vol. 25, no. 3, 287-304, 1992.

[34] E. Zuckerberger, A. Tal and S. Shlafman. *Polyhedral Surface Decomposition with Applications*, Computers & Graphics, 26(5), 2002, 733–743.