Pattern Anal Applic (2010) 13:451–468
DOI 10.1007/s10044-010-0179-5

TECHNICAL ADVANCES

3D object recognition using invariants of 2D projection curves

Mustafa Unel · Octavian Soldea · Erol Ozgur · Alp Bassa

Revised: 14 August 2009 / Accepted: 29 September 2009 / Published online: 22 May 2010
© Springer-Verlag London Limited 2010

Abstract This paper presents a new method for recognizing 3D objects based on the comparison of invariants of their 2D projection curves. We show that Euclidean equivalent 3D surfaces imply affine equivalent 2D projection curves that are obtained from the projection of cross-section curves of the surfaces onto the coordinate planes. Planes used to extract cross-section curves are chosen to be orthogonal to the principal axes of the defining surfaces. Projection curves are represented using implicit polynomial equations. Affine algebraic and geometric invariants of projection curves are constructed and compared under a variety of distance measures. Results are verified by several experiments with objects from different classes and within the same class.

Keywords Recognition · Algebraic surfaces · Implicit polynomials · Invariants · Principal axes

1 Introduction

Object recognition is a central problem in computer vision. Most object recognition systems can be classified into three categories [1] depending upon what is measured from the scene: (1) systems that use only intensity data, (2) systems that use only range data, and (3) systems that use both intensity and range data. Systems based on intensity data mostly use points and straight line segments as features. Systems based on range data use planar or quadric surfaces as the most common features, but points and line segments are also used. State-of-the-art approaches to object recognition are [2]:

- *Pose-consistency approaches*: these methods use geometric techniques to identify a sufficient number of matches between image and model features. These approaches include alignment techniques and affine and projective invariants. An advantage of invariant approach is that indexing can be done in sublinear time.
- *Template matchers*: these methods record a description of all images of each object. They have been used quite successfully in various tasks, such as face identification and 3D object recognition. Their great advantage is that, unlike purely geometric approaches, they exploit the great discriminatory power of image brightness/color information. However, they usually require a separate segmentation step that separates the objects from the background, and they are potentially sensitive to illumination changes.
- *Relational matchers*: these methods describe objects in terms of relations between templates. Usually, one looks for rather conventional image patches and then reasons about relations between them. There are two difficulties with this approach: first, some relational models are easy to match, but some can be difficult to match; second, current methods handle local patches (such as eye corners) and simple objects (such as faces) well, but it is quite challenging to build matchers that find, for example, animals based on relations between image patches.

M. Unel (✉) · O. Soldea · E. Ozgur · A. Bassa
Faculty of Engineering and Natural Sciences,
Sabanci University, Istanbul, Turkey
e-mail: munel@sabanciuniv.edu

O. Soldea
e-mail: octavian@sabanciuniv.edu

E. Ozgur
e-mail: erol@sabanciuniv.edu

A. Bassa
e-mail: bassa@sabanciuniv.edu

- *Aspect graphs*: these methods explicitly record the qualitative changes in object appearance due to changes in viewpoint. Recognition techniques based on aspect graphs lie somewhere between appearance-based and structural methods since they actually describe the appearance of an object by the evolution of its structure as a function of viewpoint. In practice, such approaches have not fulfilled their promise partly because the reliable extraction of contour features, such as terminations and T-junctions from real images is extremely difficult and partly because even relatively simple objects may have extremely complicated aspect graphs.

1.1 Summary of existing related work

Gonzalez et al. [3] recognized 3D objects by using Fourier descriptors for clustering the silhouettes of the object viewed from multiple viewpoints. Kim et al. [4] proposed a new scalable 3D object representation, which utilizes a common-frame constellation model (CFCM), and a fully automated learning method, appearance-based automatic feature clustering and sequential construction of clustered CFCMs, to recognize objects. Rothganger et al. [5] presented a novel representation for 3D objects by employing local affine-invariant descriptors of their images and defining the spatial relationships between the corresponding surface patches for recognition purpose. Lee et al. [6] implemented a fast 2-stage algorithm for recognizing 3D objects using a new feature space, built from curvature scale space images, and the matching part is realized in the eigenspaces of the feature space. Li et al. [7] integrated Algebraic Functions of Views (AFoVs) with indexing and learning methods in order to recognize 3D objects. Nagabhushan et al. [8] developed a new technique called two-dimensional principal component analysis (2D-PCA) for 3D object representation and recognition. Chen and Bhanu [9] introduced an integrated local surface descriptor, which is calculated only at the features points of the object surface for surface representation and 3D object recognition. The local surface descriptor is characterized by its centroid, its local surface type, and a 2D histogram. Mian et al. [10] performed model-based 3D object recognition via similarity measures by constructing a model of a 3D object from range images and defining tensors among their views.

Diplaros et al. [11] combined the color and shape invariants in a multidimensional color–shape context which is subsequently used as an index for discriminative purposes in object recognition. Shotton et al. [12] proposed a system which is based only on local contour features and uses a novel formulation of chamfer matching for an automatic visual recognition. Adan et al. [13] presented a strategy for 3D object recognition using a flexible similarity measure based on cone-curvature (CC) features which originates from the recent modeling wave (MW) concept. Shan et al. [14] proposed to represent each model object as a collection of shapeme histograms which enables recognition of partially observed query objects from a database of complete model objects by matching the query histogram. Several classification and recognition schemes are characterized by complex searches in exponential graphs of configurations, see for example [15, 16]. For some other older references on alignment and invariants based on moments, B-splines, superquadrics, conics, differential invariants, and Fourier descriptors, see [17]–[23].

1.2 An overview of our method

Algebraic curves/surfaces have proved to be very useful for shape representation [24–27]. Invariants associated with algebraic models have also been employed in several model-based vision and pattern recognition applications [29–33]. In the past few years, implicit representations have been used more frequently, allowing a better treatment of several problems. It is sometimes more convenient to have an implicit equation in applications, such as determining curve/surface intersections and the point classification problem, since they imply a simple evaluation of the implicit functions. Implicit polynomials (IP) are also well-suited for determining "how close" measured points on a curve/surface are to the ideal curve/surface, once the ideal surface is modeled with an algebraic equation [34].

This work develops new techniques for identifying and comparing 3D objects from their 2D projection curves. Motivation for this work comes from the fact that computing geometric invariants of 3D surfaces is not an easy task and algebraic invariants, usually obtained by tedious symbolic manipulations, are highly involved non-linear functions of the curve/surface coefficients [35], and therefore they are quite sensitive to data perturbations. This means that they cannot be used in robust object identification and comparison problems. On the other hand, there is a significant amount of work on efficient computational procedures for computing geometric and/or algebraic invariants of 2D curves [29, 30, 31]. Therefore, instead of comparing 3D algebraic surfaces using non-robust algebraic invariants, we propose to use robust algebraic/geometric invariants of 2D projection curves obtained from boundaries of objects under analysis. This way, a considerable speedup is also obtained since using invariants of curves instead of boundary surfaces is computationally much more efficient.

The proposed method begins with a range image or a tessellated representation of an object. Both of the versions compute the orientation of the analyzed object. When the input is a range image, our scheme fits an algebraic surface to the object. The eigenvectors of the second-order moment

matrix of the surface data are then computed. These eigenvectors imply three orthogonal directions in space along which surface data scatter most. They are the principal axes of the surface. Cross sections of the algebraic surface with planes orthogonal to its principal axes yield projection curves on the coordinate planes. When the input is a tessellated image, we compute the orientation of the analyzed object employing a quasi-convex hull of the boundary of the subject. The quasi-convex hull induces an orientation of the input object in terms of inertia axes, as defined by the moment of inertia tensor. Cross sections are obtained by intersecting planes perpendicular to axes of inertia and the tessellated model. Since cross sections are planar curves, we treat them as projection curves, i.e. consider reorienting objects such that inertia axes coincide with the coordinate axes.

We show that these projection curves are affine equivalent. We propose two methods for constructing both algebraic and geometric affine invariants of projection curves. For 3D object recognition, we employ an average similarity measure of the invariant vectors of certain number of projection curves on the coordinate planes. In addition, we also employ a distance measure between sets of invariants. Our method falls into the category of pose-consistency approaches described above.

This paper is organized as follows. In Sect. 2 we define cross sections and projection curves of 3D objects and establishes affine equivalence of projection curves for Euclidean equivalent surfaces in the form of a theorem. Projection curves obtained from principal cross sections are also presented. In Sect. 3 we present algebraic representations for modeling. We consider both algebraic surfaces and curves. Section 4 develops affine invariants of projection curves, defines a similarity for comparing invariant vectors, and presents the proposed method in algorithmic form. Section 5 discusses experimental results. Finally, Sect. 6 summarizes our work and concludes with some remarks.

2 Cross sections of an object and projection curves

Intersection of a 3D object with a plane in arbitrary orientation gives a *cross-section curve*. Further, we project this curve onto the coordinate planes and obtain a *projection curve*. More precisely, let Ω be a 3D object and $\partial\Omega$ be its boundary. Moreover, let Π be a plane in $\mathbb{R}^3$. Denoting the cross-section curve by Γ, we have

$$\Omega \cap \Pi = \Gamma \tag{1}$$

We will orthogonally project the curve Γ onto the $z = 0$ plane and denote the resulting projection curve by Υ. In other words,

$$\Gamma \xrightarrow{P_{z=0}} \Upsilon \tag{2}$$

In case the plane Π is orthogonal to the $z = 0$ plane, one should consider projections onto the two other coordinate planes since projection onto the $z = 0$ plane does not imply a well-defined closed-bounded curve.

2.1 Affine equivalent projection curves

Let B and $\bar{B}$ be two object boundaries related by a Euclidean/rigid transformation E, which consists of a rotation, R, and a translation, T, and let $\Pi : z = a_1x + b_1y + c_1$ and $\bar{\Pi} : z = a_2x + b_2y + c_2$ be the *corresponding planes* that are related by the same rigid transformation. Let Γ and $\bar{\Gamma}$ be the cross-section curves, and let Υ and $\bar{\Upsilon}$ be the projection of Γ and $\bar{\Gamma}$ under the projection $P_{z=0}$ onto the plane $z = 0$, respectively. The following theorem establishes affine equivalence of Υ and $\bar{\Upsilon}$.

Theorem 1 *If two surfaces B and $\bar{B}$ are Euclidean equivalent, their corresponding projection curves Υ and $\bar{\Upsilon}$ are affine equivalent.*

Proof Euclidean equivalence of B and $\bar{B}$ implies Euclidean equivalence of Γ and $\bar{\Gamma}$, namely

$$B \xrightarrow{E} \bar{B} \Rightarrow \Gamma \xrightarrow{E} \bar{\Gamma} \tag{3}$$

To prove this basic fact note that Γ and $\bar{\Gamma}$ are two planar curves on the corresponding planes $\Pi : z = a_1x + b_1y + c_1$ and $\bar{\pi} : z = a_2x + b_2y + c_2$. Since Euclidean equivalence of B and $\bar{B}$ implies Euclidean equivalence of the corresponding planes that are related by the same Euclidean transformation, it follows that Γ and $\bar{\Gamma}$ are also Euclidean equivalent.

Let $(x, y, z)^T$ and $(\bar{x}, \bar{y}, \bar{z})^T$ be two corresponding points on Γ and $\bar{\Gamma}$, respectively. In light of (3), Γ and $\bar{\Gamma}$ are related by E, and therefore it follows that

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{4}$$

Since the point $(x, y, z)^T$ lies on the plane Π, we have $z = a_1x + b_1y + c_1$. Substituting this into (4) and considering the first two equations, we obtain

$$\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} r_{11} + r_{13}a_1 & r_{12} + r_{13}b_1 \\ r_{21} + r_{23}a_1 & r_{22} + r_{23}b_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_1 + r_{13}c_1 \\ t_2 + r_{23}c_1 \end{pmatrix} \tag{5}$$

which proves affine equivalence of Υ and $\bar{\Upsilon}$.

2.2 Projection curves obtained from primary cross sections

We are interested in cross sections that are intrinsic to each 3D object. In this section, we define intrinsic cross sections and use the term primary. In order to define primary cross sections, we introduce moments.

The moment of order p, q, r of a 3D object Ω is

$$m_{p,q,r} = \int_{\Omega} x^p y^q z^r \mathrm{d}m,$$

where dm is the differential element of mass inside Ω. Moments, among other things, define physical properties of objects, such as center of mass as well as orientation of objects. For example, the mass center of Ω, G, is defined by $G = (G_x, G_y, G_z) = \left(\frac{m_{1,0,0}}{m_{0,0,0}}, \frac{m_{0,1,0}}{m_{0,0,0}}, \frac{m_{0,0,1}}{m_{0,0,0}}\right)$.

2.2.1 Primary cross sections: the cloud of points case

Let

$$\Sigma_{p,q,r} = \left(\int_{\Omega} (x - G_x)^p (y - G_y)^q (z - G_z)^r \mathrm{d}m \right),$$

where $p + q + r = 2$. We call this matrix the second-order moment matrix. When the input of our scheme is a cloud of points, we approximate the center and the second-order moment matrix using a sufficient number of data points, X_i, $i = 1, ..., N$, on it as follows:

$$G = \frac{\sum_{i=1}^{N} X_i}{N} \tag{6}$$

$$\Sigma = \frac{\sum_{i=1}^{N} (X_i - G)(X_i - G)^T}{N} \tag{7}$$

The second-order moment matrix is symmetric, $\Sigma = \Sigma^T$, and semi-positive definite, $\Sigma \geq 0$. Therefore, its eigenvalues are real and non-negative. Let these eigenvalues be denoted by λ_1, λ_2, and λ_3, and the corresponding eigenvectors by v_1, v_2, and v_3. These eigenvectors form the axes of an ellipsoid, with half-axes lengths $\sqrt{\lambda_1}, \sqrt{\lambda_2}$, and $\sqrt{\lambda_3}$. Eigenvector directions are the *principal axes* along which data scatter most [36, 37]. We would like to note that if a surface in 3D space is subject to a Euclidean transformation, its principal axes will also be subjected to the same transformation. We refer to the third axis as the primary axis.

Assume we have computed the mass center and the orientation of a 3D object. A set of primary planes is a set of planes that are perpendicular to the primary axis and they are symmetrically located around the origin. For clarity, if the cardinality of the set of primary planes is odd, the central plane passes through the mass center of the 3D object. The intersections of a set of primary planes with the 3D surface define a set of primary cross sections.

2.2.2 Primary cross sections: the tessellation case

For tessellated surfaces, however, we compute moments using explicit formulas as described in [38]. We embed the tessellated surface into a quasi-convex hull. We refer to the center of mass and orientation of the quasi-convex hull as being those of the underlying object. In this case, we define the orientation of object via inertia axes. We refer to the axis that correspond to the smallest absolute eigenvalue in the moment of inertia tensor as the primary axis.

In this context, we use the term principal axes in the same way we have defined it in Sect. 2.2.1. The intersections of a set of primary planes with the tessellated model defines a set of primary cross sections.

In the cloud of points case, we use the term primary projection curve to refer to projections of primary cross sections onto coordinate planes. In the tessellation case, we use the term primary projection curve to refer to the primary cross sections if the object's inertia axes are aligned with the coordinate axes.

3 Modeling object boundaries using algebraic representations

Algebraic curves and surfaces are among the most powerful boundary representation methods that can be used to model 2D and 3D objects. In this work, we will consider fitting of both 3D and 2D algebraic representations.

3.1 Obtaining projection curves by fitting algebraic surfaces

3D object data can be modeled by nth degree implicit surfaces of the form:

$$\begin{aligned} F_n(x,y,z) &= \sum_{0 \leq i,j,k; i+j+k \leq n} a_{ijk} x^i y^j z^k \\ &= \underbrace{a_{0,0,0}}_{H_0} + \underbrace{a_{1,0,0}x + a_{0,1,0}y + a_{0,0,1}z}_{H_1(x,y,z)} + \cdots \\ &\quad + \underbrace{a_{n,0,0}x^n + \cdots + a_{0,0,n}z^n}_{H_n(x,y,z)} \\ &= \underbrace{[1 \quad x \quad \ldots \quad z^n]}_{m^T} \underbrace{[a_{0,0,0} \quad a_{1,0,0} \quad \ldots \quad a_{0,0,n}]^T}_{a} \\ &= m^T a = 0 \end{aligned} \tag{8}$$

where each ternary form $H_r(x, y, z)$ is a *homogeneous polynomial* of degree r ($0 \leq r \leq n$) in x, y, and z, i.e. sum of the exponents in each term is r; m^T is the vector of

monomials; and a is the vector of implicit polynomial (IP) coefficients. An algebraic surface is called monic if $a_{n,0,0} = 1$. A 2D curve model is similar, but with the z terms discarded, namely

$$\begin{aligned} f_n(x,y) &= \sum_{0\le i,j;i+j\le n} a_{ij}x^i y^j = \underbrace{a_{0,0}}_{h_0} + \underbrace{a_{1,0}x + a_{0,1}y}_{h_1(x,y)} + \cdots \\ &\quad + \underbrace{a_{n,0}x^n + \cdots + a_{0,n}y^n}_{h_n(x,y)} \\ &= \underbrace{[1 \quad x \quad \ldots \quad y^n]}_{m^T}\underbrace{[a_{0,0} \quad a_{1,0} \quad \ldots \quad a_{0,n}]^T}_{a} \\ &= m^T a = 0 \end{aligned} \tag{9}$$

The problem of representing a dataset or some of its sections by an IP model is referred to as the polynomial fitting problem. Usually a cost function in the following form is minimized

$$J = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Dist}^2(P_i, Z(F_n)) \tag{10}$$

where $\mathrm{Dist}(P_i, Z(F))$ is the distance of the data point $P_i = (x_i, y_i, z_i)_{i=1}^N$ to the zero set $Z(F_n) = \{(x, y, z) : F_n(x, y, z) = 0\}$ of the IP model $F_n(x, y, z)$.

There are two main categories of IP fitting techniques: the non-linear methods, which use the true geometric distance or Euclidean approximation, and linear methods based on the algebraic distance. The non-linear approaches are invariant to transformations in the Euclidean space and are not biased [39]. However, except for very basic shapes, there is no available closed form expression for the Euclidean distance. Therefore, very time-consuming iterative optimization procedures must be carried out [24, 25].

Thus, linear techniques have been applied for the IP fitting problem [26, 27, 40]. Consequently, (10) can be formulated as a linear least squares problem, which make these methods much faster than the non-linear solutions.

In light of Sect. 2, cross section of an algebraic surface $F_n(x, y, z) = 0$ with a plane $z = ax + by + c$ yields the following cross-section curve:

$$\Gamma \stackrel{\Delta}{=} \{(x, y, z) |\ F_n(x, y, z) = 0,\ z = ax + by + c\} \tag{11}$$

The orthogonal projection of Γ onto $z = 0$ will give us the projection curve Υ as

$$\Gamma \xrightarrow{P_{z=0}} \Upsilon \stackrel{\Delta}{=} \{(x, y) |\ f_n(x, y) = 0\}$$

where $f_n(x, y) = 0$ is an algebraic curve obtained by plugging $z = ax + by + c$ into $F_n(x, y, z) = 0$.

3.2 Obtaining projection curves through fitting algebraic curves to the cross sections of tessellations

When objects have relatively complicated shapes we avoid 3D polynomial fitting. We compute intersection of the 3D input object with cross-section planes and fit polynomials to the resulting 2D intersection contours. We describe this procedure using images in Fig. 1. Consider the 3D tessellated object at the top of Fig. 1. This image shows an ant which is a tessellated model in Princeton shape benchmark [41]. We show a quasi-convex hull of the ant in violet. This quasi-convex hull is further used in computing the orientation of the object. The axes of inertia are superimposed on the image and shown in red, green, and blue. Nine primary cross sections, which are perpendicular to the red axis, i.e. x axis, intersect the body and the legs of the ant. These nine intersection curves are superimposed on the model and are shown in blue. The intersection curve undergoes 2D polynomial fitting. The results of the

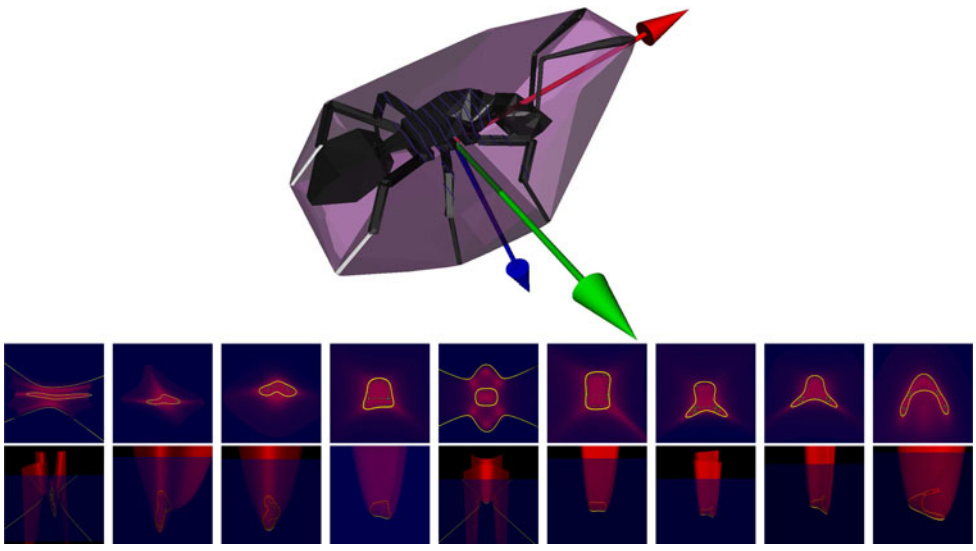

Fig. 1 Polynomial fitting for the primary cross sections of an ant, which is an object in Princeton shape benchmark. The second and third image lines represent bivariate polynomial fittings to the primary cross sections

polynomial fitting for each one of the nine intersections are shown in the second and the third line of Fig. 1.

As a conclusion for Sects. 2 and 3, we summarize the flow of computation in an algorithm. This algorithm is presented in Algorithm 1.

Algorithm 1 Primary Cross Section Computation

Input:
M_{3D} - a 3D model
$nPlanesNr$ - the number of primary planes involved in intersections
Output:
$arrPolys$ - a set of primary cross sections, represented as zero level sets of bivariate polynomial

CompPrimaryCrossSections($arrPolys, M_{3D}, nPlanesNr$)

1: **if** M_{3D} is a cloud of points **then**
2: Fit a 3D polynomial surface to M_{3D} using the fitting algorithm detailed in [27]. Let this surface be $\partial\ M_{3D}$.
3: Compute the orientation of $\partial\ M_{3D}$ as described in Section 2.2.1
4: **else**
5: $//M_{3D}$ is a tessellated model
6: Compute a 3D quasi-convex hull that includes M_{3D}.
7: Compute the orientation of the quasi-convex hull in terms of axes of inertia as described in Section 2.2.2
8: Let $\partial\ M_{3D}$ be the boundary of M_{3D}.
9: **end if**
10: Compute a set of $nPlanesNr$ primary planes as described in Section 2.2
11: **if** M_{3D} is a cloud of points **then**
12: Extract bivariate polynomials as intersections between the primary planes and $\partial\ M_{3D}$, as described in Section 3.1
13: **else**
14: $//M_{3D}$ is a tessellated model
15: Compute intersections between the primary planes and $\partial\ M_{3D}$.
16: Fit bivariate polynomials to the intersections, as described in Section 3.2
17: **end if**
18: $arrPolys$:= Bivariate polynomials computed at line 3.2 or 3.2.

4 Affine invariants of algebraic curves

Affine equivalent curves have the same affine invariants. One can compute algebraic/geometric affine invariants of projection curves using a variety of techniques. In this paper, we present two methods for constructing such invariants.

Before introducing invariants, let us recall the following important result.

Theorem 2 [31] *A non-degenerate (monic) polynomial $f_n(x, y)$ can be uniquely expressed as a finite sum of the products of real and/or complex line factors, namely*

$$f_n(x,y) = \Pi_n(x,y) + \gamma_{n-2}[\Pi_{n-2}(x,y) + \gamma_{n-4}[\Pi_{n-4}(x,y) + \cdots]] \tag{12}$$

where each $\Pi_k(x,y) = \prod_{i=1}^{k} L_i(x,y)$ is the product of k line factors and γ_k's are some real scalars.

For example, a (monic) conic ($n = 2$), cubic ($n = 3$), and quartic ($n = 4$) curve can be (line) decomposed as

$$f_2(x,y) = L_1(x,y)L_2(x,y) + \alpha = 0,$$

$$f_3(x,y) = L_1(x,y)L_2(x,y)L_3(x,y) + \alpha L_4(x,y) = 0,$$

and

$$\begin{aligned} f_4(x,y) &= L_1(x,y)L_2(x,y)L_3(x,y)L_4(x,y) \\ &\quad + \alpha L_5(x,y)L_6(x,y) + \beta \\ &= 0, \end{aligned} \tag{13}$$

respectively, where α and β are real scalars.

In the sequel, we will focus on quartic curves.

4.1 Algebraic affine invariants using canonical curves

Consider a line decomposed quartic curve as in (13). For closed-bounded curves, $L_1(x, y)$, ..., $L_4(x, y)$ will form two pairs of complex-conjugate lines, namely $\{L_1, L_2 = L_1^*\}$ and $\{L_3, L_4 = L_3^*\}$. The intersection of complex-conjugate lines will be two real points, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. The remaining two lines, $L_5(x, y)$ and $L_6(x, y)$, can be either real or complex-conjugate, but their intersection will be a third real point, $p_3 = (x_3, y_3)$. When the curve undergoes an affine transformation all the lines map to the corresponding lines in the transformed curve. The same correspondence holds for intersection points. Such intersection points are termed "related-points" in [29, 31]. These related points motivate introduction of canonical curves as follows:

Assume $f_4(x, y) = 0$ and $\bar{f}_4(x, y) = 0$ are affine equivalent quartic algebraic curves. Let the mappings of three points of $f_4(x, y) = 0$ to the corresponding related points of $\bar{f}_4(x, y) = 0$, be described by

$$(x_i, y_i) \xrightarrow{A} (\bar{x}_i, \bar{y}_i) \quad i = 1, 2, 3.$$

This mapping defines the affine transformation matrix A via the relation

$$\underbrace{\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}}_{\triangleq T} = A \underbrace{\begin{pmatrix} \bar{x}_1 & \bar{x}_2 & \bar{x}_3 \\ \bar{y}_1 & \bar{y}_2 & \bar{y}_3 \\ 1 & 1 & 1 \end{pmatrix}}_{\triangleq \bar{T}} \Longrightarrow A = T\bar{T}^{-1}. \tag{14}$$

Note that three such related points of $f_4(x, y) = 0$ defines a *canonical transformation matrix*

$$A_c \triangleq \begin{pmatrix} x_1 - x_3 & x_2 - x_3 & x_3 \\ y_1 - y_3 & y_2 - y_3 & y_3 \\ 0 & 0 & 1 \end{pmatrix} \tag{15}$$

and a corresponding monic ($a_{4,0} = 1$) *canonical curve* $f_4^c(x, y) = 0$ of $f_4(x, y) = 0$ defined by the relation

$$f_4(x,y) = 0 \xrightarrow{A_c} s_c f_4^c(x,y) = 0, \tag{16}$$

for some scalar s_c. Three corresponding related points of $\bar{f}_4(x, y) = 0$ will define a corresponding *canonical transformation matrix*

$$\bar{A}_c \triangleq \begin{pmatrix} \bar{x}_1 - \bar{x}_3 & \bar{x}_2 - \bar{x}_3 & \bar{x}_3 \\ \bar{y}_1 - \bar{y}_3 & \bar{y}_2 - \bar{y}_3 & \bar{y}_3 \\ 0 & 0 & 1 \end{pmatrix} \tag{17}$$

and a corresponding monic *canonical curve* $\bar{f}_4^c(x, y) = 0$ of $\bar{f}_4(x, y) = 0$ defined by the relation

$$\bar{f}_4(x, y) = 0 \xrightarrow{\bar{A}_c} \bar{s}_c \bar{f}_4^c(x, y) = 0, \tag{18}$$

for some scalar $\bar{s}_c$.

Theorem 3 [29] *The implicit polynomial curves* $f_4(x, y) = 0$ *and* $\bar{f}_4(x, y) = 0$ *will be affine equivalent if and only if their (monic) canonical curves* $f_4^c(x, y) = 0$ *and* $\bar{f}_4^c(x, y) = 0$ *are the same, in which case the affine transformation matrix is given by* $A = A_c \bar{A}_c^{-1}$.

Curve decomposition is detailed in [31]. After decomposing the curves, related-points can easily be determined as the real intersection points of the lines. These points will imply canonical transformation matrices defined by (15) and (17) and the corresponding canonical curves defined by (16) and (18), respectively. Therefore, we can associate a canonical curve with each projection curve. Since Theorem 2 establishes the fact that affine equivalent curves have the same canonical curve, all the coefficients of a canonical curve will be algebraic invariants, and therefore a vector of algebraic invariants can be constructed from them.

To be more specific, let

$$f_4^c(x, y) = a_{40}^c x^4 + a_{31}^c x^3 y + \cdots + a_{00}^c$$

be monic such that $a_{40}^c = 1$. The vector

$$\left(a_{40}^c [= 1], a_{31}^c, \ldots, a_{00}^c\right)$$

represents a set of algebraic invariants.

4.2 Geometric affine invariants using distance ratios

A monic ($a_{n,0} = 1$) algebraic curve of degree n has $n(n + 3)/2$ independent coefficients and therefore will have at least $n(n + 3)/2 - 6$ functionally independent invariants under the affine group [21]. Note that affine group has six transformation parameters. For example, a quartic curve ($n = 4$) has 14 independent coefficients, and therefore it has at least eight affine invariants.

Line decomposition of a quartic curve will imply that

$$\begin{aligned} f_4(x, y) &= L_1(x, y)L_2(x, y)L_3(x, y)L_4(x, y) \\ &\quad + \alpha L_5(x, y)L_6(x, y) + \beta \\ &= 0, \end{aligned}$$

with α and β representing 2 independent scalar invariants. In this case, the counting argument will imply at least $14 - 6 - 2 = 6$ additional affine invariants. Next, we will show how to construct these invariants.

First of all, lines in the decomposition of the curve can be ordered based on the real and imaginary parts of their parameters. Therefore, we can assume without loss of generality that the lines are ordered. First three lines $L_1(x, y) = 0$, $L_2(x, y) = 0$, and $L_3(x, y) = 0$ define what we call a "base triangle". $L_4(x, y) = 0$, $L_5(x, y) = 0$, and $L_6(x, y) = 0$ will intersect the lines $L_1(x, y) = 0$, $L_2(x, y) = 0$, and $L_3(x, y) = 0$. The lines $L_4(x, y) = 0$, $L_5(x, y) = 0$, and $L_6(x, y) = 0$ are called "transversals" of the base triangle (see Fig. 2).

The x and y coordinates of the intersection point P of any two non-parallel complex lines will in general be complex numbers of the form

$$x = a + bi, \quad y = c + di$$

The *distance* between two points with generally complex coordinates, $P = (a_1 + ib_1, c_1 + id_1)$ and $Q = (a_2 + ib_2, c_2 + id_2)$, is given by

$$\begin{aligned} ||(Q - P)|| &\triangleq |QP| = |PQ| \\ &= \sqrt{(a_2 - a_1)^2 + (b_2 - b_1)^2 + (c_2 - c_1)^2 + (d_2 - d_1)^2} \geq 0 \end{aligned} \tag{19}$$

Menelaus' Theorem [31] states that the product of ratios of distances measured from the intersection points to the corners of the triangle is 1, namely

$$\frac{|P_{34}P_{23}||P_{14}P_{13}||P_{24}P_{12}|}{|P_{34}P_{13}||P_{14}P_{12}||P_{24}P_{23}|} = 1 \tag{20}$$

Since the product of three distance ratios is 1, one of them can be determined automatically if the remaining two are specified. In general there is no constraint on these two distance ratios, and therefore they will be independent of each other. Since the distance ratio on a line is an affine invariant,

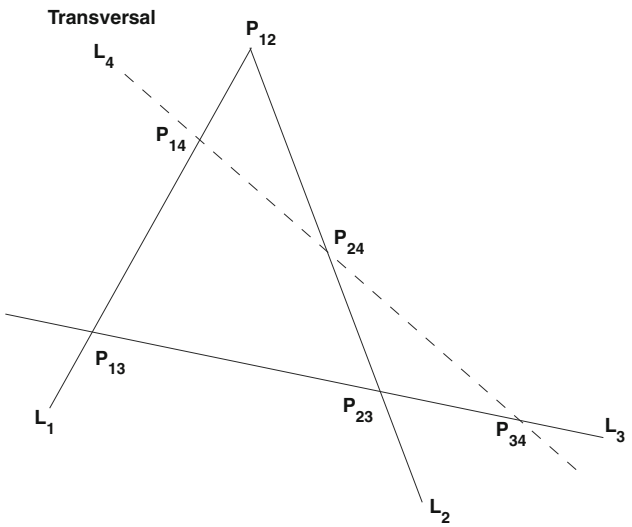

Fig. 2 A base triangle defined by L_1, L_2, and L_3 and its transversal L_4

the first transversal $L_4(x, y) = 0$ will imply two independent affine distance ratio invariants relative to the base triangle. The other two lines, $L_5(x, y) = 0$ and $L_6(x, y) = 0$, will each define two more independent ratio invariants, hence giving a total of six affine invariants as noted above. To be more specific, for the quartic ($n = 4$) case, the vector

$$\left(\frac{|P_{34}P_{23}|}{|P_{34}P_{13}|}, \frac{|P_{14}P_{13}|}{|P_{14}P_{12}|}, \frac{|P_{35}P_{23}|}{|P_{35}P_{13}|}, \frac{|P_{15}P_{13}|}{|P_{15}P_{12}|}, \frac{|P_{36}P_{23}|}{|P_{36}P_{13}|}, \frac{|P_{16}P_{13}|}{|P_{16}P_{12}|}\right)$$

represents a set of geometric invariants, where P_{ij} represents the intersection of the ith line with the jth one, $i, j \in \{1, ..., 6\}$.

4.3 Comparison of invariant vectors

Algebraic/geometric invariants of a curve can be concatenated to form a vector called "invariant vector". To compare two invariant vectors, the cosine of the angle between invariant vectors can be used as a similarity measure. The more this value is closer to one, the more similar invariant vectors are. More precisely,

$$\cos(\theta) = \frac{\langle a, b\rangle}{\|a\|\|b\|} \tag{21}$$

where θ is the angle between two invariant vectors, $\langle .,. \rangle$ denotes the inner product, $\|.\|$ denotes the Euclidean norm, and a and b are the invariant vectors of the two projection curves.

Let

$$MatchGradePointClouds\left(arrInvs^1, arrInvs^2\right) \tag{22}$$

be a procedure that computes the similarity between two input vectors $arrInvs^1$and $arrInvs^2$. We use this procedure in a matching grades routine for arrays of invariants, which is described in Algorithm 2.

Algorithm 2 Compute Matching Grades for Moderately Complex Objects

Input:
$mtrxInvs^1, mtrxInvs^2$ - two matrices of invariants, where on each line we have a set of invariants that characterize a primary cross section
Output:
$matchGrade$ - a matching grade
Comment:
The input matrices should have equal sizes. Let m be the number of rows of the input matrices and let $arrInv[i]$ denote the ith row in matrix $arrInv$.

MatchGradePointClouds$(mtrxInvs^1, mtrxInvs^2)$

1: //Create $arrInvGrades$ - which is an empty container
2: $arrInvGrades := \oslash$
3: **for all** $nI \in \{1 \ldots m\}$ **do**
4: $localMatchGrade := MatchGradePointClouds\left(mtrxInvs^1[nI], mtrxInvs^1[nI]\right)$ see Equation (22)
5: $arrInvGrades := arrInvGrades \bigcup \{localMatchGrade\}$
6: **end for**
7: //Compute the averages of matching over sets of invariants
8: $matchGrade := average\,(arrInvGrades)$

4.4 Distance between invariant matrices

Let A and B be two $m \times n$ matrices of invariants. Define the distance between these two matrices of invariants as

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{|A_{ij} - B_{ij}|}{|A_{ij}| + |B_{ij}|}. \tag{23}$$

We will use the notation

$$MatchGradeTessellation\left(mtrxInvs^1, mtrxInvs^2\right) \tag{24}$$

for a procedure that computes the distance between two input vectors $mtrxInvs^1$ and $mtrxInvs^2$ as defined in Eq. (23).

4.5 Object comparison algorithm

In this section we deep into the details of comparison of invariants. The main comparison algorithm is described in Algorithm 5. Several necessary routines are defined in Algorithms 3 and 4.

Algorithm 3 Compute Invariants of a Polynomial

Input:
$poly2D$ - a 2D polynomial
Output:
$arrInvsAlg$ - an array of algebraic invariants
$arrInvsGeom$ - an array of geometric invariants

CompInvariants$(arrInvsAlg, arrInvsGeom, poly2D)$

1: $arrInvsAlg :=$ Compute the algebraic invariants as described in Section 4.1
2: $arrInvsGeom :=$ Compute the geometric invariants as described in Section 4.2

Algorithm 4 Compute Invariants of a 3D Model

Input:
M_{3D} - a 3D model
$nPlanesNr$ - the number of primary planes involved in intersections
Output:
$mtrxInvsAlg$ - a matrix of algebraic invariants
$mtrxInvsGeom$ - a matrix of geometric invariants
Comment:
Each line in the output matrices is a set of invariants that corresponds to a primary projection curve. The number of lines of the output matrices are equal. Lines with identical ordering indices represent invariants for the same primary projection curve.

CompInvsModel3D$(mtrxInvsAlg, mtrxInvsGeom, M_{3D}, nPlanesNr)$

1: **CompPrimaryCrossSections**$(arrPolys, M_{3D}, nPlanesNr)$, see Algorithm 1
2: **for all** $poly2D \in arrPolys$ **do**
3: Let $nIndx$ be the ordering index of the bivariate primary polynomial $poly2D$ and let $mtrxInvsAlg[nIndx]$ and $mtrxInvsGeom[nIndx]$ be the $nIndx-$th lines in these output matrices.
4: CompInvariants$(mtrxInvsAlg[nIndx], mtrxInvsGeom[nIndx], poly2D)$, see Algorithm 3
5: **end for**

Algorithm 5 Compare Objects

Input:
M_{3D}^1, M_{3D}^2 - two 3D models
$nPlanesNr$ - the number of primary planes involved in intersections
Output:
$matchGradeAlg$ - an algebraic grade of matching between the two input 3D models
$matchGradeGeom$ - a geometric grade of matching between the two input 3D models

CompMatchGrade$(matchGradeAlg, matchGradeGeom, M_{3D}^1, M_{3D}^2, nPlanesNr)$

1: **CompInvsModel3D**$(mtrxInvsAlg^1, mtrxInvsGeom^1, M_{3D}^1, nPlanesNr)$, **see Algorithm 4**
2: **CompInvsModel3D**$(mtrxInvsAlg^2, mtrxInvsGeom^2, M_{3D}^2, nPlanesNr)$, **see Algorithm 4**
3: **if** M_{3D} is a cloud of points **then**
4: matchGradeAlg := MatchGradePointClouds$(mtrxInvsAlg^1, mtrxInvsAlg^2)$, see Algorithm 2
5: matchGradeGeom := MatchGradePointClouds$(mtrxInvsGeom^1, mtrxInvsGeom^2)$, see Algorithm 2
6: **else**
7: $//M_{3D}$ is a tessellated model
8: matchGradeAlg := MatchGradeTessellation$(mtrxInvsAlg^1, mtrxInvsAlg^2)$, see Equation (24)
9: matchGradeGeom := MatchGradeTessellation$(mtrxInvsGeom^1, mtrxInvsGeom^2)$, see Equation (24)
10: **end if**

5 Experimental results and discussions

In this section we present several experiments. We differentiate between moderately complicated structures and relatively complicated objects. In addition, the moderately complicated objects are grouped into classes that have relative small number of members, while the more complicated ones are included in large classes. In particular, we consider moderately complicated objects defined by clouds of points, while the relatively complicated objects represented by tessellated models, which are included in widely accessible large databases of objects, such as Princeton [41] and McGill [42] 3D shape benchmarks.

5.1 Experiments with objects represented by cloud of points

In these experiments we employ 3D surface fitting. 3D objects defined by clouds of points are modeled by fourth degree algebraic surfaces and then projection curves are extracted for comparison purposes.

5.1.1 *Experiments with objects from different classes*

Several experiments are performed for invariant recognition of 3D objects represented by implicit algebraic surfaces. Object pose is changed by applying randomly selected Euclidean transformations to 3D object data. The original and the transformed data are normalized and fitted by fourth degree (quartic) algebraic surfaces [27]. Principal axes are then calculated from the resulting surface data although they can directly be computed from raw 3D data. Seven principal planes, with 0.1 unit distance apart and orthogonal to the principal axis with the largest eigenvalue, i.e. the primary principle axis, are used to construct cross-section curves of the surfaces. One of the planes is selected to be at the center and the remaining ones are placed uniformly around the center (three on each side of the center). In principle, one can use any number of such planes for generating projection curves, but seven of them are sufficient for performing invariant computations. Since more than one projection curve is employed, the average similarity is computed and employed for object comparison.

The sample objects used in our experiments are depicted in Fig. 3. Figures 4, 5, and 6 show three example objects, their fourth degree algebraic surface models with the principal axes, v_1, v_2, and v_3, computed, and seven cross section and projection curves.

Results of various experiments are tabulated in Table 1. First two columns in the table show sample objects and the type of invariants, algebraic or geometric, used for similarity computation. The next column shows average similarities in the noise-free case, where each object is subject to a random Euclidean transformation and is compared with its transformed version based on average similarity of affine invariants of its projection curves. Next two columns in the table indicate average similarities in two different noisy scenarios, where each object is subject to a random Euclidean transformation and corrupted by a Gaussian noise with specified variances, $\sigma^2 = 0.0025$ and $\sigma^2 = 0.01$, and then compared with its transformed version based on

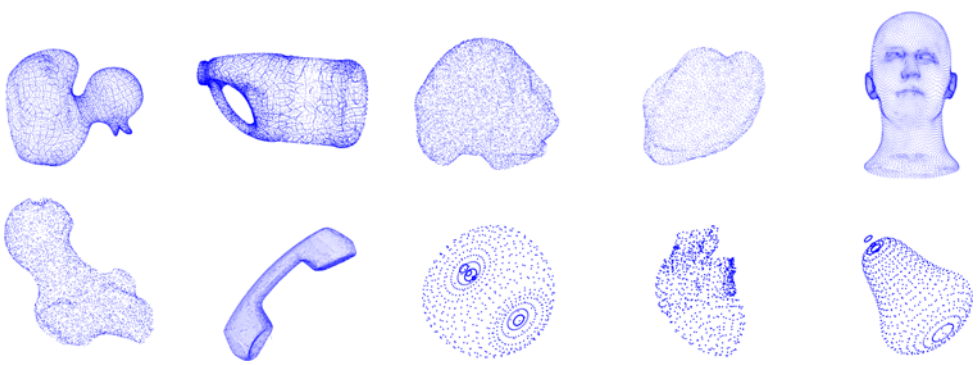

Fig. 3 Sample objects used in our experiments: Duck, bottle, venus, patella bone, mannequin, balljoint, phone, apple, heart, and pear

average similarity of affine invariants of its projection curves. Similarly, the next two columns show average similarities in two different missing data situations, where each object is subject to a random Euclidean transformation and a certain percentage, 10 and 20%, of object data are randomly discarded, and then compared with its transformed version based on average similarity of affine invariants of its projection curves. Finally, the last column in the table shows average similarities for the case, where each object is re-sampled by throwing half of the points. The resulting objects are each subject to a random Euclidean transformation and then fitted by algebraic surfaces. Their projection curves are computed and compared based on average similarity of their affine invariants.

Analysis of Table 1 reveals several important facts. In the noise-free case object self-comparisons imply average similarities near one, both for algebraic and geometric invariants, which is what we expect. In other cases, where objects are subject to data perturbations, average similarities are relatively robust although they might assume quite different values for algebraic and geometric invariants. To see how well these similarities can discriminate different objects, comparisons have also been made for different objects. A test object, phone, is compared with 10 different model objects in our database and results are tabulated in Table 2. Note that the average similarity for phone objects is very close to one, and far below one for the phone and other objects. Figure 7 depicts these

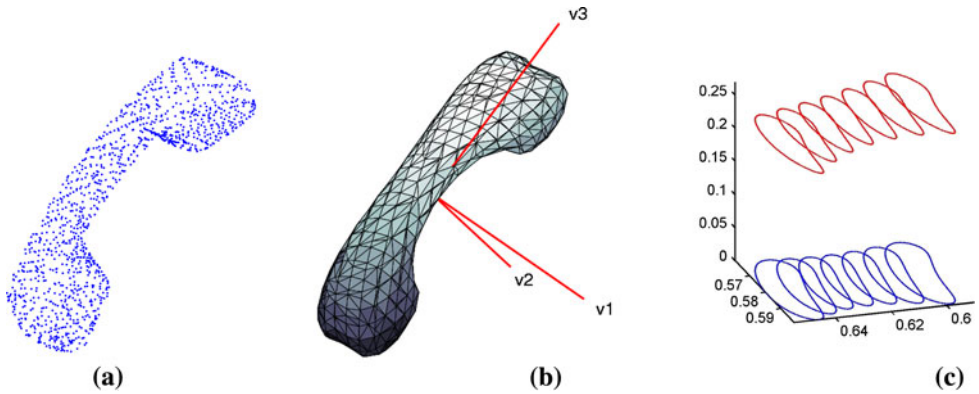

Fig. 4 **a** Phone data, **b** phone fit with axes, and **c** cross section and projection curves

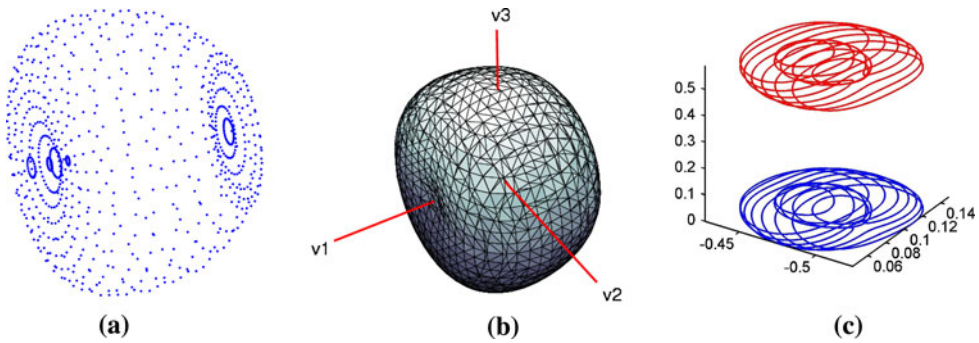

Fig. 5 **a** Apple data, **b** apple fit with axes, and **c** cross section and projection curves

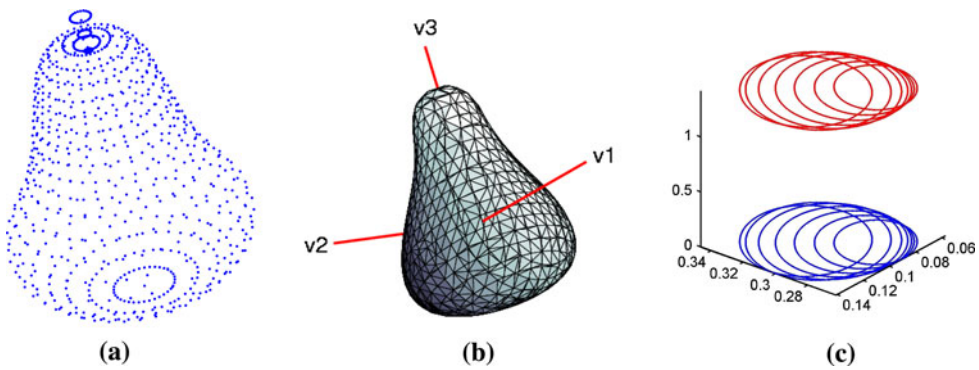

Fig. 6 **a** Pear data, **b** pear fit with axes, and **c** cross section and projection curves

Table 1 Average similarities of algebraic and geometric invariants for different objects subject to random Euclidean transformations and various data perturbations

	Noiseless	$\sigma^2 = 0.0025$	$\sigma^2 = 0.01$	%10	%20	Re-sampled
Phone						
Algebraic	0.9995	0.9972	0.9618	0.9976	0.7909	0.9999
Geometric	0.9989	0.9998	0.9944	0.9985	0.9936	0.9999
Apple						
Algebraic	0.9980	0.9874	0.7036	0.9985	0.9964	0.7190
Geometric	0.9996	0.9985	0.9894	0.9998	0.9963	0.9995
Pear						
Algebraic	0.9999	0.9957	0.9333	0.9475	0.8402	0.9861
Geometric	0.9828	0.9776	0.8368	0.9772	0.9328	0.9808
Heart						
Algebraic	0.9837	0.9044	0.8794	0.8427	0.8649	0.4961
Geometric	0.9664	0.8712	0.8763	0.9323	0.9516	0.9639
Patella						
Algebraic	0.9370	0.8898	0.8219	0.9929	0.9870	0.9679
Geometric	0.9988	0.9536	0.7575	0.9927	0.7858	0.9357
Bottle						
Algebraic	0.9987	0.9956	0.9919	0.9866	0.9403	0.9238
Geometric	0.9995	0.9989	0.9978	0.9936	0.9640	0.9746
Venus						
Algebraic	0.9917	0.9998	0.8193	0.9907	0.9885	0.8312
Geometric	0.9844	0.9998	0.9691	0.9884	0.9749	0.8672
Mannequin						
Algebraic	0.9939	0.9929	0.9899	0.9795	0.9681	0.9837
Geometric	0.9990	0.9980	0.7005	0.9301	0.8298	0.7576
Duck						
Algebraic	0.9970	0.9942	0.9875	0.9730	0.8038	0.9912
Geometric	0.9713	0.8181	0.7244	0.9684	0.9014	0.9950
Balljoint						
Algebraic	0.9433	0.9798	0.9695	0.9642	0.8569	0.9756
Geometric	0.9369	0.8719	0.7623	0.8436	0.7359	0.9394

Table 2 A test object (phone) has been compared with model objects in the database based on average similarities of invariant vectors

Model objects	Test object (phone)	
	Algebraic	Geometric
Phone	0.9764	0.9994
Apple	0.3232	0.6337
Heart	0.5980	0.6491
Pear	−0.0339	0.4938
Duck	0.5338	0.5034
Bottle	0.6809	0.4939
Venus	0.6429	0.2894
Patella	0.2268	0.6593
Mannequin	−0.0147	0.5156
Balljoint	0.4864	0.6380

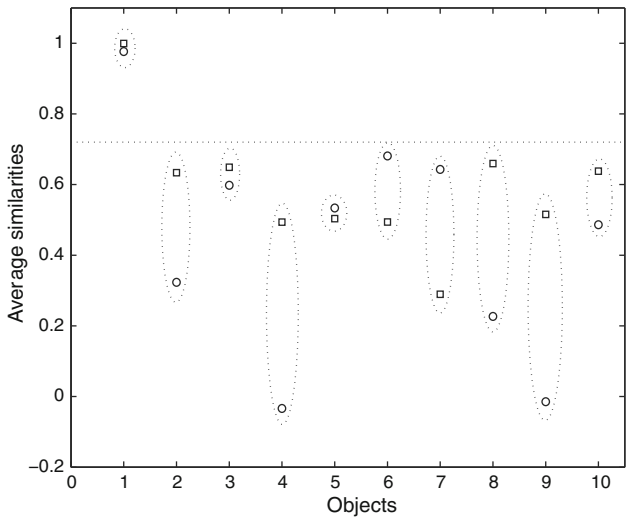

Fig. 7 Test object is the phone. Model objects are: *1* phone, *2* apple, *3* heart, *4* pear, *5* duck, *6* bottle, *7* venus, *8* patella, *9* mannequin, and *10* balljoint. Average similarities are plotted both for algebraic (*circle*) and geometric (*square*) invariants, and are included in a *dotted ellipse* for each object. The *dotted line* is a threshold value

average similarities for the test and model objects graphically. A threshold value of approximately 0.7 matches the test object (phone) to the model object (phone) and discriminates it from the remaining model objects.

5.1.2 Experiments with similar but not identical objects

To see the discrimination power of the proposed method for similar, but not identical objects represented as cloud of points, we have performed two sets of experiments with head and car datasets (see Figs. 8 and 9). Tables 3 and 4 tabulate comparison of invariant vectors for these similar, but not identical objects. Note that while identical objects have a similarity value, for both algebraic and geometric invariants, well above 0.9, i.e. very close to one, similar but not identical ones have similarity values less than this value. These experimental results show the potential of our proposed method for within-class recognition.

5.2 Experiments with tessellated objects

We present results of an implementation in C++ [43]. We have used several numerical procedures from the Numerical Recipes in C [44] library. For example, we have used the *zrhqr* routine for computing roots of polynomials toward polynomial decomposition. For visualization purposes, we have used the VTK [45] library and Mesh View [46]. We have run our implementation over a Core2Duo computer equipped with two 3 GHz processors and 2 GB of memory.

We have also used VTK for 3D data processing. For example, the quasi-convex hulls are computed employing the *vtkHull* class. Following *vtkHull* specification [45], *vtkHull* generates a hull by squeezing the planes toward the input model, until the planes touch it, and computes the intersection among these planes in a polyhedron representation. For each face of the model, we compute its normal and consider one of its boundary points. Let us call this point a representative of the face. All representatives

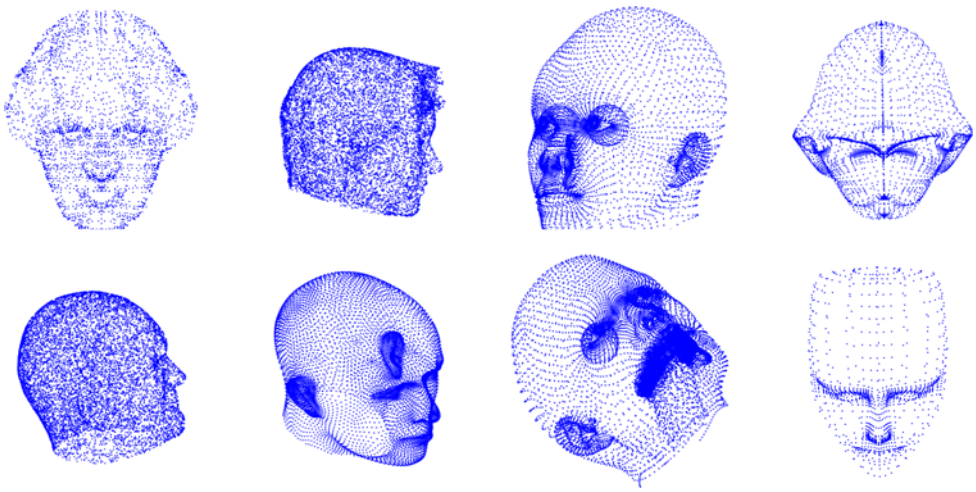

Fig. 8 Female, male, and alien heads

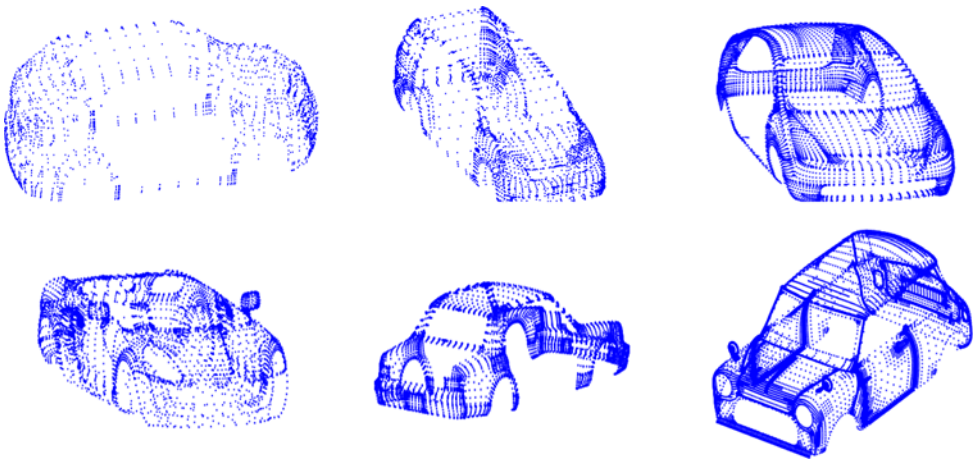

Fig. 9 Car frames for harrier-hybrid, isis, prius (*first row*), toyota, century, and mini-cooper (*second row*), respectively

Table 3 Comparison of test object (male1) with other objects

	Test object: male1	
	Algebraic	Geometric
Male1	0.9456	0.9952
Male2	0.8066	0.8428
Male3	0.7917	0.8121
Female1	0.6802	0.7850
Female2	0.6968	0.5101
Female3	0.6145	0.5020
Allien1	0.5897	0.3228
Allien2	0.5975	0.6277

Table 4 Comparison of test object (harrier-hybrid) with the other car frames

	Test Object: harrier-hybrid	
	Algebraic	Geometric
Harrier-hybrid	0.9998	0.9236
Isis	0.4768	0.8037
Prius	0.9972	0.7614
Toyota	0.7802	0.7142
Century	0.2799	0.8425
Mini-cooper	0.8236	0.8144

and normals are then given at input to *vtkHull*, which computes a quasi-convex hull. We have experimented with more initialization setups for *vtkHull*, however, the way described here is the best trade-off we have found. For clarity, we should note that *vtkHull* receives input faces and computes intersections among them only.

We present part of the objects that we considered from Princeton and McGill shape benchmarks in Fig. 10. In all experiments performed on tessellated objects, we used nine primary planes where the distance between them is 0.1.

One of the differences between the McGill and Princeton shape benchmark objects is the accuracy of tessellation. The objects in Princeton shape benchmark are usually represented by one mesh. Unlike this, the objects in McGill are represented by adjacent surfaces that intersect each other, i.e. several meshes approximates the surface of the objects, see Fig. 11.

Another problem we have coped with is articulated objects. The large variability of configurations in which articulated objects might appear induces an inherent need for a careful selection of objects for analysis, see Fig. 12.

In the next sections, we show confusion matrices for Princeton and McGill shape benchmarks. These confusion

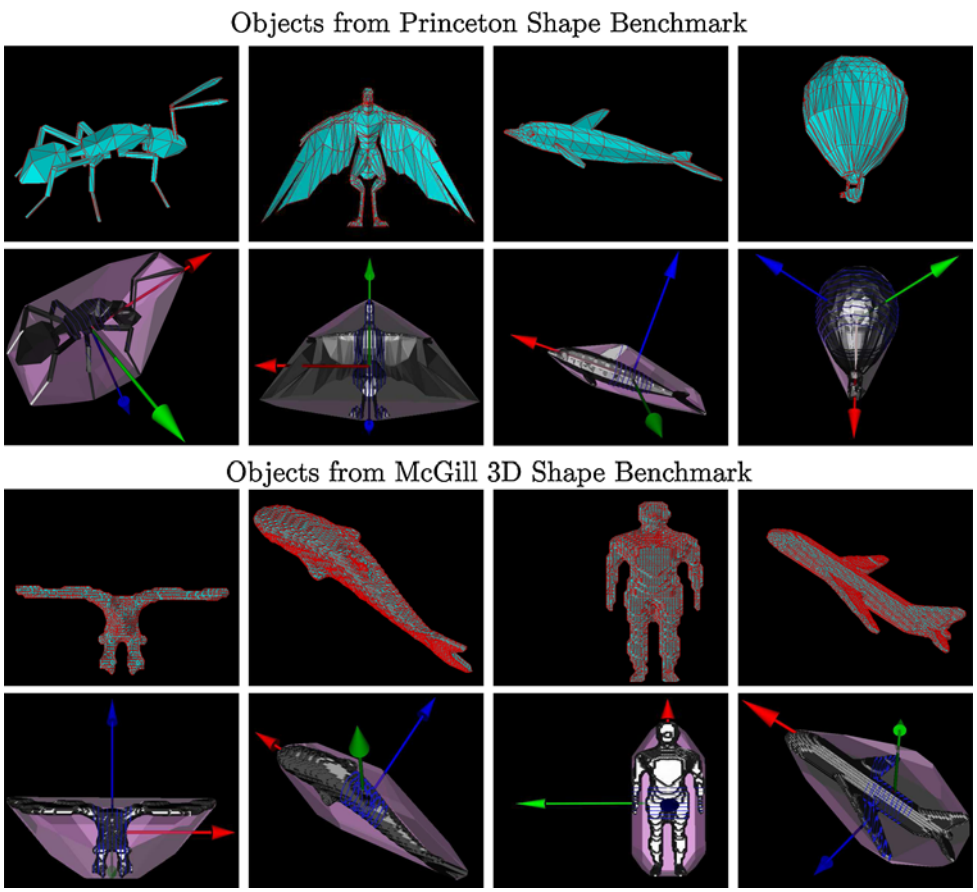

Fig. 10 Princeton and McGill 3D shape benchmarks. Each group of two lines represent tessellated objects (*top*) and objects superimposed with their axes of inertia, approximations of their convex hulls, and intersection with principal planes (*bottom*). The *red*, *green*, and *blue* axes correspond to *Ox*, *Oy*, and *Oz* coordinates, respectively. The axes *Ox*, *Oy*, and *Oz* are ordered in increasing absolute eigenvalues (of the inertia matrix) of the analyzed objects. The intersection with primary planes are represented with *blue* contours. The primary planes are perpendicular to *Ox* axis. The tessellation models are visualized with Mesh Viewer [46]

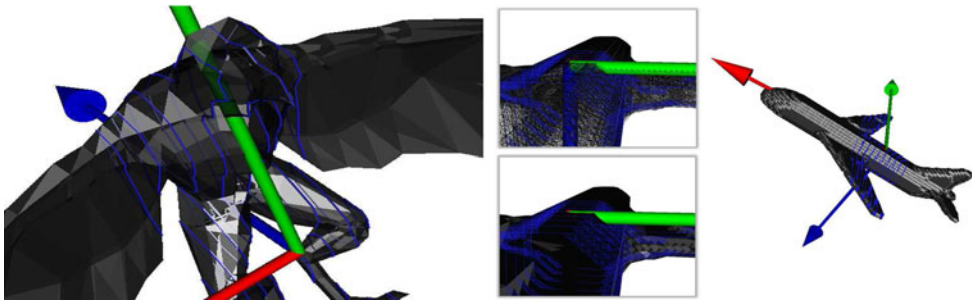

Fig. 11 Problems in tessellations of Princeton and McGill benchmarks objects. The *blue curves* represent intersection of parallel planes with the tessellation of the objects under analysis. In Princeton objects, usually, the contours are clearly defined by a unique mesh, in almost any region of the object, see the *left* image. Unlike Princeton, McGill objects are characterized by multiple meshes, and therefore, intersections with parallel planes consist of multiple contours, see the set of the three images at right. The images in the *right* represent a full object as well as two zoom-in regions. The two zoom-in regions illustrate the same part of the plane body, which is represented as wires and tessellations

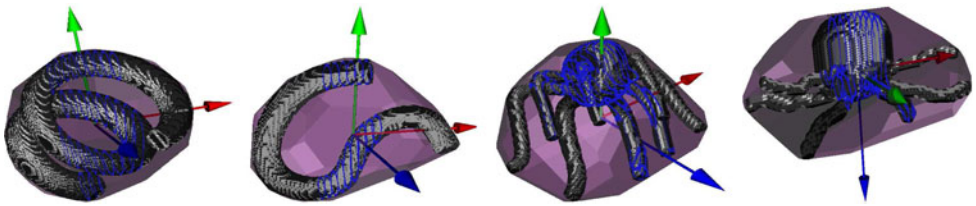

Fig. 12 Problems with articulated objects in Princeton and McGill shape benchmarks

matrices are given in tables. Each presented table involves a set of classes. For each table, on each line we consider a certain class, of which representatives belong to the same or other different classes. The number of objects correctly classified are reported on the diagonals of the table. The i, jth entry of the table represents the number of objects in the ith class that belong to the jth one.

When classifying an object we compare its invariants versus all other objects in various classes. We compute the closest object in terms of distances of invariants using Algorithm 5, which relies on Eq. (23).

5.2.1 *Experiments with McGill 3D shape benchmark objects*

We present results of three experiments performed on McGill 3D shape benchmark in Tables 5, 6, 7, 8, 9, and 10. The McGill 3D shape benchmark divides the objects into articulated and non-articulates super-classes. Each super-class is further divided into sub-classes. We have chosen less articulated classes for experiments, see also Fig. 12.

Table 5 Algebraic invariants-based confusion matrix computed for three classes from McGill 3D shape benchmark

	Humans	Cups	Fishes
Humans	9	4	4
Cups	5	9	2
Fishes	4	2	13

Table 6 Geometric invariants-based confusion matrix computed for three classes from McGill 3D shape benchmark

	Humans	Cups	Fishes
Humans	6	5	6
Cups	1	9	6
Fishes	6	5	8

Table 7 Algebraic invariants-based confusion matrix computed for four classes from McGill 3D shape benchmark

	Birds	Cups	Fishes	Four
Birds	5	1	1	6
Cups	2	7	5	2
Fishes	0	2	13	4
Four	3	6	8	14

Table 8 Geometric invariants-based confusion matrix computed for four classes from McGill 3D shape benchmark

	Birds	Cups	Fishes	Four
Birds	6	0	2	5
Cups	0	9	6	1
Fishes	1	4	10	4
Four	2	7	8	14

Table 9 Algebraic invariants-based confusion matrix computed for various classes from McGill 3D shape benchmark

	Ants	Humans	Airplanes	Birds	Cups	Dolphins	Fishes	Four	Tables
Ants	2	3	1	1	1	1	0	0	0
Humans	4	4	1	1	2	0	2	3	0
Airplanes	1	1	7	0	4	0	4	1	1
Birds	2	1	0	4	0	1	0	3	2
Cups	2	2	1	0	5	1	2	2	1
Dolphins	0	3	0	0	2	2	3	1	0
Fishes	0	1	0	0	1	3	10	3	1
Four	1	5	1	1	4	2	6	9	2
Tables	1	2	2	2	3	1	1	0	6

Table 10 Geometric invariants-based confusion matrix computed for various classes from McGill 3D shape benchmark

	Ants	Humans	Airplanes	Birds	Cups	Dolphins	Fishes	Four	Tables
Ants	3	0	1	0	0	0	0	2	3
Humans	1	1	4	0	0	0	3	5	3
Airplanes	0	1	7	0	1	1	1	4	4
Birds	1	2	2	4	0	0	2	0	2
Cups	0	0	1	0	8	1	4	0	2
Dolphins	0	3	1	0	1	2	1	0	3
Fishes	0	4	3	0	3	0	6	3	0
Four	0	7	4	0	2	1	6	10	1
Tables	0	2	6	2	2	0	0	4	2

5.2.2 *Experiments with Princeton shape benchmark objects*

We present results of three experiments performed on Princeton shape benchmark in Tables 11, 12, 13, 14, 15, and 16. In these experiments, we focus on classifying sub-classes of one super-class. In the first experiment, we compute the confusion matrix for the car super-class (see Tables 11 and 12). In the second experiment, we compute the confusion matrix for the air-vehicle super-class (see Tables 13 and 14). In the third experiment, we compute the confusion matrix for several classes that might be considered part of the super-class vehicle (see Tables 15 and 16).

Table 11 Algebraic invariants-based confusion matrix computed for the car super-class from Princeton shape benchmark

	Race-car	Sedan	Sports-car
Race-car	9	2	3
Sedan	2	14	4
Sports-car	4	7	1

Table 12 Geometric invariants-based confusion matrix computed for the car super-class from Princeton shape benchmark

	Race-car	Sedan	Sports-car
Race-car	3	8	3
Sedan	3	15	2
Sports-car	1	6	5

Table 13 Algebraic invariants-based confusion matrix computed for the air vehicle super-class from Princeton shape benchmark

	Dirigible	Hot-air-balloon
Dirigible	5	1
Hot-air-balloon	1	5

Table 14 Geometric invariants-based confusion matrix computed for the air vehicle super-class from Princeton shape benchmark

	Dirigible	Hot-air-balloon
Dirigible	3	3
Hot-air-balloon	1	5

Table 15 Algebraic invariants-based confusion matrix computed for the vehicle super-class from Princeton shape benchmark

	Antique car	Bicycle	Dirigible	Hot air balloon	Military tank	Race car	Sedan	Sports car
Antique-car	1	1	0	0	0	2	1	0
Bicycle	0	1	0	0	2	0	1	1
Dirigible	0	0	4	1	0	0	1	0
Hot-air-balloon	0	0	1	4	1	0	0	0
Military-tank	0	1	1	0	5	2	5	2
Race-car	1	0	0	0	2	8	1	2
Sedan	0	0	0	0	4	2	8	6
Sports-car	0	0	0	0	3	4	7	4

Table 16 Geometric invariants-based confusion matrix computed for the vehicle super-class from Princeton shape benchmark

	Antique car	Bicycle	Dirigible	Hot-air-balloon	Military tank	Race car	Sedan	Sports car
Antique-car	1	0	0	0	1	1	0	2
Bicycle	1	0	0	0	1	2	0	1
Dirigible	0	0	2	4	0	0	0	0
Hot-air-balloon	0	0	1	4	1	0	0	0
Military-tank	0	0	0	0	7	2	5	2
Race-car	0	0	0	0	6	1	4	3
Sedan	0	0	0	0	7	2	8	3
Sports-car	0	0	0	0	4	1	6	7

5.3 Implementation details and time performance analysis

We divide the whole processing into three stages and provide time performance analysis for each one of them. We report time requirements for polynomial fitting, polynomial decomposition, as well as invariants extraction.

We depict average time performance of polynomial fitting algorithm [28] in Fig. 13. In Fig. 13, we plot the time spent for generating polynomials of degree 4, 6, 8, and 10 depending on the number of points at input. These numbers range from 20 to over 2,000. We present time consumption for polynomial decomposition and invariants extraction in Fig. 14. In Fig. 14, we show the time performance analysis for fitted polynomials of degrees 4, 6, 8, and 10.

We considered polynomials of degree 4, 6, 8, and 10 since these polynomials are the most practical ones and usually present the best trade-off between the accuracy of

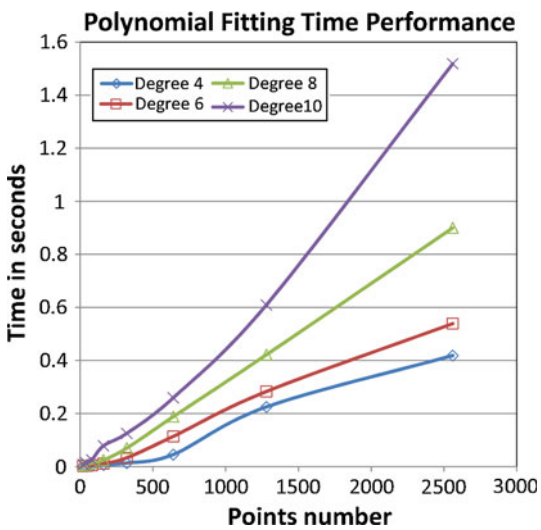

Fig. 13 Polynomial fitting times

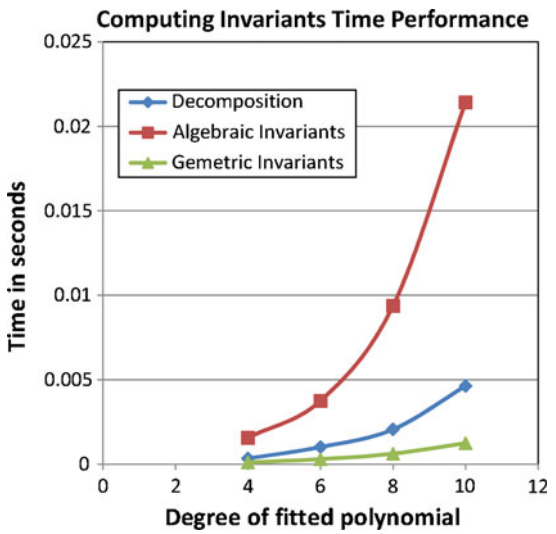

Fig. 14 Polynomial decomposition and invariants extraction times

computations and the time consumption. To the best of our knowledge, our time performance analysis is the first serious evaluation presented for a polynomial fitting algorithm, being performed in a framework of C++ implementation. Moreover, it is the first serious time performance analysis for the proposed polynomial decomposition as well as the invariants extraction.

While many classification and recognition schemes are characterized by complex searches in exponential graphs of configurations, see for example [15, 16], our scheme is fast and does not involve such heavy computations. However, one of the limitations of our scheme is articulated objects.

6 Conclusions

We have presented a new technique for recognizing 3D objects described by algebraic surfaces using affine invariants of their projection curves. The proposed method explicitly constructs robust algebraic and geometric invariants for projection curves and therefore eliminates the need for finding and computing geometric invariants of 3D surfaces, which is a non-trivial task. Our particular treatment based on algebraic curve decompositions provides significant new insight about invariants. Unlike the more classical algebraic invariants, the geometric invariants obtained here have a natural interpretation in terms of distances. Therefore, one can apply several different measures when comparing such invariants. Moreover, in terms of time performance, our algorithm is fast when compared to existing algorithms.

Experiments with cloud of points and tessellated objects have been performed in order to evaluate the utility of affine invariants of projection curves for object recognition. The robustness of such invariants has been assessed with several noisy, missing, and re-sampled datasets. In addition, time performance analysis has been performed for evaluating various stages of computation. While our method is very fast, it has limitations when applied to articulated objects. However, it can easily handle arbitrary pose variations since it is based on invariants.

References

1. Costa MS, Shapiro LG (2000) 3D object recognition and pose with relational indexing. Comput Vis Image Underst 79(3):364–407
2. Forsyth DA, Ponce J (2003) Computer vision: a modern approach. Prentice Hall
3. Gonzalez E, Adan A, Feliu V, Sanchez L (2008) Active object recognition based on Fourier descriptors clustering. Pattern Recognit Lett 29(8):1060–1071
4. Kim S, Kweon IS (2008) Scalable representation for 3D object recognition using feature sharing and view clustering. Pattern Recognit 41(2):754–773
5. Rothganger F, Lazebnik S, Schmid C, Ponce J (2006) 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. Int J Comput Vis 66(3):231–259
6. Lee TK, Drew MS (2007) 3D object recognition by eigen-scale-space of contours. In: First international conference on scale space methods and variational methods in computer vision (SSVM'07), pp 883–894
7. Li W, Bebis G, Bourbakis N (2004) Integrating algebraic functions of views with indexing and learning for 3D object recognition. In: Computer vision and pattern recognition workshop, p 102, June 2004
8. Nagabhushan P, Guru DS, Shekar BH (2006) Visual learning and recognition of 3D objects using two-dimensional principal component analysis: a robust and an efficient approach. Pattern Recognit 39(4):721–725
9. Chen H, Bhanu B (2007) 3D free-form object recognition in range images using local surface patches. Pattern Recognit Lett 28(10):1252–1262
10. Mian AS, Bennamoun M, Owens R (2006) Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans Pattern Anal Mach Intell 28(10):1584–1601
11. Diplaros A, Gevers T, Patras I (2006) Combining color and shape Information for illumination-viewpoint invariant object recognition. IEEE Trans Image Process 15(1):1–11
12. Shotton J, Blake A, Cipolla R (2008) Multiscale categorical object recognition using contour fragments. IEEE Trans Pattern Anal Mach Intell 30(7):1270–1281
13. Adan A, Adan M (2004) A flexible similarity measure for 3D shapes recognition. IEEE Trans Pattern Anal Mach Intell 26(11):1507–1520
14. Shan Y, Sawhney HS, Matei B, Kumar R (2006) Shapeme histogram projection and matching for partial object recognition. IEEE Trans Pattern Anal Mach Intell 28(4):568–577
15. Pechuk M, Soldea O, Rivlin E (2008) Learning function based classification from 3D imagery. Comput Vis Image Underst 110(2):173–191
16. Froimovich G, Rivlin E, Shimshoni I, Soldea O (2007) Efficient search and verification for function based classification from real range images. Comput Vis Image Underst 105(3):200–217
17. Taubin G, Cooper DB (1992) 2D and 3D object recognition and positioning with algebraic invariants and covariants. In: Chapter 6 of Symbolic and numerical computation for artificial intelligence. Academic Press
18. Huang ZH, Cohen FS (1996) Affine invariant B-spline moments for curve matching. Image Process 5(10):1473–1480
19. Solina F, Bajcsy R (1990) Recovery of parametric models from range images: The case for superquadrics with global deformations. IEEE Trans Pattern Anal Mach Intell 12(2):131–147
20. Ma S (1993) Conics-based stereo, motion estimation and pose determination. IJCV 10(1):7–25
21. Mundy JL, Zisserman A (1992) Geometric invariance in computer vision. The MIT Press
22. Calabi E, Olver PJ, Shakiban C, Tannenbaum A, Haker S (1998) Differential and numerically invariant signature curves applied to object recognition. IJCV 26(2):107–135
23. Wu MF, Sheu HT (1998) Representation of 3D surfaces by two-variable Fourier descriptors. IEEE Trans Pattern Anal Mach Intell 20(8):583–588
24. Taubin G, Cukierman F, Sullivan S, Ponce J, Kriegman DJ (1994) Parameterized families of polynomials for bounded algebraic curve and surface fitting. IEEE PAMI 16:287–303

25. Keren D, Cooper DB, Subrahmonia J (1994) Describing complicated objects by implicit polynomials. IEEE Trans Pattern Anal Mach Intell 16:38–53
26. Sahin T, Unel M (2005) Fitting globally stabilized algebraic surfaces to range data. In: 10th IEEE international conference on computer vision (ICCV'05), October 2005
27. Sahin T, Unel M (2008) Stable algebraic surfaces for 3D object representation. J Math Imaging Vis 32(2):127–137
28. Sahin T, Unel M (2004) Globally stabilized 3L curve fitting. In: Lecture notes in computer science (LNCS-3211), pp 495–502. Springer
29. Wolovich W, Unel M (1998) The determination of implicit polynomial canonical curves. IEEE Trans Pattern Analysis Mach Intell 20(10):1080–1089
30. Unel M, Wolovich WA (1999) A new representation for quartic curves and complete sets of geometric invariants. Int J Pattern Recognit Artif Intell 13(8):1137–1149
31. Unel M, Wolovich WA (2000) On the construction of complete sets of geometric invariants for algebraic curves. Adv Appl Math 24(1):65–87
32. Subrahmonia J, Cooper DB, Keren D (1996) Practical reliable bayesian recognition of 2D and 3D objects using implicit polynomials and algebraic invariants. IEEE Trans Pattern Anal Mach Intell 18(5):505–519
33. Vijayakumar B, Kriegman DJ, Ponce J (1995) Invariant-based recognition of complex curved 3D objects from image contours. In: International conference on computer vision, Boston, MA
34. Wolovich W, Albakri H, Yalcin H (2002) The precise modeling and measurement of free-form surfaces. J Manuf Sci Eng 2002
35. Keren D (1994) Using symbolic computation to find algebraic invariants. IEEE Trans Pattern Anal Mach Intell 16(11): 1143–1149
36. Strang G (1998) Introduction to linear algebra. Wellesley-Cambridge Press
37. Duda RO, Hart PE, Stork DG (2001) Pattern classification. Wiley
38. Sheynin SA, Tuzikov AV (2003) Moment computation for objects with spline curve boundary. IEEE Trans Pattern Anal Mach Intell 25(10): 1317–1322
39. Faber P, Fischer RB (2001) Pros and cons of Euclidean fitting. In: Proceedings of annual German symposium for pattern recognition (DAGM01, Munich). Springer LNCS 2191, Berlin, pp 414–420, Sep 2001
40. Blane M, Lei Z et al (2000) The 3L algorithm for fitting implicit polynomial curves and surfaces to data. IEEE Trans Pattern Anal Mach Intell 22(3):298–313
41. Shilane P, Min P, Kazhdan M, Funkhouser T (2004) The Princeton Shape Benchmark, Shape Modeling International, Genova, Italy, June 2004
42. Zhang J, Siddiqi K, Macrini D, Shokoufandeh A, Dickinson S (2005) Retrieving articulated 3-D models using medial surfaces and their graph spectra. In: International workshop On energy minimization methods in computer vision and pattern recognition
43. Stroustrup B (2000) The C++ programming language, special edn. Addison Wesley, Reading, Mass
44. Numerical Recipes (2007) The art of scientific computing, 3rd edn. Cambridge University Press, ISBN-10: 0521880688
45. Kitware Inc (2009) The visualization toolkit. http://www.vtk.org/
46. Viewer M (2009) http://mview.sourceforge.net/