Recognition of 3D Objects Based on Implicit Polynomials

Hilla Ben-Yaacov, David Malah, *Life Fellow, IEEE*, and Meir Barzohar

Abstract—Closed-form expressions for a new set of 3D rotation invariants that are linear, quadratic, and angular combinations of implicit polynomial (IP) coefficients are developed. Based on these invariants, we propose a 3D object recognition method that outperforms recognition based on IP fitting after pose estimation, and the MPEG-7 SSD technique.

Index Terms—Implicit polynomials, 3D object recognition, tensor contraction, rotation-invariant, 3D object fitting.

1 Introduction

Implicit polynomials (IPs) are used for efficient representation of 2D curves and 3D surfaces specified by discrete data. The ability to efficiently describe complicated boundaries using IP coefficients is attractive for object recognition [1], [2] applications, and might also be used for pose estimation [3].

Over time, different algorithms for fitting IPs to the data were developed. Early fitting algorithms were nonlinear and iterative [4], and often failed to represent or recognize relatively complicated objects. A more recent fitting algorithm is the 3L [5], which solves a Least Squares (LS) problem. However, this algorithm provided coefficients which were not stable enough for object recognition [6]. Another advanced algorithm [7], [8] used a topological approach to obtain IPs which have a simply connected zero set. State-of-the-art fitting algorithms are Gradient-One [6], Min-Max [9], Min-Var [9], and Ridge-Regression [6] (enhanced for the 3D case over the 3L technique in [10]); all of them apply a linear LS solution to the fitting problem as well. The main difference between them and 3L is that these algorithms have additional requirements on the fitting which improve the stability of the IP coefficients and alleviate the problem of spurious zero sets that typically appear in fitting of high-order IPs. Compared with older nonlinear iterative algorithms [4], the LS algorithms have a much better performance in both representation and recognition tasks, with lower complexity. These representation and recognition abilities were previously tested mainly for 2D IPs [9], [11], [12]. As the main research topic of this work is 3D recognition, rather than fitting, we chose to focus on the Gradient-One algorithm only.

Recognition of 3D objects is important in various fields, such as medical imaging, robotics, automatic security systems, etc. Previous approaches usually required pose estimation for the alignment of the new object representation with each of the dictionary object representations. In many approaches, there is also a need to identify matching key points between two object representations. These stages have high computational cost since accurate pose estimation and matching key points are usually iterative (such as the Iterative Closest Point [13] for small angle differences, or the Tensor Voting approach [14]), and each iteration performs calculations for many data points. Also, finding the correct correspondence between key points is very difficult, and its results are usually not accurate enough. An example of such an approach can be found in [15], which needs a large number of spin images (2D histograms around object surface points), from different viewpoints on the object surface, in order to perform surface matching.

The recognition in [1] and [2] is based on nonlinear low-degree (3-4) IP fitting algorithms, which are less stable. In addition, their object recognition performance was obtained for relatively small data sets. In [6], the classifier is based on Gradient-One, which is more stable, but the authors use a single high-degree (6-10) IP fitting, and results are reported for 2D recognition only.

In this work, we use a tensor representation of IPs in order to derive a set of $\lfloor \frac{n}{2} \rfloor + 1$ rotation invariants which are linear combinations of the IP coefficients, as well as $\lceil \frac{n}{2} \rceil$ quadratic and $\frac{\lceil \frac{n}{2} \rceil (\lceil \frac{n}{2} \rceil - 1)}{2}$ angular rotation invariants, from an IP of degree n. We develop new sets of 3D IP rotation invariants (linear, quadratic, and angular), as well as closed-form expressions for these invariants. Following the 2D IP recognition method Multi-Order and Fitting Error Technique (MOFET) [11], [12], we propose a classifier based on these 3D IP rotation invariants, as well as on 3D IP fitting errors, 2D IP rotation invariants and fitting errors (from the most descriptive 2D projection), and the eigenvalues of a Principal Component Analysis (PCA) decomposition. The suggested classifier uses various IP degrees (both for 2D and 3D) in order to utilize both the stability of low-degree IPs and the descriptiveness of high-degree IPs. Our proposed recognition approach is model-based, as the classifier features are combinations of IP coefficients and the IP is a parametric representation of the object surface. We explore the results of our classifier using the Gradient-One fitting algorithm. We apply the proposed method to a database of rigid objects and compare our recognition results with pose estimation methods followed by IP fitting [3] and with the Shape Spectrum Descriptor (SSD), which was adopted by the MPEG-7 standard [16], [17].

This paper is organized as follows: Section 2 describes the rotation invariants derivation. Section 3 deals with preprocessing and feature extraction. Section 4 describes the classifier design. Section 5 describes our recognition results and compares them with other methods. Section 6 summarizes and suggests future research directions.

2 Rotation-Invariant Expressions for 3D Implicit Polynomials

In order to avoid pose estimation in the recognition process, we need to use expressions that are invariant to rotation. Such IP-based expressions have been developed for the 2D case, both analytically [18] and using symbolic computation [1], [19]. Additional 2D IP-based geometric invariants were developed in [20], [21]. However, 3D IP-based rotation invariants were developed by using symbolic computation [1] only. These invariants contain sum of high-degree products of IP coefficients. These products may cause instability due to their high degree, and therefore, we prefer the analytically derived invariants which we will develop in this section for the 3D case, extending those developed for the 2D case in [18].

2.1 Separation of an IP into Forms

A 3D IP:

$$f_n(x,y,z) = \sum_{0 \le k,l,m,k+l+m \le n} a_{klm} x^k y^l z^m \tag{1}$$

can be separated into homogeneous parts $H_r(x,y,z)$, $r = 0, \ldots, n$, where $H_r(x,y,z)$ denotes a homogeneous ternary IP of degree r, also called a *"form"* of degree r, and is defined as follows:

• *The authors are with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel.*
E-mail: hilla.ben.yaacov@gmail.com, malah@ee.technion.ac.il, meirb@visionsense.com.

Manuscript received 10 Sept. 2008; revised 17 Mar. 2009; accepted 30 July 2009; published online 18 Nov. 2009.
Recommended for acceptance by D. Weinshall.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-09-0606.
Digital Object Identifier no. 10.1109/TPAMI.2009.197.

$$H_r(x,y,z) = \sum_{k+l+m=r} a_{klm} x^k y^l z^m. \qquad (2)$$

$H_n(x,y,z)$ is called *"the leading form"* of an IP of degree n. Note that translations leave the leading form unaffected, but affect the rest of the forms [3].

2.2 Derivation of Rotation Invariants Based on Tensor Representation

Our proposed method for 3D rotation invariants derivation is based on the pose estimation method of [3]. In this section, we use the notation (x_1, x_2, x_3), instead of (x,y,z), for convenience.

We start by briefly reviewing the tensor representation suggested in [3] for pose estimation, and we will then use this method for developing 3D rotation invariants.

For pose estimation purposes [3], it is more convenient to use a tensor representation for each form:

$$H_n(x_1,x_2,x_3) = \sum_{i_1=1}^{3}\sum_{i_2=1}^{3}\cdots\sum_{i_n=1}^{3} s^{i_1 i_2 \dots i_n} x_{i_1} x_{i_2} \dots x_{i_n}, \qquad (3)$$

where $(s^{i_1 i_2 \dots i_n})_{1\le i_1,i_2,\dots,i_n\le 3}$ is a symmetric tensor of order n, denoted S_n. It can also be considered as an n-dimensional array with 3^n entries. Each entry $s^{i_1 i_2 \dots i_n}$ can be expressed by the form coefficients:

$$s^{i_1 i_2 \dots i_n} = \frac{a_{klm}}{\frac{n!}{k!l!m!}}, \qquad (4)$$

where k, l, and m are the number of tensor indices $(i_1, i_2, \dots, i_n)$ that are equal to 1, 2, and 3, respectively.

Given a tensor, a contraction with respect to two indices (e.g., i_1 and i_2) is defined as a new tensor of order $n-2$:

$$s'^{i_3 i_4 \dots i_n} = \sum_{i_1=1}^{3} s^{i_1 i_1 i_3 i_4 \dots i_n}. \qquad (5)$$

In other words, we set $i_2 = i_1$, and sum over i_1. A total contraction of an even degree tensor gives a zero-order tensor (a scalar) which is an invariant. For example, for a symmetric 3×3 matrix (tensor of order 2), the tensor contraction gives the trace of the matrix, which is known to be an invariant under euclidean transformations (e.g., rotation).

The IP-based pose estimation suggested in [3] fits an even degree IP with $n = 2p$ and extracts only the leading form $(H_n(x,y,z))$ since this form is invariant to translation. In order to find the intrinsic orientation of the IP, it performs $(p-1)$ tensor contractions on the leading form, resulting in a 3×3 matrix denoted by $V(H_n)$, whose elements are linear combinations of the leading form coefficients.

Now, after the brief review of [3], we explain how to develop a novel set of closed-form 3D rotation-invariant expressions. We suggest adding a pth tensor contraction (i.e., a total of p contractions for an IP of degree $n = 2p$). This will result in a tensor of order 0 which is the trace of $V(H_n)$. For example, for a second degree form, we get:

$$V(H_2) = \begin{bmatrix} a_{200} & \frac{a_{110}}{2} & \frac{a_{101}}{2} \\ \frac{a_{110}}{2} & a_{020} & \frac{a_{011}}{2} \\ \frac{a_{101}}{2} & \frac{a_{011}}{2} & a_{002} \end{bmatrix}. \qquad (6)$$

We denote the 3D linear invariants by $L_{3D,k}$, where k is the form degree. The rotation invariant we get is

$$L_{3D,2} = trace[V(H_2)] = a_{200} + a_{020} + a_{002}. \qquad (7)$$

The advantage of this method is the simplicity with which we can derive an invariant from any even degree form, even for high degrees. For example, for a fourth degree form, the rotation invariant we get is

$$L_{3D,4} = a_{400} + a_{040} + a_{004} + \tfrac{1}{3}(a_{220} + a_{202} + a_{022}). \qquad (8)$$

In the general case of a form of degree $n = 2p$, we developed a general expression for the linear invariant obtained by p tensor contractions of a form of degree $n = 2p$:

$$\begin{aligned} L_{3D,n} &= trace[V(H_n)] = \\ &= \sum_{i_{n-1}=1}^{3} \cdots \sum_{i_3=1}^{3}\sum_{i_1=1}^{3} s^{i_1 i_1 i_3 i_3 \cdots i_{n-1} i_{n-1}}, \end{aligned} \qquad (9)$$

and, using (4), we get:

$$L_{3D,n} = \sum_{\substack{k,l,m\ even\\ k+l+m=n}} \frac{k!l!m!}{n!} \cdot \frac{(n/2)!}{(k/2)!(l/2)!(m/2)!} \cdot a_{klm}, \qquad (10)$$

where $\frac{(n/2)!}{(k/2)!(l/2)!(m/2)!}$ is the number of times each tensor element participates in the tensor contractions (the indexes are divided by 2 since the contraction sets each pair of indexes to be equal). Using the tensor approach, we can derive the complete set of $\lfloor \frac{n}{2} \rfloor + 1$ linear invariants for an IP of degree n, one linear invariant for each even degree form.

We now examine odd degree forms. Each odd degree form $n = 2p+1$, when represented as a tensor, can be contracted p times until we get a first degree tensor (a vector). The squared magnitude of this vector is invariant under rotation. In the same way, the relative angles between these vectors (derived from different forms of the same IP) are also invariant under rotation.

For a first degree form:

$$H_1(x_1,x_2,x_3) = a_{100}x_1 + a_{010}x_2 + a_{001}x_3, \qquad (11)$$

we get the tensor representation (no contraction is needed, $p = 0$):

$$V(H_1) = [a_{100}\ a_{010}\ a_{001}]^T, \qquad (12)$$

and the squared magnitude of this vector is:

$$Q_{3D,1} = \|V(H_1)\|^2_{l_2} = a^2_{100} + a^2_{010} + a^2_{001}. \qquad (13)$$

We denote the 3D quadratic invariants by $Q_{3D,k}$, where k is the form degree. For a third degree form, we get:

$$\begin{aligned} Q_{3D,3} = \|V(H_3)\|^2_{l_2} &= \left[a_{300} + \frac{a_{120}}{3} + \frac{a_{102}}{3}\right]^2 + \\ &+ \left[a_{030} + \frac{a_{210}}{3} + \frac{a_{012}}{3}\right]^2 + \left[a_{003} + \frac{a_{201}}{3} + \frac{a_{021}}{3}\right]^2. \end{aligned} \qquad (14)$$

And the angle between a first degree form and a third degree form will be:

$$\theta_{13} = cos^{-1}\left[\frac{V(H_1)\cdot V(H_3)}{\|V(H_1)\|_{l_2}\|V(H_3)\|_{l_2}}\right], \qquad (15)$$

where $\{\cdot\}$ is the dot product between two vectors.

For an nth degree form ($n = 2p+1$), we get:

$$V(H_n) = \begin{bmatrix} \displaystyle\sum_{\substack{k,l,m\ even,\\ k+1+l+m=n}} \frac{a_{k+1,l,m}}{\frac{n!}{(k+1)!l!m!}} \frac{((n-1)/2)!}{(k/2)!(l/2)!(m/2)!} \\ \displaystyle\sum_{\substack{k,l,m\ even,\\ k+1+l+m=n}} \frac{a_{k,l+1,m}}{\frac{n!}{k!(l+1)!m!}} \frac{((n-1)/2)!}{(k/2)!(l/2)!(m/2)!} \\ \displaystyle\sum_{\substack{k,l,m\ even,\\ k+1+l+m=n}} \frac{a_{k,l,m+1}}{\frac{n!}{k!l!(m+1)!}} \frac{((n-1)/2)!}{(k/2)!(l/2)!(m/2)!} \end{bmatrix}, \qquad (16)$$

$$Q_{3D,n} = \|V(H_n)\|^2_{l_2}. \qquad (17)$$

Using tensor representation, we get $\lceil \frac{n}{2} \rceil$ quadratic invariants from an nth degree IP, one for each odd degree form. Therefore,

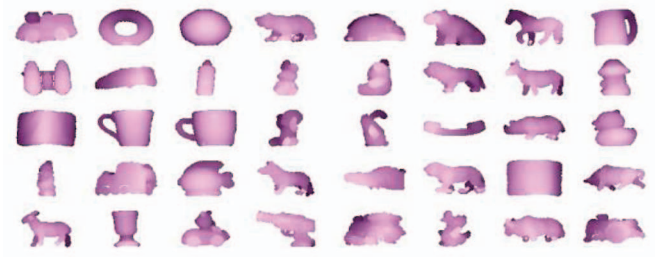
Fig. 1. One position of each of the 40 rigid objects in the database.

we can also derive $\frac{\lceil\frac{n}{2}\rceil(\lceil\frac{n}{2}\rceil-1)}{2}$ angular invariants (one between each possible pair of odd degree forms).

Note that these are explicit expressions for the derivation of 3D invariants, unlike the recursive derivation method given in [18] for 2D invariants. We can extend our explicit 3D invariants expression for the 2D invariants computation as well, in the following way:

$$L_{2D,n} = \sum_{k,l \ \ even, k+l=n} \frac{k!l!}{n!} \cdot \frac{(n/2)!}{(k/2)!(l/2)!} \cdot a_{kl}, \tag{18}$$

and similarly also for the quadratic and angular invariants. In [18], the suggested recursive scheme includes the calculations of the same binomial coefficients as in our proposed method, but in addition, in [18], it is also needed to invert a matrix of $\frac{(n+1)(n+2)}{2}$ on $\frac{(n+1)(n+2)}{2}$ for the derivation of the invariants of a form of degree n. Thus, we need fewer computations than the scheme given in [18] for the 2D linear invariants. Note, however, that if we apply this method in the 2D case, we get the same set of linear invariants, but only part of the quadratic/angular sets described in [18].

2.3 Derivation of Rotation Invariants Based on Trigonometric Expressions

In addition to the previous derivation approach, we now present another derivation method, useful mainly when dealing with low-degree IPs.

Every 3D rotation can be considered as three rotations, one around each axis (x, y, and z) with angles α, β, and γ, respectively [22]. One of the 2D invariants derivation methods described in [18] uses the 2D rotation matrix. Following the same approach in the 3D case will require proving that the expressions are invariant under each of the above rotations. For example, for a second degree IP, the original polynomial is:

$$\begin{aligned} f_2(x,y,z) = {} & a_{000} + a_{100}x + a_{010}y + a_{001}z + a_{200}x^2 + \\ & + a_{110}xy + a_{101}xz + a_{020}y^2 + a_{011}yz + a_{002}z^2. \end{aligned}$$

After substituting the relations between x, y, z and x', y', z' for a rotation around z axis, and rearranging:

$$\begin{aligned} f_2(x',y',z') = {} & \underbrace{a_{000}}_{b_{000}} + \cdots + \underbrace{\left(c^2 a_{200} - cs a_{110} + s^2 a_{020}\right)}_{b_{200}} x'^2 + \\ & + \underbrace{\left(s^2 a_{200} + cs a_{110} + c^2 a_{020}\right)}_{b_{020}} y'^2 + \cdots + \underbrace{a_{002}}_{b_{002}} z'^2, \end{aligned}$$

where $c \triangleq \cos\gamma$ and $s \triangleq \sin\gamma$. Obviously, we have the following invariant:

$$L_{3D,0} = b_{000} = a_{000}. \tag{19}$$

From examination of the new IP coefficients b_{klm} and some trigonometric identities, we find that after z-axis rotation, we also get that:

$$L_{3D,2} = b_{200} + b_{020} + b_{002} = a_{200} + a_{020} + a_{002}. \tag{20}$$

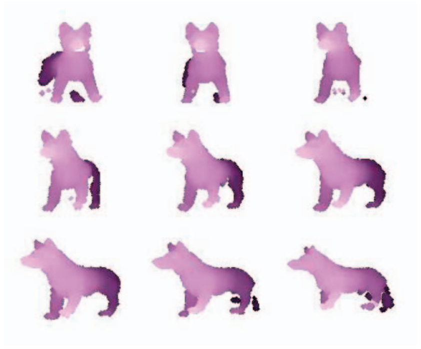
Fig. 2. Nine different positions of the object "fox."

If we follow the same procedure for each of the rotation matrices, we'll see that (19) and (20) are still invariants. This is the same linear invariant we obtained using the previously suggested derivation method in (7). The complexity of the trigonometric expressions grows considerably with the IP degree, and so the rotation matrix together with the trigonometric identities is useful only for low-degree IPs. We used this method for deriving two more quadratic invariants; the first appears in (13) and the second, which cannot be obtained using the tensor method, is (for the full proof, see [23]):

$$\begin{aligned} Q_{3D,2} = {} & a_{200}^2 + a_{020}^2 + a_{002}^2 - 2a_{200}a_{020} \\ & - 2a_{200}a_{002} - 2a_{020}a_{002} + a_{110}^2 + a_{101}^2 + a_{011}^2. \end{aligned} \tag{21}$$

3 Feature Extraction

3.1 Rigid Objects Database

The databases which we found available for 3D object recognition performance evaluation are either synthetic databases (i.e., computer graphic models, such as [24]) or small real databases (only a relatively small number of different objects, such as [25], [26]). Therefore, we created a new objects database, with acquisitions of 40 objects, each in nine different viewing angles.

This database was acquired using the equipment of the Geometric Image Processing (GIP) lab at the Computer Science Faculty of the Technion. The system is based on the "structured light" technique [27]. The projector's hardware is manipulated so that it triggers the camera. The camera frame rate is 30 Hz, and each 3D image is produced from a sequence of 12 consecutive camera frames of 12 different projected patterns. Therefore, the system acquires 2.5 images per second. For the acquisition, the object is placed on a stand, with a black background behind it. After the acquisition, each 3D frame is resampled on a uniform xy grid of 320×240 pixels, then filtered in order to smooth noise artifacts, and the background is cropped. The final number of data points per object in our database is $\sim$12,000 on average. Each pixel has three coordinates and represents a point in the 3D space.

We acquired 40 different objects, each was placed on the rotatable stand and acquired in nine consecutive positions. The difference between two consecutive positions is $\sim$12 degrees, and we acquired five frames in each position of each object. A possible application for such a setup is a factory production line with various products which need automatic sorting. The database objects appear in Fig. 1. Nine different positions of the object "fox" are shown in Fig. 2.

3.2 Preprocessing

Before the feature extraction, we have to apply some preprocessing to each object frame.

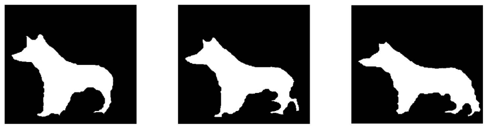
Fig. 3. xy projections of three different positions of the object "fox."

3.2.1 Translation and Scaling

For translation invariance, we locate the center of mass of the data points at the origin [23].

For scale invariance, as in [12], we chose the scaling factor to be the 75th percentile of the distances of the data points from the origin (small changes in the exact percentile hardly affect the results [11]). In other words, we sort the distances of all the data points from the origin, and choose the element that is just larger than 75 percent of the other elements as the scaling factor $S_{75\%}$. We then use it to scale each coordinate.

3.2.2 2D Projections

In order to obtain more features using IPs, we used 2D projections. We used only projections on the main planes (xy, xz, and yz) in order to avoid high-complexity calculations. If the object is acquired from a few viewing points, our experiments show that these projection planes are satisfactory. If we examine 2D projections that are obtained from several different orientations of a 3D object (e.g., Fig. 3), they are connected by a nonlinear projective transform. However, since the viewpoint differences between consecutive orientations are relatively small, there are two possible approaches: The first, to consider the projective transform as an affine transform, and transform each projection into its "mother-shape" [12]. The disadvantage of this approach is that when we deal with very similar objects, along with affine invariance, we lose some of the object properties that distinguishes it from other similar objects. The second approach is to consider this transform as some additional model error that was added to the contour, and perform the IP fitting and invariants calculation without using the "mother-shape." In our experiments, the second approach has shown better results, and so these are the results which we demonstrate in Section 5.

For the rigid objects, the most descriptive projection is the projection on xy. The two other projections (xz and yz) were not very informative, and had very noisy contours. After the 2D projection, we used morphological operators for holes filling and then extracted the 2D contour. An example for xy projection of three different positions of the object "fox" is shown in Fig. 3.

3.3 Selected Features

3.3.1 Linear Invariants

After the preprocessing, we fit IPs of degrees 2, 4, and 6 to the 3D objects, and to their 2D xy projection, using the Gradient-One fitting algorithm. By this approach, we follow the MOFET technique, introduced for 2D contours recognition using IPs [12]. Using the IP representation, we separate the IP into forms, and from each even degree form, we compute the linear rotation invariants described in Section 2 using the explicit expressions that we have developed in (10) and in (18). In both 3D and 2D cases, we have $\lfloor \frac{n}{2} \rfloor + 1$ linear invariants for an IP of degree n.

3.3.2 Quadratic Invariants

For the 3D case, using the same IP representation, we also derived the two quadratic rotation invariants described in Section 2 (see (13) and (21)). The rest of the 3D quadratic invariants, and all of the 3D angular invariants, are not used as features, as they are not stable enough. The quadratic and angular invariants in the 2D case [18] are also not stable enough [12], and therefore, were not used as features.

TABLE 1
Chosen Features

Type	Chosen Features
3D linear invariants	IP degrees 2,4,6 (2+3+4=9 features)
3D IP fitting error	IP degrees 2,4,6 (1+1+1=3 features)
two 3D quadratic invariants $Q_{3D,1}$, $Q_{3D,2}$	IP degrees 2,4 (2+2=4 features)
2D linear invariants	xy - IP degrees 2,4,6 (2+3+4=9 features)
2D IP fitting error	xy - IP degrees 2,4,6 (1+1+1=3 features)
3D PCA eigenvalues	3 eigenvalues (3 features)
Total number of features	31 features

3.3.3 Implicit Polynomial Fitting Errors as Features

Following the MOFET technique [12], we also suggest using the IP fitting error as a feature. Therefore, for each IP fitting degree, after solving the IP fitting LS problem, we calculate the fitting error for each data point and use the 75th percentile of the errors vector as a feature describing the fitting error.

3.3.4 Eigenvalues of 3D PCA

We obtain three more features: the eigenvalues of PCA on the original data points, which are obviously invariant to rotation. In the 3D case, we perform an eigenvalue decomposition of the 3×3 data scatter matrix. We sort the eigenvalues in decreasing order according to their magnitude, and use them as additional features.

Table 1 summarizes the features that were chosen. The exact features were chosen based on their robustness and informativeness. The total number of features is 31.

4 Recognition

4.1 Classifier Design

We chose a classifier that is based on the probability density functions (PDFs) of feature vectors. We assume that the feature vector has a Gaussian distribution. Each multivariate Gaussian PDF is estimated from feature vectors belonging to one or more different views of an object from the dictionary. Thus, each dictionary object is represented by one or more PDFs. In all of the simulations, we used the Gradient-One fitting algorithm.

We divide the objects database into learning and testing data sets in the following way: We have nine different positions for each object (~ 12 degrees difference between consecutive positions, denoted by #1-#9), and for each position, we have five different consecutive frames (in order to gather statistics on the features sensitivity to noise during the learning process). We use even positions for learning and odd positions for testing. Note that, in order to use the proposed method in an operational system, $\sim$15 positions per object are needed for learning in order to recognize the object from every viewing point. Also note that this is a worst-case scenario: We learned each object from positions with a difference of $\sim$24 degrees between them (#2, #4, #6, and #8), and then we tested our performance using the most different viewing positions that we could acquire—in the middle between the angles of the learning positions (#1, #3, #5, #7, and #9).

4.1.1 Learning Positions

For each learning position of each object, we do the following: Each of the five frames is perturbed 10 times by adding colored Gaussian noise (which, according to our analysis [23], is the acquisition noise model). We used an averaging filter of 11×11 (normalized so that the sum of its square coefficients is equal to 1) to filter white

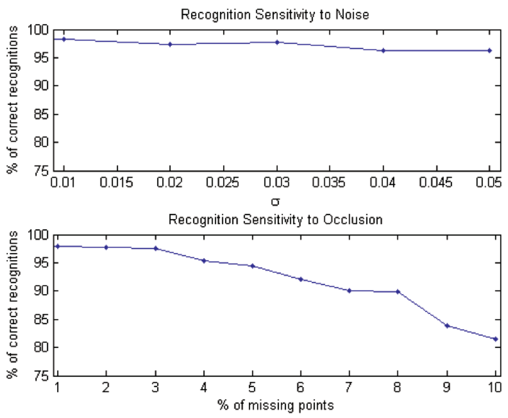

Fig. 4. Analysis of the recognition process sensitivity to noise and occlusion.

Gaussian noise with standard deviation of 0.01. The parameters were chosen empirically according to our noise model analysis. Thus, we have $40 \cdot 4 \cdot 5 = 800$ real frames in the learning data set, synthetically increased 10 times to obtain $800 \cdot 10 = 8{,}000$ frames. For each of the 8,000 perturbed instances, we perform all of the preprocessing stages from the previous section and calculate the feature vector $\underline{v}$.

Let us denote each object by O_k , $k = 1, \dots, 40$, and each learning position (view) by V_n, $n = 2, 4, 6, 8$. We examined several approaches for the classifier design [23]. The best approach, according to our experimental results, is estimating the parameters of a PDF from learning position pairs of each object. For each object, we used the $50 \cdot 2 = 100$ feature vectors of positions #2 and #4 for the estimation of one PDF, the 100 feature vectors of positions #4 and #6 for the second PDF, and the 100 feature vectors of positions #6 and #8 for the third PDF. We used the feature vectors of each learning position pair in order to compute a vector of means $\underline{\mu}$, and a covariance matrix Σ for construction of their PDF:

$$P(\underline{v}/O_k, V_n) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(\underline{v}-\underline{\mu})^T \Sigma^{-1} (\underline{v}-\underline{\mu})\right)}, \tag{22}$$

where $k = 1, \dots, 40$, $n = 2\&4, 4\&6, 6\&8$, and in case we use the entire feature vector, we have $d = 31$.

In this approach, we have $40 \cdot 3 = 120$ different PDFs for the description of 40 objects (three PDFs per object). This approach assumes that positions with very different viewing angles of the same object have very little in common, and therefore, it is better to learn similar positions together, but to separate the learning of different positions.

4.1.2 Testing Positions

We calculate the same feature vector for each of the five frames of each object testing position, resulting in $40 \cdot 5 \cdot 5 = 1{,}000$ real frames in the testing data set. We then calculated the probability of it originating from each of the objects. The object which resulted in the highest probability was chosen as the best recognition for the test vector.

We want to compare the probabilities of each object O_k in each position combination V_n, given the observation $\underline{v}$:

$$P(O_k, V_n/\underline{v}), \; k = 1, 2, \dots, 40, \; n = 2\&4, 4\&6, 6\&8. \tag{23}$$

Using Bayes rule, we get:

$$\begin{aligned} P(O_k, V_n/\underline{v}) &= \frac{P(\underline{v}/O_k, V_n)P(O_k, V_n)}{P(\underline{v})} = \\ &= \frac{P(\underline{v}/O_k, V_n)P(V_n/O_k)P(O_k)}{P(\underline{v})}, \end{aligned} \tag{24}$$

TABLE 2
Results of Object Recognition Using Different Sets of Features: Entire Feature Vector ($d = 31$); IP-Based Features Only ($d = 28$); 3D IP-Based Features Only ($d = 16$), and 3D PCA Eigenvalues Only ($d = 3$)

$d = 31$	$d = 28$	$d = 16$	$d = 3$
98.8%	96.9%	96.1%	74.2%

where $P(\underline{v})$ is the same for every k, and we assume that $P(O_k) = \frac{1}{40}$, and $P(V_n/O_k) = \frac{1}{3}$. Therefore, we can ignore these expressions in the probabilities comparison and compare only $P(\underline{v}/O_k, V_n)$, $k = 1, \dots, 40$, $n = 2\&4, 4\&6, 6\&8$.

5 Experimental Results

5.1 Synthetic Simulations

As we mentioned in Section 3.1, we have a difference of about 12 degree between consecutive positions (i.e., in each testing position, around 12 degrees/360 degrees $\simeq$ 3 percent of the closest learning positions' data points are missing), and an acquisition noise model of colored Gaussian noise with a standard deviation of about 0.01.

In order to analyze the sensitivity of the entire recognition process, we performed the following two synthetic simulations:

The first: In the learning stage, we used colored Gaussian noise, as described in Section 4.1.1, each time with a different standard deviation (0.01-0.05). Then, in the testing stage, we added to each testing frame of each object, colored Gaussian noise with the same standard deviation as in the learning stage. The recognition results appear at the top of Fig. 4. Note that the noise was added *in addition* to the acquisition noise which already exists in the testing frames.

The second: The learning stage was the same as described in Section 4.1.1. Then, in the testing stage, we chose various percentage values of points (1-10 percent) to eliminate from each testing frame of each object (a central point was randomly selected, then an appropriate number of closest points were eliminated). The results appear at the bottom of Fig. 4. Note that these data points were eliminated *in addition* to the occlusion which already exists in the testing frames.

It can be seen that, as the standard deviation of the noise or the percentage of missing points becomes larger, the recognition process performance is reduced relatively slowly.

5.2 Recognition Results

Table 2 shows the recognition results using different sets of features. By IP-based features only, we refer to 3D/2D IP-based invariants and fitting errors, and by 3D IP-based features only, we refer to 3D IP-based invariants and fitting errors. It can be seen that without the additional PCA features, the results are less good, but we still manage to classify correctly almost 97 percent of the instances. Note that, even without the 2D/PCA-based features, we classify correctly over 96 percent of the instances. For comparison, using the 3D PCA features only, resulted in correct classification of only 74.2 percent of the instances.

5.2.1 Comparison with Other Methods

We compared our proposed method results with pose estimation recognition results. The pose estimation methods we examined are PCA- and IP-based tensor approaches [3]. In both cases, after the rotation of the data points, we fit an IP and use the IP coefficients as our features. We then use an l_2 distance between feature vectors for classification. The IP degree we chose is 4 ($d = 35$), since in this case, we use a single-degree IP, and higher degree IP coefficients were less stable and showed poor recognition performance. Note that our proposed invariants are *combinations* of IP coefficients, and they were found to be useful even for IPs with degree larger than 4.

TABLE 3
Comparison between the Proposed Method, Pose Estimation, and the SSD Method for Object Recognition

Method	Results
Proposed method ($d = 31$)	**98.8%**
PCA based pose estimation (4^{th} degree IP, $d = 35$)	91.6%
IP based pose estimation (4^{th} degree IP, $d = 35$)	90.5%
SSD ($d = 27$)	90.1%

We used IP-based pose estimation with a second degree IP (i.e., the second degree IP was used for pose estimation and then a fourth degree IP was fitted and its coefficients were used as features). The pose estimation results using Gradient-One appear in Table 3. It can be seen that the PCA has a similar performance to the second degree IP pose estimation. It can also be seen that our proposed method results with $d = 31$ (98.8 percent) are better than the PCA/IP-based pose estimation methods.

We also compared the performance of the proposed method with the SSD technique [16]. This technique was adopted by the MPEG-7 standard for 3D descriptors and has a relatively low complexity. We used a histogram of 25 bins for the descriptor (the default for MPEG-7 is 100 bins, but its results were found to be a little worse) and we also used the singular and planar descriptors (i.e., we had the 25 bins plus the two descriptors, a total of 27 features). We used the l_1 norm on the difference between the feature vectors for classification. We implemented the SSD technique in Matlab. Our implementation is based on [16] and on the freely available MPEG-7 reference software [28].

The results of this comparison also appear in Table 3. It can be seen that our proposed method has better performance. The computational complexity of the methods is considered in the next section.

5.2.2 Comparison of Computational Complexity

All three methods, our proposed method, pose-estimation-based techniques, and the SSD technique, have *similar computational complexity*. If we denote the number of object data points by N, then each of the stages of each method has a complexity of $O(N)$. (The SSD stages are actually dependent on the number of triangles, which in our databases is around $2N$ for an object with N data points.) The constant that multiplies N is large in some stages (such as the Least-Squares IP fit in our case and in the pose estimation case, or the second degree explicit polynomial fit at each point in the SSD case), but is difficult to estimate. Average running times for the recognition of a single object, using a Matlab implementation, appear in Table 4. The proposed method appears to have a commensurate running time with the other two methods, while presenting better results on the examined database.

6 Summary and Conclusions

In this work, we examined the recognition abilities of 3D IPs using the Gradient-One fitting algorithm. We introduced new sets of 3D rotation invariants (linear, quadratic, and angular), which are based on IPs and their tensor representation and obtained closed-form expressions for these invariants for every IP degree.

We also developed one more quadratic invariant based on IP properties and trigonometric identities. After some preprocessing, we followed the 2D IP recognition method known as MOFET [12] and suggested the use of a feature vector that contains 3D IP rotation invariants and fitting errors, 2D IP rotation invariants and fitting errors (from a 2D projection), and the eigenvalues of a PCA decomposition (total of 31 features). The feature vector included invariants of several IP degrees (both for 2D and 3D) in order to utilize both the stability of low-degree IPs and the descriptiveness of high-degree IPs.

TABLE 4
Average Running Times of Recognition of a Single-Object (Matlab Implementation)

proposed method ($d = 31$)	pose estimation+IP coef. (PCA/4^{th}degree IP, $d = 35$)	SSD ($d = 27$)
13 Sec	15 Sec	55 Sec

We designed a PDF-based classifier and showed that the 3D IP invariants followed by fitting error approach has better performance compared with pose estimation methods followed by IP fitting [3]. We also found in our tests that our proposed method outperforms the Shape Spectrum Descriptor technique [16], which was adopted by the MPEG-7 standard for 3D descriptors.

Future work could consider other fitting algorithms, such as the Ridge-Regression of [6], [10], or the Min-Max/Min-Var algorithms of [9] and explore their recognition results with the newly derived invariants. Another possibility is exploring the use of quaternions [29] for the derivation of a full set of quadratic rotation invariants, in a similar way to the use of complex representation in the 2D case [18]. In addition, the proposed method could be examined for face recognition tasks.

Acknowledgments

The authors would like to thank the Geometric Image Processing (GIP) lab at the Computer Science Faculty of the Technion for using their equipment to acquire the rigid objects database. The authors would also like to thank Zoya Landa for providing the Matlab implementation of the 2D MOFET approach. The helpful comments by the anonymous reviewers are gratefully acknowledged.

References

[1] J. Subrahmonia, D.B. Cooper, and D. Keren, "Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 5, pp. 505-519, May 1996.

[2] M. Barzohar, D. Keren, and D. Cooper, "Recognizing Groups of Curves Based on New Affine Mutual Geometric Invariants, with Applications to Recognizing Intersecting Roads in Aerial Images," *Proc. IAPR Int'l Conf. Pattern Recognition,* vol. 1, pp. 205-209, Oct. 1994.

[3] J. Tarel, H. Civi, and D. Cooper, "Estimation of Free-Form 3D Objects Without Point Matching Using Algebraic Surface Models," *Proc. IEEE Workshop Model-Based 3D Image Analysis,* pp. 13-21, 1998.

[4] G. Taubin, "Estimation of Planar Curves, Surfaces and Nonplanar Space Curves Defined by Implicit Equations, with Applications to Edge and Range Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 11, pp. 1115-1138, Nov. 1991.

[5] M.M. Blane, Z. Lei, H. Civi, and D.B. Cooper, "The 3L Algorithm for Fitting Implicit Polynomial Curves and Surfaces to Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 3, pp. 298-313, Mar. 2000.

[6] T. Tasdizen, J. Tarel, and D. Cooper, "Improving the Stability of Algebraic Curves for Applications," *IEEE Trans. Image Processing,* vol. 9, no. 3, pp. 405-416, Mar. 2000.

[7] D. Keren and C. Gotsman, "Fitting Curves and Surfaces with Constrained Implicit Polynomials," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 1, pp. 31-41, Jan. 1999.

[8] D. Keren, "Topologically Faithful Fitting of Simple Closed Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 1, pp. 118-123, Jan. 2004.

[9] A. Helzer, M. Barzohar, and D. Malah, "Stable Fitting of 2D Curves and 3d Surfaces by Implicit Polynomials," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 10, pp. 1283-1294, Oct. 2004.

[10] T. Sahin and M. Unel, "Stable Algebraic Surfaces for 3D Object Representation," *J. Math. Imaging and Vision,* vol. 32, no. 2, pp. 127-137, Oct. 2008.

[11] Z. Landa, "2D Object Description and Classification Based on Contour Matching by Implicit Polynomials," MSc thesis, The Technion-Israel Inst. of Technology, http://sipl.technion.ac.il/siglib/FP/Zoya_Landa.pdf, Aug. 2006.

[12] Z. Landa, D. Malah, and M. Barzohar, "2D Object Description and Recognition Based on Contour Matching by Implicit Polynomials," CCIT Report #755, Electrical Eng. Dept., Technion, Feb. 2010.

[13] P. Besl and N. McKay, "A Method for Registration of 3D Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 239-256, Feb. 1992.
[14] P. Mordohai and G. Medioni, "Dimensionality Estimation and Manifold Learning Using Tensor Voting," http://iris.usc.edu/medioni/download/ndmanual.htm, 2009.
[15] A.E. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 5, pp. 433-449, May 1999.
[16] T. Zaharia and F. Preteux, "3D Shape-Based Retrieval within the Mpeg-7 Framework," *Proc. SPIE Conf. Nonlinear Image Processing and Pattern Analysis,* pp. 133-145, Jan. 2001.
[17] C. Dorai and A.K. Jain, "Shape Spectrum Based View Grouping and Matching of 3D Free-Form Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 10, pp. 1139-1145, Oct. 1997.
[18] J. Tarel and D. Cooper, "The Complex Representation of Algebraic Curves and Its Simple Exploitation for Pose Estimation and Invariant Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 7, pp. 663-674, July 2000.
[19] D. Keren, "Using Symbolic Computation to Find Algebraic Invariants," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 11, pp. 1143-1149, Nov. 1994.
[20] M. Unel and W.A. Wolovich, "A New Representation for Quartic Curves and Complete Sets of Geometric Invariants," *Int'l J. Pattern Recognition and Artificial Intelligence,* vol. 13, no. 8, pp. 1137-1149, Dec. 1999.
[21] M. Unel and W. Wolovich, "On the Construction of Complete Sets of Geometric Invariants for Algebraic Curves," *Advances in Applied Math.,* vol. 24, no. 1, pp. 65-87, Jan. 2000.
[22] G.B. Arfken and H.J. Weber, *Mathematical Methods for Physicists.* Elsevier, 2005.
[23] H. Ben-Yaacov, "3D Object Description and Classification by Implicit Polynomials," MSc thesis, The Technion-Israel Inst. of Technology, http://sipl.technion.ac.il/siglib/FP/Hilla-Ben-Yaacov.pdf, Sept. 2008.
[24] "Stuttgart Range Image Database," http://range.informatik.uni-stuttgart.de/htdocs/html/, 2009.
[25] S. Rusinkiewicz, D. DeCarlo, A. Finkelstein, and A. Santella, "Suggestive Contour Gallery," http://www.cs.princeton.edu/gfx/proj/sugcon/models/, 2009.
[26] "The Digital Michelangelo Project," www.graphics.stanford.edu/data/dmich-public/, 2009.
[27] D. Scharstein and R. Szeliski, "High-Accuracy Stereo Depth Maps Using Structured Light," *Proc. 2003 IEEE CS Conf. Computer Vision and Pattern Recognition,* vol. 1, I-195-I-202, June 2003.
[28] MPEG-7 Implementation Studies Group, "Technology—Multimedia Content Description Interface—Part 6: Reference Software, ISO/IEC FCD 15938-6/N4006, MPEG-7," Mar. 2001.
[29] B. Horn, "Closed-Form Solution of Absolute Orientation Using Unit Quaternions," *J. Optical Soc. Am.,* vol. 4, pp. 629-642, Apr. 1987.