Pose Estimation of Free-Form 3D Objects without Point Matching using Algebraic Surface Models

Jean-Philippe Tarel[†⊥]
[⊥]INRIA
Domaine de Voluceau, Rocquencourt,
B.P. 105, 78153 Le Chesnay Cedex, France.

Hakan Çivi[†‡]
[‡]Department of Industrial Engineering
Bogazici University
Bebek, Istanbul 80815, Turkey

David B. Cooper[†]
[†]LEMS, Division of Engineering
Brown University, Box D
Providence, RI, 02912-9104, USA

Abstract

This paper presents new approaches to two fundamental 3D vision problems: 3D surface representation, and fast extraction of geometric information from this representation, particularly 3D alignment. For surface representation by implicit algebraic surfaces, a non-iterative, robust, repeatable and approximately least squares fitting algorithm, 3D 3L, is proposed. The alignment problem is solved by formulating an intrinsic coordinate system determined by tensor contractions of the surface representation parameters.

Keywords:*pose estimation, 3D implicit surfaces, tensor analysis, implicit surface fitting*

1. Introduction

3D pose estimation involves determining the underlying coordinate transformation between two occurrences of an object, each in different position. In 3D, the coordinate transformation is usually a rigid motion composed of a rotation and a translation. Among several approaches proposed to solve the problem are scatter matrices [6], high degree moments [5], least-squares formulations based on a set of point correspondences [7], superquadrics [10], and model-based approach using a point set and a surface model [2]. These methods either require an accurate point correspondence (as in least squares methods) or are sensitive to occlusion (as in scatter matrix) or are iterative (as in model-based approach), or have limited representation power (as with superquadrics) and would not be effective when objects are spherically or cylindrically symmetric plus some bumps as in figure 4(a).

This paper presents new powerful approaches to two fundamental problems in 3D computer vision. First is the representation of 3D surface data by models described by modest numbers of parameters which can be fit to data with simple parallel processing. Second is the fast extraction of geometric information from these models, e.g., intrinsic coordinate systems which permit accurate 3D alignment with one shot rather than iterative computation. More specifically, the results presented are the following. A 3D implicit polynomial surface is fit to a data set which can represent a closed surface or an open 3D surface patch. The fitting is linear, approximately least-squares with the result that the computation is modest and non-iterative. The proposed fitting algorithm is a 3D version of the 3L fitting algorithm which was first used for 2D shapes with satisfying results and is extended here to the 3D case. With the algorithm, the polynomial coefficients are much more stable and repeatable, than are the results with other current fitting methods, under small changes in the data sets to be fit. Fitting is done using polynomials with degrees ranging from 1 to 12 so that either very crude or fairly high resolution shape information can be captured with a single polynomial. Using this model, to compute the shape alignment an intrinsic coordinate system is formed by computing a center and a set of three coordinate axes for the implicit surface. This is a one shot computation, and the resulting coordinate system can then be aligned by a single computation with an analogously computed intrinsic coordinate system for another 3D shape. The two aligned 3D representations or data sets can then be compared very accurately at low computational cost with our methods. Our intrinsic coordinate system is determined by tensor contractions [5]. Its desirable properties are that the covariant parameters that result and are used are linear combinations of the coefficients of the fitted polynomial and seem to be more stable and repeatable in the presence of noise than are parameters used in previous approaches.

The representations and methodology developed appear to be ideally suited to the following uses:

1. A single IP (implicit polynomial) can represent 3D surface data "well" if the surface shape is of moderate complexity, and it can provide a meaningful approximation, for many purposes including pose estimation to a complex shape. For example, the 6th degree polynomial in Figure 4 provides a good representation for the given human heart data. Furthermore, essentially all the time required for this fitting is in the computation of 84 monomials for each data point. With efficient programs, this should be less – perhaps considerably less – than a few seconds on a fast computer. Since each point is processed separately, this can also be done on a signal processing chip. Figure 1(b)-(c) illustrate a 3D data set of a human head and a 10th degree IP representation. IP models are useful for visualization and for measuring shape geometry parameters of the represented object at a level of resolution appropriate to the representation.

2. These representations are useful for comparing two shapes. Since our IP fitting approach is to fit an IP $f(x, y, z)$ to the distance transform $d(x, y, z)$ of the data set Γ_0 in the vicinity of Γ_0, for any new data point $(\tilde{x}, \tilde{y}, \tilde{z})$ $d(\tilde{x}, \tilde{y}, \tilde{z})$ gives the approximate distance of $(\tilde{x}, \tilde{y}, \tilde{z})$ to Γ_0. Given two 3D data sets Γ_0 with K points and $\hat{\Gamma}_0$ with $\hat{K}$ points along with $f(x, y, z)$ and $\hat{f}(x, y, z)$, the respective polynomial fittings, the comparison of the shapes can be done in a number of ways, two of which are the following: (i) Compute the approximate sum of squared distances from the points in Γ_0 to the IP representation $\hat{f}(x, y, z)$ for $\hat{\Gamma}_0$, or vice versa, i.e.,

$$\frac{1}{K} \sum_{(x,y,z)\in\Gamma_0} \hat{f}^2(x, y, z)$$

(ii) Compute the approximate sum of squared distances between smooth approximations f and $\hat{f}$ to Γ_0 and $\hat{\Gamma}_0$,

$$\frac{1}{K + \hat{K}} \sum_{(x,y,z)\in\Gamma_0\cup\hat{\Gamma}_0} [f(x, y, z) - \hat{f}(x, y, z)]^2$$

Note that this particular approximate distance measure, which we call PIMs (Polynomial Interpolated Measure), will be highly accurate even though corresponding coefficients of the two IPs may be very different. Both measures (i) and (ii) can be expressed as Mahalanobis distances in the polynomial coefficients (see [9] for details for the 2D case)

3. A complex object can be represented at arbitrarily fine resolution with low degree IPs by representing the data by overlapping polynomial patches, each covering only a portion of the object surface. Alignment of an entire object can be done in terms of the intrinsic coordinate systems for its patches.

4. Since the IP representations are coordinate independent, the computational cost of tracking a data set which is moving and deforming is very low. Hence, IP patches are an ideal representation for representing, tracking and registering a moving organ in medical data analysis [1].

5. For the preceding reasons, IP patches are ideal for many applications such as object recognition, positioning, and metrological inspection for manufacturing automation, and indexing into 3D databases.

2. Implicit Polynomial Model

Many 3D problems are still relatively difficult as they involve processing of huge volumes of data. A generic model fitted to data sets can dramatically ease the required processing and simplify several 3D problems. In computer vision, objects in 3D images are mostly described by their *surfaces*. Ellipsoids, quadratic surfaces, and super-quadric are used as surface models. In essence, these surfaces are special forms of the more generic *implicit algebraic surfaces*. An implicit algebraic surface is defined as the zero set of an implicit polynomial in 3 variables x_1, x_2, x_3. More formally, a 3D IP surface of degree n given by the following equation:

$$\begin{aligned} f_n(x_1, x_2, x_3) &= \sum_{0\le i,j,k;\, i+j+k\le n} a_{ijk}{x_1}^i{x_2}^j{x_3}^k \\ &= \underbrace{a_{000}}_{H_0} + \underbrace{a_{100}x_1 + a_{010}x_2 + a_{001}x_3}_{H_1(x_1,x_2,x_3)} + \\ &\underbrace{a_{200}{x_1}^2 + a_{110}x_1x_2 + a_{101}x_1x_3 + a_{020}{x_2}^2 + a_{011}x_2x_3 + a_{002}{x_3}^2}_{H_2(x_1,x_2,x_3)} \\ &+ \ldots + \underbrace{a_{n00}{x_1}^n + \ldots a_{0n0}{x_2}^n + \ldots + a_{00n}{x_3}^n}_{H_n(x_1,x_2,x_3)} \\ &= \sum_{r=0}^{n} H_r(x_1, x_2, x_3) = 0, \end{aligned} \tag{1}$$

Here $H_r(x_1, x_2, x_3)$ is a *homogeneous ternary polynomial (called a form)* of degree r in x_1, x_2, and x_3. Notice that in equation 1 the graded lexicographic ordering on monomials induced by $x_3 \prec x_2 \prec x_1$ is applied.

There are two useful representations for implicit polynomial surfaces:

- *Vector form*: (^{t} denotes matrix transpose)

$$f_n(x_1, x_2, x_3) = \mathbf{Y}^t \mathbf{A}, \tag{2}$$

where

$$\mathbf{A} = [a_{000} \quad a_{100} \quad a_{010} \ldots a_{n00} \ldots a_{0n0} \ldots a_{00n}]^t \tag{3}$$

and

$$\mathbf{Y} = [1 \quad x_1 \quad x_2 \quad x_3 \ldots {x_1}^n \ldots {x_2}^n \ldots {x_3}^n]^t \tag{4}$$

- *Tensor form*: when a new component $x_4 = 1$ is added to every 3D point (x_1, x_2, x_3) and the polynomial is rewritten as a homogeneous polynomial in 4 variables:

$$f_n(x_1, x_2, x_3, x_4) = \sum_{0 \leq i,j,k,l; i+j+k+l=n} a_{ijkl} {x_1}^i {x_2}^j {x_3}^k {x_4}^l$$

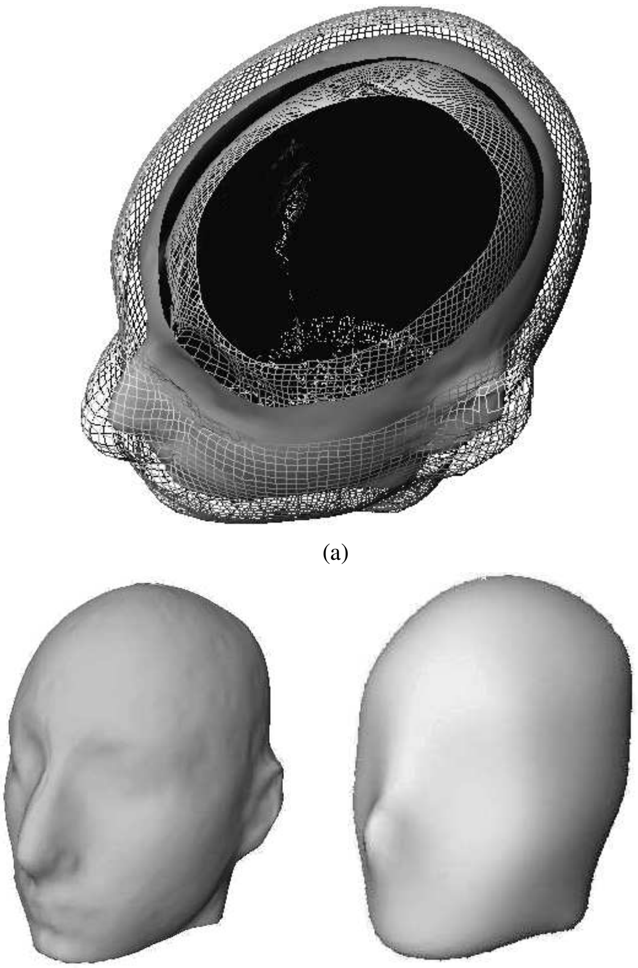
the above sum can be written in an unique way in the following form:

$$f_n = \sum_{i_1=1}^{4} \sum_{i_2=1}^{4} \cdots \sum_{i_n=1}^{4} b^{i_1 i_2 \cdots i_n} x_{i_1} x_{i_2} \cdots x_{i_n} \tag{5}$$

where $b^{i_1 i_2 \cdots i_n}$ equals a_{ijkl} divided by $\frac{n!}{i!j!k!l!}$, the multinomial coefficient in the expansion of the form $(x_1+x_2+x_3+x_4)^n$. This equation defines the covariant tensor $B_n = (b^{i_1 i_2 \cdots i_n})_{1 \leq i_1, i_2, \ldots, i_n \leq 4}$ of order n representing the polynomial (see Section 4.1). Note, here the tensor B is an n-dimensional array with 4^n entries $b^{i_1 i_2 \ldots i_n}$.

The vector form, as described in the next section, is very convenient for implicit polynomial fitting, while the tensor form provides a useful framework for pose estimation as shown in section 4.

Given $\Gamma_0 = \{(x_{1m}, x_{2m}, x_{3m}) | m = 1, ..., K\}$ a set of data points along an object surface, an implicit polynomial is said to represent this object surface if every point of Γ_0 is close to the zero set $\mathcal{Z}(f) = \{(x_1, x_2, x_3) | f(x_1, x_2, x_3) = 0\}$ of the implicit polynomial. Given such a data set Γ_0, an implicit polynomial representation is obtained through a fitting algorithm.



(a)

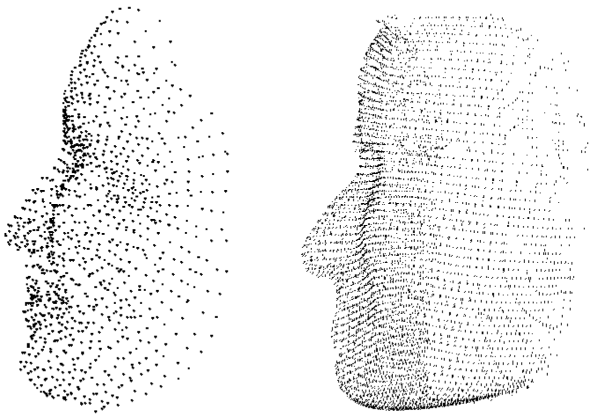
(b) (c)

Figure 1. (a) Level sets for head (b) Head data set (c) 10th degree fit (ears discarded).

3. Implicit Polynomial Fitting

The IP fitting problem can be set up as follows. Given a data set $\Gamma_0 = \{p_m = (x_{1m}, x_{2m}, x_{3m}) | m = 1, .., K\}$, find the nth degree implicit polynomial $f_n(x_1, x_2, x_3)$ that minimizes the average squared distance from the data points to the zero set $\mathcal{Z}(f)$ of the polynomial [12]. When *the geometric distance* from a point to the zero set of an implicit polynomial is minimized, an iterative process is needed because there is no explicit expression for this distance. This formulation requires nonlinear optimization. Several geometric distance approximations have been used, such as the first order approximation [12], which speed up computation considerably, but iterative nonlinear optimization is still required.

In the next section we present the 3L algorithm, a linear fitting algorithm which is of lower computation and has better polynomial estimated coefficient repeatability than all presently existing IP fitting methods.



(a) (b)

Figure 2. (a) Face (b) 12th degree fit.

3.1. 3L Fitting

The polynomial $f(x_1, x_2, x_3)$ is an explicit function at all values of x_1, x_2, x_3 and usually fitting formulations take into account only Γ_0. We get fast, stable, repeatable implicit polynomial surface fits by fitting the explicit polynomial $f(x_1, x_2, x_3)$ to a portion of the distance transform, $d(x_1, x_2, x_3)$, of Γ_0. $d(x_1, x_2, x_3)$ is the function which, at (x_1, x_2, x_3), takes on the value of the signed distance from (x_1, x_2, x_3) to Γ_0. 3L fitting, besides the original data set Γ_0, uses a pair of synthetically generated data sets Γ_{+c} and Γ_{-c} consisting of points at a distance c to either side of Γ_0. Note, Γ_{+c} and Γ_{-c} are the level sets of $d(x_1, x_2, x_3)$ at levels $+c$ and $-c$, respectively. $d(x_1, x_2, x_3)$ can be generated by a distance transform computation algorithm from Γ_0 or as described in the next subsection. For each data point in Γ_0, the Euclidean distance transform determines a point in Γ_{+c} and another one in Γ_{-c} which are at a perpendicular distance c to each side of the original curve Γ_0 [4]. Figure 1(a) shows the 3 level sets for a head shape in 3D (surface defined by 12640 points).

Let $\Gamma_0 \cup \Gamma_{+c} \cup \Gamma_{-c} = \{(x_{1m}\ x_{2m}\ x_{3m})^t : 1 \leq m \leq 3K\}$ and

$$M = [\mathbf{Y}_1\ \mathbf{Y}_2\ \ldots\ \mathbf{Y}_{3K}]^t$$

where $\mathbf{Y}_m$ is $\mathbf{Y}$ (in equation 4) evaluated at $p_m = (x_{1m}, x_{2m}, x_{3m})$. Also define $\mathbf{d}$ as a vector whose mth component is $d(x_{1m}, x_{2m}, x_{3m})$, the distance of the point p_m to Γ_0. As previously explained, the level sets we use for $d(x_{1m}, x_{2m}, x_{3m})$ are only -c, 0, and +c. Then estimating the vector of polynomial coefficients $\mathbf{A}$ (in equation 2) is minimization of $\sum_{m=1}^{3K}(d(x_{1m}, x_{2m}, x_{3m}) - \mathbf{Y}_m^t\mathbf{A})^2$, or $||\mathbf{MA} - \mathbf{d}||^2$. The least squares solution to this problem is:

$$\mathbf{A} = (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{d} \tag{6}$$

The purpose of introducing the two level sets as additional constraints is because it makes the fitting more stable and consistent with regard to transformation of data sets, and more robust to noisy or missing data. The pose estimator discussed in following sections particularly relies on consistency and robustness of fitting under Euclidean transformations and occlusion.

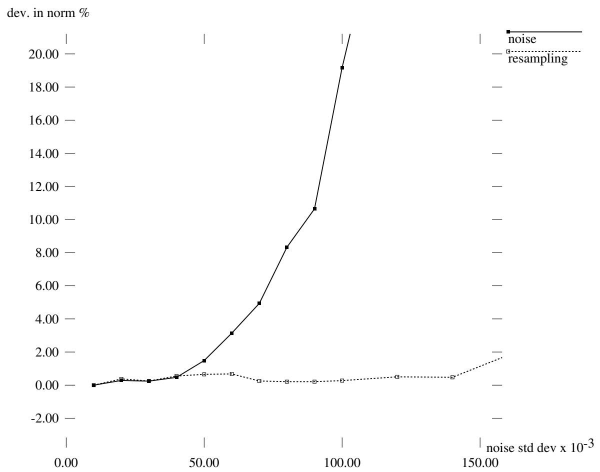
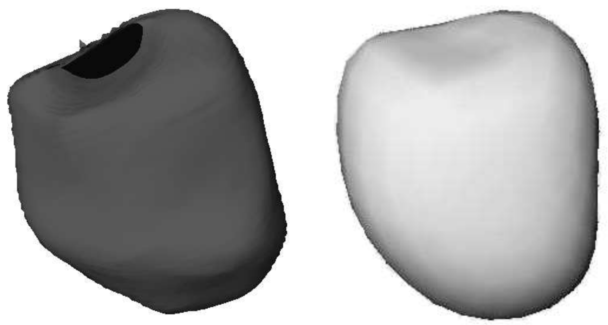


Figure 3. Relative error on the fit, in norm, due to perpendicular noise and resampling of the surface.

The 3L fitting is robust to the addition at all data set points of a random Gaussian noise in the direction perpendicular to the shape as shown in figure 3 in the case of the heart shape of figure 4. Robustness to random resampling is excellent. If a data set is too complicated to be fit accurately by a polynomial of the degree being used, then by choosing c appropriately, a meaningful smooth approximation can be obtained. This is useful for initial pose estimation and may be useful for initial low resolution object recognition.

The level sets bring in the additional benefit of forcing singularities away from the vicinity of the data set, as singularities occur at local extrema or saddle points. We refer the reader to [3] for additional information.

Figure 2 shows the data set and the 12th degree implicit surface fit for a face (with 1250 data points). Figure 4 displays the data set and the 6th degree fit for a heart (with 6480 data points). These figures along with Figure 1 illustrate the power of polynomials in representing complex shapes which are either open or closed.



(a) (b)

Figure 4. (a) Heart (b) 6th degree fit.

3.2. Level Set Generation

If the surface data is not sparse and is closed, a 3D Euclidean distance transform can be applied to obtain the level sets. However this is not the case with most 3D data sets. Moreover, computing the 3D Euclidean distance transform is computationally intensive. Other level set generation techniques which work in 2D usually fail in 3D. The principle dissimilarity between 2D and 3D lies in the fact that in 2D, the *edge chain*, a set of points forming the contour of an object, whether it is a closed or not, has a natural ordering: it can be traversed in clockwise or counterclockwise direction. Thus, vector difference of adjacent points can be used for defining an absolute normal direction, i.e., one known to be pointing towards inside or outside the curve, for all points on the curve.

In 3D, some data acquisition techniques enable the direct estimation of local topology and surface normals [13]. However in general, data consist of a set of unorganized data points. The 3D equivalent of an edge chain is a triangulation of the captured data surface. This procedure is costly and sometimes inaccurate as compared to contour tracing algorithms generating an edge chain in 2D [8]. Our approach is to fit a plane to a set of points comprising the data point and the neighboring points within a given radius from the data point. In this case, the fitted plane is an approximation to the plane tangent to the surface in the vicinity of the data point and the plane normal can be used. However since the reference frame is not known, the plane normal does not give an absolute direction. To resolve this ambiguity, for a star shape an additional reference point known to be inside or outside the surface can be used. The ray $\mathbf{R}$ joining the reference point and the data point along with the tangent plane normal $\mathbf{N}$ gives the absolute orientation. More precisely, if $\mathbf{N} \cdot \mathbf{R} < \mathbf{0}$, vectors $\mathbf{N}$ and $\mathbf{R}$ point towards opposite directions with respect to the tangent plane and towards the same direction otherwise. In our experiments, we utilized generalized eigenvector fitting for fitting local tangent planes and used the plane normal $\mathbf{N}$ along with the center of mass as the additional reference point. An alternative approach, which does not involve any additional reference point, is to benefit from the surface continuity. With this approach, for a given seed point the surface normal is computed by fitting a tangent plane and an absolute orientation is chosen arbitrarily [8]. Then, since adjacent points would have nearly parallel normals, absolute orientations of neighboring planes can be obtained by propagation from already determined ones. For our purposes, as long as the tangent plane orientations are consistent, the initial orientation can be arbitrary because, in the 3L fitting, a change in the choice of the initial orientation amounts only to a reversal of the sign of $f(x_1, x_2, x_3)$ for Γ_{+c} and Γ_{+c} which leaves the zero set of the resulting implicit polynomials the same while the coefficient vectors would differ in sign only.

4. Pose estimation

Given f_n and f'_n, two n^{th} degree IP surfaces representing the same free-form object in two different positions, how can the pose estimation be carried out in a non-iterative way? To our knowledge, the only answer to the above question is the one put forth by Taubin [11] which is effective. The approach in our paper further evolves some basic ideas in [11], appears to provide greater accuracy, and is a step in the process of trying to construct the most accurate pose estimation based on all the information in the polynomial coefficients. The proposed solution is based on the construction of two covariants providing an intrinsic reference system associated with the polynomial. One of these covariants is a vector C with 3 components each component being a polynomial function in the coefficients of forms H_n and H_{n-1} in f_n. The vector C is called the intrinsic center of the polynomial and used to estimate the object location. The second covariant is a $\mathbf{3} \times \mathbf{3}$ symmetric matrix V and each component of this matrix is a polynomial function of the coefficients of only the n^{th} degree terms H_n of the polynomial f_n. The matrix V provides the sufficient information to compute the intrinsic orientation of the polynomial.

The matrix $V(H_n)$ and the vector $C(H_n, H_{n-1})$ can be interpreted geometrically as an ellipsoid or a hyperboloid in 3D space. This quadratic form is a Euclidean covariant with respect to the original polynomial f_n. That is, upon a Euclidean transformation of the polynomial f_n, the quadratic form is transformed in the same way. This property can be used to recover the object pose.

By using the tensor representation of a polynomial, for any even degree polynomial, we are able to formulate another covariant matrix $V(H_n)$ whose components are linear with respect to the coefficients of the leading term H_n of f_n.

Because of this linearity, the proposed covariant matrix allows us to obtain what appears to be a more robust orientation estimation. We describe now how this new covariant is obtained by focusing on the case when the Euclidean transformation is solely a rotation.

4.1. Rotation

Any homogeneous polynomial, and thus the leading form H_n of f_n described by the vector $A = (a_{ijk})_{i+j+k=n}$ can be expressed in an unique way as a symmetric covariant tensor. The polynomial H_n is:

$$H_n(x_1, x_2, x_3) = \sum_{i+j+k=n} a_{ijk} {x_1}^i {x_2}^j {x_3}^k$$

By expanding the previous triangular sum (see (5)), we define a symmetric tensor $S_n = (s^{i_1 i_2 \dots i_n})_{1 \le i_1, i_2, \dots, i_n \le 3}$ such that

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

For programming purposes this tensor is represented by an n-dimensional array where each entry has 3 possible index values. Tensor representation simplifies the treatment of the pose estimation problem by providing an easy way to express the transformation of the polynomial coefficients under a Euclidean transformation of the reference frame. Upon an orthogonal transformation $R = (r_i^j)$ ((r_i^j) refers to a matrix whose entry at row i and column j is r_i^j) or more generally a linear transformation of the world (x_1, x_2, x_3) coordinate system, the tensor in the new basis is expressed as

$$s'^{j_1 \cdots j_n} = |J^{-1}| \sum_{i_1=1}^{3} \cdots \sum_{i_n=1}^{3} s^{i_1 \cdots i_n} r_{i_1}^{j_1} \cdots r_{i_n}^{j_n}$$

where J is the Jacobian of the transformation and $\mid J \mid = det(R)$ is unity for orthogonal transformation. This expression provides a direct implementation of the transformation of the polynomial coefficients through an easy array manipulation. Note, the above equation defines a *covariant tensor* as it transforms with the same transformation of the coordinate system as opposed to a *contravariant tensor* which transforms with the inverse of the coordinate transformation.

A basic tensor operation is the contraction of a tensor with respect to two indices. Given a tensor, a contraction with respect to the indices i_1 and i_2, e.g., is a new tensor which is of order $n-2$ with

$$s'^{i_3 \cdots i_n} = \sum_{i_1=1}^{3} s^{i_1 i_1 i_3 \cdots i_n}$$

A total contraction of a tensor gives a zero order tensor which is an invariant. For example, for a symmetric 2x2 matrix viewed as a tensor of order 2, the tensor contraction gives the trace of the matrix which is known to be an invariant under Euclidean transformations. Notice that a single contraction of a tensor of order n gives a tensor of order $n-2$, and components of the contracted tensor are linear functions of the original tensor. But the main property we use here is that the contracted tensor still stay an orthogonal covariant independent of how many contractions are applied. Consequently, for a tensor of even order $2p$, $(p-1)$ contraction results in a 3×3 symmetric matrix which is covariant with respect to orthogonal transformations. The diagonalization of this covariant matrix V and the orthonormal eigenvectors basis allow us to estimate the orientation of the polynomial by providing an intrinsic orientation for the polynomial.

As an example, for the leading polynomial of a 2D quartic defined by $H_4(x_1, x_2) = a_{40}{x_1}^4 + a_{31}{x_1}^3 x_2 + a_{22}{x_1}^2{x_2}^2 + a_{13}x_1{x_2}^3 + a_{04}{x_2}^4$, the tensor S_4 is defined by a 4-dimensional array, which can be decomposed in four matrix-slices (s^{ij11}), (s^{ij12}), (s^{ij21}), and (s^{ij22}) $(1 \le i, j \le 2)$ where:

$$(s^{ij11})_{1 \le i,j \le 2} = \begin{pmatrix} a_{40} & \frac{a_{31}}{4} \\ \frac{a_{31}}{4} & \frac{a_{22}}{6} \end{pmatrix}$$

$$(s^{ij12})_{1 \le i,j \le 2} = (s^{ij21})_{1 \le i,j \le 2} = \begin{pmatrix} \frac{a_{31}}{4} & \frac{a_{22}}{6} \\ \frac{a_{22}}{6} & \frac{a_{13}}{4} \end{pmatrix}$$

$$(s^{ij22})_{1 \le i,j \le 2} = \begin{pmatrix} \frac{a_{22}}{6} & \frac{a_{13}}{4} \\ \frac{a_{13}}{4} & a_{04} \end{pmatrix}$$

The contraction of tensor S_4 is equivalent to constructing a matrix from the trace of these 3 matrices and the associated covariant matrix $V(H_4)$ is:

$$V(H_4) = \begin{pmatrix} a_{40} + \frac{a_{22}}{6} & \frac{a_{31}+a_{13}}{4} \\ \frac{a_{31}+a_{13}}{4} & \frac{a_{22}}{6} + a_{04} \end{pmatrix}$$

For the quadratic form in 2D, $H_2(x_1, x_2) = a_{20}(x_1)^2 + a_{11}x_1x_2 + a_{02}(x_2)^2$, $V(H_2)$ is directly the matrix:

$$V(H_2) = \begin{pmatrix} a_{20} & \frac{a_{11}}{2} \\ \frac{a_{11}}{2} & a_{02} \end{pmatrix}$$

Since components of this covariant matrix $V(H_n)$ are linear combinations of the polynomial coefficients, we expect less sensitivity to small changes in the data than by using covariant matrices with components nonlinear in the polynomial coefficients.

Translations leave the leading term H_n unaffected. Therefore, the orientation of the polynomial can be computed directly from H_n using $V(H_n)$ as described above. Notice that the matrix $V(H_n)$ does not provide a unique solution but rather 4 solutions due to the symmetries of an ellipsoid. At this step of the computation, we are not able to disambiguate this and determine the correct rotation estimate. This is why, in the next section, we apply the translation estimation with each of the 4 rotation estimates.

4.2. Translation

This section is equivalent to Taubin's approach in [11], but presented with the tensorial notation. This notation allows us to present the approach in a more generic way and is notationally closer to the implementation. Here the transformation between f_n and f'_n is a pure translation T.

As seen before, the leading terms polynomial H_n is unaffected by a translation. This property is written in tensorial notation as $S'_n = S_n$ where S'_n is the tensor associated with the leading form after translation. The homogeneous form H_{n-1} of degree $n-1$ is transformed in the following way by a translation T:

$$S'_{n-1} = S_{n-1} + n\, S_n \odot T \qquad (7)$$

where $\odot$ is the one-time contracted product of two tensors.

Equation 7 is a linear system with respect to the translation. For a quadratic form (degree $n = 2$), solving this system is similar to using the center of a quadratic form. For example, for a conic in 2D, equation 7 is equivalent to the system:

$$\begin{pmatrix} a'_{10} \\ a'_{01} \end{pmatrix} = \begin{pmatrix} a_{10} \\ a_{01} \end{pmatrix} + 2 \begin{pmatrix} a_{20} & a_{11} \\ a_{11} & a_{02} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

which defines the well known conic center.

For higher degree (even or odd) this system is over constrained, and the solution is obtained by using the pseudo-inverse matrix of the linear system. The use of the pseudo-inverse provides the optimal solution which minimize the error on all $n-1$ degree coefficients. For a quartic in 2D (degree $n = 4$), equation 7 can be rewritten as the system:

$$\begin{pmatrix} a'_{30} \\ \frac{a'_{21}}{3} \\ \frac{a'_{12}}{3} \\ a'_{03} \end{pmatrix} = \begin{pmatrix} a_{30} \\ \frac{a_{21}}{3} \\ \frac{a_{12}}{3} \\ a_{03} \end{pmatrix} + 4 \begin{pmatrix} a_{40} & \frac{a_{13}}{4} \\ \frac{a_{13}}{4} & \frac{a_{22}}{6} \\ \frac{a_{22}}{6} & \frac{a_{31}}{4} \\ \frac{a_{31}}{4} & a_{04} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

4.3. Algorithm

Having shown how rotation and translation estimations are done, we now can answer the question posed at the beginning of section 4 by stating the Euclidean pose estimation algorithm as a whole for our polynomials f_n and f'_n:

- computation of the 4 possible rotations R_i based on the covariant $V(H_n)$ given in section 4.1 from the leading term of the polynomial f_n and one possible rotation R' based on the similar covariant matrix $V(H'_n)$ of the polynomial f'_n,
- for each i apply the rotation $R' R_i^t$ on the polynomial and compute the translation T_i based on the equation 7 (the role of f_n and f'_n can be swapped for more symmetric computations),
- for each i apply the Euclidean transformation $D_i = (R' R_i^t, T_i)$ on the original polynomial and compute the distance between the result and f'_n. The optimal Euclidean transformation D is given by the transformation minimizing the distance between the coefficient vectors f'_n and f_n transformed by D. (Alternatively, the transformed function can be compared with the data set Γ_0 for greater accuracy)

The last step allows us to obtain a unique solution by using the information in the lower degree terms. An extension is to iterate and refine the process by taking into account the information in lower degree terms in order to better estimate R' and R_i. The transformation rule for the whole polynomial is given by expressing the polynomial f_n as a symmetric tensor in homogeneous coordinates as described in section 2.

5. Experiments

To test the robustness of the proposed pose estimation technique, we ran 3 different experiments

- *perpendicular noise*: Gaussian noise in the surface normal direction locally at each point of the data set (figure 5(a) and (b)),
- *resampling*: Gaussian centered noise in the tangential plane, at each data point, and half of the points are randomly discarded (figure 5(d)),
- *occlusion*: a percentage of the data set out of a plane is chopped off (figure 7).

5.1. Noise and random resampling

In Figure 4(a) and Figure 7 the heart data set is displayed (6480 points in a box with dimensions $(3.0cm, 3.0cm, 4.2cm)$). In Figure 5(a) and (b) noisy data sets in the normal direction (standard deviation = 0.06cm and 0.12cm respectively). In (c), noisy data set in the tangential direction, i.e., each data point is perturbed in a plane tangential to the surface at the point and (d) shows a subset of 50% of these points. This is resampling of the 3D surface represented by the original data set, to be used for alignment algorithm accuracy assessment.

From Figure 6, the location error is a linear function of the noise standard deviation between 0.0 and $0.12cm$. As shown in figure 5, a noise with a standard deviation of 0.12

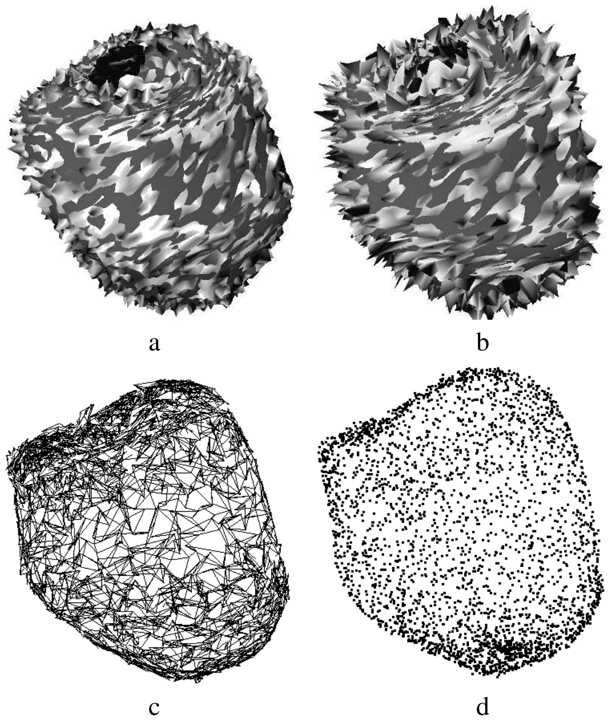


Figure 5. Heart under noise (standard deviation 0.06cm and 0.12 cm) and perturbations (random resampling).

produces major perturbation of the shape, and it is difficult to do the registration visually; nevertheless the angular error obtained by the algorithm is $0.17 rad$, and the translation error is $0.09 cm$. For higher value of the standard deviation, the algorithm is not still able to discriminate between a set of approximative symmetric solutions, and several solutions are obtained.

In these preliminary experiments, our new algorithm provides estimates at least as accurate as those in [11]. Sometimes the difference is large. A more definitive comparison requires further testing.

Finally, Figure 6 shows that our technique is quite insensitive to the variation of the standard deviation of the noise resampling, allowing us to estimate the pose between two data sets without any point-to-point matching between the two data sets. The bias is due to the fact that in these experiments the shape is resampled not along the real surface but only along the tangent plane.

5.2. Occlusion

In Figure 7, a patch consisting of 5% of the points in the heart shape is discarded. Nevertheless, the error on the ori-

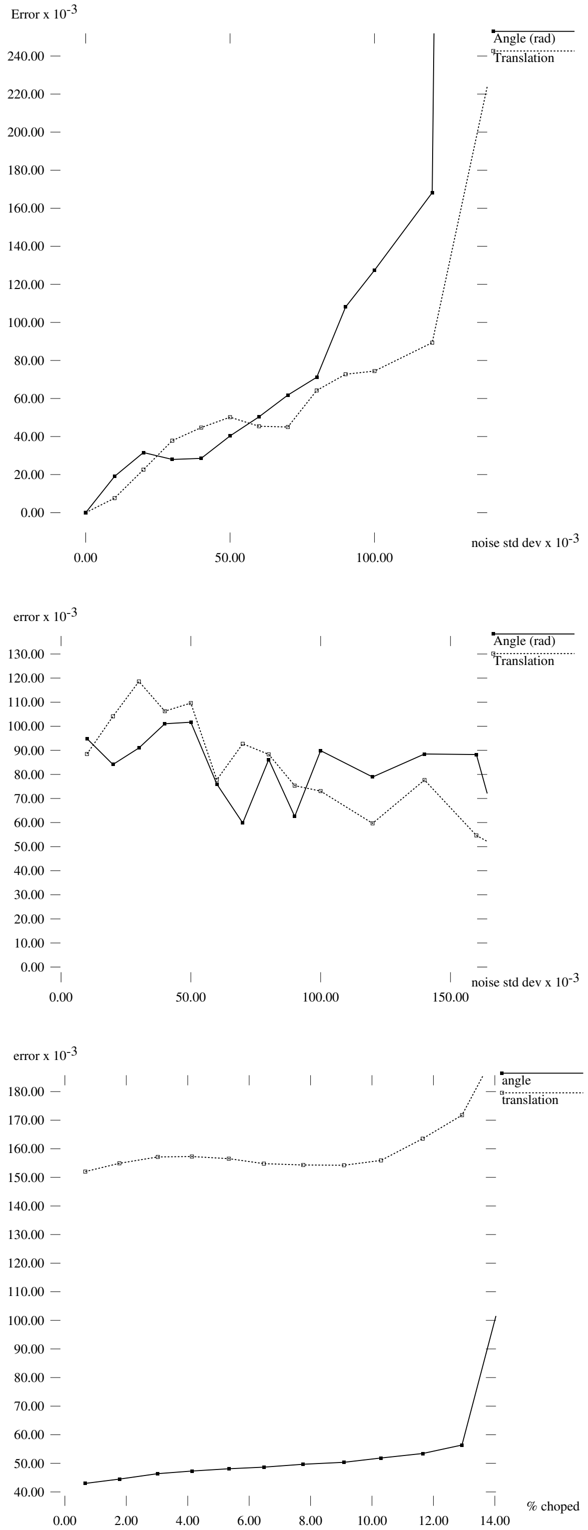
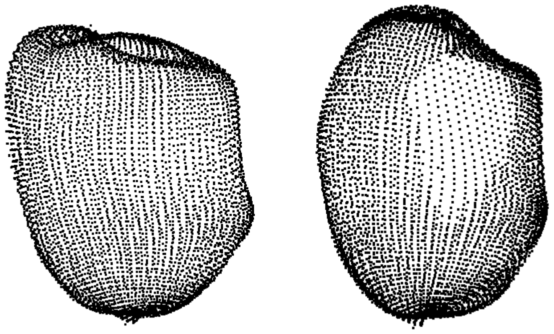


Figure 6. Error in the orientation and translation due to perpendicular noise, random resampling, and occlusion.



(a) (b)

Figure 7. (a) Original data set (b) 5% discarded with a vertical plane. The light part is the discarded region and the data points seen there belong to the far side.

entation is $0.05rad$ and the error on the pose is $0.15cm$. This result illustrates the robustness of our technique to small amounts of occlusion of the data set. In this case, if the orientation estimation is based on the classic method of using the axes of the 3D scatter matrix for the data, the error of orientation is one order of magnitude higher: $0.26rad$.

Figure 6 shows errors in the angle and translation norm estimates for occlusion between 0% and 12% where the estimation is stable. Under occlusion, it turns out that the translation is affected by a systematic bias due to fitting. But the angle is still accurate and robust.

6. Conclusions

The proposed technique does not use all the information about the pose contained in the polynomial coefficients (for example, we use only a part of the leading form to compute the orientation). We are working on pose estimation based on all the information in the polynomial coefficients.

Tensor formalism can be also used to obtain invariants for use in recognition.

Real data are localized in a finite space but a polynomial is defined on the whole space. We are exploiting V being a linear function of the coefficients to extend our PIMs measure [9] to covariant tensors and pose estimation around the data set only. This should result in even greater pose estimation accuracy.

Acknowledgments. Authors would like to thank M. Escuti for his valuable contributions in making some of the figures possible through software development and experimentation. Also the heart and the head 3D data sets used in the experiments are courtesy of C. Nastar.

References

[1] N. Ayache. Medical computer vision, virtual reality and robotics: promising research tracks. Technical report, INRIA-EPIDAURE Project, 1994.

[2] P. Besl and N. McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[3] M. M. Blane, Z. Lei, and D. B. Cooper. The 3L algorithm for fitting implicit polynomial curves and surfaces to data. Technical Report LEMS TR-160, Brown University LEMS Lab., 1997. Under review for *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[4] G. Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics and Image Processing*, 27(5):321–345, 1984.

[5] D. Cyganski and J. Orr. Applications of tensor theory to object recognition and orientation determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(6):662–674, 1985.

[6] T. Faber and E. Stokely. Orientation of 3-D structures in medical images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):626–634, 1988.

[7] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, and M. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Pattern analysis and Machine Intelligence*, 19:1426–1446, 1989.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 71–78, July 1992.

[9] Z. Lei, T. Tasdizen, and D. B. Cooper. Pims and invariant parts for shape recognition. Technical Report LEMS-163, Brown University LEMS Lab., April 1997. accepted for Sixth International Conference on Computer Vision (ICCV), Bombay, India, January 4-7, 1998.

[10] F. Solina and R. Bajcsy. Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, February 1990.

[11] G. Taubin and D. Cooper. 2D and 3D object recognition and positioning with algebraic invariants and covariants. In B. Donald, D. Kapur, and J. Mundy, editors, *Symbolic and Numerical Computation for Artificial Intelligence*. Academic Press, NYC, 1992.

[12] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 1994.

[13] G. Turk and M. Levoy. Zippered polygon meshes from range images. In A. Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 311–318. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.