Silhouette-based 2D-3D Pose Estimation Using Algebraic Surfaces

Nazar Khan
Max-Planck Institute for Computer Science

Bodo Rosenhahn
Max-Planck Institute for Computer Science

Robert Lewis
Mathematics Department, Fordham University

Joachim Weickert
Mathematical Image Analysis Group, Saarland University

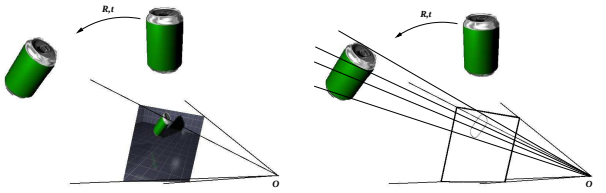
Abstract

In this contribution we present a formulation of the 2D-3D pose estimation problem using implicit algebraic surfaces. To model the projective mapping, we apply novel Dixon resultant heuristics to deal with the rapidly increasing polynomial degrees of projected image outlines. The advantages as well as disadvantages of using linearised twist coordinates for pose representation are also discussed. We also show that for pose estimation purposes, using simple algebraic distance is more practical than using a first-order approximation of the exact Euclidean distance between a point and an implicitly defined entity. While not as effective as current explicit approaches, our implicit formulation offers potential improvements in silhouette-based pose estimation.

Keywords: Pose Estimation, Algebraic Surfaces, 3L Fitting, Occluding Contours, Elimination Theory, Dixon Resultant

1. Introduction

2D-3D pose estimation means estimating the relative *position* and *orientation* of a *known* 3D model from a 2D image of the model (Figure 1(a)). Silhouette based pose estimation deals with finding the 3D pose that best matches the image silhouette. The problem of obtaining 3D estimates of orientation and translation with respect to a camera has numerous applications ranging from robotics [18] to markerless motion capture [3] to medical intervention [4]. The *explicit* approach to this problem involves registering points, lines, planes and other higher-order explicitly defined entities to obtain optimal pose estimates. Figure 1(b) shows one such explicit approach whereby 3D lines are back-projected through silhouette pixels in the image and then registered with the 3D model points [18]. In this work we explore the



(a) 2D-3D Pose Estimation Scenario (b) Silhouette-based *explicit* 2D-3D Pose Estimation

Figure 1. 2D-3D Pose Estimation

implicit approach towards the 2D-3D pose estimation problem whereby 3D objects and 2D silhouettes are modelled as implicit polynomials. In the following we will use the term *occluding contour*[1] for silhouette and *contour generator* for the 3D points whose projection forms the silhouette. This is visualised in Figure 5.

Our implicit solution to the 2D-3D pose estimation problem consists of three phases:

1. Explicit 3D mesh to implicit algebraic surface conversion.

2. Implicit contour generator to implicit occluding contour computation using elimination theory.

3. Error minimisation between explicit 2D silhouette pixels and implicit occluding contour to obtain the optimal 3D pose estimate.

The first two phases are off-line whose result is used in the third on-line phase. In the first phase, we use the 3L fitting algorithm [2] to convert a point set represented as a 3D mesh into its algebraic surface representation as shown in Figure 2. In the second phase, we use elimination theory to compute occluding contour equations from equations

[1] Occluding contour is the more widely used term in pose estimation literature

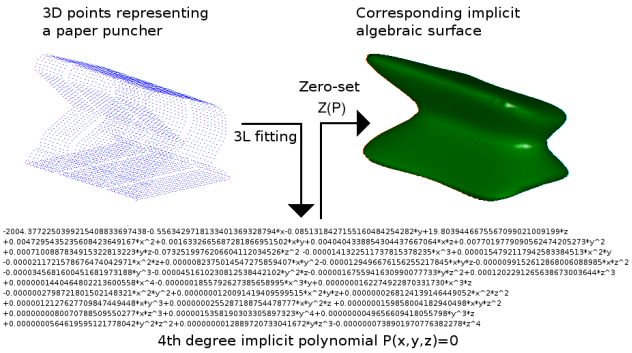


Figure 2. Conversion of 3D mesh into an implicit polynomial equation and the corresponding implicit algebraic surface.

of contour generators. Such computations are very expensive and the size of the final occluding contour equations increases very rapidly as the number of pose parameters is increased (as much as 11 MB for a 4th degree algebraic surface parametrised by 3D translation parameters). We therefore represent pose using linearised twist coordinates [3] to reduce the size of the occluding contour equations. Finally, in the third phase, we estimate the optimal linearised twist parameters by minimising the distance between image outline pixels and the occluding contour.

Representing objects algebraically (i.e., as implicit polynomials) has numerous advantages that have been exploited for 2D as well as 3D registration [20] and recognition [11, 21, 23]. Some researchers have also explored 3D reconstruction from 2D algebraic outlines [7] and vice versa [9]. However, a fundamental problem in studying the projective geometry of algebraic surfaces is that the 2D occluding contour of an algebraic surface of degree d is a 2D polynomial of degree $d(d-1)$ [6]. Table 1 illustrates how the

Table 1. Increase in degrees, computation times and number of terms of 2D occluding contours in image coordinates (s,t) for increasing degrees of the 3D algebraic surface in world coordinates (x,y,z).

Surface Degree	2	4	6	8
Occ. Contour Degree	2	12	30	56
Time (sec)	.05	.2	21	1099
Terms	6	91	496	1653

degree, computation time and number of terms of the occluding contour in 2D image coordinates increase rapidly with the degree of the algebraic surface in 3D coordinates. The study of the projective geometry of algebraic surfaces has been limited because of this unwieldy increase in degrees, computation times and sizes of projected occluding contours. We therefore limit ourselves to studying only 4th order algebraic surfaces. They can represent many useful 2D shapes and 3D real world objects [2] as can be seen in Figure 2.

We present state of the art heuristics in computing the Dixon resultant for efficiently obtaining occluding contour polynomials and show how occluding contour polynomials can be used for the 2D-3D pose estimation problem. The main idea is to parametrise the algebraic surface by the pose parameters and then obtain the corresponding parametrised occluding contour polynomial. The optimal pose for given explicit outline pixels is then obtained by minimising a distance measure between the occluding contour polynomial and the pixels. We motivate the use of the implicit pose estimation formulation in section 2. Section 3 describes the representation of 3D pose using linearised twist coordinates. It is followed by an explanation of elimination theory and novel Dixon resultant heuristics for obtaining occluding contour equations in section 4. Implicit occluding contour based pose estimation is described in section 5 which also presents some experimental results and is followed by some conclusions in section 6.

2. Motivation

Rosenhahn [18] showed that explicit approaches to the pose problem are faster than free-form approaches. However, in many cases, extracting point-based features from images is not possible. This implies the need for global descriptors. Furthermore, estimates derived from global descriptors are more accurate and robust. This work therefore explores the free-form approach whereby objects are modelled as algebraic surfaces.

A free-form approach was also used earlier by Kriegman and Ponce [9] who used elimination theory to compute occluding contour equations of parametric surfaces. They reported that the free-form approach quickly leads to "unwieldy" occluding contour equations. In that paper, they used simple elimination techniques and mentioned that advanced techniques will reduce the complexity of the problem. Occluding contour computation through elimination has since remained a bottleneck that has restricted further progress in this approach. However, due to progress in computational power, advanced elimination strategies [8][14][13] and the development of specialised computer algebra systems such as `Fermat` [12], implicit occluding contour equations have become easier (but not easy) to compute and handle. We therefore re-enter the implicit pose estimation problem. Coefficients of polynomials representing algebraic entities encode useful information about the represented entity [23]. This information includes

1. algebraic invariants that help in classifying and recognising different objects [10] [11] [19], and
2. intrinsic surface geometry that helps in registration [20].

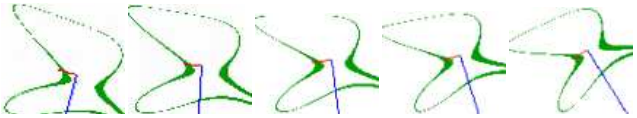
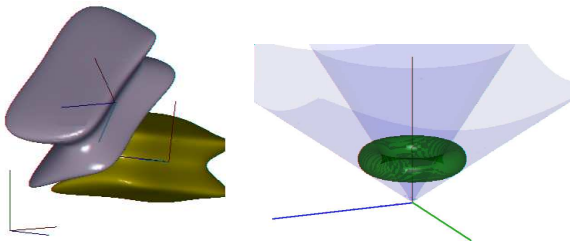


Figure 3. **Left to right:** Zero-sets of implicit polynomials (green) fitted to 2D data rotated by $0, 10, 20, 30$ and 45 degrees and the corresponding intrinsic reference frames (red and blue) extracted from the polynomial coefficients.

The motivation for this work comes from the second property, i.e., using polynomial coefficients for registration. An intrinsic reference frame of an implicit polynomial consists of the center of the polynomial combined with its orientation vectors. Figure 3 shows the intrinsic reference frames extracted from coefficients of a 2D quartic fitted to rotated data. As can be seen the extracted intrinsic reference frames are covariant with respect to the transformations. Figure 4(a) shows the intrinsic reference frame extracted from coefficients of a 4th-order algebraic surface representing a paper puncher and for the same puncher rotated by $45°$ around the x-axis. Given such intrinsic information, the problem of registering two objects reduces to alignment of their intrinsic reference frames [20]. This obviates the need for iteratively finding correspondences needed by explicit registration approaches such as ICP [1].



(a) Intrinsic reference frames for 3D-3D implicit registration. (b) Tangent cone of a torus with respect to the origin.

Figure 4. Advantages of algebraic surfaces: (a) Implicit 3D-3D registration and (b) Tangent cone computation.

Moreover, given an algebraic surface, elimination theory can be used to compute the surface's tangent cone [17] as viewed from a certain point. Figure 4(b) shows the tangent cone of a torus when viewed from the origin. Elimination theory can also be used to compute the occluding contour of an algebraic surface's projection [9]. This is explained in section 4.

3. Pose Representation

By pose estimation we mean estimation of a rigid body motion that brings a 3D model into agreement with image data. Rigid body motion consists of a rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and a translation $\mathbf{t} \in \mathbb{R}^3$ and can be written using homogeneous coordinates as

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0_{1\times3}} & 1 \end{bmatrix}. \tag{1}$$

The group of rigid body motions is denoted by $SE(3)$. Every rigid motion can also be expressed as a rotation around a 3D axis and a translation along that axis. This leads to the *twist* representation of rigid body motion. A twist can be written as

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & \mathbf{v} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{2}$$

where

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times3} \tag{3}$$

is the skew-symmetric matrix corresponding to the rotation axis $\omega = (\omega_1, \omega_2, \omega_3)^T \in \mathbb{R}^3$ and $\mathbf{v} = (v_1, v_2, v_3)^T \in \mathbb{R}^3$. $\mathbf{v}$ determines the location of the rotation axis and the amount of translation along that axis. The norm $\|\omega\|$ represents the amount of rotation. The group of twists is denoted by $se(3)$. A one-to-one map called the *exponential map* exists between $se(3)$ and $SE(3)$ whereby

$$\mathbf{M} = e^{\hat{\xi}} = \mathbf{I} + \hat{\xi} + \frac{\hat{\xi}^2}{2!} + \frac{\hat{\xi}^3}{3!} + \cdots \tag{4}$$

We will use the approximation

$$e^{\hat{\xi}} \approx \mathbf{I} + \hat{\xi} = \underbrace{\begin{bmatrix} 1 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 1 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 1 & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\hat{\xi}'} \tag{5}$$

to obtain *linearised twists* which we denote by $\hat{\xi}'$ similar to [3]. This leads to equations of rigid body motion that are *linear* in the six motion generators $(\mathbf{v}, \omega)$. Once the optimal twist is computed using these linear equations, the corresponding pose matrix M can be efficiently computed using

$$\mathbf{M} = e^{\hat{\xi}} = \begin{bmatrix} e^{\hat{\omega}} & (\mathbf{I} - e^{\hat{\omega}})\hat{\omega}v + \omega\omega^T v \\ \mathbf{0} & 1 \end{bmatrix}, \tag{6}$$

where $e^{\hat{\omega}}$ is computed using the Rodrigues' formula

$$e^{\hat{\omega}} = I + \hat{\omega}\sin(\|\omega\|) + \hat{\omega}^2(1 - \cos(\|\omega\|)). \tag{7}$$

For a detailed and accessible introduction to rigid body motions, twists and Rodrigues' formula, we refer the reader to [15] and [16].

The parametrisation of an implicit algebraic surface $P(\mathbf{x}) = 0$ by the pose parameters is given by $P(\hat{\xi}'\mathbf{x}) = 0$ which we denote by $P(\mathbf{x}, \mathbf{v}, \omega) = 0$. Using linearised twists significantly reduces the degree and computation time of parametrised occluding contours $OC(s, t, \mathbf{v}, \omega) = 0$ as will be explained in the next section.

4. Occluding Contour Computation

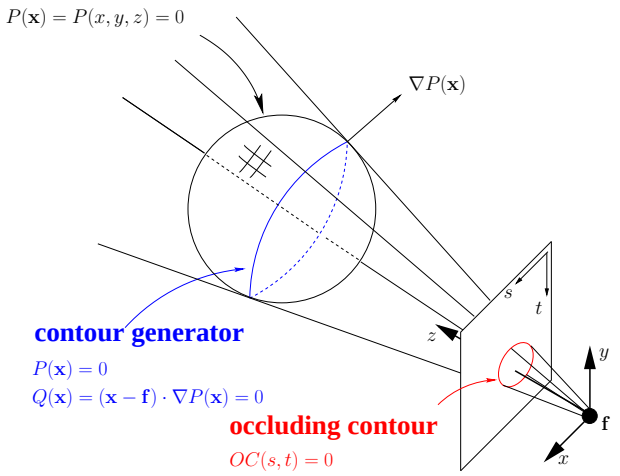


Figure 5. A contour generator (blue) w.r.t to focal point $\mathbf{f}$ and the corresponding occluding contour (red).

The occluding contour is the projection of the contour generator as shown in Figure 5. The contour generator is the locus of all points on the surface that lie on the boundary between the visible and occluded surface points. All points $\mathbf{x}$ on the contour generator therefore satisfy the following polynomial system:

$$P(\mathbf{x}) = 0 \tag{8}$$
$$Q(\mathbf{x}) = (\mathbf{x} - \mathbf{f}) \cdot \nabla P(\mathbf{x}) = 0 \tag{9}$$

$Q(\mathbf{x}) = 0$ is called the *tangency condition*. It simply states that at the contour generator points, the surface gradient is perpendicular to the tangent ray emanating from the focal point f. Since the occluding contour is the projection of the contour generator it satisfies the following polynomial system:

$$P(\mathbf{x}) = 0 \tag{10}$$
$$Q(\mathbf{x}) = 0 \tag{11}$$
$$H(\mathbf{x}, s) = 0 \tag{12}$$
$$V(\mathbf{x}, t) = 0 \tag{13}$$

where $H(\mathbf{x}, s)$ and $V(\mathbf{x}, t)$ are the horizontal and vertical projection equations. To obtain the equation of the occluding contour purely in terms of the image coordinate system (s, t), we need to **eliminate** the variables x, y, z from the system.

4.1. Elimination Theory

Elimination theory is a classical branch of mathematics [24]. Elimination of variables from a polynomial system is the algebraic counterpart of the geometric concept of projection. By eliminating variables, an algebraic variety is projected onto a lower dimensional subspace as illustrated by Figure 6. The black dots $(\sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}})$ and $(-\sqrt{\frac{1}{2}}, -\sqrt{\frac{1}{2}})$ lie at the intersection of the circle $x^2 + y^2 = 1$ and the line $y = x$ and therefore satisfy the polynomial system

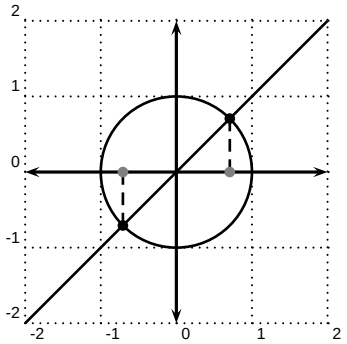


Figure 6. Elimination implies projection. The black dots at the intersection of the line and the circle satisfy the polynomial system $x^2 + y^2 - 1 = 0, x - y = 0$. Eliminating y from the system yields the projection onto the x-axis (gray dots).

$$x^2 + y^2 - 1 = 0 \tag{14}$$
$$y - x = 0 \tag{15}$$

We can eliminate the variable y from this system by substituting $y = x$ in the first equation. This yields the equation $2x^2 - 1 = 0$ which represents the gray dots $\sqrt{\frac{1}{2}}$ and $-\sqrt{\frac{1}{2}}$, i.e., the projections of the 2D black dots onto the 1D horizontal axis. Such simple elimination by substitution can work for linear equations but for higher-order polynomial systems consisting of many equations, sophisticated heuristics are required. Resultants constitute one such heuristic approach as described next.

4.1.1 Resultants

The resultant of a polynomial system is a single polynomial in the coefficients of the original polynomials. The original variables do not appear in the resultant, hence they are *eliminated*. For a polynomial system S, non-trivial solutions exist if the resultant $Res(S) = 0$. The elimination heuristic we apply for eliminating the three variables x, y, z from our system of four polynomial equations is called the *Dixon resultant* [8] which can eliminate n variables from a system of $n + 1$ polynomial equations. The basic idea is to construct a special matrix from the coefficients of the polynomials such that the determinant of this matrix is the resultant or a multiple of the resultant (called the projection operator).

Let $X = \{x_1, \ldots, x_n\}$ be a set of n variables and $P = \{p_1, \ldots, p_{n+1}\}$ a set of $n+1$ polynomials in the polynomial ring $k[x_1, \ldots, x_n]$. Form the set $\overline{X} = \{\overline{x}_1, \ldots, \overline{x}_n\}$ of n

new variables and construct the $(n+1)\times(n+1)$ cancellation matrix C_P of P:

$$C_P = \begin{bmatrix} p_1(x_1, x_2, \ldots, x_n) & \ldots & p_n(x_1, x_2, \ldots, x_n) \\ p_1(\overline{x}_1, x_2, \ldots, x_n) & \ldots & p_n(\overline{x}_1, x_2, \ldots, x_n) \\ p_1(\overline{x}_1, \overline{x}_2, \ldots, x_n) & \ldots & p_n(\overline{x}_1, \overline{x}_2, \ldots, x_n) \\ \vdots & \ldots & \vdots \\ p_1(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) & \ldots & p_n(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) \end{bmatrix}. \tag{16}$$

If we set $x_1 = \overline{x}_1$, the first two rows of C_P will become identical, thus causing the determinant $|C_P|$ to become zero. In general, setting $x_i = \overline{x}_i$ for any $1 \leq i \leq n$ will cause $|C_P|$ to become zero. Therefore, $\prod_{i=1}^{n}(x_i - \overline{x}_i)$ divides $|C_P|$ and can be cancelled out. This gives us the Dixon polynomial

$$\delta_P = \frac{|C_P|}{\prod_{i=1}^{n}(x_i - \overline{x}_i)}. \tag{17}$$

We can treat δ_P as a polynomial in $\overline{X}$ and extract the set $\overline{\mathcal{F}}$ of its coefficients which are polynomials in X. The Dixon matrix D_P is the coefficient matrix of $\overline{\mathcal{F}}$. Being a coefficient matrix, D_P will not contain any of the variables in X. Hence the variables in X will be eliminated in the expression for the Dixon projection operator $|D_P|$ provided that D_P is a square non-singular matrix. If not, then a maximal rank submatrix [8] of D_P can be used instead. For our computations, we used Arthur Chtcherba's `MAPLE` implementation [5] of [8] to compute D_P.

However, two major obstacles still remain in obtaining the true resultant. Firstly, resultant matrices tend to become very large even for moderate degree polynomial systems. Determinant computation of such large polynomial matrices is extremely challenging, even for commercial computer algebra systems (CAS) like `MAPLE`. The specialised CAS `Fermat` [12] however, has better performance. Secondly, even after successful determinant computation, very often the resultant is a factor of the determinant and needs to be factored out. Again, most CASs fail at factorising very large determinant polynomials. Therefore, further heuristics are required for (i) keeping the size of the determinant matrix small, (ii) efficient determinant computation, and (iii) extracting the resultant from the determinant. To keep matrix size small, we use successive elimination [8]:

1. Eliminate x and y from P, Q, H to get a polynomial $A(s, t, z)$.
2. Eliminate x and y from P, Q, V to get a polynomial $B(s, t, z)$.
3. Eliminate z from A, B to get the occluding contour $OC(s, t)$.

For each elimination step, the following two steps can be performed to alleviate problems (ii) and (iii):

1. Before determinant computation, identify and factor out spurious factors from the Dixon matrix using the following rule from linear algebra:

$$\begin{vmatrix} a_1\text{row}_1 \\ a_2\text{row}_2 \\ \vdots \\ a_n\text{row}_n \end{vmatrix} = a_1 a_2 \ldots a_n \begin{vmatrix} \text{row}_1 \\ \text{row}_2 \\ \vdots \\ \text{row}_n \end{vmatrix}. \tag{18}$$

   This tells us that any row factors $a_1, a_2 \ldots, a_n$ are spurious factors because the true resultant is irreducible. They can be removed from the Dixon matrix before computing the determinant. If they are not removed, they will end up as spurious factors in a comparatively bigger determinant expression. In our experiments, late removal of such factors from very large determinant expressions was often not possible in `MAPLE`. Early removal of spurious factors leads us closer to the exact resultant and significantly reduces the time for determinant computation. Our approach is a simplified version of the more powerful strategies developed by Lewis [13].

2. Use `Fermat's` multivariate Lagrange interpolation for determinant computation. It is considerably faster than `MAPLE`'s fraction-free Gaussian elimination.

In summary, given an algebraic surface parametrised by the linearised twist coordinates $(\mathbf{v}, \omega)$ of the pose (as explained in Section 3) and the projection matrix of a camera with projection center $\mathbf{f}$, the occluding contour satisfies the polynomial system

$$P(\mathbf{x}, \mathbf{v}, \omega) = 0 \tag{19}$$
$$(\mathbf{x} - \mathbf{f}) \cdot \nabla P(\mathbf{x}, \mathbf{v}, \omega) = 0 \tag{20}$$
$$H(\mathbf{x}, s) = 0 \tag{21}$$
$$V(\mathbf{x}, t) = 0 \tag{22}$$

The degree of the system in $\mathbf{v}, \omega$ is equal to the surface degree. By eliminating x, y, z from the system using the Dixon resultant, we obtain the algebraic occluding contour $OC(s, t, \mathbf{v}, \omega)$ *parametrised by the linearised twist coordinates* $\mathbf{v}$ *and* ω *of the 3D pose*. However, in our experiments we noticed that $OC(s, t, \mathbf{v}, \omega)$ obtained in this way had higher than required degrees in $\mathbf{v}$ and ω. Although this polynomial represented the correct zero-set, it had considerably larger than required size and computation time. This lead to difficulties in the numerical optimisation procedure. If instead of the contour generator equations 19 and 20, we parametrise the projection equations 21 and 22 with the twist coordinates we obtain the exact occluding contour. Such swapping of parametrisation is consistent with the observation that transforming an algebraic surface by an arbitrary transformation T is equivalent to transforming the

coordinate system by T^{-1}. Furthermore, it leads to a polynomial system that is linear in the twist coordinates instead of the degree of the surface. So the polynomial system we use is

$$P(\mathbf{x}) = 0 \tag{23}$$
$$(\mathbf{x} - \mathbf{f}) \cdot \nabla P(\mathbf{x}) = 0 \tag{24}$$
$$H(\mathbf{x}, s, \mathbf{v}, \omega) = 0 \tag{25}$$
$$V(\mathbf{x}, t, \mathbf{v}, \omega) = 0 \tag{26}$$

Table 2 shows resultant computation times for the $4th$ degree algebraic surface representing a paper puncher (Figure 2) using increasing numbers of pose parameters and viewed through a real-world projection matrix. It can be seen that computation time increases rapidly as more pose parameters are used and that multivariate Lagrange interpolation is faster then fraction-free Gaussian elimination for determinant computation. Machine used was a 2.00 GHz Intel Core2Duo with 2GB RAM.

Table 2. Column-wise: Resultant computation times for increasing numbers of pose parameters. Row-wise: Maple's Fraction-Free Gaussian Elimination (FFGE) vs. Fermat's Multivariate Lagrange Interpolation (MLI). († = No answer.)

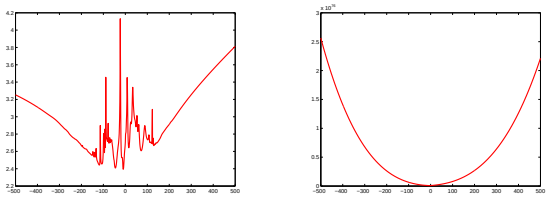
	v_1	v_1, v_2	v_1, v_2, v_3
Terms	1183	8281	40415
Maple's FFGE	4m41s	4h29m	$> 15h^{\dagger}$
Fermat's MLI	1m14s	19m30s	$3h50m$

5. Pose Estimation

Given a binary image containing the outline of the 3D object in an arbitrary pose, we need to minimise the *distance* between the set of outline pixels $\mathcal{D} = \{\mathbf{p}_1, \ldots, \mathbf{p}_q\}$ and $OC(s, t, \mathbf{v}, \omega)$ to obtain the optimal twist coordinates $\mathbf{v}, \omega$ from which the rigid body motion can be computed. No closed-form expression for the exact Euclidean distance between a point and an implicitly defined curve exists. Therefore first-order approximations like $\frac{|OC(s,t,\mathbf{v},\omega)|}{\|\nabla OC(s,t,\mathbf{v},\omega)\|}$ have to be used [22]. However, minimisation using such approximations leads to residual functions that have many local minima (Figure 7(a)). We avoid them by using simple algebraic distance $|OC(s, t, \mathbf{v}, \omega)|$ which is a smaller expression and also more smooth (Figure 7(b)). The corresponding *algebraic mean square distance* that we need to minimise becomes

$$\Delta^2_{\mathcal{D}}(\mathbf{v}, \omega) = \frac{1}{q} \sum_{i=1}^{q} OC(\mathbf{p}_i, \mathbf{v}, \omega)^2. \tag{27}$$

Since $\Delta^2_{\mathcal{D}}(\mathbf{v}, \omega)$ is non-linear in v_i and ω_i, the corresponding non-linear least squares problem involves minimising



(a) Using first-order approximation of exact distance. (b) Using algebraic distance.

Figure 7. The residual using (a) approximate distance and (b) algebraic distance as a function of a puncher's pose parameter v_1. The global minimum at 0 is surrounded by numerous local minima on each side for the case of approximate distance while algebraic distance leads to a smoother residual function.

the length of the residual vector $R = (R_1, \ldots, R_q)$:

$$\| R(\mathbf{v}, \omega) \|^2 = \sum_{i=1}^{q} R_i(\mathbf{v}, \omega)^2, \tag{28}$$

where

$$R_i(\mathbf{v}, \omega) = |OC(\mathbf{p}_i, \mathbf{v}, \omega)| \tag{29}$$

is the algebraic distance from pixel $\mathbf{p}_i$ to the zero-set $Z(OC(\mathbf{v}, \omega))$. We use the Levenberg-Marquardt algorithm to solve the non-linear minimisation problem.

Some more insight into the choice of approximate vs. algebraic distance can be gained by writing out the residual functions for the approximate and algebraic distances. For clarity, we present the case of just a single pose parameter v_1 and denote $OC(s, t, v_1)$ by f. For the case of approximate distance, the residual function and its partial derivative w.r.t. v_1 become

$$R = \sqrt{\frac{f^2}{f_s^2 + f_t^2}}, \tag{30}$$

$$R_{v_1} = \frac{1}{\sqrt{\frac{f^2}{f_s^2+f_t^2}}} \left(\frac{f f_{v_1}}{{f_s}^2 + {f_t}^2} - \frac{f^2 \left(f_s f_{s_{v_1}} + f_t f_{t_{v_1}} \right)}{\left(f_s^2 + f_t^2 \right)^2} \right). \tag{31}$$

This requires evaluating the six polynomials $f, f_{v_1}, f_s, f_t, f_{s_{v_1}}, f_{t_{v_1}}$. In contrast, when algebraic distance is used the same functions become

$$R = |f|, \tag{32}$$
$$R_{v_1} = sign(f) f_{v_1} \tag{33}$$

which requires evaluating only two polynomials f and f_{v_1}.

For a multi-camera setup with occluding contours $f_1, f_2, \ldots, f_n$, the combined residual function using algebraic distance can be defined as

$$R = \sum_{j=1}^{n} \sqrt{f_j^2} \tag{34}$$

with partial derivative w.r.t. v_i (and similarly w.r.t. ω_i)

$$R_{v_i} = \sum_{j=1}^{n} sign(f_j) f_{j_{v_i}}. \quad (35)$$

Due to approximation (5), the optimal linearised twist $\hat{\xi}'(\mathbf{v}, \omega)$ yields a linear approximation of the correct pose. The correct pose can be obtained by iterating between the following steps:

1. Compute the optimal linearised twist $\hat{\xi}'$ that minimises (27) for occluding contour f^k.
2. Transform the 3D model by $M = e^{\hat{\xi}'}$ using (6).
3. Compute an updated occluding contour f^{k+1} corresponding to the transformed 3D model.

Due to high computational cost, we perform step 3 off-line. To get closer to real-time implicit pose estimation, a quick on-line implementation of the Dixon resultant is required. So, while linearised twists lead to more efficient Dixon resultant computations, they require on-line elimination techniques in order to move towards real-time implicit pose estimation.

As an example, we attempt to compute the parameters v_1, v_2 and v_3 for the 3D model of a paper puncher as viewed in Figure 8 through a known camera projection matrix. The green outline is the initial pose while the blue outlines show the convergence behaviour of our algorithm. Convergence was achieved in only three iterations and average time per iteration was three seconds. Figure 9 shows convergence in

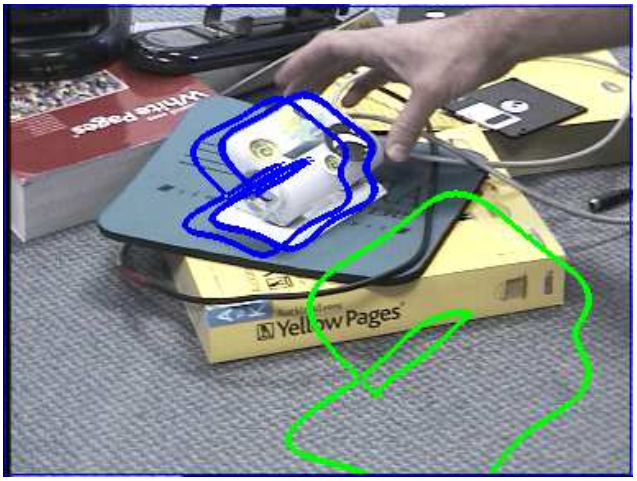


Figure 8. Convergence behaviour using algebraic distance when estimating three pose parameters v_1, v_2, v_3. (Green = initial pose, blue = estimated poses.)

presence of noise, occlusion and missing data in the image outline pixels. Figure 10 shows the variation in algebraic error of the estimated pose when noise is added to the image silhouettes in a stereo setup.

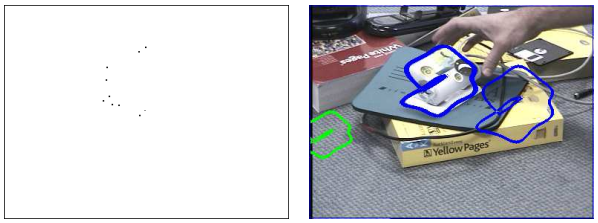


Figure 9. Convergence behaviour using algebraic distance in presence of noise, occlusion and missing data. (Green = initial pose, blue = estimated poses.)

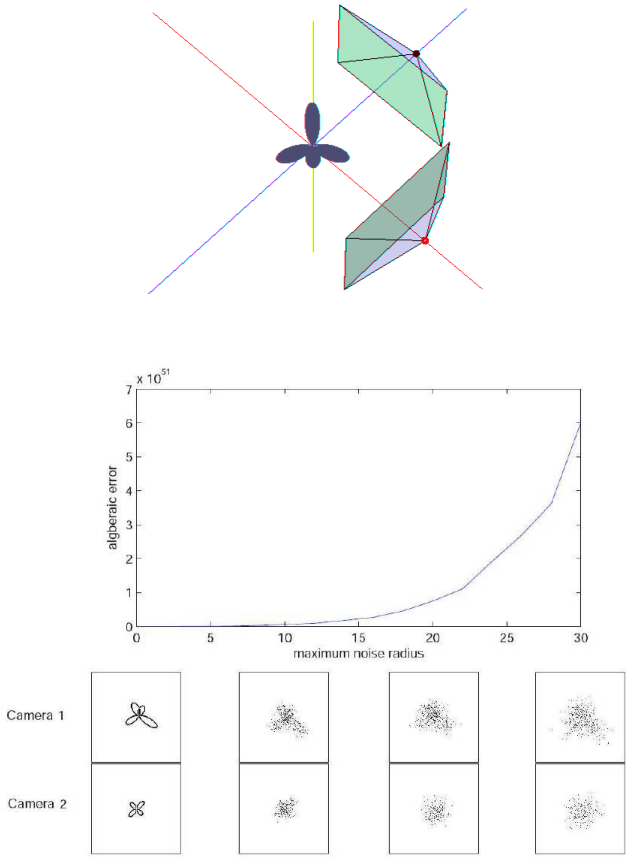


Figure 10. Algebraic error variation with increasing silhouette noise for a stereo setup viewing a 4th degree algebraic surface placed at the origin.

6. Conclusion

We have presented a formalisation of implicit 2D-3D pose estimation. We use linearised twist coordinates to represent pose. While this choice leads to iterative linear search for the optimal pose parameters in explicit pose estimation formulations, using it in our implicit formulation calls for further research in fast, on-line resultant computations. However, linearised twists do lead to efficient Dixon

resultant computations.

In our experiments, parametrising coordinate transformations instead of surface transformations by the pose parameters leads to exact outline equations. We have also shown that for numerical minimisation, using simple algebraic distance is more practical than using a first-order approximation of the exact Euclidean distance. In the results shown, we have only estimated the vector $\mathbf{v}$. Estimation of ω can be carried out using ideas from [17] who relate those properties of the image outline and surface that depend only on rotation (for instance, contour curvature of outline and Gaussian curvature of surface).

7. Acknowledgments

References

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

[2] M. M. Blane, Z. Lei, H. Civi, and D. B. Cooper. The 3L algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):298–313, 2000. 1, 2

[3] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinetics. *International Journal of Computer Vision*, 56(3):179–194, 2004. 1, 2, 3

[4] D. Burschka, M. Li, R. Taylor, and G. D. Hager. Scale-invariant registration of monocular stereo images to 3D surface models. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 2581–2586, 2004. 1

[5] A. D. Chtcherba. Maple package for dixon/bezout resultant computation. http://www.cs.panam.edu/∼cherba/Projects/maple-dixon/. 5

[6] D. Forsyth, J. L. Mundy, A. Zisserman, and C. Rothwell. *Symbolic and Numerical Computation for Artificial Intelligence*, chapter 7: Applications of Invariant Theory in Computer Vision. Computational Mathematics and Applications. Academic Press, 1992. 2

[7] K. Kang, J.-P. Tarel, R. Fishman, and D. B. Cooper. A linear dual-space approach to 3D surface reconstruction from occluding contours using algebraic surfaces. In *IEEE International Conference on Computer Vision*, volume I, pages 198–204, Vancouver, Canada, 2001. 2

[8] D. Kapur, T. Saxena, and L. Yang. Algebraic and geometric reasoning using dixon resultants. In *International Symposium on Symbolic and Algebraic Computation*, pages 99–107, 1994. 2, 4, 5

[9] D. J. Kriegman and J. Ponce. On recognizing and positioning curved 3-D objects from image contours. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 12, pages 1127–1137, Dec 1990. 2, 3

[10] Z. Lei, D. Keren, and D. B. Cooper. Computationally fast bayesian recognition of complex objects based on mutual algebraic invariants. In *IEEE International Conference on Image Processing*, volume 3, pages 635–638, Washington, DC, USA, October 1995. 2

[11] Z. Lei, T. Tasdizen, and D. B. Cooper. PIMs and invariant parts for shape recognition. In *Proceedings of Sixth International Conference on Computer Vision*, pages 827–832, Mumbai, India, 1998. 2

[12] R. H. Lewis. Computer algebra system Fermat. http://www.bway.net/∼lewis. 2, 5

[13] R. H. Lewis. Heuristics to accelerate the dixon resultant. Accepted for publication in Mathematics and Computers in Simulation, 2007. 2, 5

[14] R. H. Lewis and P. F. Stiller. Solving the recognition problem for six lines using the dixon resultant. *Mathematics and Computers in Simulation*, 49(3):205–219, 1999. 2

[15] Y. Ma, S. Saotta, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision, From Images to Geometric Models*. Springer, New York, 2003. 3

[16] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1994. 3

[17] J. Ponce and D. J. Kriegman. *Symbolic and Numerical Computation for Artificial Intelligence*, chapter 5: Elimination Theory and Computer Vision. Computational Mathematics and Applications. Academic Press, 1992. 3, 8

[18] B. Rosenhahn. *Pose Estimation Revisited*. PhD thesis, Inst. für Informatik und Praktische Mathematik der Christian-Albrechts-Universität zu Kiel, 2003. 1, 2

[19] K. Siddiqi, J. Subrahmonia, D. B. Cooper, and B. Kimia. Part-based bayesian recognition using implicit polynomial invariants. In *IEEE International Conference on Image Processing*, page 3360, 1995. 2

[20] J. P. Tarel, H. Civi, and D. B. Cooper. Pose estimation of free-form 3D objects without point matching using algebraic surface models. In *Proceedings of IEEE Workshop on Model Based 3D Image Analysis*, pages 13–21, Mumbai, India, 1998. 2, 3

[21] J.-P. Tarel and D. B. Cooper. The complex representation of algebraic curves and its simple exploitation for pose estimation and invariant recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):663–674, 2000. 2

[22] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991. 6

[23] G. Taubin and D. B. Cooper. *Symbolic and Numerical Computation for Artificial Intelligence*, chapter 6: 2D and 3D Object Recognition and Positioning with Algebraic Invariants and Covariants. Computational Mathematics and Applications. Academic Press, 1992. 2

[24] C. E. Wee and R. N. Goldman. Elimination and resultants - part 1: Elimination and bivariate resultants. *IEEE Computer Graphics and Applications*, 15(1):69–77, January 1995. 4