

# Constrained fitting in reverse engineering

Pál Benkó, Géza Kós, Tamás Várady  
Geometric Modelling Laboratory,  
Computer and Automation Research Institute,  
Hungarian Academy of Sciences  
László Andor  
CADMUS Ltd, Budapest, Hungary  
Ralph Martin  
Cardiff University, Wales, UK

## Abstract

This paper considers simultaneous fitting of multiple curves and surfaces to 3D measured data captured as part of a reverse engineering process, where constraints exist between the parameters of the curves or surfaces. Enforcing such constraints may be necessary (i) to produce models to sufficiently accurate tolerances for import into a CAD system, and (ii) to produce models which successfully reproduce regularities and symmetries required by engineering applications.

The constraints to be satisfied may be determined manually, or more likely, by an automatic process. In the latter case, typically many more constraints are generated than can all be simultaneously satisfied. We present a new numerical method able to resolve conflicts between constraints.

Secondly, reverse engineering generates large amounts of data. Constrained fitting methods are iterative in nature, and so an efficient method needs to restrict the amount of computation performed on each iteration. Our method achieves this through carefully constructed representations for objects and constraints, and approximations to distance functions.

This paper describes our approach to constrained fitting, and illustrates its usefulness with some 2D and 3D examples taken from reverse engineering.

## 1 Introduction

Reverse engineering of solid shape concerns the problem of taking point data scanned from the surface of a geometric object, for example a mechanical component, and producing a CAD model representing that object [23]. In particular, we wish to producing boundary-representation solid models. In this paper we concentrate on the problem of producing models of objects bounded by simple analytical surfaces (planes, cylinders, cones, spheres, and tori), swept surfaces, and fixed-radius rolling ball blends. The problems arising in the reverse engineering of free-form objects are rather different, although we will refer to relevant work in this area later.

We assume here that 3D point data from multiple views has already been merged into a single point cloud, and that this has been *segmented*, i.e. some process has already decided which data points belong to each of the primary surfaces of the object. We also assume that the type of each primary surface has been determined [3].

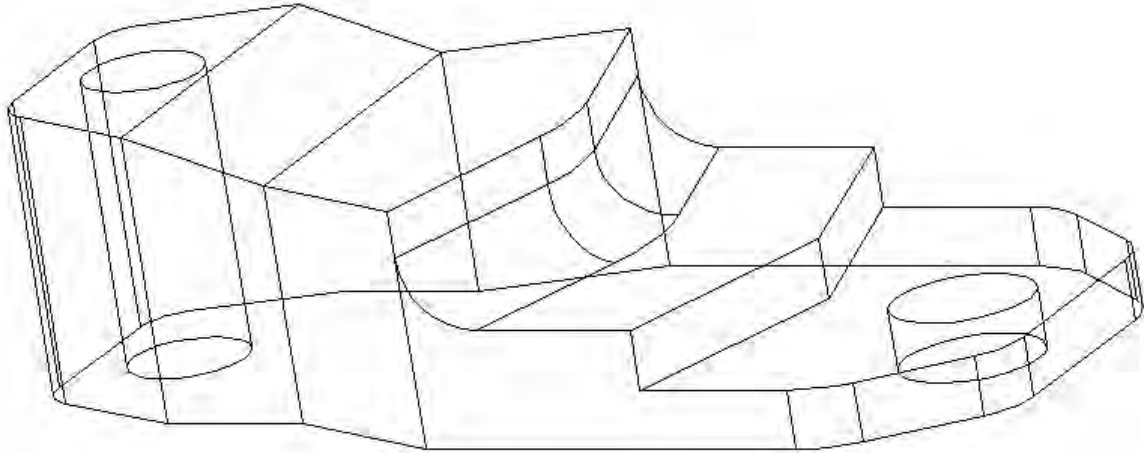


Figure 1: Test object

### 1.1 Capturing designers' intentions

Once analytical surfaces have been fitted to the points belonging to each primary surface, a boundary-representation model can be constructed by stitching these surfaces together (including blends as needed). However, real data is noisy; other errors are introduced by the numerical procedures used. As a result, the model produced will not be acceptable for real CAD/CAM purposes. The reason is that human designers usually include many deliberate relationships or constraints between the primary surfaces, for a variety of aesthetic, functional and manufacturing reasons. Such constraints will not be present in the model if each surface is fitted independently. A *beautification* process must be used to determine what the *ideal* model is likely to be.

Simple constraints which often occur include parallelism and perpendicularity of plane normals and cylinder axes, concentricity of spheres, the centre of a sphere lying on the axis of a cone, and so on. Many such constraints implicitly occur in the test object shown in Figure 1.

For a review of the types of constraints which can arise, and a survey of their relative frequency of occurrence in a range of simple mechanical components, see Mills [15]. Further extensive discussions of the topic can be found in [4].

Two possible approaches exist for ensuring that a reverse-engineered boundary-representation model includes such constraints. The approach assumed in this paper is that when surfaces are fitted to the points, a set of constraints is also imposed, so that rather than the surfaces being fitted *sequentially*, they are fitted *simultaneously*, using the constraints as a set of side-conditions which the parameters describing the surfaces must also satisfy. Model building then proceeds with the fitted surfaces.

The second approach fits each surface independently, and then builds a model. Constraints are then enforced as a *post-process* by changing the values of the surface parameters; at this stage the original point data is ignored.

In principle, the first approach is more likely to lead to an accurate and correct model, but it is more computationally intensive. The second approach is justified on the grounds that if there are sufficient regularities in the model, maybe *all* of the parameters of the model can be given suitable values by the constraints, with the input point data just providing a

nearby model to guide the reasoning process.

The constraints themselves may come from two sources. In the simplest, tedious and error-prone, case, the user controlling the reverse engineering process enters them manually, using a point and click process. This is very likely to under- or over-constrain the object. Nevertheless, this may be a useful method of including certain key design intentions.

Alternatively, the system may automatically infer the constraints [13]. Some initial fit to each surface must be made for the purposes of determining numerically almost-satisfied constraints, which are then used to generate constraint equations. This process will almost certainly produce more plausible constraints than can be simultaneously satisfied, i.e. an inconsistent, over-constrained, problem. Thus, it is important to associate a *figure-of-merit* with each constraint. This may depend on how well the constraint is initially numerically satisfied, how frequently this type of constraint occurs in typical engineering objects, etc. Some decision process is then needed to fit the surfaces under that set of *mutually consistent* constraints which gives the highest overall figure-of-merit.

In Section 5 we assume that the set of constraints has already been supplied, and that they have been ordered according to appropriate figures-of merit.

## 1.2 Smooth edges between surfaces

A second important use of constrained fitting occurs in reverse engineering. Once primary surfaces have been fitted, where two of them meet in a sharp edge, they can be intersected to produce the edge between them. This cannot be done if they meet with (almost) tangential continuity, as a surface-surface intersection calculation will not be robust, if indeed it works at all. Such a multiple smoothly-adjacent face configuration can be seen near the middle of the object in Figure 1. The requirement for tangent continuity between primary surfaces is another important source of constraints. The curves forming such edges, and positions of vertices at their ends, must be explicitly represented. These may be found by constrained fitting.

Given the set of surface types described above, most sets of adjacent surfaces with tangential continuity will form *either* a composite linear-extrusion surface, or a composite surface of revolution. The former case is illustrated by the sequence of vertical faces at the right-hand end of the object in Figure 1. In either case, sweeping a planar profile curve generates the surface. This planar curve can, in simple cases, be represented by a series of straight lines and circular arcs which meet with tangential continuity. Thus, we can reduce the surface fitting problem for a swept region to a curve fitting problem (after suitable projection of the original 3D data points), together with a set of constraint equations which enforce the desired continuity [2].

## 1.3 General considerations

A simplistic approach to finding a constrained set of geometric entities would construct them *sequentially*, with subsequent ones having restrictions placed on their parameters. Such a process is undesirable, as it leads to an uneven error distribution (objects fitted later have greater error). In extreme cases, it may even lead to a failure to find a solution, when *simultaneous* construction may be successful.

For certain problems *auxiliary* constraints may be required. For example, suppose we wish to require a set of any number of planes to be concurrent through a common point.

We could prescribe the concurrency of *all* subsets of four planes to meet in a common point. A better approach would be to choose a sufficient *independent* collection of subsets of four planes to meet in a common point; geometric reasoning software could produce this collection from the complete set of subsets. Instead, we adopt a more general solution to the problem: we introduce *auxiliary (construction)* objects. In this example, the auxiliary object is the common point through which all planes must pass; this point is not part of the original problem. This solution is more symmetric than using an independent collection of subsets of four planes; a secondary advantage is that the coordinates of the common point are also explicitly computed. In general, the use of auxiliary constraints also has the benefit of leading to a much smaller constraint problem.

## 1.4 Overview

The rest of this paper is structured as follows. Section 2 discusses previous work on constraints in surface fitting, geometric modelling, and reverse engineering. Section 3 describes a new algorithm for geometric fitting under a set of constraints. The novelty lies in solving this problem in the presence of a prioritised list of constraints, some of which may be inconsistent. It depends for efficiency on further new ideas in Sections 4 and 5 for object and constraint representation, and distance function approximation. Section 4 shows with examples how this algorithm may be used to constrain a set of line segments and circular arcs, and illustrates the fitting of tangent-continuous profile curves through 2D data. Section 5 describes how to express constraints involving surfaces, and demonstrates constrained fitting of multiple surfaces in 3D. Finally, we draw some conclusions in Section 6.

## 2 Previous work

Various authors have considered constraints in the context of surface fitting and geometric modelling; few have explicitly done so in the context of reverse engineering. We review their different approaches below. However, we first note that a generally useful framework for geometric constraints is provided by Triggs [21]; although he uses it for the solution of problems in computer vision, he notes that it is of much wider applicability.

### 2.1 Constraints in surface fitting

A wide range of work considers constraints in the context of fitting a *single* (possibly piecewise) free-form surface. Typical constraints enforced in such a context are *global* requirements for the surface fitted to be positive, convex, or monotonic. Applying constraints when fairing data is a similar application. A recent collection of papers summarises much work in both areas [10].

*Local* modifications of single surfaces may use constraints on control (or other) points during user modification of the shape of the surface [5, 25]. Such approaches are often referred to as *physics-based modelling*: changes in shape of a dynamically deformable surface are controlled by a set of virtual springs attached to it, constraining its shape [20]. A survey is provided by Gibson [8]. Later work has included other constraints such as forces caused by collision between surfaces, and is commercially available in software provided by such companies as MathEngine.

Very similar technology is widely used in computer vision for finding and tracking boundaries of objects in images. An elastic energy term is used to attract a deformable contour (a “snake”) to the desired outline. External constraints may be imposed on the shape of the snake [7].

The functional decomposition approach of Weiß [24] uses constrained surface fitting to reverse engineer free-form objects. Here, user segmentation and fitting of *primary* surfaces is followed by fitting of secondary *feature* surfaces linking the primary surfaces. The secondary surfaces are simultaneously made to fit measured data points, and are constrained to be tangential to the primary surfaces. The exact locations of the boundary curves where the surfaces meet are also found during this process.

## 2.2 Constraints in geometric modelling

Many current geometric modelling systems allow constraints to be specified on models constructed by a user. In *parametric* systems, a generic model is defined using a set of parameters. A particular realisation of the model is then instantiated by allocating particular values to the parameters. It may be possible in such systems to specify some parameters in terms of others, but there must be a unique ordering allowing the geometry to be constructed sequentially from previously determined elements.

In contrast, *variational* systems allow the user to input the geometry in a schematic way, and for the actual geometry to be determined by constraints linking the geometric elements. Various methods of solving such constraint systems exist. These may be broadly summarised as (i) analytical solvers, which rely on numerical methods (such as Newton-Raphson methods or homotopy continuation [12]), or symbolic algebra (using Gröbner bases or resultants), or both, (ii) graph-theoretical based methods [9], and (iii) rule-based methods [11]; for an extensive survey see [6]. Selecting the desired solution when the non-linear constraint system has many multiple solutions is a tricky problem; it persists *even* when the user provides an approximately correct initial geometry, and heuristics are often used. Generally, the geometric problem is translated into an algebraic equation system, which is possibly reduced by algebraic simplification.

## 2.3 Constraints in reverse engineering

Although the same sorts of objects are constrained in reverse engineering and in constrained design, there is one fundamental difference. In design, the user adds constraints manually. If the user is competent, the correct number of constraints is applied, and the unspecified geometry is neither under- nor over-constrained. In contrast, in reverse engineering, especially in a fully automated process, the reverse engineering system itself has to attempt to identify the constraints. This will almost inevitably result in far too many constraints. Some will be redundant (for example,  $n$  parallel lines can generate  $O(n^2)$  pairwise constraints), while others are likely to be contradictory.

The generation of constraints will depend on various thresholds, and it is not necessarily simple to choose them in a robust and reliable manner. For example, it might be considered that faces subtending an angle of about (whatever that means!)  $90^\circ$  should be orthogonal. However, small draught angles could have been added to a design to allow a cast component to be removed from a mould. It is not clear whether the reverse engineering process should try to capture these draught angles, or the original design before they were added.

Overall, probably the best that we can hope is that some figure of merit is associated with each constraint, assessing the likely validity of that constraint. To help do this, we have undertaken a part survey to ascertain the frequency of occurrence of various constraints and regularities in a variety of engineering parts [15].

Methods of realistic usefulness in reverse engineering thus need to cope with a set of prioritised, inconsistent constraints. In being able to do this, we believe our constraint satisfaction approach is novel.

Early reverse engineering work by Porrill [16] considered the construction of wire-frame models from stereo pairs. His constraints involved lines, and were, specifically, orthogonality, intersection, equality, and connection by a small rigid motion. His approach was based on Kalman filtering.

Werghi [26] has considered the simultaneous recovery of multiple quadric surfaces from range data under constraints which may constrain a quadric to be of a particular surface type, or may relate the orientations of two surfaces, for example. He assumes that the constraints are known in advance. His basic approach is to minimise a function containing a least-squares term and a penalty term associated with the constraints. Optimisation is based on a sequential unconstrained programming technique, where a weight is allocated to each constraint; the weights are incremented sequentially to force the constraints to tend to zero. At each iteration minimisation is performed using Levenberg-Marquardt techniques. Werghi himself notes problems with this approach, which include complex formulation of the constraint function, heavy reliance on the convexity of the constraint space, and the need for accurate initial estimates of the solution. To some extent, these are addressed by an alternative approach reported in [17], although at the expense of speed.

Another piece of work by the same group [18] fits data to elements from a library of parametrised shape models, to obtain shape and position parameters simultaneously, e.g. for an array of similar holes. By averaging shape over several geometrically similar elements, better estimates of the parameters may be obtained than if each were extracted individually. This is important in reverse engineering in cases where it is difficult to obtain reliable data, cylindrical holes being a specific example. In this work, a distinction is drawn between *domain* constraints (such as restriction on parameter size) and *relational* constraints (relative positions and orientations known a priori).

### 3 Numerical methods for constrained fitting

#### 3.1 Problem statement

We assume that there is a set of parametrised objects whose parameter values are to be estimated. These objects are classed as *primary* objects (curves, surfaces, etc.), and *auxiliary* objects, which are construction objects sometimes needed to express constraints between the primary objects. The parameter values describing both kinds of objects are collected into a vector  $\mathbf{x}$  containing  $n$  parameters in total, whose values are to be estimated.

We assume that there is a global function  $f$  to be minimised, which depends on the parameters  $\mathbf{x}$ , and in some general way on distances of the objects to be fitted from a set of data points. We suppose the data points have been segmented, i.e. each point has been allocated to one of the objects. Note that  $f(\mathbf{x})$  depends only on the parameters of the primary objects, and does not depend on the parameters describing the auxiliary objects.

For example, let us suppose we are performing a fit of 3D data points to a set of surfaces  $\{s_i\}$ . The least-squares distance of each point from the appropriate surface may be used as a measure of goodness of fit. In such a case we could formulate  $f$  as

$$f(\mathbf{x}) = \sum_i \alpha_i \sum_j d(p_{ij}, s_i)^2,$$

where  $i$  runs over the different surfaces being fitted,  $p_{ij}$  is the  $j$ th point assigned to the  $i$ th surface, and  $d(p, s)$  is the distance of point  $p$  from surface  $s$ .  $\alpha_i$  is a weighting in the sum; it can be one, or the reciprocal of the number of points belonging to a surface, or the area of the set of triangles spanning the surface sample points divided by the number of points, or anything else the application requires. (Prioritisation of the constraints is a separate issue from the weights used in  $f(\mathbf{x})$ .)

We also have a set of  $k$  constraint equations  $\{c_l\}$  linking the parameter values of the primary and auxiliary objects. These constraints can be written in the form

$$\mathbf{c}(\mathbf{x}) = 0.$$

The problem to be solved is to find those parameter values which minimise  $f$  subject to  $\mathbf{c} = 0$ .

However, in practice, the constraints may involve redundant or inconsistent equations. We will return to this issue later.

### 3.2 Lagrangian multipliers

The standard method for solving a minimisation problem subject to a set of equality constraints is to use Lagrangian multipliers. We can view the constraint  $\mathbf{c}(\mathbf{x}) = 0$  as a hypersurface  $S$  upon which we wish to minimise  $f$ .

Let us assume  $f$  and  $\mathbf{c}$  are smooth. Then at any point of this hypersurface, the gradient of  $f$  can be computed; this can be decomposed into components perpendicular to the hypersurface, and in the tangent plane of the hypersurface. If the component in the tangent plane is non-zero, this means that as we go across the surface in the direction of  $-f'(\mathbf{x})$ ,  $f$  decreases. Conversely, at a local extremum of  $f$  on the surface, the component in the tangent plane is zero. Thus, at such a point,  $f'(\mathbf{x})$  is orthogonal to  $S$ , and so there is a  $\mathbf{\Lambda} \in \mathbb{R}^k$  such that  $f'(\mathbf{x}) = \mathbf{\Lambda}^T \mathbf{c}'(\mathbf{x})$ . Using this fact we can now construct a system of equations with  $n + k$  variables (the coordinates of  $\mathbf{x}$  and  $\mathbf{\Lambda}$ ) and  $n + k$  equations ( $f'(\mathbf{x}) = \mathbf{\Lambda}^T \mathbf{c}'(\mathbf{x})$  gives  $n$  equations,  $\mathbf{c}(\mathbf{x}) = 0$  gives  $k$  more). The components of  $\mathbf{\Lambda}$  are called Lagrangian multipliers. The system may be solved by multidimensional Newton-Raphson methods. The number of solutions of the linear system set up during this process can be used to determine if all constraints may be satisfied, if certain constraints are contradictory, or if the constraints are not independent.

### 3.3 Sequential constraint satisfaction

The Lagrangian-multiplier method works adequately when the constraints are independent, but is less useful when they are not. Werghi's original approach [26] also fails to address this issue; later they adopted a genetic algorithm partly for that reason. As an alternative, we have devised a method particularly suited to the case when there are multiple constraints, not all of which it may be possible to satisfy simultaneously, and some of which may repeat information provided by others.

We assume that the constraints have been sorted into an order of priority. We wish to find a minimum for  $f$  by sequentially attempting to satisfy the constraints in priority order, so that any constraint not inconsistent with those previously satisfied is also imposed. In what follows, we assume that  $\mathbf{c}(\mathbf{x}) = (c_1(\mathbf{x}), \dots, c_k(\mathbf{x}))$ , where  $c_1$  is the highest priority constraint, and  $c_k$  the lowest.

The two problems:  $\mathbf{c}(\mathbf{x}) = 0$ , and  $f(\mathbf{x})$  is minimal, are solved simultaneously by a modified Newton iteration. Given an initial estimate of the desired parameter vector, in principle a single iteration step computes its projection onto the constraint surface and then optimises it within the tangent plane. Computing the projection itself requires an iterative method. For efficiency reasons and due to the fact that the optimal vector in the tangent plane steps away from the constraint surface anyway, the exact projected point is not computed, but only the first step of the projection iteration is performed. Similar techniques are used in optimisation theory [1].

At each iteration step we use a linear approximation for  $\mathbf{c}$ , and a quadratic one for  $f$  (because we are solving  $\mathbf{c} = 0$  and minimising  $f$  requires solution of an equation in  $f'$ ). Each step of the iteration computes an update to the parameter values:  $\mathbf{d}$  is the update to be added to the current parameter vector  $\mathbf{x}_0$ . To compute  $\mathbf{d}$ , we use the following Taylor series expansions about  $\mathbf{x}_0$ :

$$\mathbf{c}(\mathbf{x}_0 + \mathbf{d}) \approx \mathbf{c}(\mathbf{x}_0) + \mathbf{c}'(\mathbf{x}_0)\mathbf{d} \quad (1)$$

$$f(\mathbf{x}_0 + \mathbf{d}) \approx f(\mathbf{x}_0) + f'(\mathbf{x}_0)\mathbf{d} + \frac{1}{2}\mathbf{d}^T f''(\mathbf{x}_0)\mathbf{d} \quad (2)$$

Using these expansions, the problem to be solved may be written in the following form:

$$C\tilde{\mathbf{d}} = 0 \quad (3)$$

$$\tilde{\mathbf{d}}^T A \tilde{\mathbf{d}} \quad \text{to be minimised,} \quad (4)$$

where  $\tilde{\mathbf{d}} = (d_1, \dots, d_n, 1)^T$ , and  $C$  is formed by concatenation:  $C = [\mathbf{c}'(\mathbf{x}_0)|\mathbf{c}(\mathbf{x}_0)]$ . The last coordinate 1 in  $\tilde{\mathbf{d}}$  allows for the constant term in Eqn. (1). The constant term in Eqn. (2) is irrelevant when minimising, allowing  $A$  to be an  $(n+1) \times (n+1)$  matrix of the following form:

$$A = \begin{array}{|c|c|} \hline f''(\mathbf{x}_0) & f'(\mathbf{x}_0) \\ \hline f'(\mathbf{x}_0)^T & 0 \\ \hline \end{array}$$

To compute  $\tilde{\mathbf{d}}$ , we first use Eqn. (3) to reduce  $\tilde{\mathbf{d}}$  to a lower dimensional vector  $\mathbf{d}^*$  of *independent* variables (essentially, the constrained variables are expressed in terms of unconstrained ones):

$$\tilde{\mathbf{d}} = M\mathbf{d}^*. \quad (5)$$

We will explain how to compute  $M$  shortly;  $M$  is a matrix of appropriate size chosen so that  $CM = 0$ . The exact dimension of  $\mathbf{d}^*$  depends on how many independent constraints are there.

Then, substituting Eqn. (5) into Eqn. (4), we now simply have an unconstrained minimisation problem: minimise  $\mathbf{d}^{*T} A^* \mathbf{d}^*$ , where  $A^* = M^T A M$ . The solution for  $\mathbf{d}^*$ , and hence  $\tilde{\mathbf{d}}$  and thus  $\mathbf{d}$ , is found by solving a linear equation, that the derivative of that quadratic

function is zero. This equation gives a minimum if the upper left corner of  $A'$  is positive semi-definite, which holds if the upper left corner of  $A$ , i.e.  $f''$  is positive semi-definite.

In our case,  $f$  is the (weighted) sum of squared errors. Let us take a closer look at the second derivative of a squared quantity:  $(q^2)'' = (2qq')' = 2(q'q'^T + qq'')$ . In this sum the first term is positive semi-definite and the second may be presumed to be small due to multiplication by  $q$ , which gives the signed distance of a point from the surface. Later in Section 3.5 we will see that in certain cases, a more rigorous justification can be given, particularly in the case of simple representations, which includes all planar objects.

Let us now consider how to compute  $M$ . This is done by working through the rows of  $C$  in turn (from top to bottom) using a process similar to that used in Gaussian elimination.

Suppose firstly that all constraints are independent of the previous ones, and consistent, for the purposes of explanation; we will explain later what happens if they are not. Starting with the first row, we choose the variable  $d_l$  whose coefficient has the largest absolute value (for stability reasons) and express  $d_l$  in terms of the others. Suppose the equation for this row corresponds to

$$C_1 d_1 + \dots + C_n d_n + C_{n+1} = 0,$$

where  $C_l$  is the largest coefficient (ignoring  $C_{n+1}$ ). Then we may write

$$d_l = -\frac{C_1}{C_l} d_1 - \dots - \frac{C_{l-1}}{C_l} d_{l-1} - \frac{C_{l+1}}{C_l} d_{l+1} - \dots - \frac{C_n}{C_l} d_n - \frac{C_{n+1}}{C_l}.$$

We may use this to get rid of  $d_l$  by writing  $\tilde{\mathbf{d}} = M_1 \mathbf{d}_1$ , where

$$M_1 = \begin{array}{c} \begin{array}{|ccc|ccc|} \hline 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ \hline -\frac{C_1}{C_l} & \cdots & -\frac{C_{l-1}}{C_l} & -\frac{C_{l+1}}{C_l} & \cdots & -\frac{C_{k+1}}{C_l} \\ \hline 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \\ \hline \end{array} \\ , \end{array}$$

the “special” row of  $M_1$  being row  $l$ . Note that  $\mathbf{d}_1$  is of one dimension less than  $\tilde{\mathbf{d}}$ , and is constructed as

$$\mathbf{d}_1 = (d_1, \dots, d_{l-1}, d_{l+1}, \dots, d_n, 1).$$

We now construct  $CM_1$ , and consider row 2 of  $(CM_1)$  in the analogous equation  $(CM_1)\mathbf{d}_1 = 0$  to eliminate another variable. This process is repeated for each row in turn. In this way,  $M$  is built up as  $M = M_1 \dots M_j$  where  $j$  is the number of independent constraint equations.

Now, in practice, the constraint equations may not be independent. Two cases need to be considered.

If *all* the entries in a given row  $m$  currently being considered are zero (within prescribed tolerance), then the constraint equation follows from the previous ones. In this case, we cannot eliminate one of the variables, and a matrix  $M_m$  cannot be constructed. We simply proceed to the subsequent row without further processing. (This is one reason why  $j$  above may be less than  $k$ , the number of constraints.)

If the constant term in a given row  $m$  is not zero but all the others are, then the constraint contradicts the previous ones, and so this constraint equation is discarded. Again, this simply

means that we proceed to the next row without further processing, as in the previous case. However, this time, we may well wish to note in a separate list that this constraint could not be satisfied, so that we can report this fact to the user.

One final subtlety remains. A single *geometric* constraint may lead to more than one constraint *equation*, as will be seen in Section 4.2.2. For example, constraining two circles to be concentric needs two separate equations linking  $x$  and  $y$  coordinates in 2D. When the set of constraints is processed sequentially, such a geometric constraint may be inconsistent with the previous constraints. However, the first constraint equation of the geometric constraint may not be inconsistent, but only the second. When we remove this inconsistent constraint equation, we must also *backtrack* and remove any other constraint equations belonging to the same geometric constraint, by setting all appropriate rows to zero. We must then rewind the computation to the row after the highest row which was discarded, and subsequent  $M_i$  must be recomputed afresh.

### 3.4 Initial values for the iteration

As with any iterative scheme, good initial values are required. In practice, for the *primary* objects, we have found in our experiments that using initial parameter values provided by an unconstrained fit to each is adequate.

However, *auxiliary* objects cannot be initialised this way. Our experiments show that results are insensitive to initial values for auxiliary *point* objects, but other auxiliary objects must be chosen more carefully. In principle, estimates for any auxiliary objects can be generated from the primary objects which are constrained by these auxiliary objects or (in the 3D case) from an underlying triangulation of the point cloud. Whether such estimates are always adequate requires further investigation.

### 3.5 Efficient representation

During constrained fitting we aim to minimise an error function, a weighted sum of squared distances between points and associated objects. The first and second derivatives of this error function have to be evaluated during each iteration step. To make iteration efficient, we wish to evaluate as much as possible in a once-only preprocessing step before iteration commences. We achieve this by using suitable object and constraint representations, combined with appropriate approximations to distance functions. On the other hand, we have a separate aim, unfortunately often contradicting the previous one, of using an approximate distance function close to the true Euclidean distance.

An important idea is to use a *faithful* distance function [14]. An approximate distance function is faithful if and only if: (i) it is zero where the true distance function is zero, i.e. on the object (curve or surface), and (ii) if its derivative *on the object* is equal to the derivative of the true distance function. In other words, a distance function is faithful if and only if its zero set is the object itself, and its derivative is unit and perpendicular to the object there.

We wish to write the approximate signed distance function between an object (described by a parameter vector  $\mathbf{x}$ ) and a point  $p$  in the form

$$d(p, \mathbf{x}) = S(\mathbf{x})^T P(p) = P(p)^T S(\mathbf{x}), \quad (6)$$

where  $S$  and  $P$  are vector valued functions (of the same dimension). If we do this, the function

to be minimised (for a single object) is

$$e(\mathbf{x}) = \sum_p d(p, \mathbf{x})^2 = S(\mathbf{x})^T \left( \sum_p P(p)P(p)^T \right) S(\mathbf{x}).$$

Using such a separable form, we only need to compute the matrix  $M = \sum P(p)P(p)^T$  once, before iteration commences, as it depends only on the data points and not on the parameters of the object.

The first and second derivatives of  $e(\mathbf{x})$  can then be computed using

$$e' = 2S^T MS', \quad e'' = 2(S'^T MS' + S^T MS'').$$

These may still be time consuming to compute, so we attempt to find descriptions where  $S$  is *simple*. If  $S$  is just a row vector of parameters, i.e.  $S(\mathbf{x}) = (x_1, x_2, x_3, \dots)$ , rather than a vector containing more complicated functions of the parameters, then the derivative formulae are simple, and take the forms

$$e'(\mathbf{x}) = 2M\mathbf{x}, \quad e''(\mathbf{x}) = 2M.$$

Note that by construction,  $M$  is positive semi-definite, that is,  $e''$  is positive semi-definite in the whole search space. In fact, one may always choose a new representation  $\tilde{\mathbf{x}} = S(\mathbf{x})$  in this way, but there may be more variables, linked by potentially complicated constraints.

We illustrate the above ideas using a concrete example. Suppose we wish to fit points to a circle. The Euclidean distance between a circle with centre  $o$  and radius  $r$ , and a point  $p$ , is  $\|p - o\| - r$ . Because of the implied square root in this function, it is not in the separable form given in Eqn. (6). However, we may replace it with the faithful approximation

$$((p - o)^2 - r^2) / 2r = (p^2 - 2\langle p, o \rangle + o^2 - r^2) / 2r.$$

If we choose  $S(\mathbf{x}) = S(o_x, o_y, r) = (1/2r, -o_x/r, -o_y/r, (o^2 - r^2)/2r)$ ,  $P(p) = (p^2, p_x, p_y, 1)$ , we now *do* have the desired form in Eqn. (6). However, we may go further. We now change the parameters used to describe the circle from  $\mathbf{x} = (o_x, o_y, r)$  to  $\mathbf{x}' = S(\mathbf{x}) = (\kappa, o'_x, o'_y, A)$ . We can then write  $S(\mathbf{x}') = \mathbf{x}'$ ; the extra variable used makes a constraint necessary. It can be derived from the equations  $o' = -2\kappa o$  and  $A = (o^2 - r^2)/2r$ , giving  $o'^2 - 4\kappa A = 1$ . Computing the derivatives is now simpler; in particular, no divisions are required. Furthermore, this choice of variables works in the singular case where the circle degenerates to a line. In such a case, using the original set of parameters, the radius becomes infinite, but using the new parameters,  $\kappa = 0$ ,  $o'$  becomes the normal to the line (the normalisation condition ensures that it is a unit normal) and  $A$  becomes the signed distance of the line from the origin. So (as will be seen below) this is the representation of our choice.

## 4 Constrained fitting in two dimensions

We now consider how to solve constrained fitting problems in 2D where the curves are line segments and circular arcs. We start by defining a fairly general set of constraints for use in this context. Such constraints have a range of applications, and could be useful, for example, for inspection tasks involving manufactured objects. In reverse engineering, an important application is the reconstruction of translationally and rotationally swept surfaces [3]. These

are generated by sweeping a planar profile curve, which in many simple cases is made up of series of straight lines and circular arcs with tangential continuity. An example of this application will be given later in this Section.

As noted previously, we assume that the data points have already been segmented, and that the constraints to be imposed have been deduced automatically (for example, in a reverse engineering context), or with user assistance (for example, for an inspection task).

#### 4.1 Planar constraints

Here we enumerate various constraints which we have identified as the most important for problems involving straight line segments and circular arcs.

The following are *simple* constraints:

- *circle* has a fixed radius
- *circle* has a fixed centre
- two *circles* are concentric
- two *circles* are tangent
- two *circles* intersect orthogonally
- *line* passes through the centre of *circle*
- *line* is tangent to *circle*
- two *lines* are at a given angle
- three *lines* are concurrent (share a common point)

Other more general constraints may also be required, e.g. ‘three circles are tangent at a common point, and a fourth circle and two lines also pass through that point’. As we noted in the Introduction, in order to describe such constraints, *auxiliary* objects of the following three types are used:

- *number*
- *point*
- *point-with-orientation*, i.e. a point with an associated unit direction vector

These are not in general fitted to data points (although point entities may be). Additional constraints defined in terms of the auxiliary objects may be chosen from the following:

- the cosine of the angle between two *lines* is *number*
- the square of the distance between two *points* is *number*
- *line* passes through *point*
- *circle* passes through *point*
- *circle* is centred at *point*
- *line* meets *point-with-orientation*, i.e. it passes through the point and its normal is oriented accordingly
- *circle* meets *point-with-orientation*, i.e. it passes through the point and its normal there is oriented accordingly
- *point* equals the point part of *point-with-orientation*

The last constraint is useful to avoid having to introduce extra constraints such as ‘*line* passes through the point of *point-with-orientation*’. The first two constraints are useful for solving problems like fitting regular polygons.

As an example, we show how these auxiliary objects and extra constraints can be used to formulate the problem ‘three circles are tangent at a common point, and a fourth circle and two lines also pass through that point’. We introduce two auxiliary objects, a point and a point-with-orientation, and prescribe the following seven constraints: each of the first three circles passes through the point-with-orientation; the point is the point part of the point-with-orientation; the fourth circle and each of the two lines pass through the point.

Note that some subtleties must be observed in using the constraints, and different effects may be obtained when using constraints in the first and second groups. For example, with the object and constraint representations we give below, the ‘three lines are concurrent’ constraint in the first group is taken in a projective sense: three parallel lines satisfy that constraint. Expressing the same requirement by introducing an auxiliary point and demanding that each line passes through it solves the corresponding Euclidean problem; also, as a by-product, we get the intersection point, which we do not in the former case.

## 4.2 Representation of planar objects and constraints

Objects and constraints can be represented in many different ways. We aim to describe objects so that the error functions of objects (circles and lines) behave in a similar way: the representation is chosen so that a similar geometric error causes a similar numerical error in the representation. This is important to ensure that errors in different kinds of objects are comparable.

For efficiency reasons, as noted above, we use a faithful approximation to the true Euclidean distance function for circles; our representation for straight lines is a special case of our representation for circles. Such representations were introduced for natural quadrics in [14], where the representation used was also optimal in the sense of using the lowest possible number of parameters. Our representation for lines and circles is not optimal in that sense, as an extra constraint is needed for each object, but the implementation is simpler as it uses only polynomials, not trigonometric functions. Similar use of extra parameters may be found in [28].

### 4.2.1 Representation of lines and circles

A *line*  $l$  is described by its normal and distance from the origin using the usual implicit equation representation:

$$l_0x + l_1y + l_2 = 0.$$

Thus, we can write  $d(p, \mathbf{x}) = l_0x + l_1y + l_2$  and so  $S(\mathbf{x}) = (l_0, l_1, l_2)$  while  $P(p) = (x, y, 1)$ .

Following Pratt [19] we write the equation of a *circle*  $c$  as

$$c_0(x^2 + y^2) + c_1x + c_2y + c_3 = 0.$$

Its radius is  $1/|2c_0|$ ; its centre is  $(-c_1/2c_0, -c_2/2c_0)$ . If  $c_0 = 0$ , the circle degenerates to a line.  $S(\mathbf{x})$  and  $P(p)$  for a circle were given in Section 3.5.

The representation of a *number*  $n$  is obvious, while a *point*  $p$  is represented by its coordinates  $(p_0, p_1)$ . A *point-with-orientation*  $v$  is represented by  $(v_0, v_1; v_2, v_3)$  where the first two components give the point, and the other two give the unit orientation vector.

The normal to a line is a unit vector:	$l_0^2 + l_1^2 - 1 = 0.$
Pratt's normalisation condition for a circle [19]:	$c_1^2 + c_2^2 - 4c_0c_3 - 1 = 0.$
The normal of a point-with-orientation is a unit vector:	$v_2^2 + v_3^2 - 1 = 0.$

Table 1: Normalisation constraints in 2D.

#### 4.2.2 Representation of planar constraints

As well as the externally imposed constraints, other constraints are required internally to ensure that various normalisation conditions required by the representation are satisfied, and to ensure that only objects of the desired shape are allowed. Note that we choose constraint functions that are easy to compute and differentiate, rather than ones which express some sort of geometric distance from the ideal state. Normalisation constraints are given in Table 1.

External geometric constraints are represented as shown in Table 2. Certain cases marked (\*) in Table 2 need a little further explanation: the correct signs for the  $\pm$  in the first two cases are chosen by considering the initial values of the variables. In the case of a line tangent to a circle, the  $\pm$  sign is chosen depending on the half plane that contains the circle. If the signs of the parameters of either the line or the circle are changed, the  $\pm$  sign must also be changed. Finally, in the case of two circles being tangent, the  $\pm$  sign determines whether the circles are tangent from the inside or outside, provided that  $c_0$  and  $c'_0$  have the same sign. If we incorrectly prescribe an outside tangency for two circles which are tangent from the inside, then we may get the expected result (with great error), or one of the circles will change sign, and the geometric result will be an inner tangency (with small error).

Note how certain geometric constraints are represented by *multiple* constraint equations.

#### 4.3 Degenerate cases

As noted, a circle may degenerate to a line: if  $c_0 = 0$ , the circle equation becomes that of a line, and its normalisation condition does likewise. We now consider what happens in the case of other degenerate constraints.

In the case of ‘line goes through the centre of a circle’, when the circle degenerates to a line, the constraint equation requires the two lines to be orthogonal. ‘Line is tangent to a circle’ in the degenerate case requires the two lines to be parallel.

The constraints requiring that a ‘circle is tangent to or orthogonally intersects a circle’, when the second circle is degenerate, leave just the normalisation condition of the degenerate circle. Thus we may prescribe any number of circles with differing tangency and orthogonality conditions, and the constraints will not be considered contradictory. Instead, some of the circles will be forced to be degenerate in the solution returned.

The condition ‘circles are concentric’, when one of the circles is degenerate, reduces to the requirement that the normal of the line (degenerate circle) is zero, which contradicts the previously asserted ‘unit normal’ constraint. However, two degenerate circles are considered to be concentric.

Note however, that with this representation, contradictory constraints can be satisfied by converging to a singular case: if two circles degenerate to lines then they are considered to be tangent, concentric and orthogonally intersecting at the same time!

The square of the distance between two points $p$ and $p'$ is the number $n$ :	$(p_0 - p'_0)^2 + (p_1 - p'_1)^2 - n = 0.$
The cosine of the angle between two lines $l$ and $l'$ is the number $n$ (*):	$l_0 l'_0 + l_1 l'_1 \pm n = 0.$
A line $l$ passes through a point $p$ :	$l_0 p_0 + l_1 p_1 + l_2 = 0.$
A circle $c$ passes through a point $p$ :	$c_0(p_0^2 + p_1^2) + c_1 p_0 + c_2 p_1 + c_3 = 0.$
The point of a point-with-orientation $v$ is equal to point $p$ :	$p_0 - v_0 = 0$ $p_1 - v_1 = 0.$
A circle $c$ is centred at point $p$ :	$2c_0 p_0 + c_1 = 0$ $2c_0 p_1 + c_2 = 0.$
A circle $c$ has fixed curvature $\kappa$ :	$c_0 - \kappa/2 = 0.$
A line $l$ meets a point-with-orientation $v$ :	$l_0 v_0 + l_1 v_1 + l_2 = 0$ $l_0 v_3 - l_1 v_2 = 0.$
A circle $c$ meets a point-with-orientation $v$ :	$c_0(v_0^2 + v_1^2) + c_1 v_0 + c_2 v_1 + c_3 = 0$ $(2c_0 v_0 + c_1)v_3 - (2c_0 v_1 + c_2)v_2 = 0.$
Two lines $l$ and $l'$ are parallel:	$l_0 l'_1 - l_1 l'_0 = 0.$
Two lines $l$ and $l'$ are orthogonal:	$l_0 l'_0 + l_1 l'_1 = 0.$
Two lines $l$ and $l'$ intersect at an angle given by a fixed value $\alpha$ (*):	$l_0 l'_0 + l_1 l'_1 \pm \cos \alpha = 0.$
Three lines $l, l'$ and $l^*$ go through the same point (projectively speaking):	$l_0(l'_1 l_2^* - l_2 l_1^*) + l_1(l_2 l_0^* - l_0 l_2^*) + l_2(l_0 l_1^* - l_1 l_0^*) = 0.$
A line $l$ goes through the centre of a circle $c$ :	$l_0 c_1 + l_1 c_2 - 2l_2 c_0 = 0.$
A line $l$ is tangent to a circle $c$ (*):	$l_0 c_1 + l_1 c_2 - 2l_2 c_0 \pm 1 = 0.$
Two circles $c$ and $c'$ are tangent (*):	$(c'_0 c_1 - c_0 c'_1)^2 + (c'_0 c_2 - c_0 c'_2)^2 - (c_0 \pm c'_0)^2 = 0.$
Two circles $c$ and $c'$ intersect orthogonally:	$(c'_0 c_1 - c_0 c'_1)^2 + (c'_0 c_2 - c_0 c'_2)^2 - c_0^2 - c_0'^2 = 0.$
Two circles $c$ and $c'$ are concentric:	$c'_0 c_1 - c_0 c'_1 = 0$ $c'_0 c_2 - c_0 c'_2 = 0.$

(\*) See text for notes on choice of signs.

Table 2: External geometric constraints in 2D.

## 4.4 Examples in 2D

### 4.4.1 Simple examples

Some simple examples are now presented showing enforcement of a variety of planar constraints.

The first example uses four constraints, one of which is inconsistent with the other three.

The highest priority constraint requires two straight lines to be orthogonal, the next highest priority constraint, in contradiction requires them to be parallel, and two further constraints require one line to pass through the centre of a circle, and the other line to be tangent to it. The initial unconstrained approximation to the solution and the final state after directly enforcing constraints can be seen in Figures 2 and 3. As was expected, the contradictory second constraint was rejected, and the other three satisfied. In Figures 4 and 5 one can see the same configuration where auxiliary objects have been used. In this example, simulated data was used, generated with a density 0.1 (i.e. about 10 points per unit length along the line or circle) and noise 1 unit (i.e. positional error). For comparison, the circle has a radius of about 10 units. Although more steps were required in the second case due to the random initialization of the auxiliary variables, the compensating advantage is that we explicitly obtain their values, for example, for the common tangent point, which may be needed as a construction item in an application.

The second example is a vet's horse<sup>1</sup> involving almost all the constraints above, including tangential lines and circles, concentricity, perpendicularity, and numerically related radii. See Figures 6 and 7, produced from data generated with density 0.1 and noise 2 units.

The final problem is fitting a regular triangle by prescribing the equality of its angles to an auxiliary number and the lengths of its sides to another number. The results are shown in Figures 8 and 9, produced from data generated with density 0.1 and noise 1 unit.

The initial fit has sides of length 1.7649, 2.2268 and 2.3602 units and angles of 63.42, 71.43 and 45.14 degrees. After 18 steps the lengths of the final edges were 2.039805, 2.039808 and 2.039808 (all of the angles were 60 degrees).

### 4.4.2 Examples of profile curve fitting

A simple example of fitting a closed profile such as might be done in reverse engineering can be seen in Figures 10 and 11. Each geometric element is tangent continuous to the adjacent ones and the two smaller circles have the same radii. Here the data were generated with a density 0.1 and noise 0.3 units; real measured data are usually much more exact.

A more complex example coming from the test object shown in Figure 1 is considered next. The simulated data were taken from the vertical faces at the right hand end of the object, with added noise. This point set represents a translational sweep, whose profile elements are tangentially connected straight lines and circles. The recovered construction elements and the final profile can be seen in Figures 12 and 13 respectively.

The computed radii of the bottom and top circles are different due to the relatively small number of data points fitted to them. In a real reverse engineering situation recognising symmetry and enforcing equal radii would be important.

---

<sup>1</sup>A Hungarian expression meaning an example showing all possible problems, as might be used for teaching veterinary students!

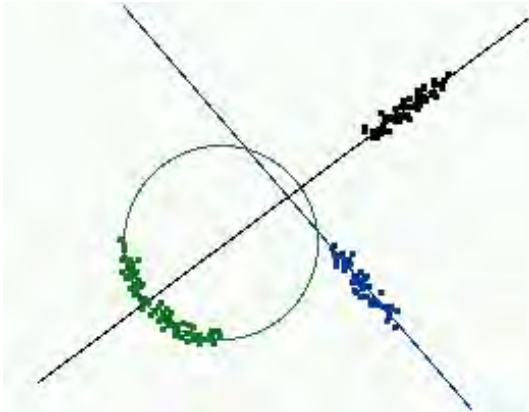


Figure 2: Initial state

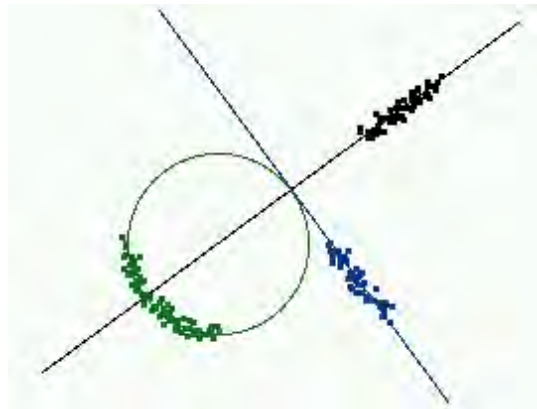


Figure 3: after 6 steps

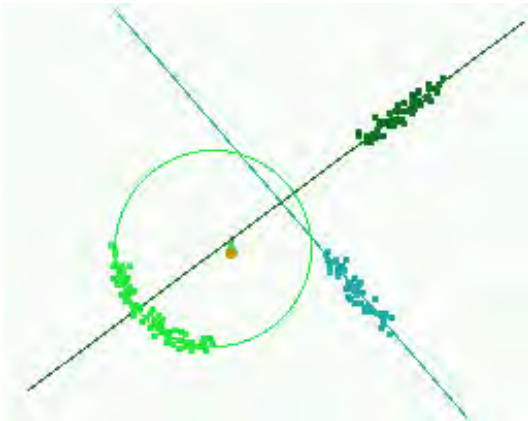


Figure 4: Initial state

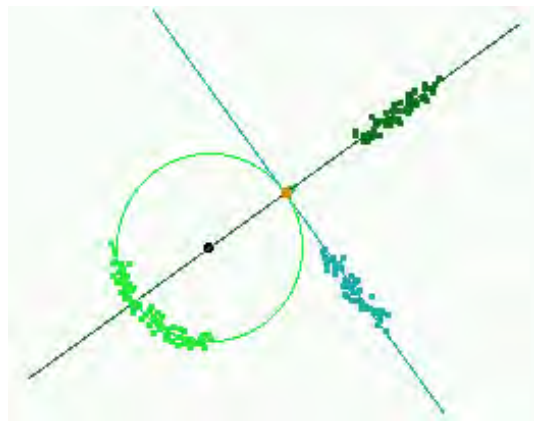


Figure 5: after 27 steps

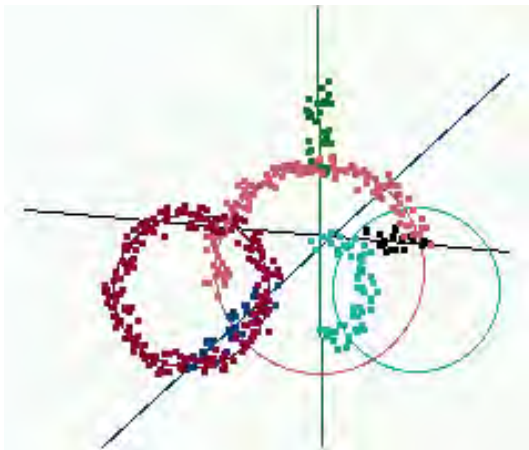


Figure 6: Initial state

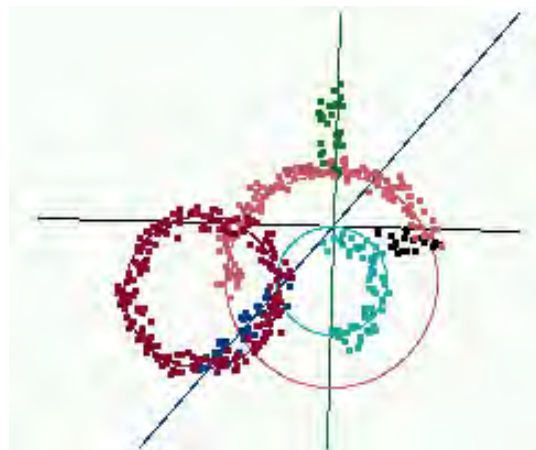


Figure 7: after 14 steps

## 5 Constrained fitting in three dimensions

Using the same principles as in two dimensions, we now suggest representations for surfaces and constraints in 3D. Here it is even more important, due to the larger quantity of data, to

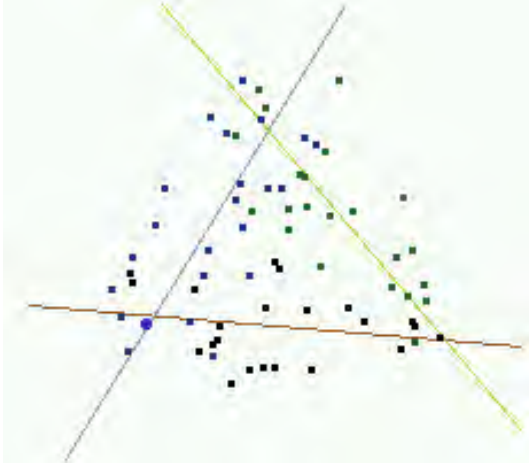


Figure 8: Initial state

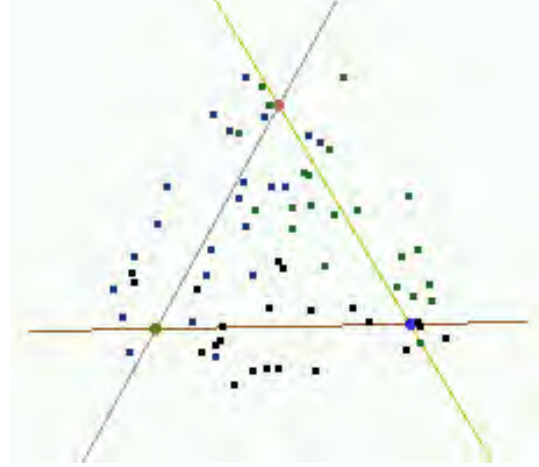


Figure 9: after 8 steps

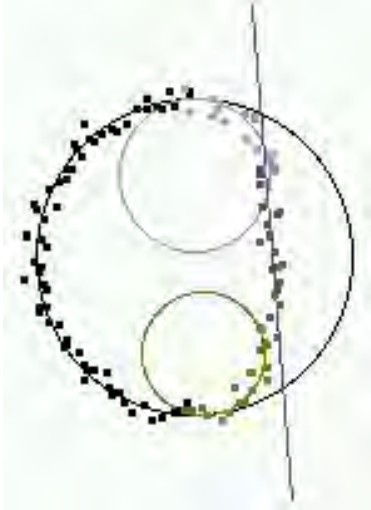


Figure 10: Initial state

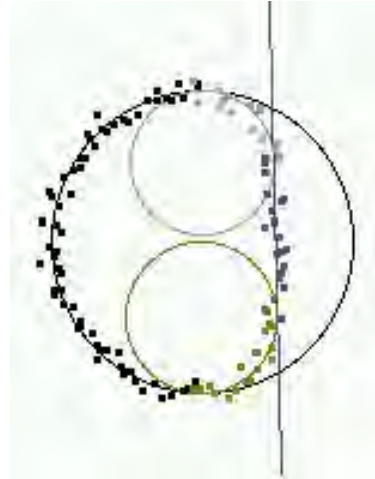


Figure 11: after 12 steps

use representations which allow as much as possible to be precomputed in terms of a suitable approximate distance function before iteration starts. As before,  $S(\mathbf{x})$  contains the unknown parameters for the surfaces, while  $P(p)$  is determined in advance from the point data.

## 5.1 Representation of 3D objects

### 5.1.1 Points and lines

The representations for the objects point, and point-with-orientation follow those used in 2D: a *point*  $p$  is  $(p_0, p_1, p_2)$ , a *point-with-orientation*  $v$  is  $(v_0, v_1, v_2; v_3, v_4, v_5)$ .

The object type *line* may be useful in some constraints. We describe it by its unit direction and its nearest point to the origin:  $(l_0, l_1, l_2; l_3, l_4, l_5)$  (constraints are simpler using this representation than they would be if Plücker coordinates were used).

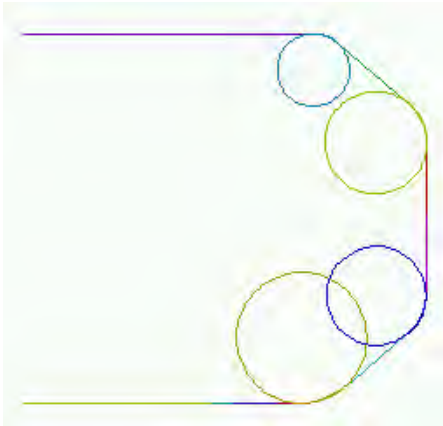


Figure 12: Fitted circles and lines

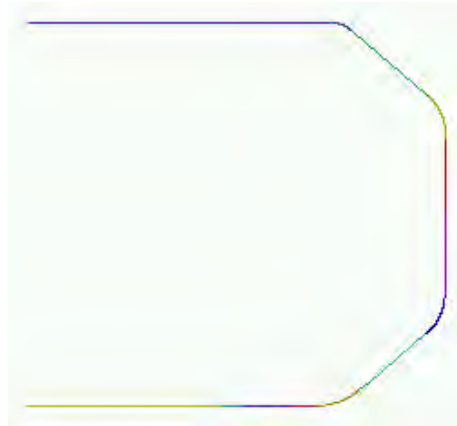


Figure 13: Fitted profile curve

### 5.1.2 Planes, spheres and cylinders

A *plane*  $q$  is described by its normal and distance from the origin, using the usual implicit equation:

$$q_0x + q_1y + q_2z + q_3 = 0.$$

Thus,  $d(p, \mathbf{x}) = q_0x + q_1y + q_2z + q_3$  and so  $S(\mathbf{x}) = (q_0, q_1, q_2, q_3)$  while  $P(p) = (x, y, z, 1)$ .

Following Pratt [19], the equation of a *sphere*  $s$  can be written as

$$s_0(x^2 + y^2 + z^2) + s_1x + s_2y + s_3z + s_4 = 0.$$

Its radius is  $1/|2s_0|$ ; its centre is  $(-s_1/2s_0, -s_2/2s_0, -s_3/2s_0)$ . Thus  $S(\mathbf{x}) = (s_0, s_1, s_2, s_3, s_4)$  while  $P(p) = (x^2 + y^2 + z^2, x, y, z, 1)$ .

Our representation for a *cylinder* is derived from the formula for the squared distance from the axis, in a similar way to the formula for a sphere, which uses the squared distance from the centre:

$$c_3(x^2 + y^2 + z^2 - (c_0x + c_1y + c_2z)^2) + c_4x + c_5y + c_6z + c_7 = 0.$$

The cylinder's radius is  $1/|2c_3|$ ; the nearest point on its axis to the origin is  $-(c_4, c_5, c_6)/2c_3$ ; the direction of its axis is  $(c_0, c_1, c_2)$ . Thus

$$S(\mathbf{x}) = (c_3(1 - c_0^2), c_3(1 - c_1^2), c_3(1 - c_2^2), -c_0c_1c_3, -c_0c_2c_3, -c_1c_2c_3, c_4, c_5, c_6, c_7)$$

and

$$P(p) = (x^2, y^2, z^2, 2xy, 2xz, 2yz, x, y, z, 1).$$

### 5.1.3 Cones

The representations given above for planes, spheres and cylinders are efficient and faithful; they subsume simpler surface types as special cases, not as singularities.

Unfortunately, a *cone* representation cannot be efficient *and* faithful, as well as subsuming simpler surfaces. However, the third property can be coupled with either of the first two. Lukács [14] gave a *faithful* representation; two new *efficient* ones are presented here. The

same approach is used as in the example in Section 3.5, where efficient representation was introduced.

Let us represent the axis by its nearest point to the origin  $o$ , and by a unit vector  $d$  pointing from  $o$  to the apex  $a$  of the cone located at  $a = o + \delta d$ . We denote the apex semi-angle by  $\alpha$ . The distance of a point  $p$  from the cone is its distance from the ruling of the cone lying in the plane containing  $p$  and the axis. As a coordinate system in that plane we choose  $o$  as origin and  $d$  as the local  $x$  axis. The coordinates of  $p$  in this system are  $x = \langle p - o, d \rangle = \langle p, d \rangle$  and  $y = \|p - o - d\langle p, d \rangle\|$ . The signed distance of  $p$  from the ruling is

$$\begin{aligned} \sin \alpha x + \cos \alpha y - \delta \sin \alpha &= \\ \cos \alpha \|p - o - d\langle p, d \rangle\| - \sin \alpha (\delta - \langle p, d \rangle) &= \\ \frac{\cos^2 \alpha (p - o - d\langle p, d \rangle)^2 - (\sin \alpha (\delta - \langle p, d \rangle))^2}{\cos \alpha \|p - o - d\langle p, d \rangle\| + \sin \alpha (\delta - \langle p, d \rangle)} &= \\ \frac{\cos^2 \alpha p^2 - \langle p, d \rangle^2 + \langle p, 2\delta \sin^2 \alpha d - 2 \cos^2 \alpha o \rangle + \cos^2 \alpha o^2 - \sin^2 \alpha \delta^2}{\cos \alpha \|p - o - d\langle p, d \rangle\| + \sin \alpha (\delta - \langle p, d \rangle)}. \end{aligned} \quad (7)$$

Rational functions (and square roots) do not give an efficient representation, so we approximate the denominator to simplify it. Consider the circle example again. There the Euclidean signed distance is  $\|p - c\| - r$ , which implicitly involves a square root, so we try a faithful approximation. We can rewrite  $\|p - c\| - r$  as  $((p - c)^2 - r^2)/(\|p - c\| + r)$ . Now for points close to the circle  $\|p - c\| \approx r$ , so the denominator can be approximated by  $2r$ , giving Pratt's formula. Applying a similar approach to the cone, we can eliminate the square root from Eqn. (7) obtaining

$$d(\mathbf{x}, p) = \frac{\cos^2 \alpha p^2 - \langle p, d \rangle^2 + \langle p, 2\delta \sin^2 \alpha d - 2 \cos^2 \alpha o \rangle + \cos^2 \alpha o^2 - \sin^2 \alpha \delta^2}{2 \sin \alpha (\delta - \langle p, d \rangle)}. \quad (8)$$

However, the denominator depends on the point  $p$  as well as the surface parameters  $\mathbf{x}$ , so the fraction cannot be separated into  $S(\mathbf{x}), P(p)$  form.

We give two different approaches to overcome this problem. Firstly, we further approximate the already approximated denominator by its average. Secondly, we approximate the original denominator of Eqn. (7) by twice the other term, and keep its point part fixed at the value computed from the initial guess. If the initial estimates for the surface parameters are quite close to the actual values, then this second approach may be acceptable. This may be dangerous to do if the initial estimate is poor: practical experiments demonstrate that we end up with poor estimates of the axis direction. Nevertheless, the second approach is clearly much more efficient. In the following, we refer to the second approach as the ‘‘fast’’ method, and the first method as the ‘‘safe’’ method. We now describe both approaches in further detail.

In the safe approach, we assume that the barycentre of the points is the (3D) origin, and approximate the denominator of Eqn. (7) by twice the average of the second term:  $2\delta \sin \alpha$ . Introducing  $\kappa = 1/(2\delta \sin \alpha)$  we obtain

$$d(\mathbf{x}, p) = \kappa \cos^2 \alpha p^2 - \kappa \langle p, d \rangle^2 + \langle p, \sin \alpha d - 2\kappa \cos^2 \alpha o \rangle + \kappa \cos^2 \alpha o^2 - \frac{1}{4\kappa}.$$

Introducing further new variables  $\lambda = \kappa \cos^2 \alpha$ ,  $o' = \sin \alpha d - 2\lambda o$  and  $A = \lambda o^2 - 1/(4\kappa)$ , the distance function can be written as

$$\lambda p^2 - \kappa \langle p, d \rangle^2 + \langle p, o' \rangle + A. \quad (9)$$

Let us now rewrite the formula for  $A$  using  $o'^2 = \sin^2 \alpha + 4\lambda^2 o^2 = 1 - \lambda/\kappa + 4\lambda^2 o^2$ :

$$\begin{aligned} A &= \lambda o^2 - \frac{1}{4\kappa} \\ &= \frac{o'^2 - 1 + \lambda/\kappa}{4\lambda} - \frac{1}{4\kappa} \\ &= \frac{o'^2 - 1}{4\lambda} \end{aligned} \tag{10}$$

From the definition of  $o'$  we have  $\langle d, o' \rangle = \sin \alpha$ ; squaring this gives:

$$\langle d, o' \rangle^2 = 1 - \cos^2 \alpha = 1 - \lambda/\kappa. \tag{11}$$

Finally we can eliminate  $\kappa$  by introducing a new variable  $d' = \sqrt{\kappa}d$  (so  $\kappa = d'^2$ , as  $d$  is a unit vector). Putting all of these ideas together, we arrive at the following distance function and constraints:

$$d(p, \mathbf{x}) = \lambda p^2 - \langle p, d' \rangle^2 + \langle p, o' \rangle + A \tag{12}$$

$$o'^2 - 4\lambda A = 1 \tag{13}$$

$$\langle d', o' \rangle^2 = d'^2 - \lambda \tag{14}$$

Eqn. (12) comes from Eqn. (9); Eqn. (13) is a rearranged form of Eqn. (10) and has the same basic form as Pratt's condition; Eqn. (14) is a rearrangement of Eqn. (11) (while it is not as simple as the corresponding condition for a cylinder, this is the price to pay for the rather simple error function being similar to the one for a cylinder).

If the term  $\langle p, d' \rangle^2$  is left out of Eqn. (12), then we obtain the equation of a sphere, and the normalisation condition in Eqn. (13) is just its normalisation condition. The cone and the sphere intersect where the omitted term is zero, that is, on the plane through the origin orthogonal to the axis of the cone. Moreover, because of the squaring, they meet there tangentially, so the centre of the sphere, at  $-o'/(2\lambda)$ , is on the axis, in the opposite direction from  $o$  to the apex, and at a distance from the apex of  $1/(2\lambda \sin \alpha)$ .

Now let us consider the second, fast, approach. Assume we have a good initial estimate for the cone. We approximate the denominator of Eqn. (7) by twice the first term *computed from the initial estimate*. We do not update this value in subsequent iterations. That term is the distance  $d(p)$  of  $p$  from the axis multiplied by  $\cos \alpha$ . For the time being we forget about the  $2d(p)$  in the denominator (it will be incorporated in  $P(p)$ , so we simply divide the numerator by  $\cos \alpha$ ). Introducing the new variables  $\lambda = \cos \alpha$ ,  $d' = d/\sqrt{\lambda}$ ,  $o' = (2\delta \sin^2 \alpha d - 2 \cos^2 \alpha o)/\lambda$ ,  $A = \lambda o^2 - \sin^2 \alpha \delta^2/\lambda$ , we obtain

$$d(p, \mathbf{x}) = \lambda p^2 - \langle p, d' \rangle^2 + \langle p, o' \rangle + A,$$

which is the same as Eqn. (12), but with different constraints:

$$\begin{aligned} \lambda d'^2 &= 1 \\ (1 - \lambda^2)(o'^2 - 4\lambda A) &= \lambda \langle d', o' \rangle^2 \end{aligned}$$

(verifying the latter is simple but tedious).

In either representation,  $S(\mathbf{x})$  takes the form:

$$S(d', \lambda, o', A) = (\lambda - d'_x{}^2, \lambda - d'_y{}^2, \lambda - d'_z{}^2, -\lambda d'_x d'_y, -\lambda d'_x d'_z, -\lambda d'_y d'_z, o'_x, o'_y, o'_z, A)^T.$$

In the safe representation  $P(p)$  is

$$P(p) = (p_x^2, p_y^2, p_z^2, 2p_x p_y, 2p_x p_z, 2p_y p_z, p_x, p_y, p_z, 1),$$

while in the fast representation this must be divided by  $2d(p)$  (and the constraints are different).

In summary, we use the representation below for the cone:

$$c_3(x^2 + y^2 + z^2) - (c_0x + c_1y + c_2z)^2 + c_4x + c_5y + c_6z + c_7 = 0.$$

At times, one may need a faithful representation rather than an efficient one. Of course, the distance function will be rational at best; applying similar techniques as above to Eqn. (8) we obtain

$$d(\mathbf{x}, p) = \frac{o_3(x^2 + y^2 + z^2) - o_4(o_0x + o_1y + o_2z)^2 + o_5x + o_6y + o_7z + o_8}{2((o_5 - x)o_0 + (o_6 - y)o_1 + (o_7 - z)o_2)}.$$

#### 5.1.4 Torus

We now consider representations for a *torus*. Take the plane containing the axis of the torus and a data point  $p$ . For our coordinate system in that plane, we choose an origin at the centre  $c$  of the torus, as a local  $x$  axis we use a unit vector  $d$  in the torus axis direction, and the local  $y$  axis points towards the half plane containing  $p$ . In this system, the coordinates of  $p$  are  $x = \langle p - c, d \rangle$  and the distance  $y = d(p)$  of  $p$  from the axis ( $d(p)^2 = (p - c)^2 - \langle p - c, d \rangle^2$ ). The centre of the corresponding circular section of the torus is  $(0, R)$ , where  $R$  is the major radius of the torus; the radius  $r$  of the circular section is the minor radius of the torus. Using Pratt's form the signed distance is

$$\begin{aligned} d(\mathbf{x}, p) &= \kappa(p - c)^2 + A + R'd(p) = \kappa p^2 - 2\kappa\langle p, c \rangle + \kappa c^2 + A + R'd(p) \\ &= \frac{(\kappa p^2 - 2\kappa\langle p, c \rangle + B)^2 - R'^2((p - c)^2 - \langle p - c, d \rangle^2)}{\kappa p^2 - 2\kappa\langle p, c \rangle + B - R'd(p)} \end{aligned} \quad (15)$$

(as usual,  $\kappa = 1/2r$ ,  $R' = -2\kappa R$ ,  $R'^2 - 4\kappa A = 1$ ,  $B = \kappa c^2 + A$ , so  $R'^2 - 4B\kappa = 1 - 4\kappa^2 c^2$ ).

We now follow the approach taken in the cone case, and approximate the denominator. For a faithful approximation we use  $2(\kappa p^2 - 2\kappa\langle p, c \rangle + B)$ ; for a fast approximation we use  $2R'd(p)$ .

For the faithful version, we introduce  $d' = R'd$ , so Eqn. (15) becomes

$$d(\mathbf{x}, p) = \frac{1}{2} \left( \kappa p^2 - 2\kappa\langle p, c \rangle + B - \frac{d'^2(p - c)^2 - \langle p - c, d' \rangle^2}{\kappa p^2 - 2\kappa\langle p, c \rangle + B} \right),$$

and the only constraint is

$$d'^2 - 4B\kappa = 1 - 4\kappa^2 c^2.$$

Note that in this representation  $r = 1/2\kappa$ , and  $R = r\|d'\|$ .

In turn, let us consider the fast version. Introducing  $\kappa' = \kappa^2/R'$ ,  $d' = \sqrt{R'}d$ ,  $B' = 2B\kappa/R' - R'$  and  $C = B^2/R' - R'c^2 + \langle c, d' \rangle^2$  we obtain

$$d(\mathbf{x}, p) = \frac{1}{2d(p)} (\kappa' \langle p, p - 2c \rangle^2 + \langle p, d' \rangle \langle p - 2c, d' \rangle + B' \langle p, p - 2c \rangle + C).$$

This time two constraints must be imposed:

$$\begin{aligned} d'^2 \left( 4\kappa' c^2 - d'^2 - 2B' \right) &= 1, \\ 4\kappa' \left( C - \langle c, d' \rangle^2 \right) + 1 &= B'^2. \end{aligned}$$

Note that in this representation  $r = 1/2\sqrt{\kappa'd'^2}$ ,  $R = rd'^2$ .  $S(\mathbf{x})$  and  $P(p)$  now take the form

$$\begin{aligned} S(d', \kappa', c, B', C) &= S(t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8) \\ &= (t_3, -t_3t_4, -t_3t_5, -t_3t_6, 4t_3t_4^2 + t_0^2 + t_7, 4t_3t_5^2 + t_1^2 + t_7, 4t_3t_6^2 + t_2^2 + t_7, \\ &\quad 4t_3t_4t_5 + t_0t_1, 4t_3t_4t_6 + t_0t_2, 4t_3t_5t_6 + t_1t_2, -t_4t_7, -t_5t_7, -t_6t_7, t_8); \\ P(p) &= ((x^2 + y^2 + z^2)^2, 4x(x^2 + y^2 + z^2), 4y(x^2 + y^2 + z^2), 4z(x^2 + y^2 + z^2), \\ &\quad x^2, y^2, z^2, 2xy, 2xz, 2yz, 2x, 2y, 2z, 1)/(2d(p)). \end{aligned}$$

### 5.1.5 Representation of 3D constraints

As in the case of the planar objects, our representations for the sphere, cylinder, cone and torus are over-parametrised, and shape constraints linking the parameters must be imposed both for normalisation and to obtain surfaces of the correct type. Constraints in three dimensions are thus again of two types: internal constraints on the parameters of the objects, and external constraints expressing relationships between objects. The internal normalisation and shape conditions are given in Table 3. External geometric constraints are represented as in Table 4. Again, where we say an entity *meets* a point-with-orientation we mean that the entity passes through the point part, and at that point has normal parallel to the vector part. Constraints for tori have been omitted for reasons of space, but can easily be obtained from the preceding discussion.

There is a subtle difficulty not present in the planar case which arises in the rows of Table 4 marked (\*). Parallelism of two vectors should be expressed by one equation less than the dimension of the space, which is simple in the planar case but not directly possible in 3D. There are several alternative ways to enforce parallelism. One can set the dot product of the vectors to zero, which leaves a degree of freedom. One can introduce an auxiliary *number* object and prescribe that one of the vectors is a multiple *number* of the other (which breaks down if the other vector is a zero vector). For unit vectors, one can overconstrain the system by making two vectors equal (apart from sign). We choose the last option for two unit vectors. In other cases, if one of the vectors is known to be non-zero, we chose the ‘multiple’ approach. In that case, given the constraint  $a = nb$ , the auxiliary number may be initialised to be  $n = \langle a, b \rangle / b^2$ ; if  $b$  is known to be a unit vector, then we simply use  $n = \langle a, b \rangle$ .

## 5.2 Degenerate cases

A sphere or a cylinder may degenerate to a plane if  $s_0 = 0$  or  $c_3 = 0$  respectively. In these cases the equation and the normalisation condition become those of a plane, in a similar way to the planar case where a circle becomes a line. The constraints also behave similarly. Note that the axis-direction of a degenerate cylinder is some arbitrary direction in the plane.

A safe cone representation can degenerate to a cylinder if  $o_0^2 + o_1^2 + o_2^2 = o_3$ ; in that case the description is faithful. A description of a plane given by  $o_0 = o_1 = o_2 = o_3 = 0$  is also faithful. If  $o_3 = 0$  but  $(o_0, o_1, o_2) \neq 0$ , then we have an unhelpful (overparametrised, and not

The direction of a point-with-orientation is a unit vector—	$v_3^2 + v_4^2 + v_5^2 - 1 = 0.$
For a line— its direction is a unit vector: its point is the one nearest the origin:	$l_0^2 + l_1^2 + l_2^2 - 1 = 0,$ $l_0 l_3 + l_1 l_4 + l_2 l_5 = 0.$
The normal to a plane is a unit vector—	$q_0^2 + q_1^2 + q_2^2 - 1 = 0.$
Pratt's normalisation condition for a sphere [19]—	$s_1^2 + s_2^2 + s_3^2 - 4s_0 s_4 - 1 = 0.$
For a cylinder— its axis direction is a unit vector: the axis is defined by its point nearest to the origin: a Pratt-like condition:	$c_0^2 + c_1^2 + c_2^2 - 1 = 0,$ $c_0 c_4 + c_1 c_5 + c_2 c_6 = 0,$ $c_4^2 + c_5^2 + c_6^2 - 4c_3 c_7 - 1 = 0.$
For the safe/fast cone representation— a Pratt-like condition: a further algebraic condition:	$o_4^2 + o_5^2 + o_6^2 - 4o_3 o_7 - 1 = 0,$ $(o_0 o_4 + o_1 o_5 + o_2 o_6)^2 - o_0^2 - o_1^2 - o_2^2 + o_3 = 0.$
For the fast cone representation— two algebraic conditions:	$o_3(o_0^2 + o_1^2 + o_2^2) - 1 = 0,$ $(1 - o_3^2)(o_4^2 + o_5^2 + o_6^2 - 4o_3 o_7) -$ $o_3(o_0 o_4 + o_1 o_5 + o_2 o_6)^2 = 0.$
For the faithful cone representation— the axis direction is a unit vector: a Pratt-like condition:  a further algebraic condition:	$o_0^2 + o_1^2 + o_2^2 - 1 = 0,$ $o_5^2 + o_6^2 + o_7^2 - 4o_3 o_8 -$ $4o_4(o_0 o_5 + o_1 o_6 + o_2 o_7)^2 = 0,$ $4o_4(o_4 - o_3) - 1 = 0.$
For the fast torus representation— two algebraic conditions:	$(t_0^2 + t_1^2 + t_2^2)4t_3((t_4^2 + t_5^2 + t_6^2) -$ $(t_0^2 + t_1^2 + t_2^2) - 2t_7) - 1 = 0,$ $4t_3(t_8 - (t_0 t_4 + t_1 t_5 + t_2 t_6)^2) - t_7^2 + 1 = 0.$
For the faithful torus representation—	$t_0^2 + t_1^2 + t_2^2 - 4t_3 t_7 + 4t_3^2(t_4^2 + t_5^2 + t_6^2) - 1 = 0.$

Table 3: Normalisation and shape constraints in 3D.

A plane $q$ goes through a point $p$ :	$q_0p_0 + q_1p_1 + q_2p_2 + q_3 = 0.$
A sphere $s$ goes through a point $p$ :	$s_0(p_0^2 + p_1^2 + p_2^2) + s_1p_0 + s_2p_1 + s_3p_2 + s_4 = 0.$
A cylinder $c$ goes through a point $p$ :	$c_3(p_0^2 + p_1^2 + p_2^2 - (p_0c_0 + p_1c_1 + p_2c_2)^2) + c_4p_0 + c_5p_1 + c_6p_2 + c_7 = 0.$
A cone $o$ goes through a point $p$ :	$o_3(p_0^2 + p_1^2 + p_2^2) - (p_0o_0 + p_1o_1 + p_2o_2)^2 + o_4p_0 + o_5p_1 + o_6p_2 + o_7 = 0.$
A plane $q$ meets a point-with-orientation $v$ . (*):	$q_0v_0 + q_1v_1 + q_2v_2 + q_3 = 0$ $v_3 \pm q_0 = 0$ $v_4 \pm q_1 = 0$ $v_5 \pm q_2 = 0.$
A sphere $s$ meets a point-with-orientation $v$ (*):	$s_0(v_0^2 + v_1^2 + v_2^2) + s_1v_0 + s_2v_1 + s_3v_2 + s_4 = 0$ $2s_0v_0 + s_1 \pm v_3 = 0$ $2s_0v_1 + s_2 \pm v_4 = 0$ $2s_0v_2 + s_3 \pm v_5 = 0.$
A cylinder $c$ meets a point-with-orientation $v$ (*):	$c_3(v_0^2 + v_1^2 + v_2^2 - (v_0c_0 + v_1c_1 + v_2c_2)^2) + c_4v_0 + c_5v_1 + c_6v_2 + c_7 = 0$ $2c_3(v_0 - (v_0c_0 + v_1c_1 + v_2c_2)c_0) + c_4 \pm v_3 = 0$ $2c_3(v_1 - (v_0c_0 + v_1c_1 + v_2c_2)c_1) + c_5 \pm v_4 = 0$ $2c_3(v_2 - (v_0c_0 + v_1c_1 + v_2c_2)c_2) + c_6 \pm v_5 = 0.$
A cone $o$ meets a point-with-orientation $v$ (expressed with the help of an auxiliary number $n$ ):	$o_3(v_0^2 + v_1^2 + v_2^2) - (v_0o_0 + v_1o_1 + v_2o_2)^2 + o_4v_0 + o_5v_1 + o_6v_2 + o_7 = 0$ $2o_3v_0 - 2(v_0o_0 + v_1o_1 + v_2o_2)o_0 + o_4 - nv_3 = 0$ $2o_3v_1 - 2(v_0o_0 + v_1o_1 + v_2o_2)o_1 + o_5 - nv_4 = 0$ $2o_3v_2 - 2(v_0o_0 + v_1o_1 + v_2o_2)o_2 + o_6 - nv_5 = 0.$
A point $p$ is on the axis of a cylinder $c$ (expressed with the help of an auxiliary number $n$ ):	$2c_3p_0 + c_4 - nc_0 = 0$ $2c_3p_1 + c_5 - nc_1 = 0$ $2c_3p_2 + c_6 - nc_2 = 0.$
A plane $q$ is parallel to the axis of a cylinder $c$ :	$q_0c_0 + q_1c_1 + q_2c_2 = 0.$
A plane $q$ is tangent to a cylinder $c$ :	$q_0c_0 + q_1c_1 + q_2c_2 = 0$ $2q_3c_3 - q_0c_4 - q_1c_5 - q_2c_6 = 0.$
A plane $q$ is orthogonal to a cylinder $c$ (*):	$q_0 \pm c_0 = 0$ $q_1 \pm c_1 = 0$ $q_2 \pm c_2 = 0.$
A plane $q$ is tangent to a cone $o$ :	$o_0(q_0 \pm o_4) + o_1(q_1 \pm o_5) + o_2(q_2 \pm o_6) = 0$ $q_0o_4 + q_1o_5 + q_2o_6 - 2q_3o_3 \pm 1 = 0.$
A sphere $s$ is centred on the axis of a cylinder $c$ (expressed with the help of an auxiliary number $n$ ):	$c_3s_1 - s_0c_4 - nc_0 = 0$ $c_3s_2 - s_0c_5 - nc_1 = 0$ $c_3s_3 - s_0c_6 - nc_2 = 0.$
Two cylinders $c$ and $c'$ are coaxial:	$c'_3c_4 - c_3c'_4 = 0$ $c'_3c_5 - c_3c'_5 = 0$ $c'_3c_6 - c_3c'_6 = 0$ $c_0 \pm c'_0 = 0$ $c_1 \pm c'_1 = 0$ $c_2 \pm c'_2 = 0.$
A cylinder $c$ and a cone $o$ are coaxial (expressed with the help of two auxiliary numbers $n$ and $n'$ ):	$o_0 - nc_0 = 0$ $o_1 - nc_1 = 0$ $o_2 - nc_2 = 0$ $o_3c_4 - c_3o_4 - n'e_0 = 0$ $o_3c_5 - c_3o_5 - n'e_1 = 0$ $o_3c_6 - c_3o_6 - n'e_2 = 0.$

(\*) See text for comment.

Table 4: External geometric constraints in 3D.

faithful) representation of two parallel planes. Both  $(o_0, o_1, o_2)$  and  $(o_4, o_5, o_6)$  give the unit normal (they may be opposite); their bisector goes through  $(o_4, o_5, o_6)/2$  and the distance between the planes is  $\sqrt{1+4o_7}$ . If  $o_7 = -1/4$ , the two planes coincide; if  $o_7 < -1/4$ , the two planes are complex, that is, no real points satisfy the equation.

We now consider constraints between a sphere and a cylinder when either or both are degenerate. An equality constraint on the radii implies that either *both* of the objects are degenerate or *neither* of them. A degenerate sphere is always centred-on-the-axis of a degenerate cylinder. A degenerate sphere centred-on-the-axis of a non-degenerate cylinder means that the plane is orthogonal to the cylinder. A non-degenerate sphere centred-on-the-axis of a degenerate cylinder is contradictory, as it means that the direction in the plane given by the axis-direction is orthogonal to the plane.

### 5.3 Examples in 3D

Convincingly demonstrating the results of constrained fitting in three dimensions is rather tricky. On the one hand, large numerical tables of final parameter values are rather tedious and difficult to interpret. On the other hand, showing pictures of the final geometry from a single view also makes it hard to be certain that the constraints are properly satisfied. Perhaps the strongest argument that correct geometry is produced is that *valid* ACIS models can be built using the constrained geometric elements and auxiliary quantities, which would not be possible if the constraints were not satisfied to a high accuracy.

Here we present two examples, firstly a relatively simple object showing a few constraints, and secondly a more realistic one from our test object in Figure 1.

#### 5.3.1 Simple 3D example

This object, shown in Figure 14, consists of a hemisphere with an aligned cylindrical through-hole of radius half that of the sphere. This leads to the following constraints: the centre of the sphere lies on the axis of the cylinder, the axis of the cylinder is parallel to the normal of the base plane, and the radii of the sphere and cylinder are in the ratio 2:1. 4730 simulated sample points were computed with uniform noise of standard deviation of 0.15 units (the radius of the sphere is 2 units). The geometry found after constrained fitting, using 6 iteration steps, has the following numerical values; the initial estimates for each quantity found using unconstrained fitting are shown in brackets. The distance between the centre of the sphere and the axis of the cylinder after constrained fitting was  $9.1 \times 10^{-10}$  ( $8.7 \times 10^{-4}$ ), the angle between the axis of the cylinder and the plane normal was 0 ( $7.2 \times 10^{-2}$ ), the ratio of the radii of the sphere and cylinder was 2 (2.007).

#### 5.3.2 A more complex 3D example

Here the input data was generated by simulation from the middle part of the test object in Figure 1, where multiple surfaces meet with tangential continuity. These faces must be simultaneously fitted under constraints that they meet tangentially, because fitting each one separately and intersecting them to find the edges only works in cases when the edges between adjacent surfaces are sharp. To fit the surfaces with tangential continuity, auxiliary elements—the smooth edge curves and vertices—must be explicitly created. After solving the constrained fitting problem, the auxiliary entities must be directly transferred to the solid modelling kernel

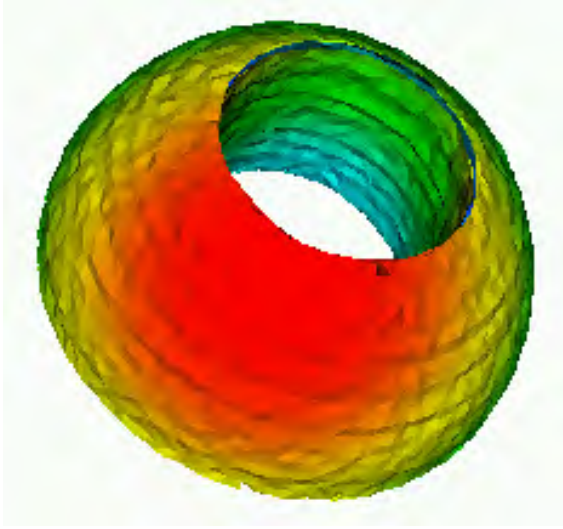


Figure 14: Simulated noisy data

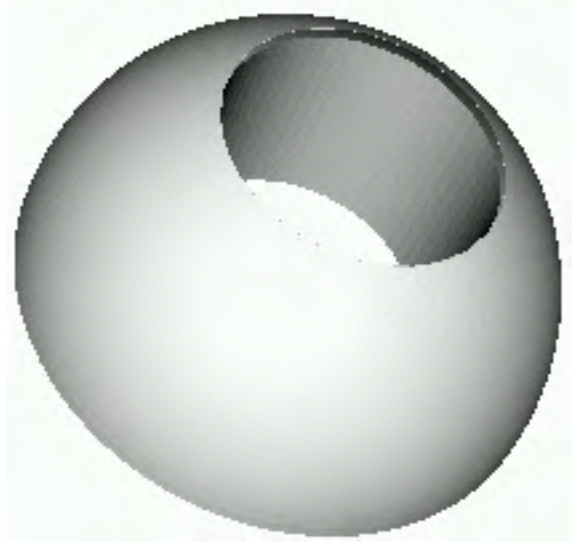


Figure 15: Reconstructed simple object

to be directly built into the bounding geometry. Note that the constraints must be satisfied to within the modelling kernel's tolerances if a valid model is to be successfully created.

The segmented input data is shown in Figure 16. To obtain the desired geometry in addition to the primary surfaces, many constraints and auxiliary objects must be introduced. The primary surfaces are three planes, three cylinders and a torus (in this example the faithful representation was used). We mention just some of the auxiliary geometry. There are two types of cylinder-torus join. In both cases we must define the plane containing the shared smooth edge which is a circular arc. This plane must be orthogonal to the axis of the cylinder and should contain either the centre of the torus (for the top cylinder) or the axis of the torus (for the side cylinders). In the former case, the radius of the cylinder must equal the difference of the major and minor radii of the torus; also the cylinder and the torus must be coaxial. In the latter case the radius of each cylinder must equal the minor radius of the torus. Furthermore, the plane mentioned above must contain the vertices of the edge. The tangential continuity constraint is expressed by requiring that the surfaces meet at vertices with a common point-with-orientation. The final system we produced for this relatively simple example consisted of the surprisingly large numbers of 141 variables in 37 objects and 192 equations in 90 constraints. To determine a minimal set of constraints for this configuration was not attempted, and seems a difficult problem in general.

Unsurprisingly, although we *did* succeed in solving the system for this example, we found it trickier than solving the two dimensional examples. Also, this example is more sensitive to initial estimates. We used ad-hoc initialisations: initial estimates for points and points-with-orientation were generated from a triangulation of the simulated point data, numbers denoting equal radii were initialised as the average of the nearly equal radii, etc. The iteration process converged after 99 iterations; when it was complete all constraints were satisfied within a tolerance value of  $10^{-12}$ . The time taken to solve this system was 10 seconds on an 800 MHz Pentium III processor.

In detail, the following results were obtained after constrained fitting. Cylinders are given by the axis point nearest to the origin, the axis direction and the radius; the torus is given by its centre, axis direction, major and minor radius; each plane is given by its normal and

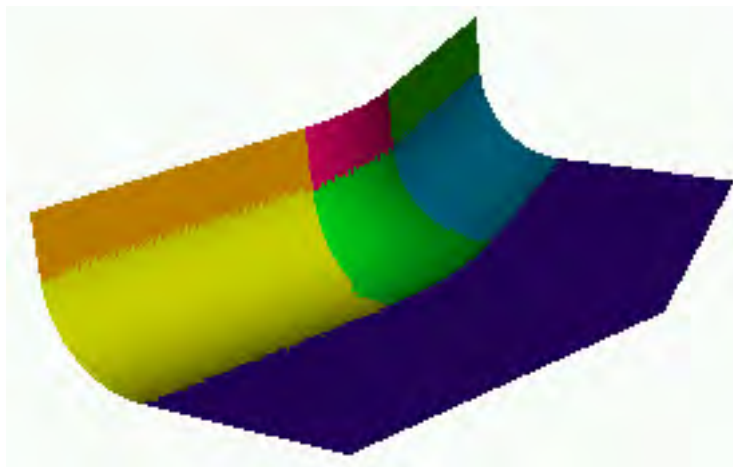


Figure 16: Segmented multiple region

distance from the origin. The final surfaces were (note the agreement of various radii, and the agreement of parameters for symmetrically placed geometry.):

$(243.059, -66.2914, 80.0411)$ ; $(0.263113, 0.964765, 4.45433 \times 10^{-5})$ ; 30.0496	left cylinder
$(146.933, -0.000582371, 80.025)$ ; $(-0.000167365, -5.25969 \times 10^{-7}, 1)$ ; 110.181; 30.0496	torus
$(243.059, 66.2905, 80.0411)$ ; $(-0.26311, 0.964766, -4.35279 \times 10^{-5})$ ; 30.0496	right cylinder
$(-0.000167365, -5.25969 \times 10^{-7}, 1)$ ; -49.9508	bottom plane
$(146.947, -0.000540294, 0.0245937)$ ; $(-0.000167365, -5.25969 \times 10^{-7}, 1)$ ; 80.1315	top cylinder
$(0.964765, -0.263113, 0.000161329)$ ; -221.901	left plane
$(0.964766, 0.26311, 0.000161606)$ ; -221.9	right plane.

The parameters of the auxiliary planes containing the cylinder-torus smooth boundary curves were found to be almost symmetric:

$$\begin{aligned} & (0.263113, 0.964765, 4.45433 \times 10^{-5}); -38.663 \\ & (-0.26311, 0.964766, -4.35279 \times 10^{-5}); 38.6636 \\ & (0.000167365, -5.25969 \times 10^{-7}, 1); -80.0004 \end{aligned}$$

as were the positions of the four internal vertices:

$$\begin{aligned} & (224.241, -21.0842, 80.0379) \\ & (224.241, 21.0828, 80.038) \\ & (253.237, -28.9906, 49.9932) \\ & (253.237, 28.9892, 49.9932). \end{aligned}$$

In a real reverse engineering context, it is quite possible that further constraints would also be used. For example, certain radii in the above example should probably have integer values.

## 6 Conclusion

We have considered the problem of simultaneously fitting several geometric elements to point data, with constraints between these elements. This problem is particularly relevant to reverse

engineering. Firstly, certain geometric configurations, such as surfaces meeting in smooth edges, cannot be successfully reconstructed without constrained fitting. Secondly, real mechanical components usually display regularities and symmetries which must be present in the reconstructed model if it is to be of real use to an engineer.

In reverse engineering, it is likely that the constraints will be automatically generated; under such circumstances it is unlikely they will all be mutually consistent. However, we assume that they can be prioritised using figures-of-merit. We have given a method to fit curves and surfaces to measured data while simultaneously satisfying as many constraints as possible which do not contradict constraints of higher merit.

Furthermore, in reverse engineering, very large numbers of data points are typically processed. Since fitting methods are generally iterative, the processing time for each iteration is crucial. We have shown how to overcome this problem by determining special approximate distance functions. In these, terms containing the point data are separated from those containing the parameters of the geometric elements being fitted. The terms containing the data points need only be evaluated once before iteration starts. The iterations themselves merely involve the terms containing the parameter values.

Experimental results have been presented to demonstrate the effectiveness and efficiency of our procedure, using examples from reverse engineering, including reconstruction of smooth profile curves, and reconstruction of a set of tangentially continuous surfaces.

Future work is directed towards finding good initial estimates for the parameters describing the auxiliary objects, automatically deriving constraints for a given set of data, and reducing sets of constraints for a given configuration to a minimal set. Rejecting contradictory constraints was found to be more sensitive to initial data than expected; this also needs some further investigation.

## Acknowledgements

This work was supported by the National Science Foundation (OTKA) of the Hungarian Academy of Sciences, grant 26203, and UK EPSRC grant GR/M 78267. R. Martin also wishes to thank CADMUS Ltd., Budapest for financial support.

The useful comments of the anonymous referees are highly appreciated.

## References

- [1] J. Abedie and J. Carpentier, “Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints”, *Optimization*, Ed: R. Fletcher, Academic Press, London, 1969, pp. 37–47
- [2] P. Benkő, T. Várady, “Best Fit Translational and Rotational Surfaces for Reverse Engineering Shapes”, *The Mathematics of Surfaces IX*, Springer, 2000, pp. 70–81
- [3] P. Benkő, R. R. Martin and T. Várady, “Algorithms for reverse engineering boundary representation models”, *Computer Aided Design*, to appear.
- [4] B. Brüderlin, D. Roller, (Eds.), “Geometric constraint solving and applications”, Springer, 1998.
- [5] G. Celniker and D. Gossard, “Deformable curve and surface finite-elements for free-form shape design”, *Computer Graphics (SIGGRAPH 91)*, Vol 25, No 4, 1991, pp. 257–266

- [6] C. Durand, *Symbolic and numerical techniques for constraint solving*, Ph.D. thesis, Purdue University Computer Science Department, 1998
- [7] P. Fua and C. Brechbuler, “Imposing hard constraints on deformable models through optimisation in orthogonal subspaces”, *Computer Vision and Image Understanding*, Vol 65, No 2, 1997, pp. 148–162
- [8] S. F. F. Gibson and B. Mirtich, *A survey of deformable modeling in computer graphics*, Mitsubishi Electric Research Laboratory, Technical Report TR-97-19, 1997
- [9] C. M. Hoffmann and R. Joan-Arinyo, “Symbolic constraints in constructive geometry”, *Journal of Symbolic Computation*, Vol 23, 1997, pp. 287-300
- [10] J. Hoschek and P. Kaklis (Eds.), *Advanced course on FAIRSHAPE*, B. G. Teubner, 1996
- [11] R. Joan-Arinyo and A. Soto, “A correct rule-based geometric constraint solver”, *Computers and Graphics*, Vol 21, No 5, 1997, pp. 599–609
- [12] H. Lamure and D. Michelucci, “Solving geometric constraint systems by homotopy”, In: *Third ACM Symposium on Solid Modeling and its Applications*, ACM Press, 1995, pp. 263–269
- [13] F. C. Langbein, B. I. Mills, A. D. Marshall, R. R. Martin, “Recognizing Geometric Patterns for Beautification of Reconstructed Solid Models”, *Proc. Int. Conf. on Solid Modelling and Applications*, 10–19, IEEE Computer Society Press, 2001. ISBN 0-7695-0853-7
- [14] G. Lukács, A. D. Marshall and R. R. Martin, “Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation”, In: *Proc. ECCV 98 (Vol.1)*, Lecture Notes in Computer Science, Eds: H. Burkhardt and B. Neumann, Springer-Verlag, 1998, pp. 671–686
- [15] B. I. Mills, F. C. Langbein, A. D. Marshall and R. R. Martin, “Estimate of frequencies of geometric regularities for use in reverse engineering of simple mechanical components”, Submitted to *International Journal of Shape Modeling*, 2001
- [16] J. Porrill, “Optimal combination and constraints for geometrical sensor data”, *International Journal of Robotics Research*, Vol 7, No 6, 1988, pp. 66–78
- [17] C. Robertson, R. B. Fisher, D. Corne, N. Werghi and A. P. Ashbrook, “Investigating evolutionary optimisation of constrained functions to capture shape descriptions from range data”, In: *Advances in Soft Computing—Engineering Design and Manufacturing*, Eds: R. Roy, T. Furuhashi and P. K. Chawdhry, Springer-Verlag, 1999, pp. 455–466
- [18] C. Robertson, R. B. Fisher, N. Werghi and A. P. Ashbrook, “Fitting of constrained feature models to poor 3D data”, In: *Proc. Adaptive Computing in Design and Manufacture (ACDM 2000)*, 2000, pp. 149-160
- [19] V. Pratt, “Direct Least-squares Fitting of Algebraic Surfaces”, *Computer Graphics (SIGGRAPH 87)*, Vol 21, No 4, 1987, pp. 145–152
- [20] D. Terzopoulos and K. Fleischer, “Deformable models”, *The Visual Computer*, Vol 4, No 6, 1988, pp. 306–331
- [21] B. Triggs, “Optimal estimation of matching constraints”, In: *3D Structure from Multiple Images of Large-scale Environments (SMILE 98)*, Eds: R. Koch and L. Van Gool, Lecture Notes in Computer Science, Springer-Verlag, 1998

- [22] T. Várady, P. Benkó and G. Kós, “Reverse engineering regular objects: simple segmentation and surface fitting procedures”, *Int. Journal of Shape Modeling*, Vol 4 (1998), pp. 127–141
- [23] T. Várady, R. R. Martin and J. Cox, “Reverse engineering of geometric models — an introduction”, *Computer-Aided Design*, Vol 29, No 4, 1997, pp. 255–268
- [24] V. Weiß, G. Renner and T. Várady, “Reconstruction of swept free-form features from measured data points”, In: *Proc. 32nd CIRP International Seminar on Manufacturing Systems*, 1999, pp. 33–41
- [25] W. Welch and A. Witkin, “Variational surface modeling”, *Computer Graphics (SIGGRAPH 92)*, Vol 26, 1992
- [26] N. Werghi, R. B. Fisher, C. Robertson and A. P. Ashbrook, “Modelling objects having quadric surfaces incorporating geometric constraints”, In: *Proc. 5th European Conference on Computer Vision*, Vol. II, 1998, pp. 185–201
- [27] N. Werghi, R. B. Fisher, C. Robertson and A. Ashbrook, “Object reconstruction by incorporating geometric constraints in reverse engineering”, *Computer-Aided Design*, Vol 31, 1999, pp. 363–399
- [28] N. Werghi, R. B. Fisher, C. Robertson and A. Ashbrook, “Improvement of quadric surface estimation by global fitting”, *International Journal of Shape Modelling*, Vol 6, No 1, 2000, pp. 65–78