Object reconstruction by incorporating geometric constraints in reverse engineering

Naoufel Werghi, Robert Fisher, Craig Robertson and Anthony Ashbrook
University of Edinburgh
Division of Informatics
5 Forrest Hill, Edinburgh EH1 2QL, UK
Tel: 44 131 650 4504
FAX: 44 131 650 6899
Email: {naoufelw, rbf, craigr, anthonya}@dai.ed.ac.uk

Abstract

This paper deals with the constrained reconstruction of 3D geometric models of objects from range data. It describes a new technique of global shape improvement based upon feature positions and geometric constraints. It suggests a general incremental framework whereby constraints can be added and integrated in the model reconstruction process, resulting in an optimal trade-off between minimization of the shape fitting error and the constraint tolerances. After defining sets of constraints for planar and special case quadric surface classes based on feature coincidence, position and shape, the paper shows through application on synthetic model that our scheme is well behaved. The approach is then validated through experiments on different real parts. This work is the first to give such a large framework for the integration of geometric relationships in object modelling. The technique is expected to have a great impact in reverse engineering applications and manufactured object modelling where the majority of parts are designed with intended feature relationships.

Keywords

Geometric constraints, constrained shape reconstruction, shape optimization.

INTRODUCTION AND RELATED WORK

The use of constraints in object modelling is an important topic in the CAD literature. In this area engineering concepts and shape constraints are transformed into shape models through mechanisms of checking, incorporating and solving constraints in the modelling process. Constraints in this area include specification of the geometric relationships between object features as well as engineering constraints (dimensions, material strength and machining parameters) [2, 17].

Finding geometric configurations that satisfy the constraints is the crucial issue and much research has been dedicated to different mechanisms for constraint solving. There are two main strategies for solving constraint problems according to the classification mentioned in [33]. The first strategy, referred as the instance solver, uses specific values of the constraints and looks for geometric configurations satisfying these constraints. In the second strategy, the generic solver investigates first whether the geometric elements could be placed given the constraints independently of their values. After checking that the problem is well-constrained the specific placements of the geometric elements are then determined. In CAD literature, these two strategies have been implemented through different approaches.

The numerical approaches given in [9, 32, 42, 58] are typical instance solvers. Constraints are translated into a set of algebraic equations and are usually solved simultaneously by means of iterative techniques, for instance the Newton-Raphson algorithm. This approach can deal with general cases, over-constrained systems and inconsistent constraint problems. A good initial value is required for such solvers and the algorithm should be applied with care since it may face an ill-conditioned problem.

Symbolic methods [1, 13, 39, 60] are hybrid methods in the sense that they can involve both the generic solver strategy and instance solver strategy. These methods also transform the geometric constraints into algebraic equations but instead of numerical techniques, general symbolic methods are first used to put the set of equations into a new form which is easy to solve. The set of equations is sequentially reduced by solving the simplest one at each step as far as possible. The final set can be then solved numerically. Compared to numerical approaches, they are not subject to numerical instabilities and can locate all solutions to the constraint equations. However, they tend to be computationally expensive. This often restricts the types of geometric elements and types of constraints allowed to be involved.

A more recent approach solves the constraints through sequential geometric constructions, as most configurations in engineering drawing are solvable by ruler, compass and protractor. These approaches can be roughly divided into two categories: the rule-based [3, 57, 66] and graph-based [10, 16, 24, 25, 33, 34, 35] approaches. In the first category constraints are expressed by rules or predicates. The procedure starts from an initial set of predicates defining the constraints and sequentially derives a new set of predicates by applying logical reasoning techniques, with the predicates converging towards defined positions for all the characteristic features. However since only constructive geometries can be handles by these methods they may not be very efficient for large systems of constraints.

The graph-based approaches handle the problem in a more methodical way. They start by forming a graph representation of the problem. In this graph each node represents a geometric element and the edges linking these nodes indicate the constraints between the associated geometric elements. Each edge is labelled with the constraint's type. In a first phase the graph is analysed and if it is well-constrained a set of sequential construction steps are derived from it. This phase depends only on the type and the number of constraints, so it is considered a generic constraint solver. In the second phase the construction steps are carried out integrating the actual values of the constraints to derive the solution shape.

In the Computer Vision community, constraints are mainly used in model-based recognition and localization of objects or environments more generally. They are used as *a priori* information to reduce the search space between, for example, the model features (already stored and known CAD models) and the extracted features from visual sensor output (grey level image edges, 3D range data, etc.) [5, 7, 8, 23, 27, 43]. Some of the approaches for object recognition in particular [30, 53] use a notion of graph representation close to the one used in the graph-based approaches for constraint solving, where the nodes represent object primitives (e.g. points, lines, etc.) and the arcs present geometric relationships between them (e.g. adjacency, parallelism, perpendicularity, etc.).

Constraints can be defined over the geometric and topological relationships between the the object model features (the *a priori* information) and the extracted features from the input data. These relationships are derived either from the properties of the geometric transformation between the vision sensor frame and the scene frame or the transformation between two vision sensor frames (stereo-vision) or the intrinsic structure of the objects [31].

So we can conclude that when computer vision applications deal with model-based recognition and localization, the definition and the concept of constraints are wider than those considered in CAD applications, although they may share the same terminology.

There is one area where Computer Aided Design and Computer Vision share a similar interpretation of geometric constraints, namely reverse engineering referred to as 3D geometric model reconstruction within the vision community. Reverse engineering is typically concerned with parts and industrial objects, whereas 3D geometric model reconstruction is a larger field which includes built environments. But the two terms point to the same goal, which is the transformation of a real object (in the large sense of the word) to a model and concept. In parts manufacturing reverse engineering deals with measuring an existing object so that a surface or solid model can be deduced in order to take advantage of CAD/CAM technologies. It is also often necessary to produce a copy of a part when no original drawings or documentation are available. In other cases we may want to re-engineer an existing part, when analysis and modifications are required to construct a new improved product. Even though it is possible to turn to a computer-aided design to fashion a new part, it is only after the real object is made and evaluated that we can see if the object fits with real world. For this reason designers rely on real 3D objects (real scale wood or clay models) as starting points. This procedure is particularly important to areas involving aesthetic design e.g automobile industry or generation of custom fits to human surfaces such as helmets, space suits or prostheses.

A review of the main research in the CAD community [20, 51, 54, 67] and the Vision community [11, 22, 40] (for reconstruction from single range images) and [14, 55, 56, 62] (for reconstruction from multiple range images) revealed that the exploitation of geometric constraints has not been fully investigated. This lack was noted in the survey work of Varady *et al*[61].

The first motivation behind considering geometric constraints in this work is that models needed by industry are generally designed with intended feature relationships so this aspect should be exploited rather than ignored. The consideration of these relationships is actually necessary because some attributes of the object would have no sense if the object modelling scheme did not take into account these constraints. For example, take the case when we want to estimate the distance between two parallel planes: if the plane fitting results gave two planes which are not parallel, then the distance measured between them would have no significance. Furthermore exploiting the available known relationships would be useful for reducing the effects of registration errors and mis-calibration, thus improving the accuracy of the estimated part features' parameters and consequently the quality of the modelling.

The second motivation is that generally in the manufacturing process, once the part is produced many improvement are carried manually to optimize the part and make it fit with the real world (e.g. fit with another part, adjust the part to fit particular customer). These improvements could be represented by new constraints on the shape of the part. By integrating these constraints into the CAD process the work piece optimization would be reduced and hence many cycles in the part production process would be saved. In other cases, such improvement could not be achieved by hand due to the complexity of the object or when we want to extend the application of the process to complex environments such as buildings or industrial plants.

From a CAD viewpoint the way with which the constraint problem is handled is close to the numerical constraint solver. However it differs radically from this scope on two levels. First on the level of the components of the problem. In our case we have already a real object whose shape we are trying to reconstruct, hence the object real data is used to constraint the shape. Thus, the solution has to satisfy proximity to measured points as well as the constraints. Second the numerical technique used to find the solution overcomes ill-conditioning problems.

The approach for incorporating geometric relationships in object modelling has to tackle two problems. The first is how to represent the constraints. The second is how to integrate these constraints into the shape fitting process. These two aspects are not entirely independent, the shape fitting technique imposes restrictions on the constraint representation and vice versa. Besides, the approach should handle the maximum satisfaction of the constraints.

A first step in the direction of incorporating constraints for assuring the consistency of the reconstruction was done by Porrill [46]. He linearized a set of nonlinear constraints and combined them with a Kalman filter, as applied to wire frame model construction. Porrill's method takes advantage of the recursive linear estimation of the Kalman filter, but guarantees satisfaction of the constraints only to linearized first order. Additional iterations are needed at each step if more accuracy is required.

This last condition has been taken into account in the work of De Geeter *et al* [15] by defining a "Smoothly Constrained Kalman Filter". The key idea of their approach is to replace a nonlinear constraint by a set of linear constraints applied iteratively and updated by new measurements in order to reduce the linearization error. However, the characteristics of Kalman filtering make these methods essentially adapted for iteratively acquired data and many data samples. Moreover, there was no mechanism for determining how successfully the constraints were satisfied and only lines and planes were considered in both of the above works.

The constraints considered by Bolle *et al* [6] in their approach to 3D object position covered only the shape of the surfaces. They chose a specific representation for the treated features: plane, cylinder and sphere.

Compared to Porrill's and De Geeter's work, our approach avoids the drawbacks of linearization, since the constraints are completely implemented. Moreover, our approach covers a larger category of feature shapes. Regarding the work of Bolle [6], the type of constraints which can be held by our approach go beyond the restricted set of surface shapes and cover also the geometric relationships between object features. The proposed approach has been successfully applied first on polyhedral objects [65]. To our knowledge the work appears the first to give such a large framework for the integration of geometric relationships for object reconstruction in the field of reverse engineering.

Although this work is mainly intended for object modelling, it can also find many other many useful applications, e.g. in object localisation. In registration tasks, the features represented in different views need to be put into a single reference frame. For this purpose the transformation between different views is recovered by matching between the related frames. Since a reference frame is built from object features, e.g. normals of surfaces which are supposed to be orthogonal, the estimation of the surfaces has to satisfy the orthogonality constraints. The proposed paradigm may be extended as well to any constrained built environment application like creating "as built" CAD models of a plant for planning new building work. A current method uses a motorised camera head to create highly detailed panoramic images which are then used to extract CAD models. Since the different captured parts of a plant (pipes, reservoirs, etc.) have many geometric relationships between them, using these constraints in the reconstruction process will help to have a consistent whole model. The same is true as well for modelling different compartments of buildings or cities. The current methods of extracting, matching and estimation of large scale buildings' features from aerial images have reached reasonable level. This make the application of our method for modelling different compartments of buildings or cities possible as well.

The organisation of the rest of paper will be as follows: the next section gives some preliminaries on planes and quadric surfaces and gives the parametrization of such surfaces. The aim is to make clear the relationship between the constraint formulation and the surface representations. We then state the problem and develop the proposed approach. Next we define and classify the different types of constraints. Lastly, we demonstrate the process on several synthetic and real objects to evaluate the accuracy, the convergence, repeatability and consistency of the approach.

PRELIMINARIES

This section gives a brief overview about constraining planes, general quadrics and some particular quadric shapes. A full treatment of these surfaces can be found in [4]. While the material contained here is is largely elementary geometry, we present it in order to make clear how the set of constraints used for each surface type and relationship relate to the parameters of the generic quadric.

The line

A line is defined by the following equations :

$$\frac{x - x_0}{l} = \frac{y - y_0}{m} = \frac{z - z_0}{n} \tag{1}$$

where $\vec{X}_0 = [x_0, y_0, z_0]^T$ is an arbitrary point of the line and the vector $\vec{p} = [l, m, n]^T$ defines the orientation of the line.

The plane

A plane surface can be represented by this equation:

$$f(x, y, z) = n_x x + n_y y + n_z z + d = 0 \tag{2}$$

where $\vec{n} = [n_x, n_y, n_z]^T$ is the unit normal vector to the plane and d is the distance to the origin. A plane can have two different representations $(\vec{n}, d)$ and

$(-\vec{n}, -d)$. This ambiguity is easily removed by orienting the normal toward the outside of the object.

Given N data points the best parameters which satisfy (2) in the least squares sense are those minimizing the criterion:

$$\sum_{i=1}^{N} f(x_i, y_i, z_i)^2 = \vec{p}^T H \vec{p} \tag{3}$$

where $\vec{p} = [n_x, n_y, n_z, d]^T$ is the parameter vector and H is the data matrix defined by

$$H = \sum_{i=1}^{N} \vec{h_i}\vec{h_i}^T, \quad \vec{h_i} = [x_i, y_i, z_i, 1]^T \tag{4}$$

H is symmetric and positive definite.

The quadrics

A general quadric surface is represented by the following quadratic equation:

$$f(x, y, z) = ax^2 + by^2 + cz^2 + 2hxy + 2gxz + 2fyz + 2ux + 2vy + 2wz + d = 0 \tag{5}$$

which can be written :

$$X^T A X + 2X^T B + C = 0 \tag{6}$$

where

$$A = \begin{bmatrix} a & h & g \\ h & b & f \\ g & f & c \end{bmatrix}, \quad B = [u, v, w]^T, \quad C = d; \quad X = [x, y, z]^T \tag{7}$$

The type of the quadric depends on the discriminant of the quadric Δ and the cubic discriminant $\mathcal{D}$:

$$\Delta = \begin{vmatrix} a & h & g & u \\ h & b & f & v \\ g & f & c & w \\ u & v & w & d \end{vmatrix} \quad \mathcal{D} = \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & c \end{vmatrix} \tag{8}$$

and the cofactors of $\mathcal{D}$:

$$\begin{aligned} \mathcal{A} &= bc - f^2, & \mathcal{F} &= gh - af, \\ \mathcal{B} &= ac - g^2, & \mathcal{G} &= hf - bg, \\ \mathcal{C} &= ab - h^2, & \mathcal{H} &= gf - ch, \end{aligned} \tag{9}$$

Similarly to the plane case, the best parameters which satisfy (5) for N data points in the least squares sense are those minimizing the criterion:

$$\sum_{i=1}^{N} f(x_i, y_i, z_i)^2 = \vec{p}^T (\sum_{i=1}^{N} \vec{h_i}\vec{h_i}^T) \vec{p} = \vec{p}^T H \vec{p} \tag{10}$$

where $\vec{p} = [a, b, c, h, g, f, u, v, w, d]^T$ and $h_i^T = [{x_i}^2, {y_i}^2, {z_i}^2, 2x_iy_i, 2x_iz_i, 2y_iz_i, 2x_i, 2y_i, 2z_i, 1]$

The cylinder

The quadric is a cylinder when $\Delta = \mathcal{D} = 0$, $u\mathcal{A} + v\mathcal{H} + w\mathcal{G} = 0$ and $\mathcal{A}+\mathcal{B}+\mathcal{C} > 0$. The equation of the cylinder axis is

$$\frac{x - \frac{uf}{\mathcal{F}}}{1/\mathcal{F}} = \frac{y - \frac{vg}{\mathcal{G}}}{1/\mathcal{G}} = \frac{z - \frac{wh}{\mathcal{H}}}{1/\mathcal{H}} \tag{11}$$

This means that the cylinder axis has the direction vector $[1/\mathcal{F}, 1/\mathcal{G}, 1/\mathcal{H}]^T$ and passes through the point $\vec{X}_o = [\frac{uf}{\mathcal{F}}, \frac{vg}{\mathcal{G}}, \frac{wh}{\mathcal{H}}]^T$. The axis orientation corresponds to the eigenvector of the matrix A related to the null eigenvalue. The two other eigenvalues are positive.

The circular cylinder

For a circular cylinder, we can show that the parameters of the quadric should also satisfy the following conditions:

$$\begin{aligned} agh + f(g^2+h^2) &= 0 & cfg + h(f^2+g^2) &= 0 \\ bhf + g(h^2+f^2) &= 0 & \frac{u}{f} + \frac{v}{g} + \frac{w}{h} &= 0 \end{aligned} \tag{12}$$

The matrix A (see (7)) has two identical eigenvalues λ and the radius can be expressed by

$$r^2 = (u^2 f/\mathcal{F} + v^2 g/\mathcal{G} + w^2 h/\mathcal{H} + d)/\lambda \tag{13}$$

A circular cylinder may be also represented by the canonical form:

$$(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 - (n_x(x-x_0) + n_y(y-y_0) + n_z(z-z_o))^2 - r^2 = 0 \tag{14}$$

where $\vec{X}_o = [x_0, y_0, z_0]^T$ is an arbitrary point on the axis, $\vec{n} = [n_x, n_z, n_y]^T$ is a unit vector along the axis and r is the radius of the cylinder.
This form has the advantage of having a minimal number of parameters. However its implementation in the optimization algorithm may cause some complexity, indeed it is not possible with this form to get separate terms for the data and the parameters as in (10) (which allows the data terms to be computed off line). Consequently this may increase the computational cost dramatically.

The expansion of (14) and the identification with (5) yields

$$\begin{aligned} a &= 1 - n_x^2 & h &= -n_x n_y \\ b &= 1 - n_y^2 & g &= -n_x n_z \\ c &= 1 - n_z^2 & f &= -n_y n_z \end{aligned} \tag{15}$$

The cone

A cone surface satisfies $\Delta \neq 0, \mathcal{D} = 0$. The summit of the cone is given by:

$$\vec{X}_o = A^{-1}B \tag{16}$$

The axis of the cone corresponds to the eigenvector related to the negative eigenvalue of the matrix A. The two other eigenvalues are positive.

Circular cone

For a circular cone the parameters of the quadric equation have to satisfy the following conditions

$$\frac{af - gh}{f} = \frac{bg - hf}{g} = \frac{ch - fg}{h} \tag{17}$$

As for the cylinder case, a circular cone equation has a more compact form:

$$[(x-x_o)^2+(y-y_o)^2+(z-z_o)^2]cos^2(\alpha)-[n_x(x-x_o)+n_y(y-y_o)+n_z(z-z_o)]^2 = 0 \tag{18}$$

where $[x_o, y_o, z_o]^T$ is the summit of the cone, $[n_x, n_y, n_z]^T$ is the unit vector defining the orientation of the cone axis and α is the semi-vertical angle. The quadric equation parameters can thus be expressed explicitly as a function of the above terms by :

$$\begin{aligned} a &= n_x^2 - cos^2\alpha & \qquad h &= n_x n_y \\ b &= n_y^2 - cos^2\alpha & \qquad g &= n_x n_z \\ c &= n_z^2 - cos^2\alpha & \qquad f &= n_y n_z \end{aligned} \tag{19}$$

For the same reasons as mentioned in the cylinder case, the compact form of the cone equation is not adequate for the optimization algorithm. Nevertheless it is useful to implicitly impose the conic circularity constraints.

The sphere

A sphere is characterized by equal coefficients for x^2, y^2 and z^2 terms and vanishing coefficients for the cross product terms xy, xz and yz so the parameters h, g and f are all equal to zero. The equation of a sphere can be written as:

$$a(x^2 + y^2 + z^2) + 2ux + 2vy + 2wz + d = 0 \tag{20}$$

The centre of the sphere is:

$$\vec{X}_o = [-u/a, -v/a, -w/a]^T \tag{21}$$

and the radius is

$$r^2 = \frac{u^2 + v^2 + w^2 - ad}{a^2} \tag{22}$$

THE GEOMETRIC CONSTRAINTS

The set of constraints associated with a given object can be divided mainly into two categories. The first one is the surface intrinsic constraints covering the geometric properties which reflect the specific shapes of the surfaces. Examples of these constraints will be given in the next subsection. The second category named the feature extrinsic constraints, defines the geometric and topological relationships between the different object features.

Specific shape constraints

In the text below, when we say that an equation (or set of equations) can be used as a constraint, we mean that the property $f(\vec{p}) = 0$ can be used to define to define a constraint $C(\vec{p})$ on the object parameters $\vec{p}$ by letting

$$C(\vec{p}) = f(\vec{p})$$

Circularity of a cylinder

The circularity of a cylinder can be imposed using either equations (12) or equations (15). These equations have the advantage of imposing implicitly the circularity constraints of the cylinder and avoid the problem when (12) has problem if one of the parameters (f, g, h) vanishes. Besides, they make concrete the geometric relationships between the cylinder and other object features as we will see in Section 6 (the half cylinder).

Circularity of a cone

This property can be expressed using either equations (17) or (19). Similarly to the cylinder case the last equations are more convenient.

Sphere Constraint

To require that an ellipsoidal patch represents a perfect sphere, equation (20) can be used.

Feature extrinsic constraints

These constraints reflect the geometric or topological relationships between the different features of one object. Table 1 summarizes the relationships that we have considered. We notice here that points and lines in this table may be either physical features of the object like cone apexes and edges or implicit features like centres, axes of symmetry. This list is not exhaustive and the classification may not be unique. Nevertheless it covers a large number of constraints in manufactured objects.

	point	line	plane	quadric surface
point	coincident separation	inclusion separation	inclusion separation	inclusion separation
line	-	coincident relative orientation separation	inclusion relative orientation separation	inclusion relative orientation separation
plane	-	-	coincident relative orientation separation	relative orientation separation
quadric surface	-	-	-	coincident relative orientation separation

Table 1: Relationships between features.

Coincidence constraints

It is common that a part contains features which are associated with the same geometric entity (Figure.1.a) or which coincide at the same position (Figure.1.b). In the first case these constraints are implicitly imposed by considering the same parameters for each feature. In the second case the parameters associated to each feature are equated and the resulting equations have then to be satisfied.

Inclusion constraints

A particular feature point may be included in an object feature e.g line, plane or quadric patch. The inclusion constraint requires that the point satisfies the feature's equation.

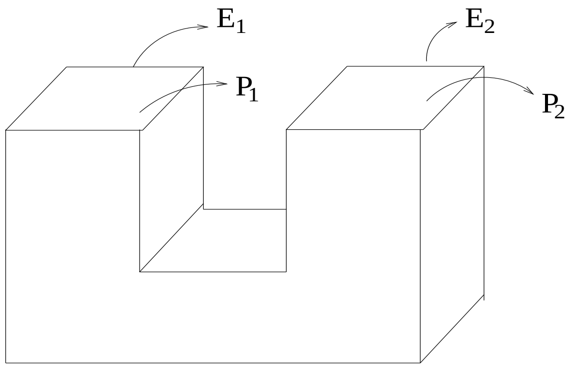
A feature line may be included in a plane or a particular quadric surface. Fig.2 shows an example of this in cylinders. By considering Equations (1) and (2), the condition that a line should lie in a plane is:

$$\begin{cases} n_x l + n_y m + n_z n = 0 \\ n_x x_0 + n_y y_0 + n_z z_0 + d = 0 \end{cases} \tag{23}$$

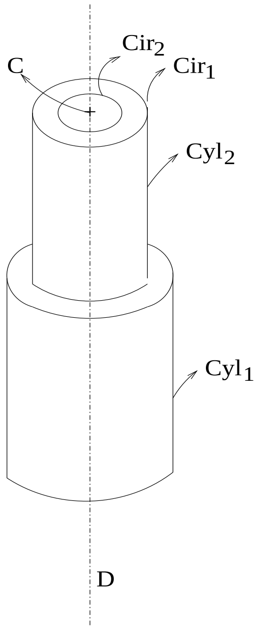
A necessary and sufficient condition that a line be included in a cylinder surface is that the line and the cylinder have the same orientation and an arbitrary point of the line $(X_0, Y_0, Z_0)^T$ satisfies the cylinder equation. Thus, from equations (1), (5) and (11) these conditions can be expressed by

$$\begin{cases} [l, m, n]^T = [1/\mathcal{F}, 1/\mathcal{G}, 1/\mathcal{H}]^T \\ f(X_0, Y_0, Z_0)^T = 0 \end{cases} \tag{24}$$

A line is included in a cone if and only if the orientation vector of the line satisfies the homogeneous equation of the cone (Equation (5) without the



(a)



(b)

Figure 1: (a): The two edges E_1 and E_2 belongs to the same infinite line. The two faces P_1 and P_2 lie in the same infinite plane. (b) The centres of the circles Cir_1 and Cir_2 coincide at the same point C. The cylinders Cyl_1 and Cyl_2 have a common axis.

u, v, w and d terms) and it passes through the cone summit. This is formulated then by

$$\left\{ \begin{array}{l} f_{homogeneous}(\vec{p}) = 0 \\ (X_0, Y_0, Z_0) = \text{cone summit}; \end{array} \right. \tag{25}$$

Relative orientation constraint

There are many orientation relationships which can be deduced and exploited in a given part. In particular, the two common particular cases of parallelism and orthogonality (Fig.3.a). The presence of these two characteristics is easily detected in an object. More generally, given a pair of features (F_i, F_j) whose orientations are defined respectively by two vectors $(\vec{n}_i, \vec{n}_j)$ which make an angle α, the relative orientation constraint is expressed by

$$\vec{n}_i^T \vec{n}_j = cos(\alpha) \tag{26}$$

Relative separation constraint

The relative separation between features can be exploited when the distance between parallel features (Fig.3.b) is already known or needs to be imposed or when the object presents a symmetry aspect leading to some separation

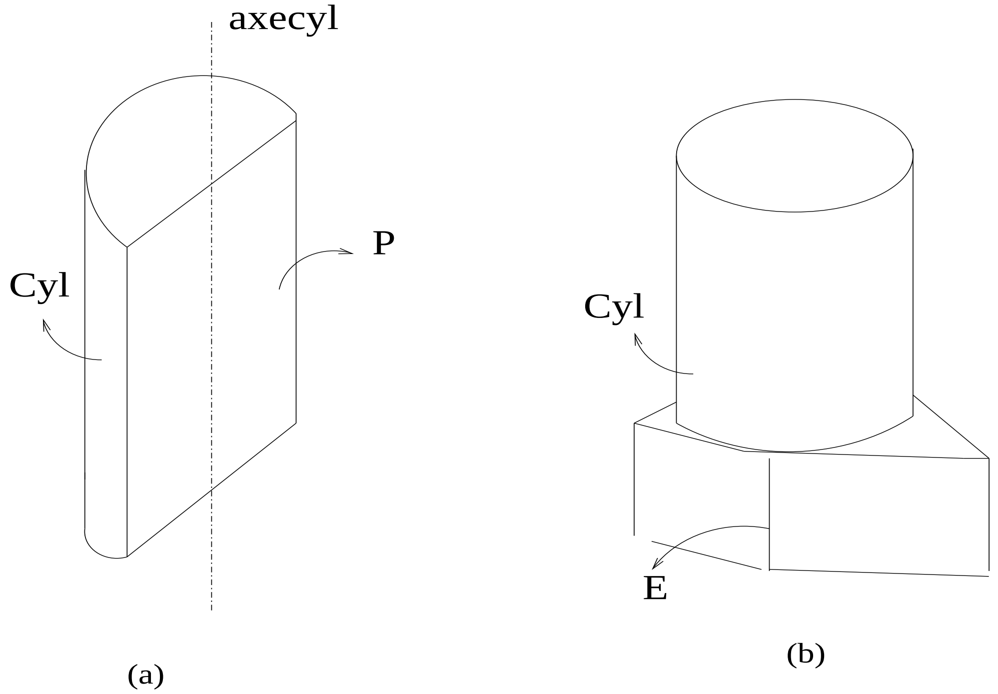


Figure 2: (a): The axis of the cylinder patch Cyl is included in the plane P. (b) The line associated with the edge E is included in the cylinder Cyl.

distance relationships (Fig.3.c). We will take as example the case of planes. Given a pair of parallel planes (P_i, P_j) separated by the algebraic distance d (Fig.3.b), this constraint is expressed by:

$$d_i + d_j = d \tag{27}$$

d_i and d_j are the distance parameters associated respectively to P_i and P_j. The planes are oriented in opposite directions.

Given two pairs of parallel planes (P_i, P_j) and (P_k, P_l) separated by the same distance (Fig.3.c), the constraint is expressed then by:

$$d_i + d_j = d_k + d_l \tag{28}$$

Other constraints

There are also other type of constraints like those imposed directly on the surface parameters as a consequence of the surface representation e.g. the representation of a plane by Equation (2) requires that the sum of the squared elements of the normal be equal to one. Such constraints will be referenced as the unit constraints.

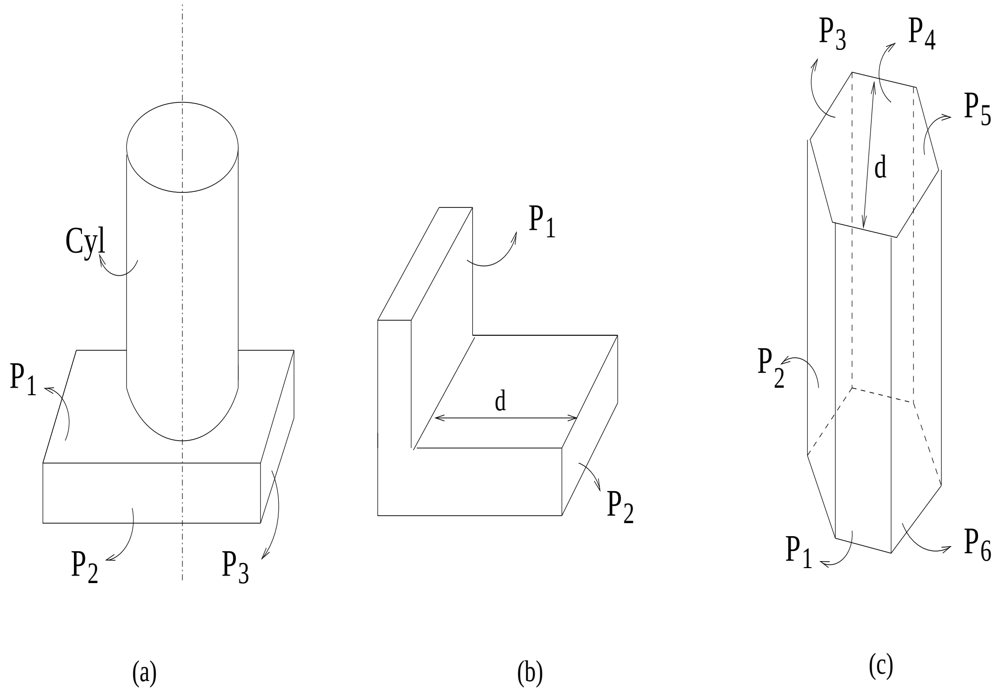


Figure 3: (a): Each pair of planes $(P_1, P2, P3)$ makes an angle of 90^o, the axis of the cylinder Cyl is orthogonal to P_1. (b): The planes (P_1, P_2) are separated by distance d. (c): Each pair of parallel planes of the hexagonal prism is separated by the same distance.

OPTIMIZATION OF SHAPE SATISFYING THE CONSTRAINTS

Given sets of 3D measurement points representing surfaces belonging to a certain object, we want to estimate the different surface parameters, taking into account the geometric relationships between these surfaces and the specific shapes of surfaces as well.

A state vector $\vec{p}$ is associated to the object, which includes all set of parameters related to the different patches. The vector $\vec{p}$ has to best fit the data while satisfying the constraints. Consider $F(\vec{p})$ to be an objective function defining the relationship between the measured data points and the parameters. Such function is generally a minimization criterion (e.g. sum of least squares residuals, maximum likelihood function, etc.).

Consider $C_k(\vec{p})$, $k = 1..M$, the set of constraint functions defining the geometric constraints where $C_k(\vec{p})$ is a vector function associated with constraint k. The problem can be then stated as follows:

$$\begin{array}{cc} \text{minimize} & F(\vec{p}) \\ \text{subject to the constraints} & C_k(\vec{p}) \leq \tau_k, \quad k = 1..M \end{array} \tag{29}$$

Here τ_k represents the tolerance related to the constraint C_k. Ideally the tolerances have zero values, but practically, for geometric constraints they are assigned certain values which reflects the geometric inaccuracies in the relative locations and shapes of features. It is up to the designer to set the tolerances, however an appropriate definition of the tolerances for a given object can be set up by using the scheme developed by Requicha [48].

When faced with an optimization problem it is necessary to know the characteristics of the components of the problem since techniques that solve the problem more efficiently depend mainly on these characteristics. The components of the problem are the objective function and the constraint functions. The characteristics to be investigated are the properties of these functions which include, linearity, smoothness or continuity, differentiability and up to what degree and the form of these functions, quadratic, sum of squared terms, etc.

The computation time of the technique should be taken into account as well. For a reverse engineering task that uses an interactive user environment, designers could not afford to spend hours waiting to get the optimized shape. So a reasonable processing time (in the order of minutes) is a necessary requirement for the optimization technique.

In order to define the appropriate approach let's examine first the components of the problem, the objective function and the constraint functions.

The objective function

Consider $S_1, ., .S_N$ the set of surfaces and $\vec{p_1}, ., .\vec{p_N}$ the set of parameter vectors related to them. Each vector $\vec{p_i}$ has to minimize a given surface fit error criterion J_i associated with the surface S_i. The set of the parameter vectors has then to minimize the following object function:

$$J = J_1 + J_2 + ...J_N \tag{30}$$

By considering a polynomial description of the surfaces, each surface S_i can be represented by:

$$\vec{h_i}^T \vec{p_i} = 0 \tag{31}$$

where $\vec{h_i}$ is the measurement vector with each component of the form $x^\alpha y^\beta z^\gamma$ for some (α, β, γ).

The advantage of this formulation is that it leads to a compact quadric expression of the objective function because of the linearity (with respect to the parameters) of surface equation (31). Indeed, given m_i measurements, the least squares criterion related to this equation is

$$J_i = \sum_{l=1}^{m_i} (\vec{h_i^l}^T \vec{p_i})^2 = \vec{p_i}^T H_i \vec{p_i} \quad H_i = \sum_{l=1}^{m_i} (\vec{h_i^l} \vec{h_i^l}^T) \tag{32}$$

H_i represents the sample covariance matrix of the surface S_i. By concatenating all the vectors $\vec{p_i}^T$ into one vector $\vec{p} = [\vec{p_1}^T, \vec{p_2}^T, ., ., ., \vec{p_N}^T]^T$ equation (30) can be written as a function of the parameter vector $\vec{p}$ and we get the following objective function:

$$F(\vec{p}) = J = \vec{p}^T \mathcal{H} \vec{p}, \qquad \mathcal{H} = \begin{bmatrix} H_1 & (0) & . & (0) \\ (0) & H_2 & . & (0) \\ (0) & . & . & (0) \\ (0) & . & (0) & H_N \end{bmatrix} \tag{33}$$

Under the above form, the objective equation contains separate terms for the data and the parameters. The data matrix $\mathcal{H}$ can be thus computed off-line before the optimization.

The inconvenience of the polynomial representation (31) of the surfaces is that it may over-parametrize the surface. For example a circular cone and circular cylinder have 10 parameters if they are represented by the quadric equation (5) whereas they actually need only 7 parameters (see (14) and (18)). Furthermore, the reduced representation imposes implicitly the circularity constraint consequently there is no need to formulate this constraint within a constraint function. However, the implementation of the reduced form in the optimization algorithm may cause some complexity, indeed because of the nonlinearity of the these forms, it has not been possible to get an objective function with separated terms for the data and the parameters. Thus, the data terms could not be computed off-line. This may increase the computational cost dramatically.

The objective function could be taken as the likelihood of the range data given the parameters (with a negative sign since we want to minimize). The likelihood function has the advantage of accounting for the statistical aspect of the measurements. As a first step, we have chosen the least squares function. The integration of the data noise characteristics in the LS function can be done afterwards with no particular difficulty, leading to the same estimation of the likelihood function in the case of the Gaussian distribution.

The constraint functions

The geometric constraints include some linear constraints (e.g. the relative separation constraint) and mainly non-linear constraints (e.g. relative orientation constraint).

A matrix representation can hold all the types of the constraints mentioned earlier. It leads to a compact form and avoid expressions with many variables. As it will be shown later in the experiment sections a close examination of the non-linear constraints shows that they can be represented by expressions containing cross-product terms of at most 2 parameters. Thus they can represented by the quadratic vector function:

$$\vec{p}^T A \vec{p} + B^T \vec{p} + C \tag{34}$$

where A and B are respectively a square matrix and a vector having the same dimension than the parameter vector $\vec{p}$, C is a scalar. This representation

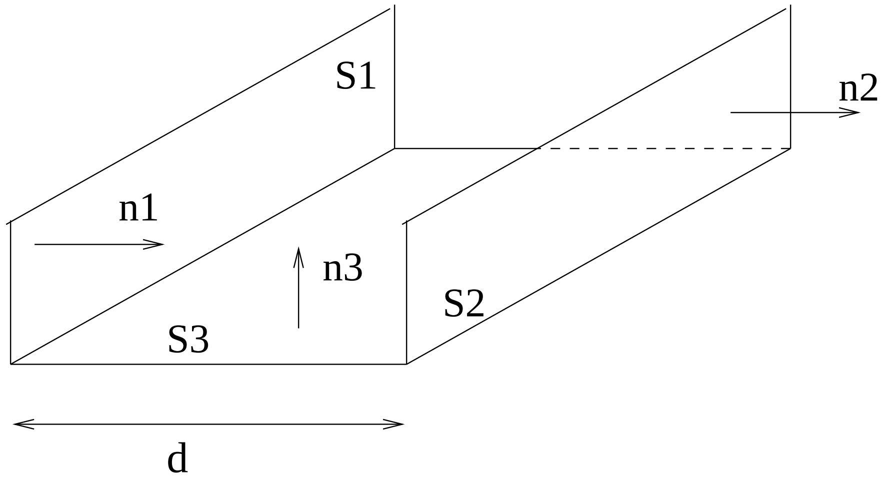


Figure 4: A slot with two parallel planes orthogonal to a third plane.

can also include linear constraints by setting the matrix A to zero. In the next sections the constraint functions will use the matrix and vector notation defined in Appendix.1.

Example

The slot shown in Figure 4 contains three surfaces. The two parallel surfaces (S_1, S_2) have been associated with a single normal vector $\vec{n_1}$ and the surface S_3 is oriented by the normal $\vec{n_3}$. The three surfaces are then defined respectively by $(\vec{n_1}, d_1)$, $(\vec{n_1}, d_2)$ and $(\vec{n_3}, d_3)$. The parameters of the slot can be then encapsulated in the vector $\vec{p} = [\vec{n_1}^T, d_1, d_2, \vec{n_3}^T, d_3]^T$. The fixed distance constraint between the surfaces S_1 and S_2 and the orthogonality constraint between (S_1, S_2) and S_3 are represented respectively by :

$$\begin{aligned} d_2 - d_1 &= d \\ \vec{n_1}^T \vec{n_3} &= 0 \end{aligned}$$

The first constraint is linear and can be put into the form

$$B^T \vec{p} + C = 0, \;\; B = [0, 0, 0, -1, 1, 0, 0, 0, 0]^T, \; A = [0], \; C = -d$$

The second constraint is non-linear and can written under the quadratic form:

$$\vec{p}^T A \vec{p} = 0 \;\; A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \;\; B = [0], \; C = 0$$

The optimization techniques

Optimization techniques fall into two broad branches namely Operation Research techniques and the recent evolutionary techniques.

Evolutionary computation techniques [28, 44] have been having increasing attraction for their potential to solve complex problems. In short they are stochastic optimization methods. They are conveniently presented using the metaphor of natural evolution: they start from a randomly generated set of points or solutions of the search space (population of individuals). Then this set evolves following a process close the natural selection principle. At each stage a new set of population is generated using simulated genetic operations such as mutation or crossover. The probability of survival of the new solutions depends on how well they fit a given evaluation function. The best are kept with high probability and the worst are discarded. This process is repeated until the set of solutions converges to the one best fitting the evaluation function.

The main advantages of the evolutionary techniques is that they do not have much mathematical requirements about the optimization problem. They are 0-order methods, in the sense that they operate only on the objective function and they can handle linear or nonlinear problems, constrained or unconstrained.

The main drawback of these techniques is that they are highly time consuming. This is due to the fact that to ensure convergence, the number of generated solutions has to be high, and at each iteration all the solutions have to be evaluated. This increases the computation time dramatically.

In CAD applications these techniques, and in particular the genetic algorithms have been used in product shape design [59], manufacturing feature extraction [38], description capture from range data [49] and design specification and evaluation [63].

The second branch of the optimization techniques are the classical operation research techniques. They are more mature than the evolutionary techniques. They involve search techniques, numerical analysis and differential tools. Most of these techniques use an iterative scheme. A reasonable initialisation causes significant speedup in convergence. A detailed review and analysis of these optimization techniques could be found in [21, 26]. Descent methods, for instance the Newton-Raphson minimization was used in constraint solving [32, 42] and surface meshing [41]. Quadratic programming and sequential quadratic programming were used for curve and surface optimization [45, 64].

Which technique should be adopted ?

We believe that the evolutionary techniques make them suitable mainly to the optimization cases where objective functions and constraints are very complex, presenting hard-handled aspects such nonlinearity, non-differentiability, or do have not explicit forms. Indeed the earlier mentioned characteristics of these techniques allow them to by-pass these problems.

As our optimization problem does not have these problems, the operational research techniques are more appropriate. This argument is supported by the time-consuming characteristic of the evolutionary techniques, where the average scale of the processing time is on the order of hours. This characteristic makes these methods not appropriate for interactive user environment and impractical for a static verification and checking of the results when experiments have to be repeated many times.

The other important reason for opting for search techniques is that we can obtain a reasonable initial estimate of the model parameters. This initial solution is the estimation of the model parameters without considering the constraints. This estimation is not far away from the optimal one since it is obtained from the real object prototype.

The optimization algorithm

Theoretically a solution of the problem stated in (29) is given by finding the set $(\vec{p}, \lambda_1, \lambda_2, ., ., \lambda_k)$ minimizing the following equation:

$$
\begin{aligned}
E(\vec{p}) &= F(\vec{p}) + \sum_{k=1}^{M} \lambda_k C_k(\vec{p}) \\
F(\vec{p}) &= \vec{p}^T \mathcal{H} \vec{p} \\
C_k(\vec{p} &= \vec{p}^T A_k \vec{p} + B_k^T \vec{P} + C_k
\end{aligned}
\tag{35}
$$

Under the Khun-Tucker conditions [21](Chapter 9), namely that the objective function and the constraint functions are continuously differentiable and the gradients of the constraint functions are linearly independent, the optimal set $(\vec{p}, \lambda_1, \lambda_2, ., ., \lambda_k)$ minimizing (35) is solution of the system:

$$
\frac{\partial F}{\partial \vec{p}} + \sum_{k=1}^{M} \lambda_k \frac{\partial C_k}{\partial \vec{p}} = 0 \tag{36}
$$

In some particular cases it is possible to get a closed form solution for (36) such as the generalized eigenvalues methods. This depends on the characteristics of the constraint functions and whether it is possible to combine them efficiently with the objective function. When the constraints are linear (having the form $A\vec{p} + B = 0$) the standard quadratic programming methods could be applied to solve this system.

However the geometric constraints are mainly non-linear. Generally it is not trivial to develop an analytical solution for such problem. In this case an algorithmic numerical approach could be of great help taking into account the increasing capabilities of computing.

Now if we look to the objective function and the constraint functions in (35) we see that they are explicitly defined in function of the parameters, they are smooth, differentiable and they both have a quadratic structure. From (32) we can notice that each submatrix H_i of $\mathcal{H}$ in (33) is the sum of cross-product terms $\vec{h_i^l}\vec{h_i^l}^T$. Thus H_i as well as H is positive definite. Consequently the objective function is convex. Such functions could be efficiently minimized. Besides it has the important property that its minimum is global. If the constraint functions are also convex, the optimization problem (35) would be a convex optimization problem for $\lambda_k > 0$. For such problem an optimal solution exists, moreover this solution corresponds to the solution of the system (36) defined by the Khun-Tucker conditions [50](section 27,28).

The constraint functions are not necessarily convex since their related matrix A is not necessarily positive definite. However the squared constraint function will have a Hessian matrix which is positive and definite, so is a convex function. The whole optimization function $E(\vec{p})$ in (35) will be then a convex function. So by considering the squared constraint function the problem would be to determine the set $(\vec{p}, \lambda_1, \lambda_2, ., ., \lambda_k)$ minimizing:

$$E(\vec{p}) = F(\vec{p}) + \sum_{k=1}^{M} \lambda_k (C_k(\vec{p})^2),\ \lambda_k > 0 \tag{37}$$

To provide a numerical solution of this problem we have been investigating an approach in the framework of sequential unconstrained minimization. The basic idea is to attach different penalty functions to the objective function $F(\vec{p})$ in such a way that the optimal solutions of successive unconstrained problems approach the optimal solution of the problem (37). Indeed the term $\sum_{k=1}^{M} \lambda_k (C_k(\vec{p})^2)$ could be seen as a penalty function controlling the constraints satisfaction. The scheme is then increment the set of λ_k iteratively, at each step minimize (37) by a standard non-constrained technique, update the solution $\vec{p}$, and repeat the process until the constraints are satisfied. For equal values of λ_k, Fiacco and McCormick [19] have shown that the solutions of (37) converge towards the same solution of the problem (29) when λ_k tends to infinity.

In more detail the proposed algorithm is: We start with a parameter vector $\vec{p}^{[0]}$ that minimizes the least squares objective function and attempt to find a nearby vector $\vec{p}^{[1]}$ that minimizes (37) for small values λ_k, in which the constraints are weakly expressed. Then we iteratively increase the set of λ_k slightly and solve for a new optimal parameter $\vec{p}^{[n+1]}$ using the previous $\vec{p}^{[n]}$. At each iteration n, the algorithm increases each λ_k by a certain amount and a new $\vec{p}^{[n]}$ is found such that the optimization function is minimized by means of the standard Levenberg-Marquardt algorithm (see Appendix 3). The parameter vector $\vec{p}^{[n]}$ is then updated to the new estimate $\vec{p}^{[n+1]}$ which becomes the initial estimate at the next values of λ_k. The algorithm stops when the constraints are satisfied to the desired degree or when the parameter vector remains stable for a certain number of iterations.

A simplified version of the algorithm is illustrated in Figure 5.a in which a single λ is associated to the constraints.

A computational problem associated with this algorithm emerges when λ_k

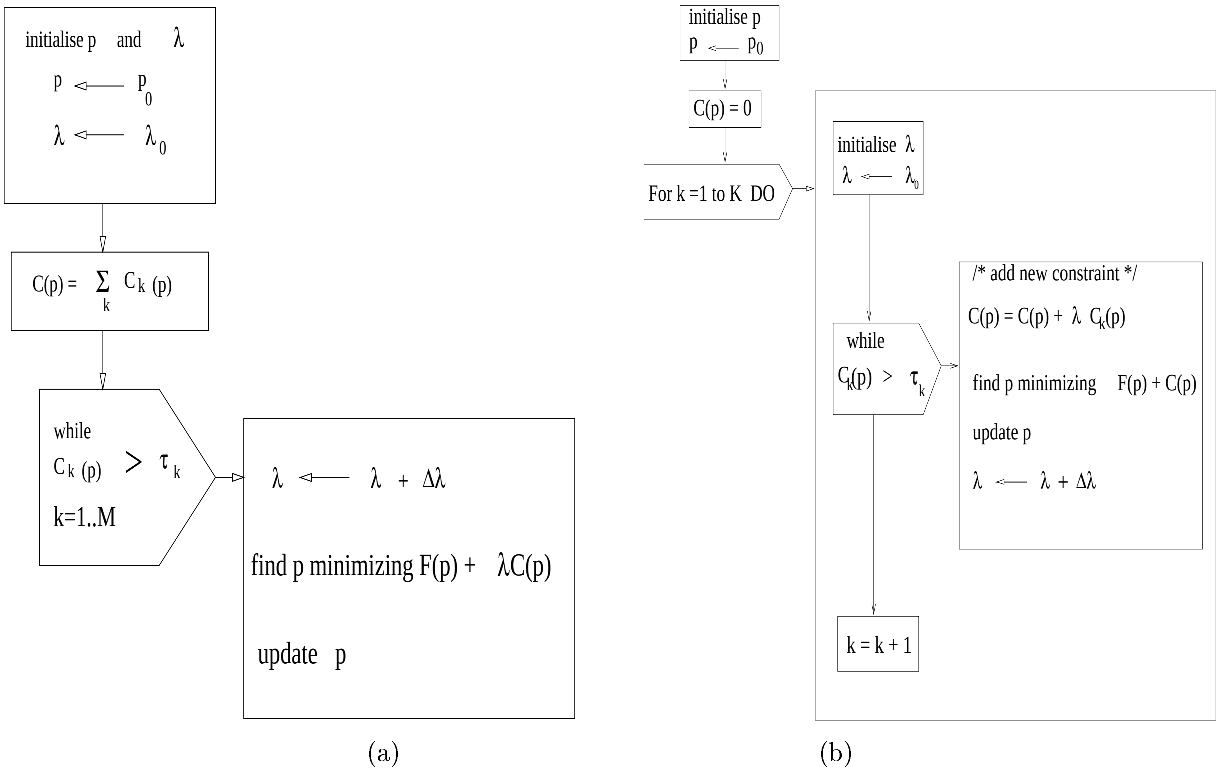


Figure 5: (a): *optim*1 - batch constraint optimization algorithm. (b): *optim*2 - sequential constraint introduction optimization algorithm.

become too large. This problem arises in the Hessian matrix of the optimization function (37) involved in Levenberg-Marquardt algorithm. This matrix become ill-conditioned for high values of λ_k. This aspect could be detected from the expression of this matrix:

$$Hess(E(\vec{p})) = 2\mathcal{H} + \sum_{k=1}^{M} 4\lambda_k C_k(\vec{p}) A_k + R^T D R \tag{38}$$

where

$$\begin{aligned} R^T &= \begin{bmatrix} \frac{\partial C_1}{\partial \vec{p}} & \frac{\partial C_2}{\partial \vec{p}} & \cdots & \frac{\partial C_M}{\partial \vec{p}} \end{bmatrix} \\ D &= \begin{bmatrix} 2\lambda_1 & 0 & \cdots & 0 \\ 0 & 2\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 2\lambda_M \end{bmatrix} \end{aligned}$$

The rank of R is equal to M since we assume that the derivatives of the constraint functions are linearly independent. $R^T D R$ will have also a rank equal to M and since D is a diagonal matrix, the M non-null eigenvalues values of $R^T D R$ will depend of λ_k. More exactly, each eigenvalue has the form $\sigma_k \lambda_k$ where σ_k is some coefficient. Thus the norm of $R^T D R$ will increase as λ_k increases. This is not the case with the other terms of the Hessian matrix (38). Indeed, $\mathcal{H}$ is independent of λ_k and the product $\lambda_k C_k(\vec{p})$ in the other term is expected either to vanish or to remain stable since the constraint value $C_k(\vec{p})$ decreases as λ_k increases. So M eigenvalues of the Hessian matrix (38) will increase with λ_k whereas the $N - M$ others remain independent and not affected. Consequently as λ_k values increase and become large the condition number of the Hessian matrix of the optimization function increases and the matrix become ill-conditioned. Consequently the computation of the inverse of the Hessian matrix in the Levenberg-Marquardt algorithm will suffer from high numerical instability and this approach will no longer be appropriate. Broyden *et al* [12] have developed a method to overcome this numerical problem. Their method is applied with penalty function having equal weight for all the constraints. We have extended the application of this method to different weights of the constraints. The details are developed in Appendix 4.

The initialization of the parameter vector is crucial to guarantee the convergence of the algorithm to the desired solution. For this reason the initial vector was the one which best fitted the set of data in the absence of constraints. This vector can be obtained by estimating each surface's parameter vector separately and then concatenating the vectors into a single one. Naturally, the option of minimizing the objective function $F(\vec{p})$ alone has to be avoided since it leads to the trivial null vector solution. On another hand, the initial values λ_k have to be not too small in order to avoid the above trivial solution and to give the constraints a certain weight. Practically this condition should be applied only to the unit constraints (e.g. the normals of the plane surfaces or quadric axis have to be unit). A convenient value for the initial λ_k is :

$$\lambda_k^0 = \frac{F(\vec{p}^{[0]})}{C_k(\vec{p}^{[0]})} \tag{39}$$

where $\vec{p}^{[0]}$ is the initial parameter estimation obtained by concatenating the unconstrained estimates. This ensures the objective function and the penalty functions have similar values at the first minimization.

Another option of the algorithm consists of adding the constraints incrementally. At each step a new constraint is added to the constraint function $C(\vec{p})$ and then the optimal value of $\vec{p}$ is found according to the scheme shown in Figure 5.b. For each new added constraint $C_k(\vec{p})$, λ_k is initialized at λ_k^0, whereas $\vec{p}$ is kept at its current value.

EXPERIMENTS

The experiments were carried out on both synthetic and real data. The real data was acquired from test objects with a 3D triangulation range sensor. The range measurements were already segmented into point sets associated with surfaces by means of the *rangeseg* [36] program.

The first experiments aimed to check the behaviour and the convergence of the algorithm. These experiments were applied on surfaces extracted from a single view of polyhedral objects. Through these experiments the performances of the batch version of the algorithm (*optim*1) and the sequential version (*optim*2) were compared (see section "The step model object").

In the second series of experiments (second subsection) we have gone further in complexity, firstly on the level of features types. Thus, objects containing quadric features were examined. Secondly the range data was collected and registered from different views. Thus, the data was additionally corrupted by the registration errors. So one objective was to test the robustness of the algorithm toward the complexity of the features (thus the diversity of the constraints) and the registration errors.

At first objects containing single quadric feature were studied. Section "half cylinder" and Section "the cone object" deal with the cylinder and the cone case respectively. Multi-quadric objects were examined afterwards (Sections "Multi-quadric object 1" and "Multi-quadric object 2"). For the first category we have compared results issued from a single view with those obtained from multiple views. For both categories we checked the impact of constraint satisfaction on the quality of object shape attribute estimation.

Other tests were carried out in order to give answers to the following questions:

1. What happens when some features are left unconstrained? What is the impact on the other constrained features and more generally on the object shape estimation? Reciprocally what is the impact of the constrained features on the non-constrained ones?

2. How stable is the algorithm?
3. How optimal is the solution?
4. What happens if some constraints are invalid or inconsistent?

Experiments carried on the synthetic polyhedral object (step model object) will give preliminary answers to question 1. Trials on real multi-quadric objects (Sections "Multi-quadric object 1" and "Multi-quadric object 2") will bring additional confirmation.

Answers to questions 2, 3 4 will be developed in the experiments of Section "multi-quadric object 2".

Application to polyhedral objects

Polyhedral objects involve mainly relative orientation constraints and relative separation constraints. Consider N plane surfaces defining a polyhedral object represented by a parameter vector $\vec{p}$.

Given a pair of planes (P_i, P_j) whose normals $(\vec{n}_i, \vec{n}_j)$ make an angle $\alpha_{i,j}$, the angle constraint (26) is expressed by:

$$C_{angle_{(i,j)}}(\vec{p}) = (\vec{p}^T \mathcal{A}_{i,j}\vec{p} - 2cos(\alpha_{i,j}))^2 = 0 \tag{40}$$

where $\mathcal{A}_{i,j}$ is an $N \times N$ matrix which according to the notation of Appendix.1 is defined by $\mathcal{A}_{i,j} = L_{(r,s,2)}$ where r and s are respectively the indices of the first element of $\vec{n_i}$ and $\vec{n_j}$ in the parameter vector $\vec{p}$.

The separation constraints (27) and (28) are respectively expressed by (see Appendix 1) :

$$C_{dist_{(i,j)}}(\vec{p}) = (\vec{i}^T_{(r,s)}\vec{p} - d)^2 = 0 \tag{41}$$

$$C_{dist_{(i,j,k,l)}}(\vec{p}) = (\vec{i}^T_{(r,s,t,l)}\vec{p})^2 = 0 \tag{42}$$

where r, s, t, l represent the indices of the distance parameters d_i, d_j, d_k, d_l in the parameter vector $\vec{p}$.

Additionally, the unit constraint has to be taken into account since the plane's orientation is defined by a unit normal. For a given surface plane P_i, whose normal is $\vec{n}_i$, this constraint is expressed by:

$$C_{unit_i}(\vec{p}) = (\vec{p}^T \mathcal{U}_i \vec{p} - 1)^2 = 0 \tag{43}$$

where $\mathcal{U}_i$ is an $N \times N$ matrix which, according to the notation of Appendix 1 is defined by $\mathcal{U}_i = U_{(r,r+2)}$, where r is the index of the first element of $\vec{n_i}$ in the parameter vector $\vec{p}$.

The step model object

The first series of tests used a synthetic step model object. This object contains sets of parallel planes. The prototype object is composed of eight faces. We have studied the case when five faces are visible (Fig.6.a). For this view we assigned a single normal to each set of parallel planes. Three normals $\vec{n_1}, \vec{n_2}, \vec{n_5}$ are associated respectively to surfaces (S_1, S_4), (S_2, S_3), and S_5. So, there are three angle constraints (orthogonality of each two normals) and the three unit vector constraints.

The set of visible surfaces is defined by the parameter vector

$$\vec{p} = [\vec{n_1}^T, d_1, d_4, \vec{n_2}^T, d_2, d_3, \vec{n_5}^T, d_5]^T$$

Using the paradigm the Section "Representation of the objective function", the objective function associated with this object is expressed by:

$$F(\vec{p}) = \vec{p}^T \mathcal{H} \vec{p}, \quad \mathcal{H} = \begin{bmatrix} G_{1,4} & (0)_{5,5} & (0)_{5,4} \\ (0)_{5,5} & G_{2,3} & (0)_{5,4} \\ (0)_{4,5} & (0)_{4,5} & G_5 \end{bmatrix} \tag{44}$$

where

$$\begin{aligned} G_{1,4} &= \begin{bmatrix} H_1 + H_4 & h_1 & h_4 \\ h_1^T & N_1 & 0 \\ h_4^T & 0 & N_4 \end{bmatrix}, \quad G_{2,3} = \begin{bmatrix} H_2 + H_3 & h_2 & h_3 \\ h_2^T & N_2 & 0 \\ h_3^T & 0 & N_3 \end{bmatrix} \\ G_5 &= \begin{bmatrix} H_5 & h_5 \\ h_5^T & N_5 \end{bmatrix} \end{aligned} \tag{45}$$

and $H_k = \sum_i^{N_k} (X_i^k)(X_i^k)^T, \quad h_k = \sum_i^{N_k} X_i^k$.
X_i^k is a data point which belongs to the plane surface S_k and N_k is the number of points of the plane S_k.

The normals $\vec{n_1}, \vec{n_2}, \vec{n_5}$ are orthogonal and have to be unit so we set the following constraint functions

$$C_{unit_1}(\vec{p}) = (\vec{p}^T \mathcal{U}_1 \vec{p} - 1)^2 = 0 \tag{46}$$
$$C_{unit_2}(\vec{p}) = (\vec{p}^T \mathcal{U}_2 \vec{p} - 1)^2 = 0 \tag{47}$$
$$C_{unit_5}(\vec{p}) = (\vec{p}^T \mathcal{U}_5 \vec{p} - 1)^2 = 0 \tag{48}$$
$$C_{angle_1}(\vec{p}) = C_{angle_{(1,2)}}(\vec{p}) = (\vec{p}^T \mathcal{A}_{1,2} \vec{p} - 2cos(\pi/2))^2 = 0 \tag{49}$$
$$C_{angle_2}(\vec{p}) = C_{angle_{(1,5)}}(\vec{p}) = (\vec{p}^T \mathcal{A}_{1,5} \vec{p} - 2cos(\pi/2))^2 = 0 \tag{50}$$
$$C_{angle_3}(\vec{p}) = C_{angle_{(2,5)}}(\vec{p}) = (\vec{p}^T \mathcal{A}_{2,5} \vec{p} - 2cos(\pi/2))^2 = 0 \tag{51}$$

Since the unit constraints are used mainly to avoid the null solution, a single λ is associated to them. The optimization function is then

$$\begin{aligned} E(\vec{p}) &= \vec{p}^T \mathcal{H} \vec{p} + \lambda_1 (C_{unit_l} + C_{unit_2} + C_{unit_5})(\vec{p}) \\ &\quad + \lambda_2 C_{angle_1}(\vec{p}) + \lambda_3 C_{angle_2}(\vec{p}) + \lambda_4 C_{angle_3}(\vec{p}) \end{aligned}$$

The results shown below are the average of 100 trials. At each trial the surfaces' points are randomly corrupted with a Gaussian noise of 3 mm standard deviation. Then *optim*1 and *optim*2 are applied to the same set of data points.

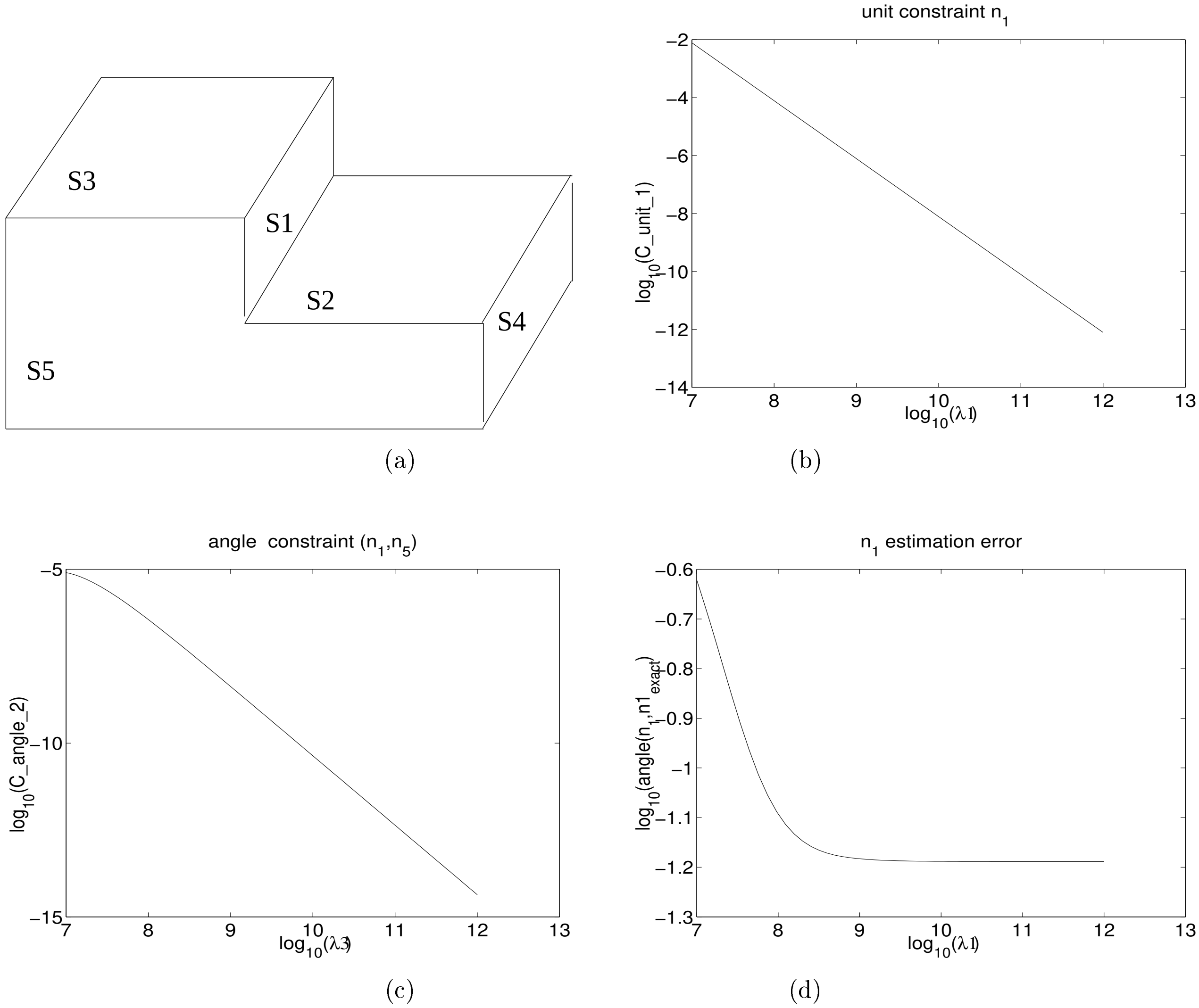


Figure 6: (a): The step model object. (b): variation of the unit constraint error as a function of the related λ. (c): Variation of the angle constraint error ($\vec{n_1}, \vec{n_2}$) as a function of the related λ. (d): error of the estimated orientation of the normal $\vec{n_1}$ (the error is represented by the angle between the estimated normal and the actual one).

Figure 6 shows some results obtained with the algorithm *optim*1. These results represent the variation and the behaviour of some constraint functions during the algorithm with respect of their associated λ. Other results represent the variation of the estimation error of some of the object parameters e.g. one surface's normal. The actual normals are known since the object is simulated.

Figure 6.b shows the decrease of the unit constraint function (46) as λ_1 increases, similarly for the angle constraint function (50) which decreases as the associated weight λ_3 increases in Figure 6.c. We notice that both functions are decreasing nearly linearly at a logarithmic scale. This suggests that it is possible to enforce the constraint to any level of tolerance until the numerical accuracy of the algorithm is compromised. The orientation error related to the surface normal $\vec{n_1}$ and represented here by the angle formed by the actual normal and the estimated one decreases and stabilizes to a relatively low value (around $0.06 degree$) in figure 6.d. This error is computed as follows: at each iteration of the algorithm *optim*1 the estimated normal $\vec{n_1}$ is extracted from the solution $\vec{p}$ and then the error with respect to the actual simulated $\vec{n_1}$ is computed. At each iteration the values of the different λ change, but the orientation error is mapped as function of λ_1 just to show its variation although it is not depending on λ_1 in particular.

Similar behaviour is observed for the other parameter vectors but they are not shown here to save space. This first observation of the constraints behaviour and the parameter estimation is encouraging because it means that the part's shape and position stabilizes as a whole. This fact will be confirmed in next experiments with other objects.

The three figures Fig.7, Fig.8 and Fig.9 illustrate respectively the variation of the angle constraints values C_{angle_1} (49) C_{angle_2} (50) and C_{angle_3} (51) during the application of the sequential version *optim*2. The optimization process has four steps, first the unit constraints are inserted then the three angle constraints are applied one by one. So that at the first step the optimization function is

$$E(\vec{p}) = \vec{p}^T \mathcal{H} \vec{p} + \lambda_1 (C_{unit_l} + C_{unit_2} + C_{unit_5})(\vec{p}) \tag{52}$$

In the second step it will be

$$E(\vec{p}) = \vec{p}^T \mathcal{H} \vec{p} + \lambda_1 (C_{unit_l} + C_{unit_2} + C_{unit_5})(\vec{p}) + +\lambda_2 C_{angle_1}(\vec{p}) \tag{53}$$

and so on.

The figures shows clearly the significant decrease of the constraint value when the related constraint function is added to the optimization function. It is seen also that once the constraint is satisfied the addition of the other constraints only affects the level of tolerance previously reached by a very small degree. In Figure 7, it is noticed that at the end of step 2 (Fig.7.b) the constraint C_{angle_1} is well satisfied although the two others are not yet. Similarly, Fig.8.(c) shows that at the end of step 3 the constraint C_{angle_2} is well satisfied while the constraint C_{angle_3} is not yet implemented. Figure 9 shows that during step 2 and step 3 (when C_{angle_1} and C_{angle_2} are applied)

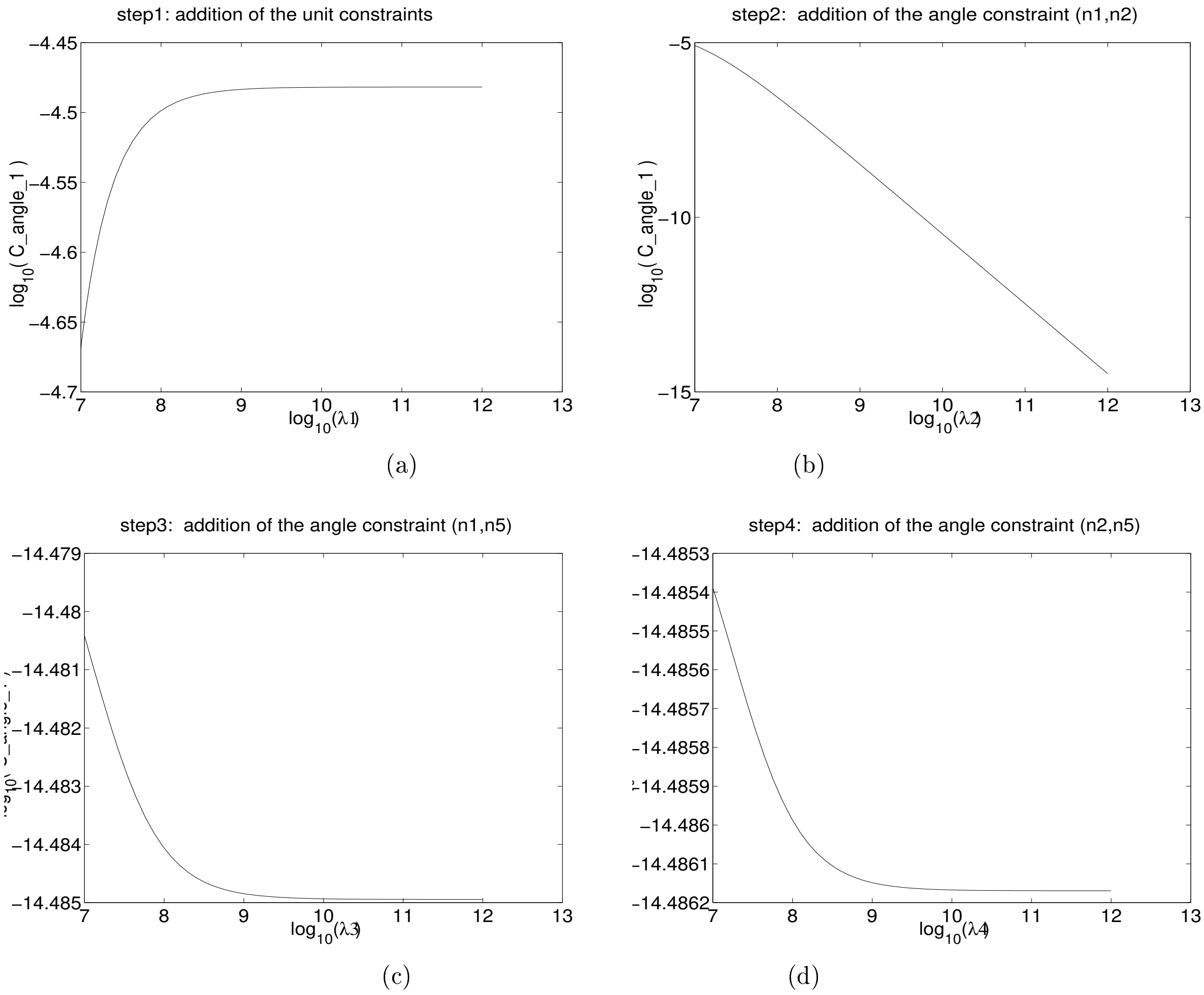


Figure 7: Variation of the angle constraint value C_{angle_1} (50) at the four steps of the algorithm. Step1 (a) : Only the unit constraints are considered in the optimization function (see (52)). Step2 (b) : the angle constraint function C_{angle_1} is added to the optimization function (see (53). Step3 (c): addition of the constraint function C_{angle_2} and Step4 (d): addition of the constraint function C_{angle_3}.

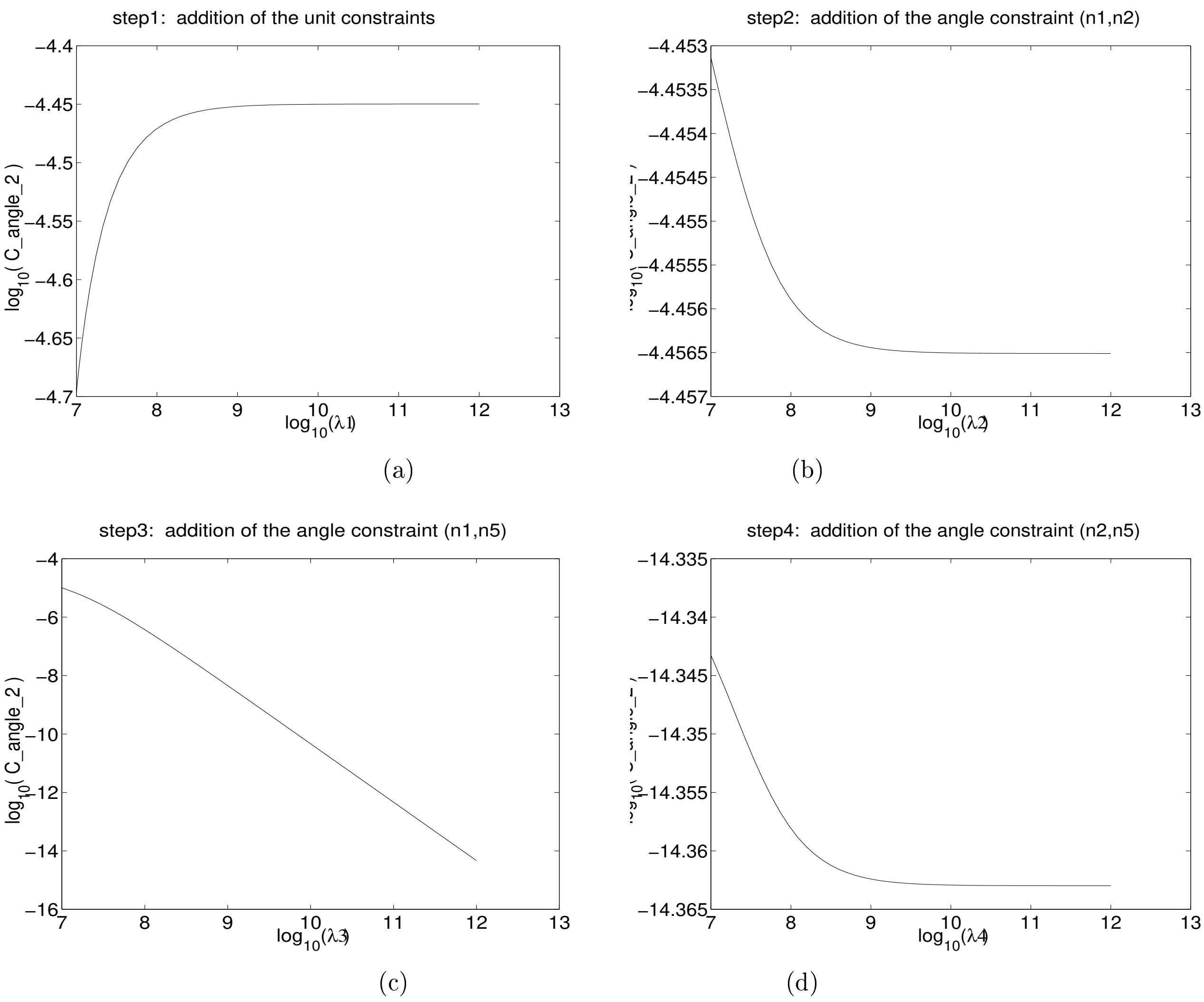


Figure 8: Variation of the angle constraint value C_{angle_2} (50) related to $(\vec{n_1}, \vec{n_5})$ at the four steps of the algorithm *optim*2.

the constraint C_{angle_3} almost keeps stable at a reasonable value. This means that the satisfaction of some constraints is not performed at much cost to the unconstrained features.

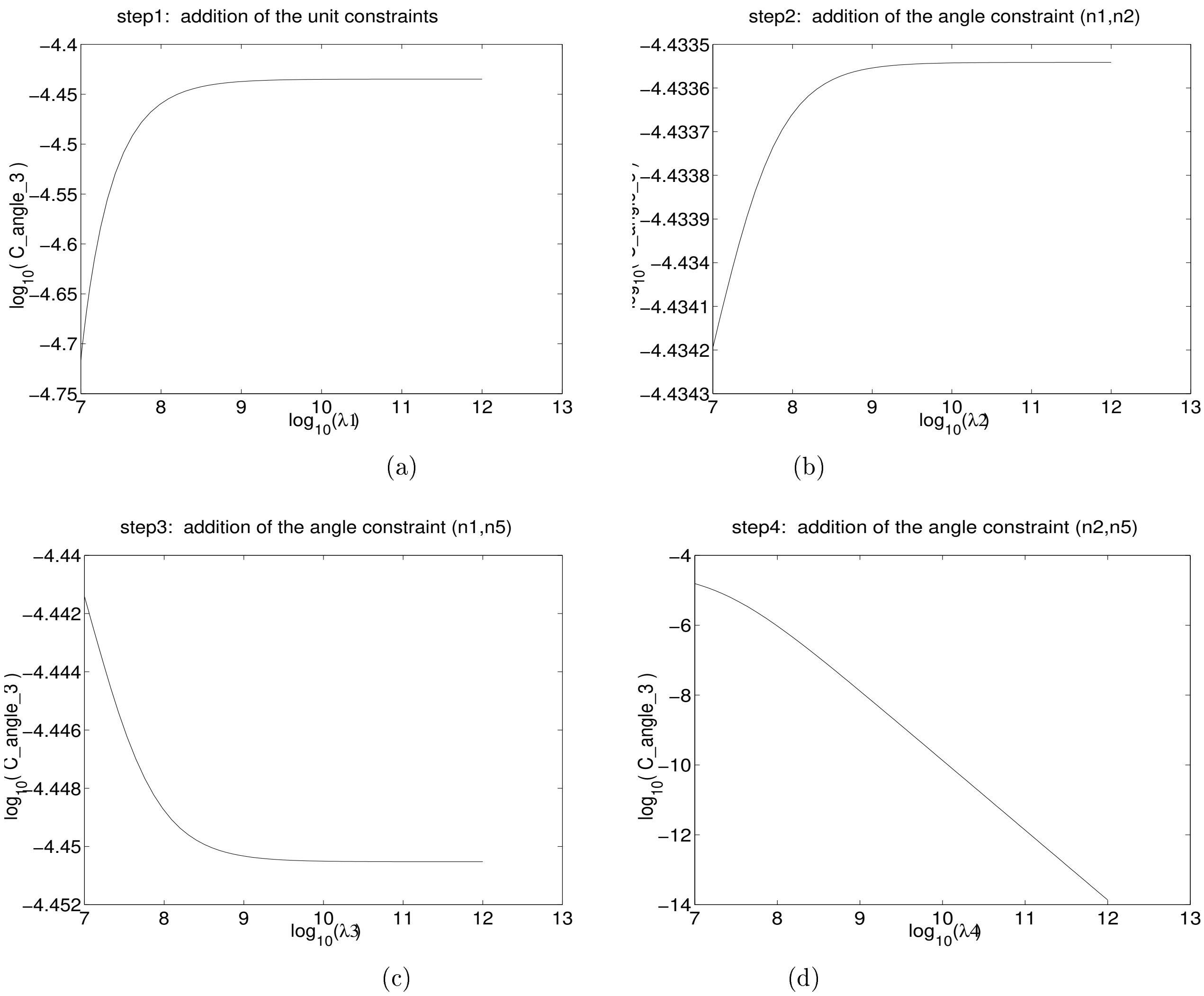


Figure 9: Variation of the angle constraint value C_{angle_3} (51) at the four steps of the algorithm *optim*2.

Figure 10 shows the the variation of the estimation error of one normal $\vec{n_1}$ along the four steps of the algorithm. Similar results are obtained for the other normals. Similar to experiment with *optim*1, figure 10.d shows that at the end of the optimization the error in $\vec{n_1}$ estimation stabilizes at a low value. The same is noticed for the other normals.

So the experiments carried out on the step model object have provided evidence of the applicability of both versions *optim*1 and *optim*2 of the algorithm. Both versions offers high satisfaction of the constraints, moreover the estimated orientation of the object surfaces extracted from the algorithm's solutions are close to the actual one in both versions. This goes towards saying that the satisfaction of object shape requirements is not performed at the expense

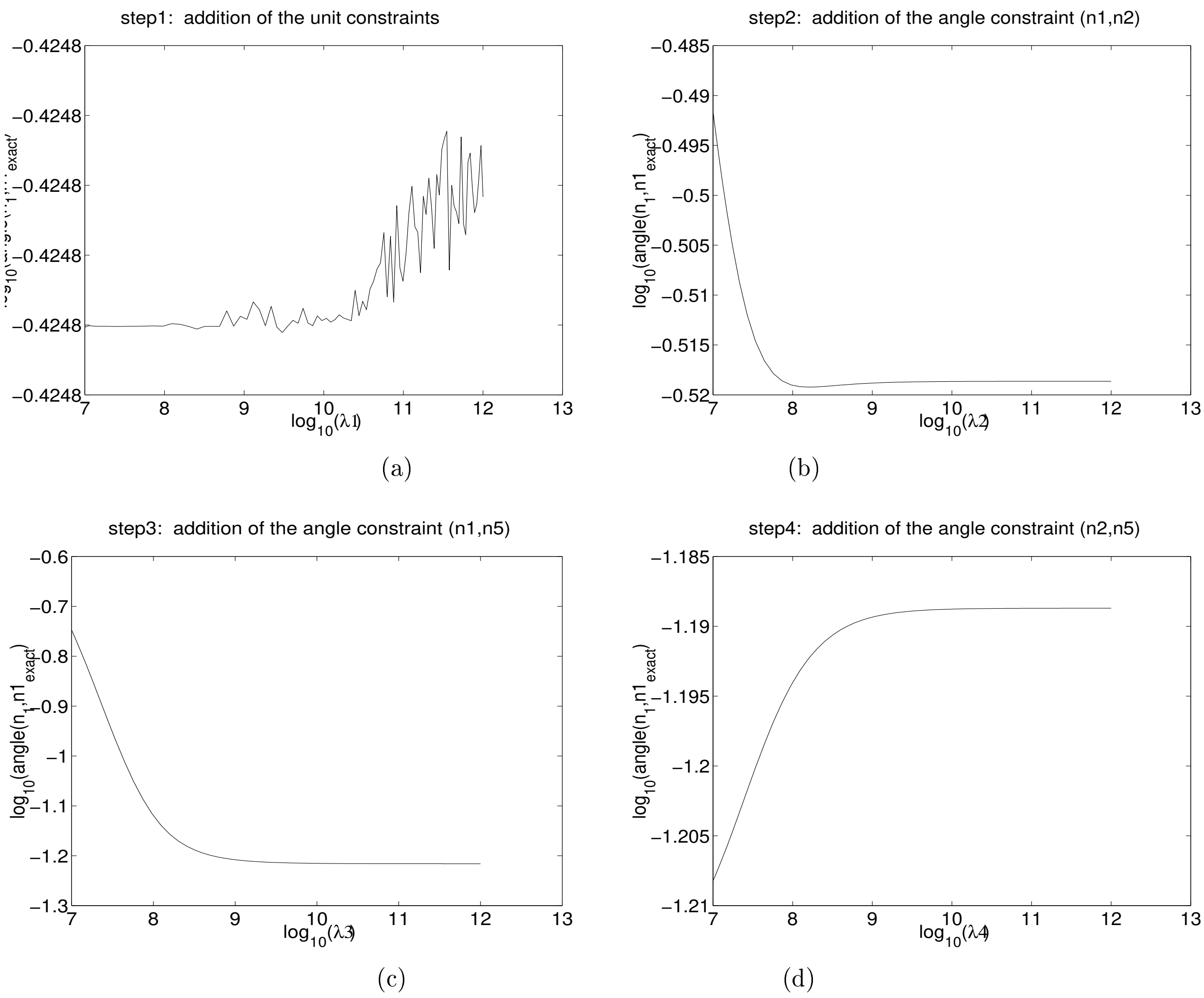


Figure 10: Variation of the orientation error in the estimation of $(\vec{n_1})$ at the four steps of the algorithm *optim*2.

of object localization, although the purpose of the algorithm is not to recover the object localization.

However, *optim*2 is more time-consuming than *optim*1 (around N times, where N is the number of constraints). So, since both estimations of *optim*1 and *optim*2 are acceptable, we have preferred to use *optim*1 for the rest of the work.

The tetrahedron

The second polyhedral object tested is a real tetrahedron. The data has been extracted from a view representing three visible faces S_1, S_2, S_3 (Figure 11). The parameter vector is $\vec{p} = [\vec{p_1}^T, \vec{p_2}^T, \vec{p_3}^T]^T$.

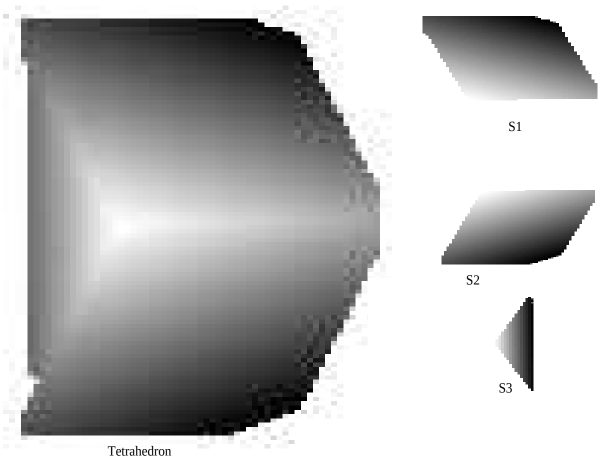


Figure 11: A top view of the tetrahedron and the extracted surfaces.

In this view, the object surfaces have three angle constraints represented by the three angles 90^o , 90^o and 120^o between the three surface normals, as well as the unit vector constraints. So we define the following constraint functions

$$
\begin{aligned}
C_{unit_1}(\vec{p}) &= (\vec{p}^T \mathcal{U}_1 \vec{p} - 1)^2 = 0 && (54)\\
C_{unit_2}(\vec{p}) &= (\vec{p}^T \mathcal{U}_2 \vec{p} - 1)^2 = 0 && (55)\\
C_{unit_3}(\vec{p}) &= (\vec{p}^T \mathcal{U}_3 \vec{p} - 1)^2 = 0 && (56)\\
C_{angle_1}(\vec{p}) = C_{angle_{(1,2)}}(\vec{p}) &= (\vec{p}^T \mathcal{A}_{1,2} \vec{p} - 2cos(2\pi/3))^2 = 0 && (57)\\
C_{angle_2}(\vec{p}) = C_{angle_{(1,3)}}(\vec{p}) &= (\vec{p}^T \mathcal{A}_{1,3} \vec{p} - 2cos(\pi/2))^2 = 0 && (58)\\
C_{angle_3}(\vec{p}) = C_{angle_{(2,3)}}(\vec{p}) &= (\vec{p}^T \mathcal{A}_{2,3} \vec{p} - 2cos(\pi/2))^2 = 0 && (59)
\end{aligned}
$$

The application of the paradigm developed in Section "Representation of the objective function" is straightforward for this object and we get easily the following optimization function:

$$\vec{p}^T \mathcal{H} \vec{p} + \lambda_1 \sum_{l=1}^{3} Unit_l(\vec{p}) + \sum_{l=2}^{4} \lambda_l C_{angle_{l-1}}(\vec{p}) \qquad (60)$$

where

$$\mathcal{H} = \begin{bmatrix} G_1 & (0)_4 & (0)_4 \\ (0)_4 & G_2 & (0)_4 \\ (0)_4 & (0)_4 & G_3 \end{bmatrix}$$

and G_k have the same structure as in equation (45). All the constraints were applied simultaneously according to algorithm *optim*1. The results are the average of 100 trials. At each trial the initial vector $\vec{p}^{[0]}$ is corrupted by a uniform deviation of scale 5%. These 100 trials are a quantitative criterion for testing the stability of the algorithm with respect to the perturbations in the initial value of the solution. Here again all the different constraints values decrease during the optimization. This is illustrated through the two examples shown in Fig.12.(a,b) where the unit constraint C_{unit_1} (54) and the angle constraints C_{angle_1} (57) are mapped in function of their associated weighting values λ_1 and λ_2. Figure 12.c represent the variation of the objective function (the least squares function) $\vec{p}^T\mathcal{H}\vec{p}$ during the optimization process; it increases slightly then it stabilizes. Figure 12.d shows the evolution of the sum of all the constraints during the algorithm application. Since at each iteration of the algorithm *optim*1 the λ_k values increase, the variation of the objective function and the sum of the constraints during the optimization is mapped in function of one of the λ_k, (λ_2). It is seen that the sum of the constraint values converges to zero at the end of the optimization. It is also noticed that the constraint values could be decreased further while the least squares error remains stable. Thus, the final part shape now satisfies the shape constraints at a slight increase in the least squares fitting error.

Application to surfaces having quadric surfaces

Compared to polyhedral objects this category has more complex constraints since the objects contain different types of surfaces and consequently more geometric features. So, besides the constraints related to the plane surfaces other constraints defining properties and relationships between quadric features could be defined as well as relationships between quadric features and plane features. The objects studied in this section contain cylindrical, conical and spherical patches. In this section, the constraints' expressions will use the notation of Appendix 1.

Also, for all objects, the results of the proposed approach have been compared with object estimation methods which do not consider constraints, in particular the least squares technique applied to each surface separately.

The half cylinder

This object is composed of four surfaces. Three patches S_1, S_2 and S_3 have been extracted from two views represented in Figure 13(a,c). These surfaces correspond respectively to the base plane S_2, lateral plane S_1 and the cylindrical surface S_3 (Figure 13.b). The parameter vector is $\vec{p} = [\vec{p_1}^T, \vec{p_2}^T, \vec{p_3}^T]^T$,

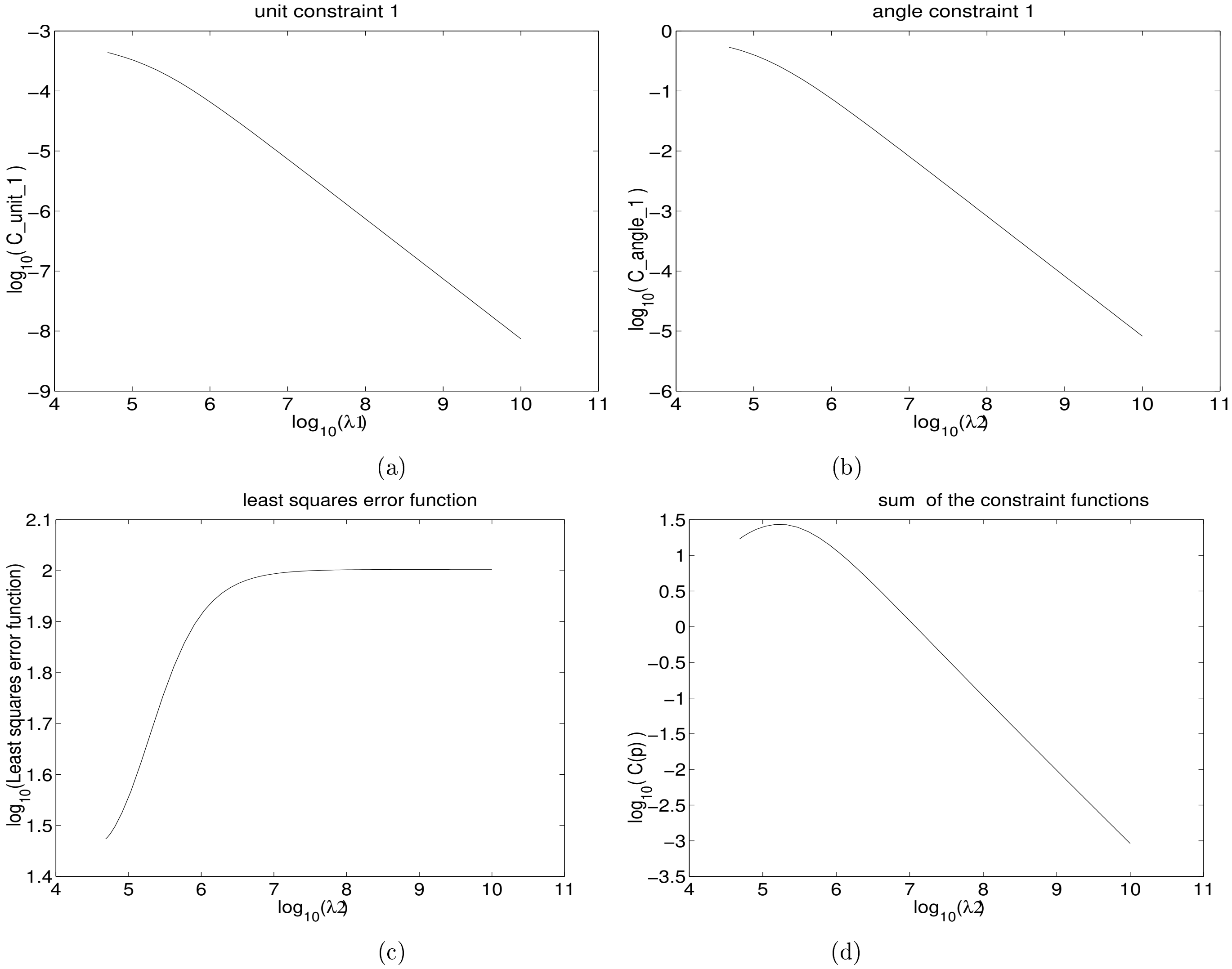
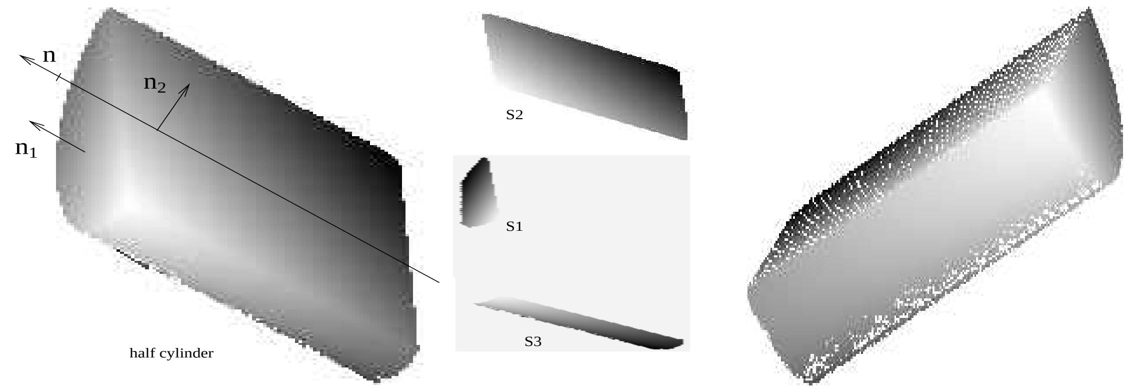


Figure 12: (a): Decrease of the unit constraint function (54) with respect to λ_1. (b): Decrease of the angle constraint function with respect to λ_2. (c) and (d) variation of the objective function $\vec{p}^T\mathcal{H}\vec{p}$ and the sum of all the constraint functions $C(\vec{p}) = \sum_{l=1}^{3} Unit_l(\vec{p}) + \sum_{l=2}^{4} Angle_{l-1}(\vec{p})$ during the optimization. These functions are mapped in function of λ_2 just to show their evolution all along the optimization process but they do not depend specifically on λ_2.



(a) (b) (c)

Figure 13: Two views of the half cylinder and the extracted surfaces.

where $\vec{p_1} = [\vec{n_1}^T, d_1]^T$, $\vec{p_2} = [\vec{n_2}^T, d_2]^T$ and $\vec{p_3} = [a, b, c, h, g, f, u, v, w, d]^T$. The least squares error function is given by:

$$F(\vec{p}) = \vec{p}^T H \vec{p}, \quad H = \begin{bmatrix} H_1 & O_{(4,4)} & O_{(4,10)} \\ O_{(4,4)} & H_2 & O_{(4,10)} \\ O^T_{(4,10)} & O^T_{(4,10)} & H_3 \end{bmatrix} \tag{61}$$

where H_1, H_2, H_3 are the data matrices related respectively to S_1, S_2, S_3.

This object has the following constraints:

1. S_1 and S_2 are perpendicular.
2. The cylinder axis is parallel to S_1's normal.
3. The cylinder axis lies on the surface S_2.
4. The cylinder is circular.

Constraint 1 is expressed by the following condition

$$C_{ang}(\vec{p}) = (\vec{n_1}^T \vec{n_2})^2 = (\vec{p}^T L_{(1,5,2)} \vec{p})^2 = 0$$

Constraint 2 is satisfied by equating the unit vector $\vec{n}$ in (14) to S_1's normal $\vec{n_1}$. Constraints 3 and 4 are represented respectively by:

$$\begin{aligned} C_{axe}(\vec{p}) &= (\vec{i_8}^T \vec{p} - \vec{p}^T L_{(5,15,2)} \vec{p})^2 = 0 \\ C_{circ}(\vec{p}) &= \sum_{k=1}^{6} C_{circ_k}(\vec{p}) = 0 \end{aligned}$$

(see Appendix 2 for details)

Finally the normals $\vec{n}_1$ and $\vec{n}_2$ have to be unit. This is represented by:

$$C_{unit}(\vec{p}) = (\vec{p}^T U_{(1,3)} \vec{p} - 1)^2 + (\vec{p}^T U_{(5,7)} \vec{p} - 1)^2 = 0$$

Thus, the optimisation function is:

$$E(\vec{p}) = \vec{p}^T H \vec{p} + \lambda_1 C_{unit}(\vec{p}) + \lambda_2 C_{ang}(\vec{p}) + \lambda_3 C_{axe}(\vec{p}) + \lambda_4 C_{circ}(\vec{p})$$

Experiments

In the first test, algorithm *optim*1 has been applied to data extracted from a single view (Figure 13.c). In Figure 14 the behaviour of the different constraints during the optimization have been mapped as a function of the associated λ_k as well as the least squares residual (61) and the sum of the constraint functions. The figures show a nearly linear logarithmic decrease of the constraints. It is also noticed that at the end of the optimization all the constraints are highly satisfied. The least squares error converges to a stable value and the constraint function vanishes at the end of the optimization. The figures also show that it is possible to continue the optimization further until a higher tolerance is reached, however this is limited by the numerical accuracy of the machine.

In the second test, registered data from view 1 (Figure 13.a) and view 2 (Figure 13.c) was used. The registration was carried out by hand. Results similar to the first test were obtained for the constraints.

Tables 2 and 3 present the values of some object characteristics obtained from an estimation without considering the constraints and from the presented optimization algorithm. These are shown for the first and second test respectively.

view2	angle($S_1, S2$)(degree)	distance(X_o, S_2)(mm)	radius(mm)
without constraints	90.84	6.32	26.98
with constraints	90.00* [1]	0.00*	29.68
actual values	90	0	30

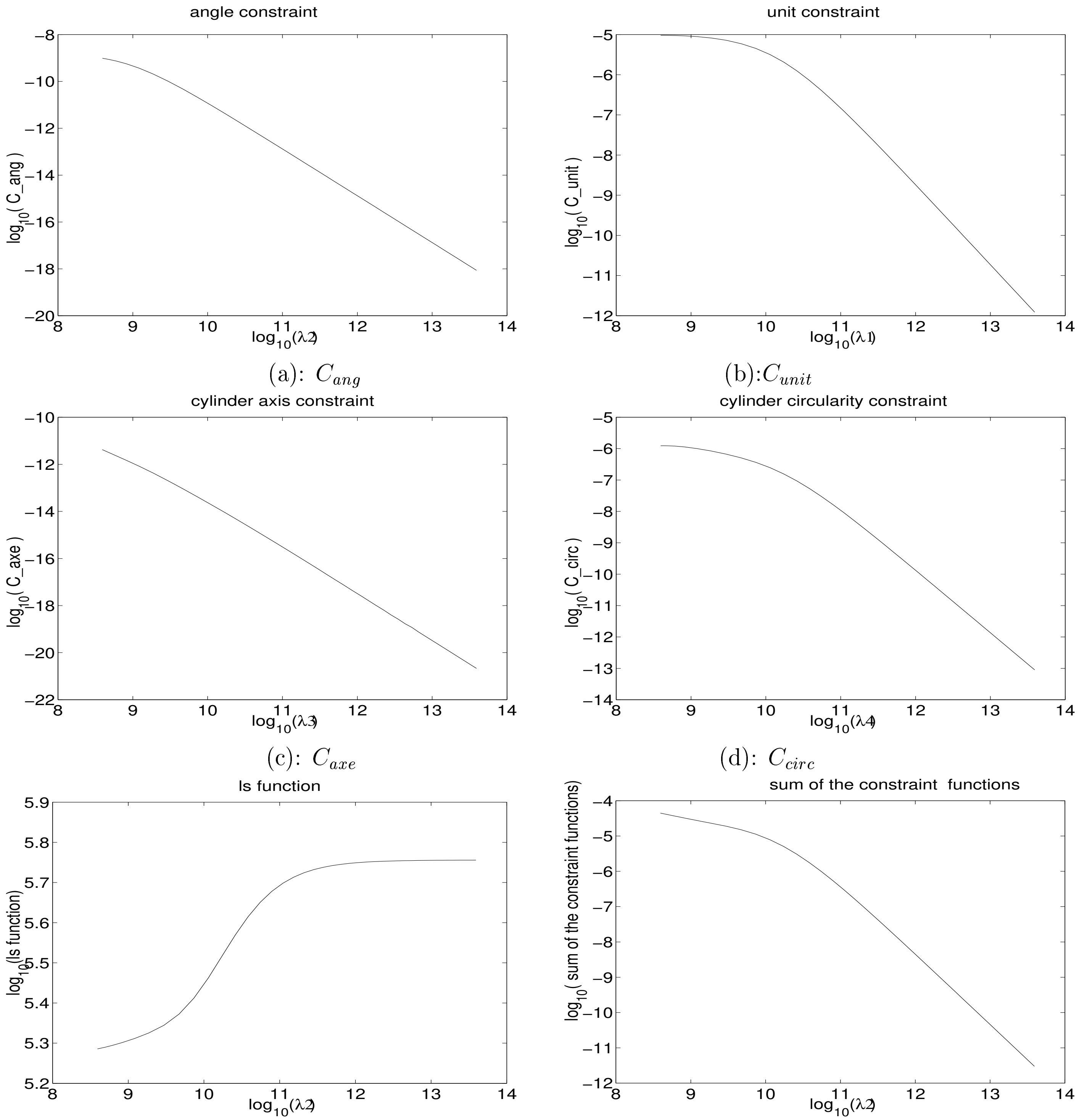
Table 2: Improvement in shape and placement parameters with and without constraints from data from single view of the half cylinder object.

registered view1 and view2	angle($S_1, S2$)(degree)	distance(X_o, S_2)(mm)	radius(mm)
without constraints	89.28	2.23	30.81
with constraints	90.00*	0.00*	30.06
actual values	90	0	30

Table 3: Improvement in shape and placement parameters with and without constraints from data merged from two views of the half cylinder object.

The characteristics examined are the angle between plane S_1 and plane S_2, the distance between the cylinder axis's point X_o (see Section "The cylinder" (14)) and the plane S_2 and the radius of the cylinder. The comparison of the tables' values for the two approaches show the clear improvement made by the proposed technique. This is noticed in particular for the radius for which the actual value is $30mm$, although the extracted surface covers considerably less than a half of a cylinder. As we constrained the angle and distance

[1]* means that the estimated value has been constrained to be the true value



(a): C_{ang} (b): C_{unit}

(c): C_{axe} (d): C_{circ}

(e): LS error (f): $\sum Constraint(\vec{p})$

Figure 14: Shape Optimization of the half cylinder object. (a),(b),(c),(d): decrease of the different constraints with respect to the related λ. (e),(f): variation of least squares function and the constraint function.

relations, we expect these to be satisfied and they are to almost an arbitrarily high tolerance, as seen in Fig.14. The radius was not constrained but the other constraints on the cylinder have allowed the least squares fitting of the unconstrained parameters to achieve a much more accurate estimation of the cylinder radius in both cases.

Multi-quadric objects

The third series of tests have been carried out on more complicated objects with several quadric surface patches. For these objects, all of the surfaces have been considered. The registration of the different views was done manually, thus the registered is expected to be corrupted by an additionally systematic error. By this way we can judge the performances of the algorithm in the presence of such errors.

Multi-quadric object 1

This object (Fig.15) comprises two lateral planes S_1 and S_2, a back plane S_3, a bottom plane S_4, a cylindrical surface S_5 and a conic surface S_6. The cylindrical patch is less than a half cylinder (40% arc), the conic patch occupies a small area of the whole cone (less then 30%). The vector parameter for this

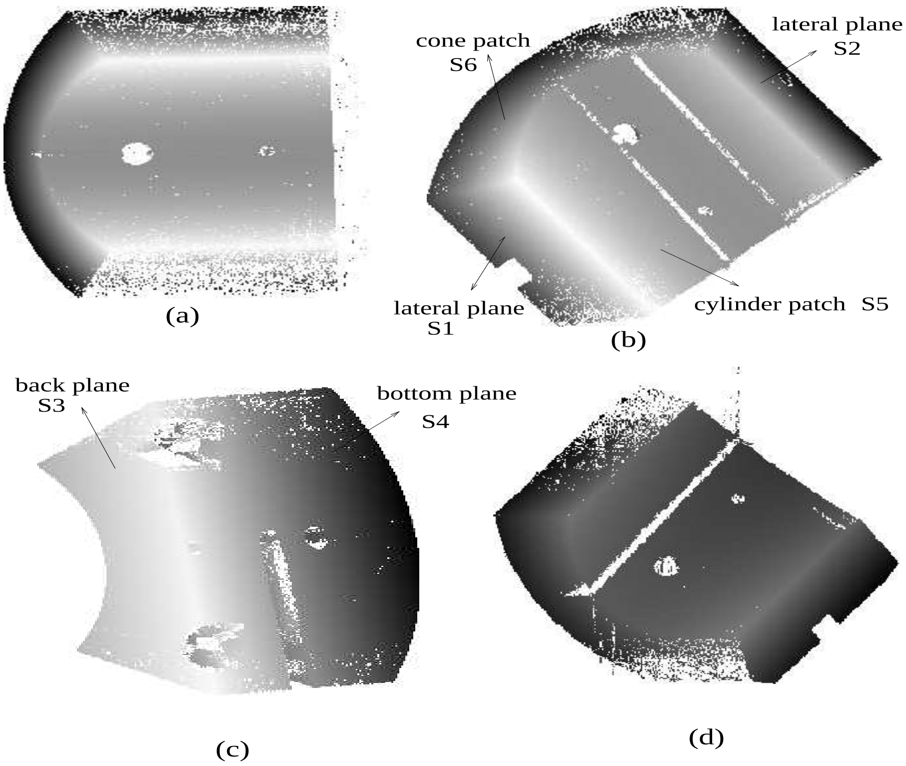


Figure 15: Four views of the multi-quadric object 1.

object is $\vec{p}^T = [\vec{p_1}^T, \vec{p_2}^T, \vec{p_3}^T, \vec{p_4}^T, \vec{p_5}^T, \vec{p_6}^T]$ where $\vec{p_i}$ is the parameter vector associated to the surface S_i.

The surfaces of the object have the following constraints:

1. S_1 makes an angle of 120^{o2} with S_2.
2. S_1 and S_2 are perpendicular to S_3.
3. S_1 and S_2 make an angle of 120^o with S_4.
4. S_3 is perpendicular to S_4.
5. The axis of the cylindrical patch S_5 is parallel to S_3's normal.
6. The axis of the cone patch S_6 is parallel to S_4's normal.
7. The cylindrical patch is circular.
8. The cone patch is circular.

The first four angle constraints are grouped into a single angle constraint function:

$$C_{angl}(\vec{p}) = \sum_{i}^{2} C_{angl_i}(\vec{p})$$

Constraints 5 and 6 are imposed by associating the normals $\vec{n}_3$ and $\vec{n}_4$ respectively to the unit vectors of the cylinder axis and the cone axis (see paragraphs circular cylinder and circular cone in Section "Preliminaries").

The circularity of the cylinder and the cone are expressed respectively by:

$$C_{circ_{cyl}}(\vec{p}) = \sum_{k=1}^{6} C_{circ_{cyl_k}}(\vec{p})$$
$$C_{circ_{cone}}(\vec{p}) = \sum_{k=1}^{6} C_{circ_{cone_k}}(\vec{p})$$

see Appendix 2 for the development of all these constraints.

Finally the unit constraints on the surface normals have to be taken into account. This leads to the following unit constraint function:

$$C_{unit}(\vec{p}) = (\vec{p}^T U_{(1,3)}\vec{p}-1)^2+(\vec{p}^T U_{(5,7)}\vec{p}-1)^2+(\vec{p}^T U_{(9,11)}\vec{p}-1)^2+(\vec{p}^T U_{(13,15)}\vec{p}-1)^2$$

The complete optimisation function is then given by the expression:

$$E(\vec{p}) = \vec{p}^T H\vec{p} + \lambda_1 C_{unit}(\vec{p}) + \lambda_2 C_{ang}(\vec{p}) + \lambda_3 C_{circ_{cyl}}(\vec{p}) + \lambda_4 C_{circ_{cone}}(\vec{p})$$

Experiments

Since the surfaces cannot be recovered from a single view, four views (Fig.15) have been registered by hand. 100 estimations were carried out. At each trial 50% of the surface's points are selected randomly. Thus the algorithm starts with a different initialization each time. The results shown in this section are the average of these estimations. So by examining the mean of the estimations we can have a judgement on the algorithm convergence.

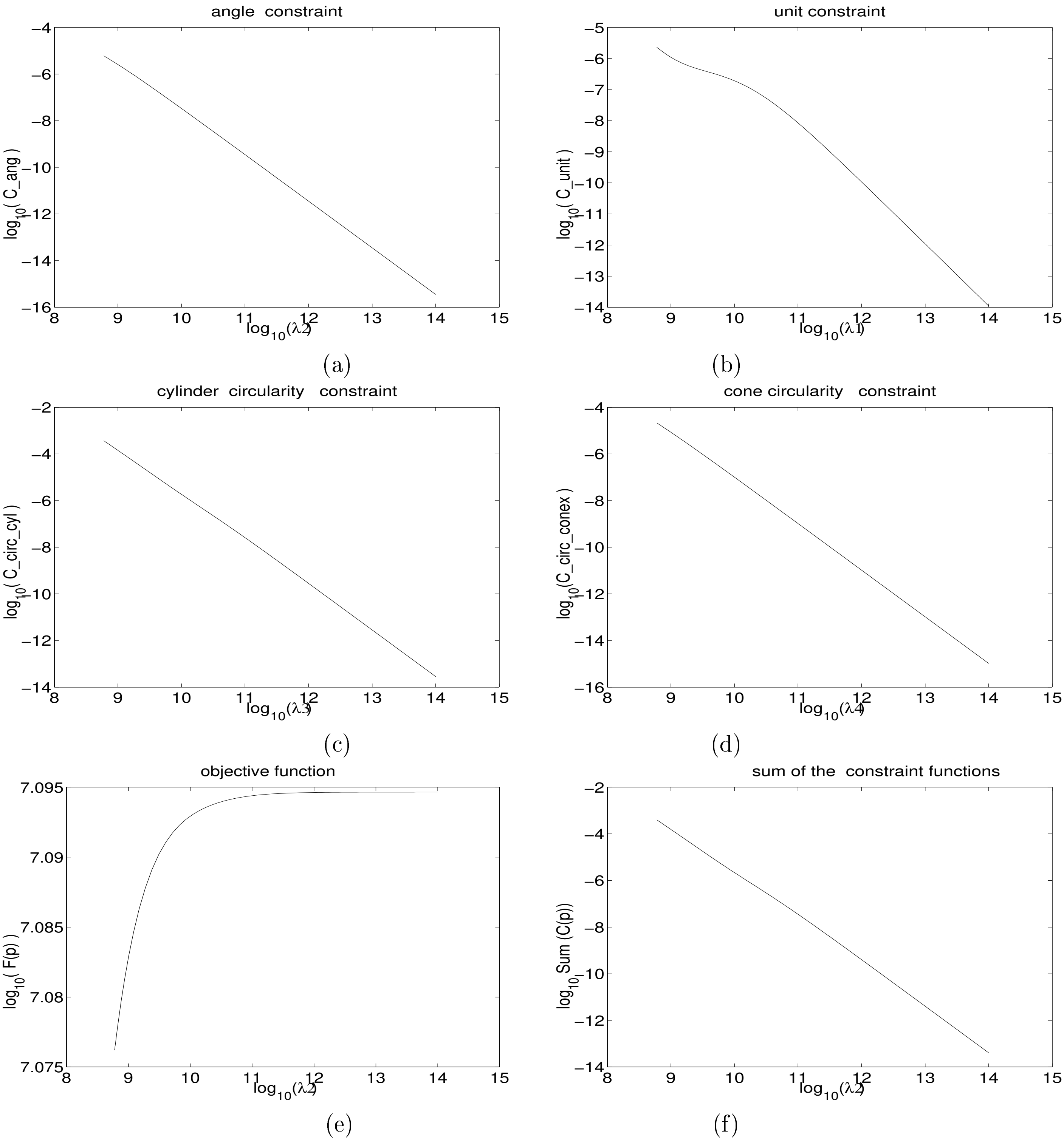


Figure 16: Shape optimization of multi-quadric object 1. a,b,c,d : Decrease of the different constraint functions with respect to the associated λ_k. e,f : Variation of least squares error and the sum of all the constraint values during the optimization.

The results regarding the algorithm convergence are shown in Figure 16. All of the constraint functions vanish and are highly satisfied.

The angles between the different fitted planes are presented in Table 4. It should be noticed that all the angles converge to the actual values. Table 5 and Table 6 contain the estimated values of some attributes of the cylinder and the cone. The values show that each of the axis constraints are perfectly satisfied, the estimated radius and the cone half angle α improve when the constraints are introduced. We notice the good shape improvement, relative to the unconstrained least squares method, given by a reduction of bias of about $22mm$ and 3^0 respectively in the radius and the half angle estimation. The standard deviation of the estimations have been reduced as well.

angle	(S_1, S_2)	(S_1, S_3)	(S_1, S_4)	(S_2, S_3)	(S_2, S_4)	(S_3, S_4)
without constraints	119.76	92.08	121.01	87.45	119.20	90.39
with constraints	120.00*	90.00*	120.00*	90.00*	120.00*	90.00*
actual values	120	90	120	90	120	90

Table 4: The surface's relative angle estimation with and without constraints.

cylinder parameters	angle(axis, S_3's normal)	radius	standard deviation of radius
without constraints	2.34	37.81	0.63
with constraints	0.00*	59.65	0.08
actual values	0	60	0

Table 5: The cylinder characteristic estimates with and without constraints.

cone attributes	angle(axis, S_4's normal)	α	standard deviation of α
without constraints	6.08	26.01	0.30
with constraints	0.00*	31.83	0.13
actual values	0	30	0

Table 6: The cone characteristic estimates with and without constraints.

The radius estimation is within the hoped tolerances, a systematic error of about $0.5mm$ is quite nice. However the cone half angle estimation involves a larger systematic error (about 1.8^o). Two factors may contribute to this. The registration error may be too large since the registration was done by hand and the limited area of the cone patch which covers less then 30 % of the whole cone. It is known that when a quadric patch does not contain enough information concerning the curvature, the estimation is very biased, even when robust techniques are applied, because it is not possible to predict the variation of the surface curvature.

[2] We consider the angle between normals.

Leaving some features unconstrained

We have also investigated whether leaving some of the features unconstrained affects the estimation since one can worry that the satisfaction of the other constraints may push the unconstrained surfaces away from their actual positions. To investigate this, we have left the angles between the pair of planes (S_1, S_2) and (S_1, S_3) unconstrained. The results are shown in Table 7. We see that the estimated unconstrained angles are still close to the actual ones and the accuracy is improved compared to the non-constrained method.

angle	(S_1, S_2)	(S_1, S_3)	(S_1, S_4)	(S_2, S_3)	(S_2, S_4)	(S_3, S_4)
without constraints	119.76	92.08	121.48	87.45	119.20	90.39
with constraints	119.99	90.33	120.00*	90.00*	120.00*	90.00*
actual values	120	90	120	90	120	90

Table 7: Improvement of non-constrained angle estimates.

Multi-quadric object 2

This object (Fig.17) contains six plane surfaces $S_1, S_2, S_3, S_4, S_5, S_6$, a cylindrical surface S_7 and a spherical surface S_8. The surfaces S_1, S_2, S_3, S_4, S_5 form a square prism, the surface S_5 is a square plane surface. The cylindrical patch is a whole cylinder and the spherical patch occupies a half sphere.

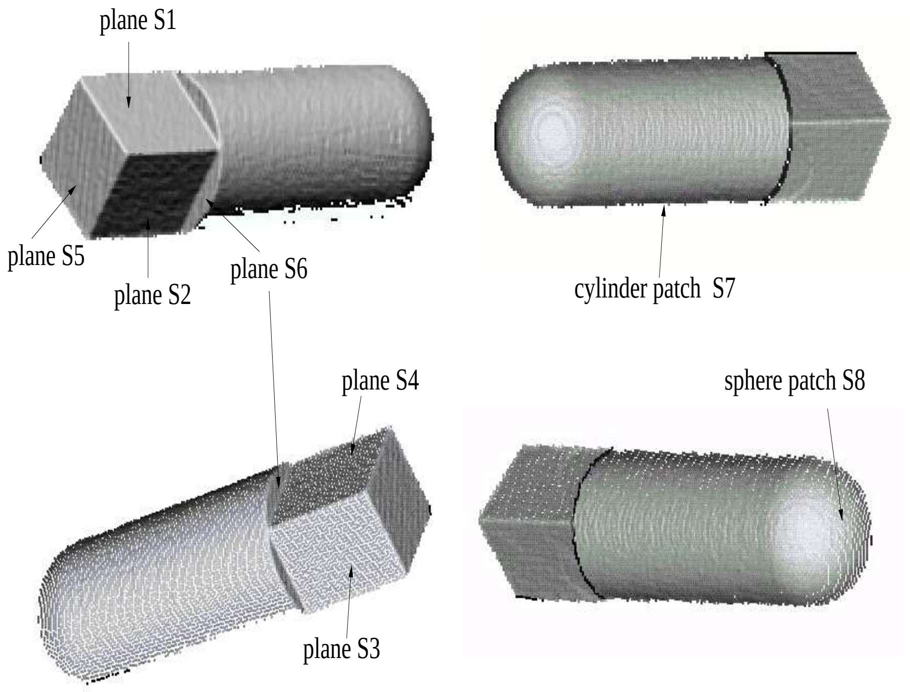


Figure 17: Four views of the multi-quadric object 2.

The surfaces of the object have the following relationships:

1. S_1, S_3 are parallel.
2. S_2, S_4 are parallel.
3. S_5, S_6 are parallel.
4. S_1, S_3 are orthogonal to S_2, S_4.
5. S_5, S_6 are orthogonal to S_1, S_3 and S_2, S_4.
6. S_1, S_3 and S_2, S_4 are separated by the same distance.
7. The cylinder axis is parallel to S_1, S_2, S_3 and S_4 and orthogonal to S_5, S_6.
8. The cylinder axis is located midway between S_1 and S_3 and midway between S_2 and S_4.
9. The cylindrical patch is circular.
10. The sphere centre lies on the cylinder axis.
11. The radius of the cylinder is equal to the radius of sphere.
12. The length diagonal of surface S_5 is equal to the cylinder diameter.

The constraints $1, 2$ and 3 allow a single normal to be associated with each of the pair of planes (S_1, S_3), (S_2, S_4) and (S_5, S_6). Consequently the parameter vector of the object could be defined as:

$$\vec{p} = [\vec{n_1}^T, d_1, d_3, \vec{n_2}^T, d_2, d_4, \vec{n_5}^T, d_5, d_6, \vec{p_7}^T, \vec{p_8}^T]^T$$

where $\vec{n_1}$ is the normal associated to the pair of planes (S_1, S_3), d_1 is the parameter distance of S_1, d_3 is the parameter distance of S_3, $\vec{n_2}$ is the normal associated to the pair of planes (S_2, S_4), d_2 is the parameter distance of S_2, d_4 is the parameter distance of S_4, $\vec{n_5}$ is the normal associated to the pair of planes (S_5, S_6), d_5 is the parameter distance of S_5, d_6 is the parameter distance of S_6, $\vec{p_7}$ is the parameter vector associated to the cylindrical patch S_7 and $\vec{p_8}$ is the parameter vector associated to the spherical patch S_8.

The constraints 4 and 5 are expressed by:

$$C_{angl}(\vec{p}) = \sum_{i=1}^{3} C_{angl_i}(\vec{p})$$

The 6^{th} constraint is formulated by:

$$C_{dist}(\vec{p}) = (\vec{i}^T_{(4,5,9,10)}\vec{p})^2 = 0$$

The 7^{th} constraint is imposed by associating the normal $\vec{n_5}$ to the unit vector of the cylinder axis (see paragraph circular cylinder in Section "Preliminaries").

The constraints $8, 9, 10, 11$ and 12 are expressed respectively by:

$$C_{axe\_pos}(\vec{p}) = (-2\vec{p}^T L_{(1,22,2)}\vec{p} + \vec{i}^T_{4,-5}\vec{p})^2 + (-2\vec{p}^T L_{(6,22,2)}\vec{p} + \vec{i}^T_{9,-10}\vec{p})^2 = 0$$

$$
\begin{aligned}
C_{circ}(\vec{p}) &= \sum_{k=1}^{6} C_{circ_k}(\vec{p}) = 0 \\
C_{sph_{center}}(\vec{p}) &= (\vec{p}^T T_{(11,12,22,23)}\vec{p})^2 + (\vec{p}^T T_{(11,13,22,24)}\vec{p})^2 + (\vec{p}^T T_{(12,13,23,24)}\vec{p})^2 = 0 \\
C_{equ_{radius}}(\vec{p}) &= (\vec{i}^T_{(25,30)}\vec{p} + \vec{p}^T U_{(27,29,22,24)}\vec{p})^2 = 0 \\
C_{median}(\vec{p}) &= (\vec{p}^T (I_{(4,1)} - 2U_{(22,24)})\vec{p} + 2\vec{i}^T_{25}\vec{p})^2 = 0
\end{aligned}
$$

Finally the unit constraints related to the planes' normals and the unit constraint of the coefficient a of the sphere are grouped into the following unit constraint

$$
C_{unit}(\vec{p}) = (\vec{p}^T U_{(1,3)}\vec{p}-1)^2+(\vec{p}^T U_{(6,8)}\vec{p}-1)^2+(\vec{p}^T U_{(11,13)}\vec{p}-1)^2+(\vec{p}^T U_{(26,26)}\vec{p}-1)^2
$$

The optimization function is then:

$$
\begin{aligned}
E(\vec{p}) = \vec{p}^T H\vec{p} &+ \lambda_1 C_{unit}(\vec{p}) + \lambda_2 C_{angl}(\vec{p}) + \lambda_3 C_{dist}(\vec{p}) + \lambda_4 C_{axe\_pos}(\vec{p}) \\
&+ \lambda_5 C_{circ}(\vec{p}) + \lambda_6 C_{sph_{center}}(\vec{p}) + \lambda_7 C_{equ_{radius}}(\vec{p}) + \lambda_8 C_{median}(\vec{p})
\end{aligned}
$$

The details concerning the formulation of all the above constraints are in Appendix 2.

Experiments

The surfaces of the objects were recovered from 4 views shown in Figure 17 and the registration of the range data was done by hand. Similarly to the previous object 100 trials were performed. At each of them 50% of the surfaces's points are selected randomly leading to a different initialisation each trial. In all the trials, the decrease of all the constraint errors and the high level of satisfaction of the constraints at the end of the optimization for a slight increase of the least squares error is essentially similar to that observed in the previous experiments and so similar graphs are not shown here.

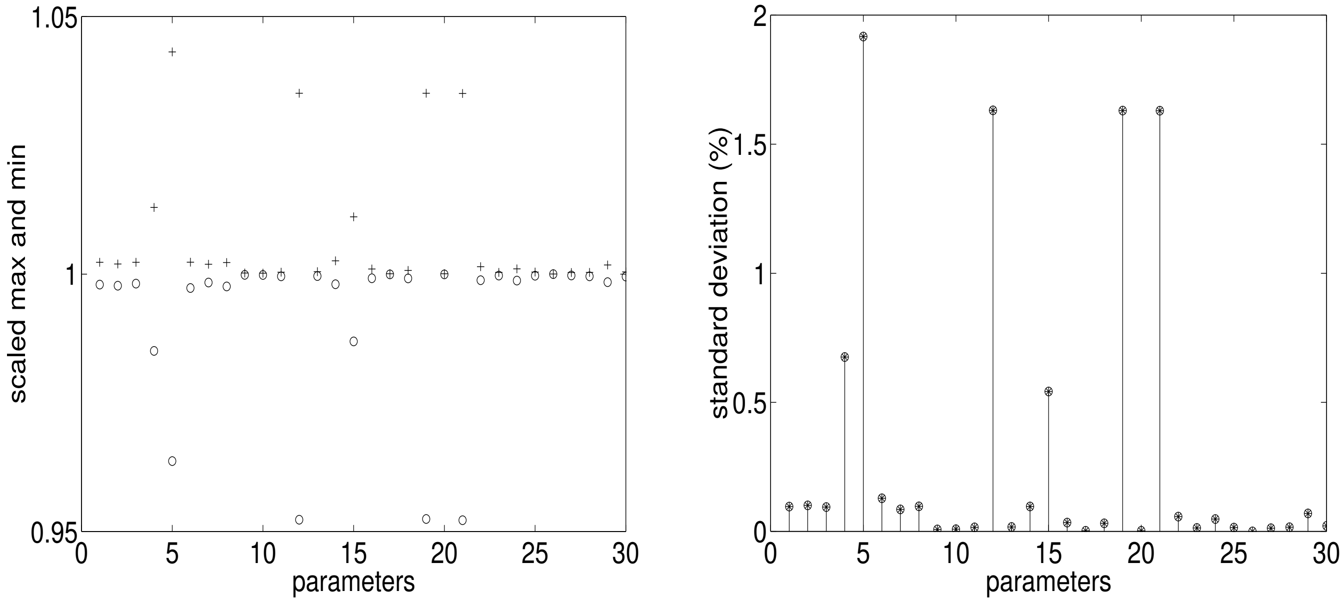
Through these different tests and trials we have been investigating:
1. How stable is the convergence of the algorithm ?
2. How close is the estimation to the actual optimal value ?
3. What are the effects of leaving some features unconstrained ?
4. What is the effect of constraint invalidity ?
5. What is the effect of constraint inconsistency ?

Lastly, some results concerning the global shape improvement of the the object model will be presented.

Stability of the convergence

The previous experiments performed each over 100 trials have shown that the mean of the estimated shapes obtained form these trials converges close to the actual solution which satisfies the constraints. The initial solution in each trial has a different value since the the data points are selected randomly.

This experiment aims to check the sensitivity of the algorithm with respect to the initial value, to test the stability of the convergence of the algorithm with respect to changes in the initial estimation. One way is to do so is to compute the difference between the maximum and the minimum value of each parameter in the set of different solutions. A second way is to examine statically the "closeness" of the different estimates to the mean solution, known in the statistic terminology as the distribution of the solutions. This distribution could be obtained by computing the variance of each parameter from the solutions issued from the 100 trials. Figure 18.a shows the maximum and the minimum value (scaled by the absolute value of the mean) for each parameter. The extrema of the different parameters vary at a very low scale around the mean solution, in a range lower than 2%. The same is noticed in the standard deviations of the parameters illustrated in Figure 18.b. This aspect is further confirmed in the distribution of the least squares errors of the different estimations shown in Figure 19. The related relative variance is 1.94%.



(a) (b)

Figure 18: (a): maximum (+) and minimum (o) value for each parameter scaled by the absolute value of the mean. (b): relative standard deviation of the parameters.

Closeness to the actual optimal solution

By actual optimal solution we mean the estimate obtained from a process where the constraints are implicitly built into the least squares model. The solution provided in this case always completely satisfies the constraints. So one may ask how close is the estimate issued from our approach to this optimal solution. As we have mentioned previously, such an ideal and elegant formulation is difficult or impossible to achieve for many objects due to the complexity

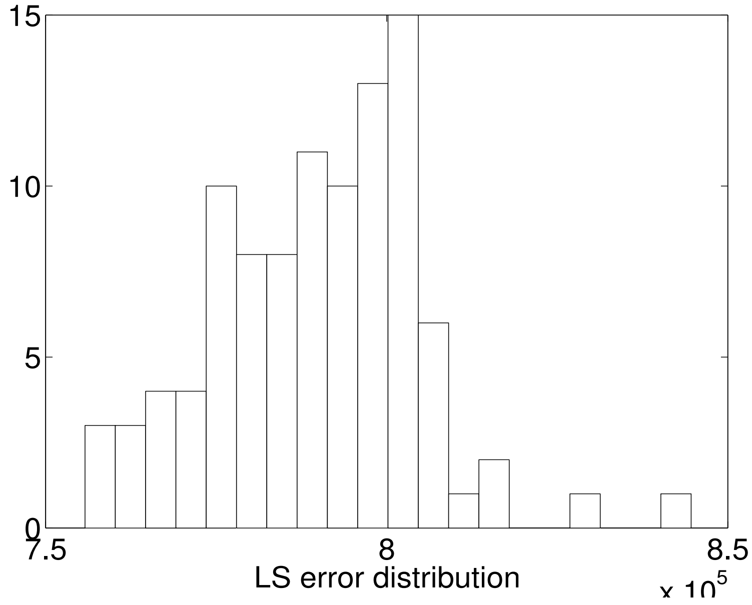


Figure 19: distribution of the least squares errors.

and to the non-linearity of the geometric constraints. In fact one purpose and motivation of our approach is to overcome this problem. Nevertheless it is possible for some simple particular cases to combine the constraints with the least squares error.

So, in order to make a comparison with the optimal solution a sub-part of the multi-quadric object 2 was considered. It is composed of the two parallel planes S_1 and S_3. The objective is to estimate the planes' orientation taking into account the parallel constraints. For the first case, the parallel constraint is implicitly considered by associating one normal to both planes. The optimization function is then:

$$\vec{n}^T H \vec{n} + \lambda(1 - \vec{n}^T \vec{n})$$

where H is the appropriate data matrix. The second term of the function is the unit constraint. A closed form solution is provided by the eigenvalue method.

In the second case each plane was assigned a different normal vector. The equality of the two normals has to be satisfied through the optimization process. According to our approach the objective function is:

$$\vec{n_1}^T H_1 \vec{n_1} + \vec{n_3}^T H_3 \vec{n_3} + \lambda_1(1 - \vec{n_1}^T \vec{n_1})^2 + \lambda_2(1 - \vec{n_3}^T \vec{n_3})^2 + \lambda_3(1 - \vec{n_1}^T \vec{n_3})^2$$

100 tests were applied for each of the two cases. The average of the results are summarized in Table 8. The estimations are similar in the two cases. This shows that both solutions converge to the same value and almost equally

	$\vec{n_i}$	$\vec{n_1}$	$\vec{n_3}$	angle($\vec{n_1}, \vec{n_3}$) (degree)	LS error
1^{st} case	0.5316 0.6733 0.5139	-	-	-	9.07
2^{nd} case	-	0.5316 0.6733 0.5139	0.5316 0.6733 0.5139	0.00	9.06

Table 8: mean estimates of S1 and S3 normal and LS error in the two type of solutions.

minimize the least squares error. The LS of the second solution is slightly lower than the optimal solution one. This is because in the optimal case the constraint is perfectly satisfied so the least squares error has to absorb all the error. The same convergence of the two solutions is further confirmed from the distribution of the angle $(\vec{n}, \vec{n_c})$ where $\vec{n_c}$ is the mean of $\vec{n_1}$ and $\vec{n_3}$ and the distribution of the difference between the LS error related to each of them, LS and LS_i (Fig.20). These dis tributions are issued from 100 trials.

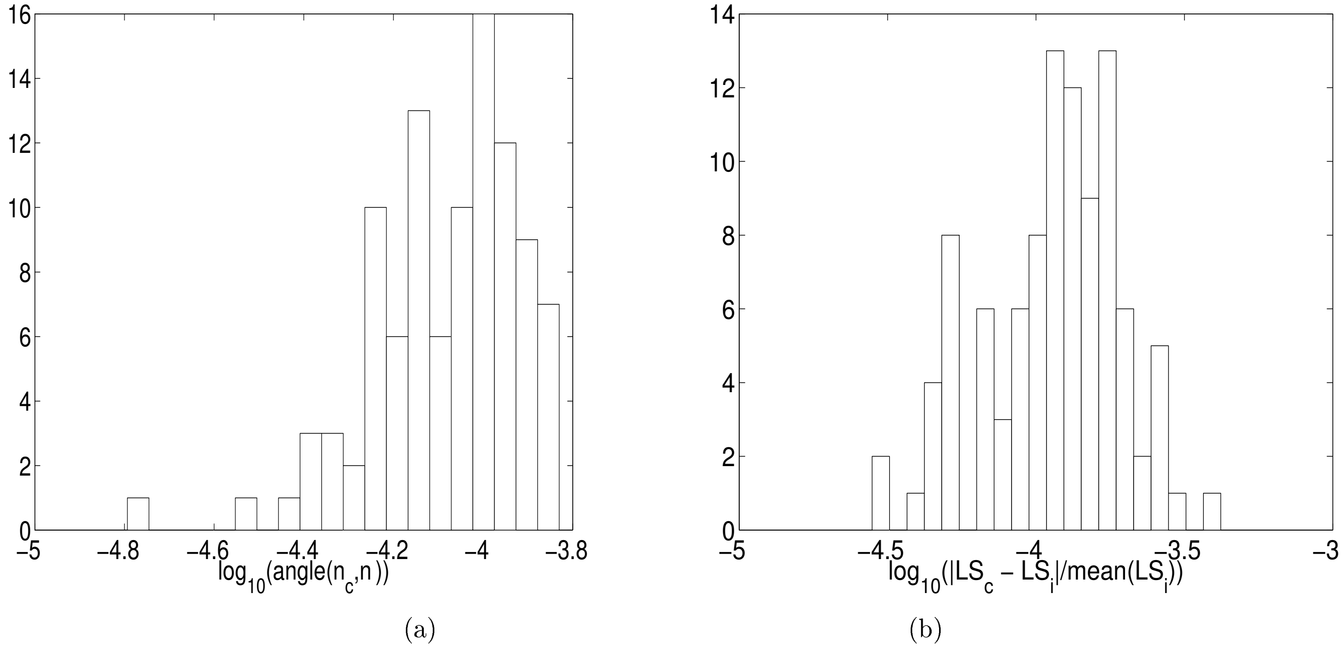


Figure 20: (a): distribution of the estimation difference. (b): distribution of the LS residuals difference.

Leaving some features unconstrained

Another series of tests has been performed without considering the median constraint (constraint 12). This is in order to check if this will affect the position of the four plane surfaces with respect to the cylinder axis and therefore the estimation of the edge of the square surface S_5. Results are shown in

Table 9 with the previous results for comparison. It is noticed that the radius estimation is not affected but the incorporation of the additional constraints slightly reduces the diagonal length error.

	distance(S_1, S_3)	distance(S_2, S_4)	diagonal of S_5	cylinder radius
without constraints	–	–	–	14.64
with all constraints	21.17	21.17	29.94	14.97
without median constraint	21.15	21.15	29.91	14.97
actual values	21.28	21.28	30.02	15.01

Table 9: comparison of the estimation without median constraints with previous results.

Invalidity of the constraints

Suppose that one or more constraints do not reflect the actual relationships between features and therefore are invalid. What would be the behaviour of the algorithm? Will these "false constraints" be satisfied? What could be the resulting estimated model ?

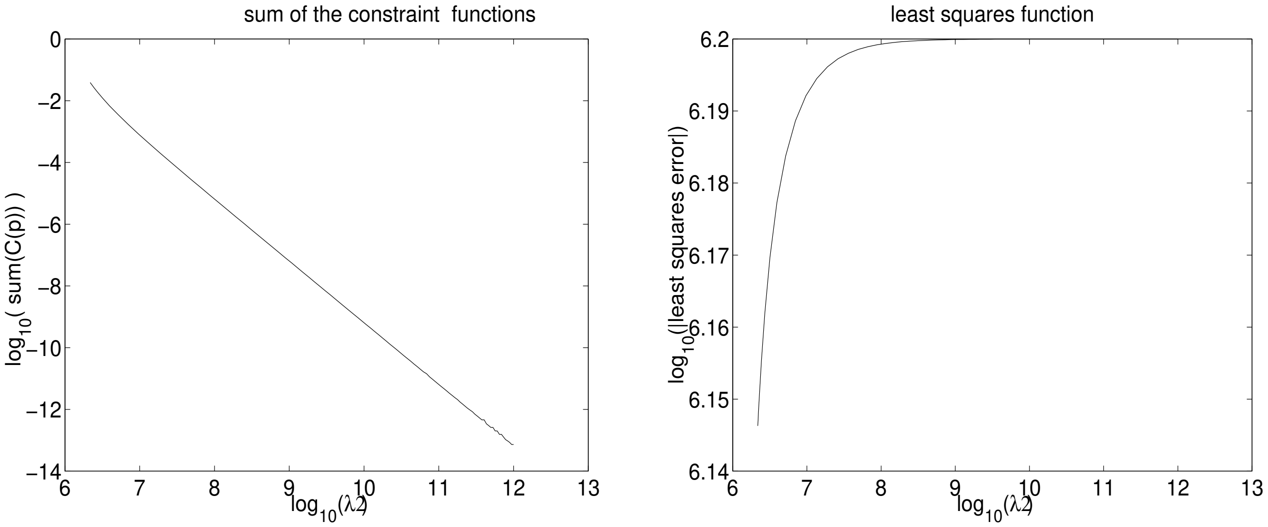
To answer these questions, some angle constraints were set to an incorrect values. Three tests was carried out, in the first the angle $(\vec{n_1}, \vec{n_2})$ was set to $\pi/3$, in the second the angle $(\vec{n_1}, \vec{n_5})$ was set to $\pi/3$ and in the third test both angles $(\vec{n_1}, \vec{n_5})$ and $(\vec{n_2}, \vec{n_5})$ were set to $\pi/3$ (the right values are $\pi/2$ for both angles).

In all these tests the behaviour and the convergence of the algorithm were qualitatively similar to those of the previous experiments. The algorithm converges, the least squares error stabilizes and all the constraints are satisfied at the end of the process although the least squares error is greater than the valid constraints case (Figure 21). Table 10 summarizes the estimated model characteristics in each of the three tests.

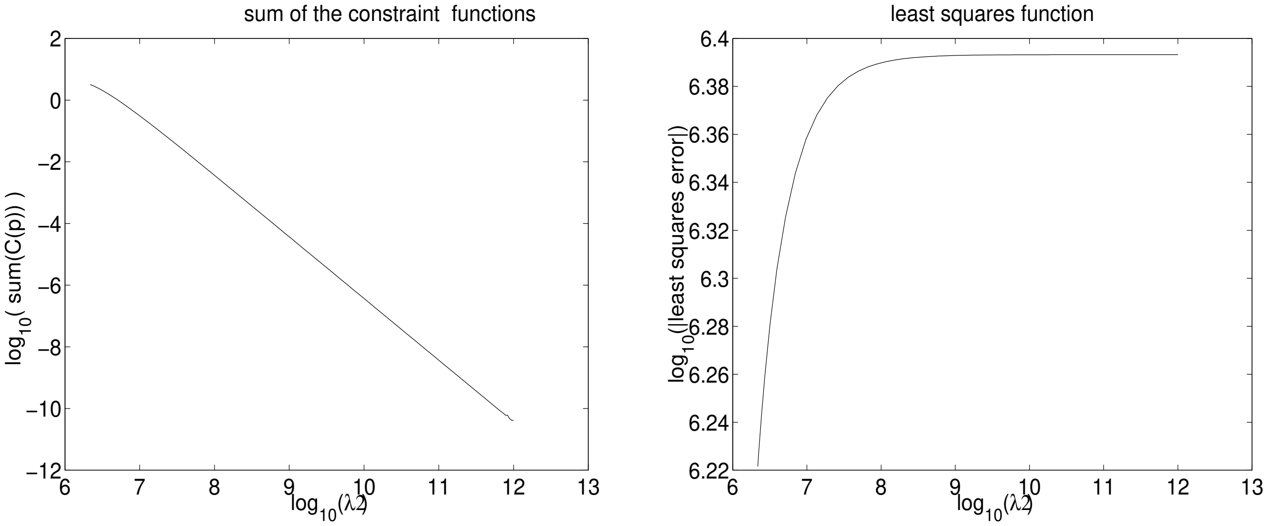
An examination of Table 10 leads to the following observations:

1. In all of the three tests the cylinder and the sphere characteristics are not affected by the invalid constraints.
2. The normal $\vec{n_1}$ which is involved in each of the invalid constraints is affected in three tests.
3. The normal $\vec{n_2}$ is changed in the first and third test where it is involved in the invalid constraints whereas it is unchanged in the second test where it is not involved.
4. The normal $\vec{n_5}$ is kept unchanged in all the tests even in those where it is involved in the inconsistent constraints.

From these observations we can deduce that invalid constraints affect the object feature's locations by shifting the involved features toward positions where the invalid constraints are satisfied. Consequently, this will increase the least squares error (Figure 21). The locations and the characteristics of the surfaces which are not involved in the invalid constraints are not affected (the



(a)



(b)

Figure 21: (a):Constraint error function and least squares error function function for valid constraints. (b):Constraint error and least squares error function for invalid constraints (3^{rd} test).

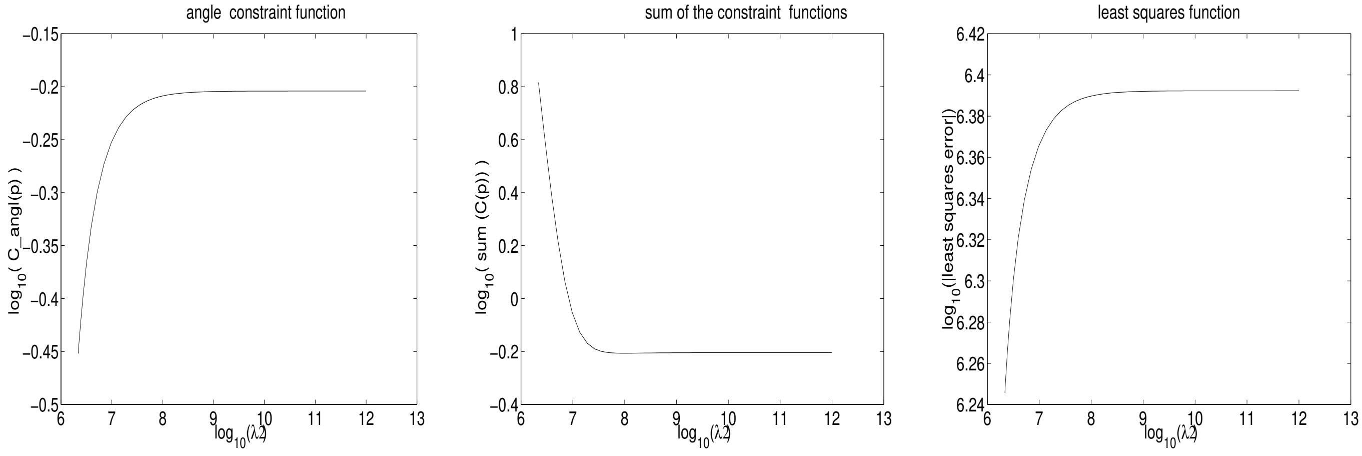
	$\vec{n_1}$	$\vec{n_2}$	$\vec{n_5}$	R_{cyl}	R_{sph}	axe_{cyl}	$Center_{sph}$
$(\vec{n_1}, \vec{n_2}) = \pi/3$	-0.61 -0.47 -0.62	-0.58 0.52 -0.62	0.72 -0.02 -0.69	14.97	14.97	0.72 -0.02 -0.69	86.30 -87.38 17.44
$(\vec{n_1}, \vec{n_5}) = \pi/3$	-0.08 -0.60 -0.78	-0.46 0.72 -0.50	0.72 -0.02 -0.69	14.97	14.97	0.72 -0.02 -0.69	86.31 -87.41 17.44
$(\vec{n_1}, \vec{n_5}) = \pi/3$ $(\vec{n_2}, \vec{n_5}) = \pi/3$	-0.02 -0.68 -0.72	0.05 0.72 -0.68	0.72 -0.02 -0.69	14.97	14.97	0.72 -0.02 -0.69	86.31 -87.42 17.44
true constraints	-0.52 -0.67 -0.51	-0.45 0.73 -0.50	0.72 -0.02 -0.69	14.97	14.97	0.72 -0.02 -0.69	86.30 -87.38 17.44

Table 10: The object characteristic estimates for invalid constraints and true constraints (last row).

sphere and the cylinder). However the normal $\vec{n_5}$ seems not to satisfy this rule since its orientation stays unchanged for all the cases where it is involved in an inconsistent constraint. This is explained by the fact that unlike $\vec{n_1}$ and $\vec{n_2}$, $\vec{n_5}$ is also involved in other constraints, in particular it is constrained to have the same orientation as the cylinder axis. The satisfaction of this constraint keeps it collinear to the cylinder axis and prevents its orientation from being affected. Thus the algorithm satisfies the invalid constraints in which $\vec{n_5}$ is involved by acting on the other normals involved in these constraints.

Inconsistency of the constraints

In this test we investigated what would be the behaviour of the algorithm when some constraints are inconsistent and have a conflict between them. For this purpose we introduced two additional inconsistent angle constraints by imposing the angles $(\vec{n_1}, \vec{n_2})$ and $(\vec{n_1}, \vec{n_5})$ to be $\pi/3$, which conflicts with the two other consistent constraints defining each pair of $(\vec{n_1}, \vec{n_2})$ and $(\vec{n_1}, \vec{n_5})$ as orthogonal vectors. The trial carried out with these inconsistent constraints revealed that the algorithm converges normally (Figure 22) both the least squares and the constraint functions stabilizing at the end of the algorithm. From Figure 22.a we notice that the angle constraints are not satisfied. This is obvious because it is not possible to satisfy conflicting constraints simultaneously. The converging values of the constraint function (the sum of all the constraints) Figure 22.b and the angle constraints error are practically equal at the end of the optimization process. This shows that the other consistent constraints are satisfied. This suggests that an inconsistent set of constraints can be detected by observing the convergence of the constraint error rather than its reduction to zero.



(a) (b) (c)

Figure 22: Results for inconsistent constraints (a): The sum of the angle constraints' errors. (b): the sum of all the constraint functions. (c) the least squres error.

Global shape improvement

The different tables shown in this section compare the geometric characteristics of the object issued from an optimization with constraints and an optimization without and show the improvement of the object characteristics estimates when constraints are applied. The results presented in the tables are the average of the 100 estimations. The angles between each pair of surfaces $(S_1, S_2), (S_1, S_5)$ and (S_2, S_5) were set as constraints and the constraints were nearly perfectly satisfied. From Table 11 we notice the satisfaction of the square property of the prism, illustrated by the equality of the two distances separating (S_1, S_3) and (S_2, S_4), their values which is close to the actual length of the edge of the square plane S_5 and closeness of the estimated value of the diagonal of S_5 to the actual value when the constraints are considered. The distances between these last surfaces for an optimization without constraints is not mentioned in this table since the estimated surfaces are not parallel.

The improvement of the quadric surfaces estimation is confirmed again for this object (Table 12 and Table 13). The radius estimation error is less than $0.04mm$ for both the cylinder and the sphere. The standard deviations of the cylinder and the sphere radius have been significantly reduced as well.

	distance(S_1, S_3)	distance(S_2, S_4)	diagonal of S_5
with constraints	21.17	21.17	29.95
standard deviation/mean	0.03 %	0.03%	0.03%
actual values	21.28	21.28	30.02

Table 11: Improvement of the prism characteristic estimates.

cylinder parameters	angle(axis, S_5's normal)	radius (mm)	σ/mean (radius)
without constraints	1.55	14.64	0.12%
with constraints	0.00*	14.97	0.03 %
actual values	0	15.01	0

Table 12: Improvement of the cylinder characteristic estimates.

sphere parameters	distance(center, cylinder axis)	radius (mm)	σ/mean (radius)
without constraints	1.36	16.02	0.11%
with constraints	0.00*	14.97	0.03 %
actual values	0	15.01	0

Table 13: Improvement of the sphere characteristic estimates.

CONCLUSION

This work presents a framework for the reconstruction of object models incorporating geometric constraints. It can hold a large number of varied constraint types and incorporates them integrally without the need for linearization. The geometric constraints are formulated in quadratic matrix functions which are continuous, differentiable and ensure a compact expression of the constraints and easy handling by the optimization process.

The proposed optimization algorithm belongs to sequential nonlinear programming. Theoretically, the characteristics of the objective function and the constraint functions satisfy the requirements for an efficient application of the algorithm. The availability of a good initial solution obtained from the measurement data ensures the convergence of the algorithm towards the optimal solution. However, the last condition make it inappropriate for constrained object design applications where a reasonable initial solution is not available. The practical difficulties of the algorithm manifested in the ill-conditioned Hessian matrix in the Levenberg-Marquardt algorithm is overcome by using an appropriate numerical technique.

The constraints can be integrated in a batch form at once or sequentially. In the sequential version the addition of a new constraint does not affect the satisfaction of the previously implemented constraints.

The experiments carried out on the different objects empirically confirm the convergence of the algorithm. The parameter optimization search does produce shape fitting that satisfies almost perfectly the constraints. They show in particular that the least squares error grows slightly as the constraints are applied and the weighting values increased, but this stabilizes above certain values of the λ_k while the constraint errors are still decreasing. Thus it is possible to satisfy the constraints up to the desired tolerance without seriously affecting the quality of the data fitting.

The above observations suggest that the proposed approach allows flexibility in the incorporation of the constraints, as well as in their satisfaction. The sequential version of the constraint implementation allows a human reverse engineer to supply them interactively whereas the batch form of constraint incorporation is suitable for being inferred by a knowledge-based system reasoning from general engineering principles. The stabilization of the increase of the least squares error while the constraint errors are still decreasing as λ_k increases offers the possibility that the user can control the degree of satisfaction of the constraints and to set the tolerances as high as necessary.

Regarding the slight increases of the LS error, we have to bear in mind that the increase of the least squares residuals value may not reflect a bad estimation in the case when measurement errors are systematic, e.g. miscalibration and registration error. This last type of error is expected in our data since the registration process is performed by hand. We believe that the slight increase of the least squares error as a consequence of the constraints satisfaction is a result of the object being located more accurately. Future work could investigate a more robust form for the objective function involving the data noise statistics.

The different trials applied on the multi-quadric objects empirically confirm the stability of the convergence of the algorithm. The low values of the parameters' variances illustrates the stability of the solution provided by the optimization search process. On the level of the object shape, this aspect is reflected by the small values of the standard deviations of the object shape characteristics. The tests have shown as well that the proposed approach leads to an estimate which is close to the optimal solution (e.g. the solution given when the constraints could be combined with the least squares error). The experiments also show that applying the constraints to only some features does not seriously affect the estimation of the unconstrained surfaces. The estimation is still improved compared to the case of unconstrained optimization.

The examination of some constraint invalidity cases has shown the constraints are always satisfied whether they are valid or not and the behaviour of the algorithm is typically the same. The satisfaction of invalid constraints leads to the relocation of the involved and less constrained features (having more degrees of freedom) toward positions where the inconsistency is removed. However this will result in a false object model. The trial performed with constraint inconsistencies case revealed the same behaviour regarding the convergence of the algorithm but the inconsistent constraints are not satisfied at the end of the optimization. This suggests that constraint validity and consistency checking have to be done before starting the optimization process.

Regarding the model estimation accuracy, the comparison of the object dimension estimates with those from unconstrained fitting confirms that the proposed approach improves the quality of the model construction to a high degree. For the second-quadric object the radius of the cylinder and the sphere have an estimation error in the range of $0.04mm$, the edge of the square prism has an estimation error around $0.1mm$. The radius of the cylinder patch estimated from the registered half cylinder has an estimation error around

$0.01mm$. For a single a view it is less than $0.5mm$. The same range of error is obtained for the radius of the cylinder patch of the first multi-quadric object.

Results for the cone patches are reasonable for the cone object, the half angle estimation error is less than 0.5^0, but less satisfactory for the first multi-quadric object. This is mainly due to the relatively small area of the conic patch. In fact, the comparison of the estimation error for the quadric surfaces shows that the larger the quadric patch, the smaller the estimation error. We intentionally chose to work with small patches because unconstrained fitting surface techniques fail to give reasonable estimates in this case (see the radius estimate in Table 5) even with robust algorithms due to the "poorness" of the information embodied in the patch.

Regarding the constraint representation, it is noticed that some constraints involve a large number of equations, in particular for the circularity constraint. One solution is to implicitly impose those constraints through the representation of the quadric equation $((X - X_o)^T(I - \vec{n}\vec{n}^T)(X - X_o) - r^2 = 0$) for the cylinder and $((X - X_o)^T(\vec{n}\vec{n}^T - cos^2(\alpha))(X - X_o) = 0)$ for the cone, where $\vec{n}$ is the unit orientation vector of the cylinder or the cone axis, X_o is a point on the cylinder axis in the cylinder case and is the apex for the cone case. The main problem encountered with this representation is the complexity of the related objective function and the difficulty of separating the data terms from the parameter terms. It will be also worthwhile investigating some topological constraints between surfaces which have a common intersection.

Although the experiments presented in this work were performed on single objects, the proposed approach can optimize multiple objects simultaneously. Generally industrial parts are designed to fit to each other, so geometric relationships between the parts may be considered and the resulting constraints can be incorporated as well in the optimization process.

Another area we are starting to investigate is how one might automatically identify inter-surface relationships that can have a constraint applied. In manufacturing objects, simple angular and spatial relationships are given by design. So, it should be straightforward to define simple Mahalanobis distance tests that hypothesize standard feature relationships, subject to the feature's statistical position distribution. With this analysis, a computer program could propose a variety of constraints that a human could either accept or reject, after which shape reconstruction could occur.

It is very likely that the consideration of the constraints tends to shift the object localization towards the actual position. The experiments carried out with the synthetic polyhedral objects provides evidence for this. It seems that the incorporation of the constraints compensate up to certain degree for the effect of the systematic errors and allows better estimation, although the authors have not yet a theoretical proof of this interpretation. This issue was partially justified in the work of Bolle *et al* [6], but only for the intrinsic constraints, namely the circularity of the cylinder and perfect sphere. By considering a larger set of constraints, the proposed framework generalizes the concept of object localization considering the constraints and make a step toward a framework which unifies object localization and object modelling.

All the algorithm procedures have been implemented with C++. The computation time for the reconstruction on a $200Mhz$ sun Ultrasparc workstation is typically few minutes or less (1-5 minutes), which is suitable for CAD work.

ACKNOWLEDGEMENT

The work presented in this paper was funded by UK EPSRC grant GR /L25110.

References

[1] R. Anantha, G.A. Kramer, R.H. Crawford. Assembly modelling by geometric constraint satisfaction. CAD, Vol.28, No.9, pp 707-722, 1996.

[2] R. Anderl, R Mendegen. Modelling with constraints: Theoretical foundation and application. CAD, Vol. 28, No 3, pp 155-168 1996.

[3] F. Arbab, B. Wang. Reasoning about geometric constraints. In H. Yoishikawa and T.Holden (eds), *Intelligent CAD II,* North Holland, pp. 93-107, 1990.

[4] R.J.Bell. *An elementary treatise on coordinate geometry.* McMillan and Co, London, 1910.

[5] P.J Besl, R.C. Jain. Three dimensional object recognition. ACM Computing Surveys, Vol.17, No.1, pp.75-145, 1985.

[6] R.M.Bolle, D.B.Cooper. On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data. IEEE Trans. PAMI, Vol.8, No.5, pp.619-638, September 1986.

[7] R.C. Bolles, R.A. Cain. Recognizing and locating partially visible objects. Int. Journal of Robotics Research, Vol.1, No.3, pp. 57-82, 1982.

[8] R. C. Bolles, P. Horaud. 3DPO, a three dimensional part orientation system. Int. Journal of Robotics Research, Vol.5, No.3, pp. 3-26, 1986.

[9] A.H. Boring. The programming language aspect of ThingLab, a constrained oriented simulation laboratory. ACM TOPALS, Vol.3, No.4, pp.353-387, 1981.

[10] W. Bouma, I. Fudos, C. Hoffman, J. Cai, R. Paige. Geometric constraint solver. CAD, Vol. 27, No.6, pp 487-501, 1995.

[11] K.L.Boyer, M.J.Mirza, G.Ganguly. The Robust Sequential Estimator. IEEE Trans. PAMI, Vol.16, No.10, pp.987-1001, October 1994.

[12] C.G Broyden, N.F. Attia. Penalty Functions, Newton's Method and Quadratic Programming. Journal of optimization theory and applications, Vol.58, No.3, 1988.

[13] B. Buchberger. Application of Grobner basis in non-linear computational geometry. In D. Kapur and and J. Mundy (eds), *Geometric Reasoning,* MIT press, 1989.

[14] Y.Chen, G.Medioni. Object Modeling by Registration of Multiple Range Images. Proc. IEEE Int. Conf. Robotics and Automation, Vol.2 pp.724-729, April 1991.

[15] J.De Geeter, H.V.Brussel, J.De Schutter, M. Decreton. A Smoothly Constrained Kalman Filter. IEEE Trans. PAMI pp.1171-1177, No.10, Vol.19, October 1997.

[16] L. Eggli, C.Y. Hsu, B.D. Bruderlin, G. Elbert. Inferring 3D models from freehand sketches and constraints. CAD, Vol.29, No.2 pp. 101-112, 1997.

[17] C.X Feng, A. Kusiak. Constraints-based design of parts. CAD, Vol 27 No 5 1995 pp 343 -352, 1995.

[18] M. Ferri, F. Magnilli, G. Viano. Projective pose estimation of linear and quadratic primitives in monocular computer vision. Image understanding, Vol.58, No.1, pp.66-84, 1993.

[19] A.V Fiacco, G.P McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, NewYork 1968.

[20] A.F. Fitzgibbon, D.W. Eggert, R.B. Fisher. High-level CAD model acquisition from range images. CAD, Vol.29, No.4, pp.321-330, 1997.

[21] R.Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, 1987.

[22] P.J.Flynn, A.K.Jain. Surface Classification: Hypothesizing and Parameter Estimation. Proc. IEEE Comp. Soc. CVPR, pp. 261-267. June 1988.

[23] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, C. Rothwell. invariant descriptors for 3D object recognition and pose. IEEE PAMI, Vol.13, No.10, pp. 971-991, October 1991.

[24] I. Fudos, C.Hoffman. Constraints-based parametric conics for CAD. CAD, Vol.28, No.2 pp. 91-100, 1996.

[25] X.S. Gao, S.C.Chou. Solving geometric constraint systems. I. A global propagation approach. CAD, Vol.30, No.1, pp.47-54, 1988.

[26] P.E.Gill, W.Murray, M.H.Wright. *Practical Optimization.* Academic Press, 1981.

[27] W.E. Grimson. *The role of geometric constraints.* MIT press, London, 1990.

[28] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[29] G.H. Golub, C.F. Van Loan *Matrix Computations.* John Hopkins University Press, 3rd edition, 1996.

[30] E.Gmur, H.Bunke. 3D Object Recognition Based on Subgraph matching in Polynomial time. In Sanfeliu, Mohr, Pavildis (eds), *Structural Pattern Analysis,* pp.131-147, World Scientific Publications. 1990.

[31] I. Herman. The use of projective geometry in computer graphics. Lecture notes in computer science 564. Springer Verlag. 1992.

[32] A. Heydon, G.Nelson. The Juno-2 Constraint-Based-Drawings Editor. SRC research report 131a, 1994.

[33] C.M.Hoffmann, P.J. Vermeer. Geometric constraint solving in $\mathcal{R}^2$ and $\mathcal{R}^3$ In Du Dingzhu and Hwang Frank (eds), *Computing in Euclidean Geometry,* 2nd edition, World Scientific Publishing, 1995.

[34] C.M.Hoffmann, P.J. Vermeer. A spatial constraint problem. Proc. 2nd Workshop on Computational Kinematics, pp.83-92, Sophia Antipolis, 1995.

[35] C.M. Hoffmann, A. Lomonosov, M. Sitharan. Finding Solvable Subsets of Constraint Graphs. Springer LNCS 1330, G. Smoka eds, pp.463-477, 1997.

[36] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke. D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, R. Fisher. An Experimental Comparison of Range Segmentation Algorithms. IEEE Trans. PAMI, Vol.18, No.7, pp.673-689, July 1996.

[37] S.L.S. Jacoby, J.S Kowalik, J.T.Pizzo. *Iterative Methods for Nonlinear Optimization Problems.* Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1972.

[38] D.Jian, S.Vijayan. Manufacturing feature determination and extraction, Part I: optimal volume segmentation. CAD, Vol.29, Vol.6, 1997,pp.427-440.

[39] K. Kondo. Algebraic method for manipulation of dimensional relationships in geometric models. Geometric Aided Design, Vol.24 No.3 pp 141-147, 1992.

[40] S.Kumar, S.Han, D.Goldgof, K.Boyer. On Recovering Hyperquadrics from Range data. IEEE Trans. PAMI, Vol.17, No.11, pp.1079-1083, November 1995.

[41] K.L Lawrence, S.N Muthukrishna, R.V Nambiar. Refinement of 3D meshes at surface intersections. CAD, Vol. 27, No. 8 , 1995.

[42] R.A Light, D.C Gossard. Modification of geometric models through variational geometry. CAD, Vol. 14, No. 4 , pp. 209-214, 1982.

[43] D.G. Lowe. Fitting parameterized three dimensional models to images. IEEE PAMI, Vol.13, No.5, pp. 441-450, 1991.

[44] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1996

[45] C.J.Ong, Y.S.Wong, H.T. Loh, X.G.Hong. An optimization approach for biarc curve-fitting of B-spline curves. CAD, Vol. 28 , No. 12, pp. 951-959, 1996.

[46] J.Porrill. Optimal Combination and Constraints for Geometrical Sensor Data. International Journal of Robotics Research, Vol.7, No.6, pp.66-78, 1988.

[47] L. Quan. Conic reconstruction and correspondence from two views. IEEE PAMI, Vol.18, No.2, pp. 151-160, 1996.

[48] A.A.G. Requicha. Reprsentation of Tolerances in Solid Modelling: Issues and Alternative Approaches. In J.W.Boyse amd M.S.Pickett (Eds), *Solid Modelling by Computers: from Theory to Applications,* New York: Plenum, 1984, pp.3-22.

[49] C. Robertson, R.B. Fisher, D. Corne, N. Werghi, A.P. Ashbrook. Investigating Evolutionary Optimisation of Constrained Functions to Capture Descriptions from Range Data. Proc. 3rd On-line World Conference on Soft Computing (WSC3), 1998.

[50] R.T. Rockafellar. *Convex Analysis.* Princton University Press, 1970.

[51] A.P. Rockwood, J. Winget. Three-dimensional object reconstruction from two-dimensional images. CAD, Vol.29, No.4, pp.279-286, 1997.

[52] R. Safee-Rad, I. Tchoukanov, K.C. Smith, B. Benhabib. Constraints on quadratic-curved features under perspective projection. Image and Vision computing, Vol.10, No.8, pp. 532-548, 1992.

[53] L.G Shapiro, R.M Haralick. Structural descriptions and inexact matching. IEEE PAMI, Vol.3, No.5, pp.504-419, september 1981.

[54] B.S. Shin, Y.G. Shin. Fast 3D solid model reconstruction from orthographic views. CAD, Vol.30, No.1, pp.63-76, 1998.

[55] H.Y.Shun, K.Ikeuchi, R.Reddy. Principal Component Analysis with Missing Data and its Application to Polyhedral Object Modelling. IEEE Trans. PAMI, Vol.17, No.9, pp.855-867. 1995.

[56] M.Soucy, D.Laurendo. Surface Modelling from Dynamic Integration of Multiple Range Views. Proc 11th Int. Conf. Pattern Recognition, pp.449-452, 1992.

[57] G. Sunde. Specification of shape by dimensions and other geometric constraints. In M.J Vozny *et al*(eds) *Geometric Modelling for CAD applications,* pp. 199-213, North Holland, 1988.

[58] I.E Sutherland. *Sketchpad: A man-machine graphical communication system.* Proc. Joint Computer Conference, pp.329-346, 1963.

[59] T.Taura, I.Nagasaka, A.Yamagishi. Application of evolutionary programming to shape design. CAD, Vol.30, No.1, pp. 29-35, 1988.

[60] W.T Wu. *Basic principles of mechanical theorem proving in geometry.* Springer Verlag, 1993.

[61] T. Varady, R. R. Martin, J. Cox. Reverse engineering of geometric models, an introduction. CAD, Vol.29, No.24, pp. 255-268, 1997.

[62] B.C.Vemuri, J.K Aggrawal. 3D Model Construction from Multiple Views Using Range and Intensity Data. Proc. CVPR, pp.435-437, 1986.

[63] D.R Wallace, M.J Jakiela, W.C.Flowers. Design search under probabilistic specifications using genetic algorithms. CAD, Vol.28, No.5, pp. 405-421, 1996.

[64] X.Wang, F. Cheng, B. Brian. Energy and B-spline interproximation. CAD, Vol.29, No.7, pp. 405-421, 1997.

[65] N. Werghi, R.B.Fidher, C.Robertson, A.Ashbrook. Improving Model Shape Acquisition by Incorporating Geoemetric constraints. Proc. BMVC, pp.530-539, Essex, September 1997

[66] R.C. Wetkamp. Geometric constraint management with quanta. in D.C Brown *et al*(eds) *Intelligent Computer Aided Design,* pp. 409-426, North Holland, 1992.

[67] Q.W. Yan, C.L.P. Chen, Z. Tang. Reconstruction of 3D objects from orthographic projections. CAD, Vol.26, No.9, 1994.

Appendix 1: Notation

$\vec{i_r}$ is a vector in which all the elements are zero except the r^{th} element which is equal to 1.
$\vec{i}_{(r,s)}$ is a vector in which all the elements are zero except the r^{th} and the s^{th} elements which are equal to 1
$\vec{i}_{(r,-s)}$ is a vector in which all the elements are zero except the r^{th} and the s^{th} elements which are equal to 1 and -1 respectively.
$\vec{i}_{(r,s,t,l)}$ is a vector in which all the elements are zero except for the r^{th}, s^{th}, t^{th} and l^{th} elements which are equal to $1, 1, -1$ and -1 respectively.
$M_{(r,s)}$ is a diagonal matrix in which all the elements are zero except the r^{th} and the s^{th} elements which are equal to 1 and -1 respectively.
$U_{(r,s)}$ is a diagonal matrix defined by:

$$U_{(r,s)} = \begin{cases} U(i,i) = 1 & \text{if } r \leq i \leq s \\ U(i,i) = 0 & \text{otherwise} \end{cases}$$

$I_{(r,s)}$ a symmetric matrix defined by:

$$I_{(r,s)} = \begin{cases} I(i,j) = I(j,i) = 1 & \text{for } r \leq i \leq s, \quad r \leq j \leq s \\ I(i,j) = 0 & \text{otherwise} \end{cases}$$

$U_{(r,s,p,t)}$ is a diagonal matrix defined by:

$$U_{(r,s,p,t)} = \begin{cases} U(i,i) = 1 & \text{if } r \leq i \leq s \\ U(i,i) = -1 & \text{if } p \leq i \leq t \\ U(i,i) = 0 & \text{otherwise} \end{cases}$$

$L_{(r,s,p)}$ a symmetric matrix defined by:

$$L_{(r,s,p)} = \begin{cases} L(i,j) = L(j,i) = 1/2 & \text{fot } r \leq i \leq r+p, \quad s \leq j \leq s+p \\ L(i,j) = L(j,i) = 0 & \text{otherwise} \end{cases}$$

$T_{(r,s,p,t)}$ a symmetric matrix defined by:

$$T_{(r,s,p,t)} = \begin{cases} T(r,t+5) = T(t+5,r) = T(s,p) = T(p,s) = 1/2 \\ T(r,t) = T(t,r) = T(s,p+5) = T(p+5,s) = -1/2 \end{cases}$$

Appendix 2: Constraints definition

The half cylinder

Constraint 3 is represented by two conditions: axis vector $\vec{n}$ is orthogonal to S_2's normal $\vec{n_2}$, and one point of the axis satisfies S_2's equation. The first condition is guaranteed by constraint 2 since $\vec{n_2}$ is orthogonal to $\vec{n_1}$. For the second condition the point X_o in Section "The cylinder" has to satisfy the equation:

$$C_{axe}(\vec{p}) = (X_o^T \vec{n_2} + d_2)^2 = 0$$

Using equations (9) and (15) this equation can be written as

$$C_{axe}(\vec{p}) = (-[u, v, w]^T \vec{n_2} + d_2)^2 = (\vec{i_8}^T \vec{p} - \vec{p}^T L_{(5,15,2)} \vec{p})^2 = 0$$

The cylinder circularity constraint is implicitly defined by the equations (15). From these equations we extract the following constraints on the parameter vector $\vec{p}$:

$$C_{circ_1}(\vec{p}) = (\vec{i_9}^T \vec{p} + \vec{p}^T U_{(1,1)} \vec{p} - 1)^2 = 0 \qquad C_{circ_4}(\vec{p}) = (\vec{i_{12}}^T \vec{p} + \vec{p}^T L_{(1,2,0)} \vec{p})^2 = 0$$
$$C_{circ_2}(\vec{p}) = (\vec{i_{10}}^T \vec{p} + \vec{p}^T U_{(2,2)} \vec{p} - 1)^2 = 0 \qquad C_{circ_5}(\vec{p}) = (\vec{i_{13}}^T \vec{p} + \vec{p}^T L_{(1,3,0)} \vec{p})^2 = 0$$
$$C_{circ_3}(\vec{p}) = (\vec{i_{11}}^T \vec{p} + \vec{p}^T U_{(3,3)} \vec{p} - 1)^2 = 0 \qquad C_{circ_6}(\vec{p}) = (\vec{i_{14}}^T \vec{p} + \vec{p}^T L_{(2,3,0)} \vec{p})^2 = 0$$

We group these six constraints into a single one:

$$C_{circ}(\vec{p}) = \sum_{k=1}^{6} C_{circ_k}(\vec{p}) = 0$$

The cone object

Eliminating $cos^2\alpha$ from the cone circularity equations (19) and taking into consideration constraint 1, the circularity constraints are formulated as:

$$\begin{aligned} a - b &= n_{1_x}^2 - n_{1_y}^2 & \qquad h &= n_{1_x} n_{1_y} \\ a - c &= n_{1_x}^2 - n_{1_z}^2 & \qquad g &= n_{1_x} n_{1_z} \\ b - c &= n_{1_y}^2 - n_{1_z}^2 & \qquad f &= n_{1_y} n_{1_z} \end{aligned}$$

A matrix formulation of these constraints as a function of the parameter vector $\vec{p}$ is:

$$C_{circ_1}(\vec{p}) = (\vec{j_{(5,-6)}}^T \vec{p} - \vec{p}^T M_{(1,2)} \vec{p})^2 = 0 \qquad C_{circ_4}(\vec{p}) = (\vec{i_8}^T \vec{p} - \vec{p}^T L_{(1,2,0)} \vec{p})^2 = 0$$
$$C_{circ_2}(\vec{p}) = (\vec{j_{(5,-7)}}^T \vec{p} - \vec{p}^T M_{(1,3)} \vec{p})^2 = 0 \qquad C_{circ_5}(\vec{p}) = (\vec{i_9}^T \vec{p} - \vec{p}^T L_{(1,3,0)} \vec{p})^2 = 0$$
$$C_{circ_3}(\vec{p}) = (\vec{j_{(6,-7)}}^T \vec{p} - \vec{p}^T M_{(2,3)} \vec{p})^2 = 0 \qquad C_{circ_6}(\vec{p}) = (\vec{i_{10}}^T \vec{p} - \vec{p}^T L_{(2,3,0)} \vec{p})^2 = 0$$

which are grouped into a single constraint:

$$C_{circ}(\vec{p}) = \sum_{k=1}^{6} C_{circ_k}(\vec{p}) = 0$$

Multi-quadric object 1

The first four angle constraints lead to six equations involving the surface normals:

$$\begin{array}{rclcrcl}
\vec{n_1}^T\vec{n_2} & = & cos(2\pi/3) = -0.5 & \quad & \vec{n_2}^T\vec{n_3} & = & cos(\pi/2) = 0 \\
\vec{n_1}^T\vec{n_3} & = & cos(\pi/2) = 0 & & \vec{n_2}^T\vec{n_4} & = & cos(2\pi/3) = -0.5 \\
\vec{n_1}^T\vec{n_4} & = & cos(2\pi/3) = -0.5 & & \vec{n_3}^T\vec{n_4} & = & cos(\pi/2) = 0
\end{array}$$

A vector formulation of these equations as a function of $\vec{p}$ is:

$$\begin{array}{rclcrcl}
C_{angl_1}(\vec{p}) & = & (\vec{p}^T L_{(1,5,2)}\vec{p} + 0.5)^2 = 0, & \quad & C_{angl_4}(\vec{p}) & = & (\vec{p}^T L_{(5,9,2)}\vec{p})^2 = 0 \\
C_{angl_2}(\vec{p}) & = & (\vec{p}^T L_{(1,9,2)}\vec{p})^2 = 0, & & C_{angl_5}(\vec{p}) & = & (\vec{p}^T L_{(5,13,2)}\vec{p} + 0.5)^2 = 0 \\
C_{angl_3}(\vec{p}) & = & (\vec{p}^T L_{(1,13,2)}\vec{p} + 0.5)^2 = 0, & & C_{angl_6}(\vec{p}) & = & (\vec{p}^T L_{(9,13,2)}\vec{p})^2 = 0
\end{array}$$

These equations are then grouped into:

$$C_{angl}(\vec{p}) = \sum_{i}^{6} C_{angl_i}(\vec{p}) = 0$$

The circularity of the cylinder and the cone are ensured using the set of equations (15) and (19) respectively. This gives the following constraints on the parameter vector $\vec{p}$ for the cylinder:

$$\begin{array}{rcl}
C_{circ_{cyl_1}}(\vec{p}) = (\vec{i_{17}}^T\vec{p} + \vec{p}^T U_{(9,9)}\vec{p} - 1)^2 & \quad & C_{circ_{cyl_4}}(\vec{p}) = (\vec{i_{20}}^T\vec{p} + \vec{p}^T L_{(9,10,0)}\vec{p})^2 \\
C_{circ_{cyl_2}}(\vec{p}) = (\vec{i_{18}}^T\vec{p} + \vec{p}^T U_{(10,10)}\vec{p} - 1)^2 & & C_{circ_{cyl_5}}(\vec{p}) = (\vec{i_{21}}^T\vec{p} + \vec{p}^T L_{(9,11,0)}\vec{p})^2 \\
C_{circ_{cyl_3}}(\vec{p}) = (\vec{i_{18}}^T\vec{p} + \vec{p}^T U_{(11,11)}\vec{p} - 1)^2 & & C_{circ_{cyl_6}}(\vec{p}) = (\vec{i_{22}}^T\vec{p} + \vec{p}^T L_{(10,11,0)}\vec{p})^2
\end{array}$$

and the following constraints for the cone:

$$\begin{array}{rcl}
C_{circ_{cone_1}}(\vec{p}) = (\vec{i}^T_{(27,-28)}\vec{p} - \vec{p}^T M_{(13,14)}\vec{p})^2 & \quad & C_{circ_{cone_4}}(\vec{p}) = (\vec{i}^T_{30}\vec{p} - \vec{p}^T L_{(13,14,0)}\vec{p})^2 \\
C_{circ_{cone_2}}(\vec{p}) = (\vec{i}^T_{(27,-29)}\vec{p} - \vec{p}^T M_{(13,15)}\vec{p})^2 & & C_{circ_{cone_5}}(\vec{p}) = (\vec{i}^T_{31}\vec{p} - \vec{p}^T L_{(13,15,0)}\vec{p})^2 \\
C_{circ_{cone_3}}(\vec{p}) = (\vec{i}^T_{(28,-29)}\vec{p} - \vec{p}^T M_{(14,15)}\vec{p})^2 & & C_{circ_{cone_6}}(\vec{p}) = (\vec{i}^T_{32}\vec{p} - \vec{p}^T L_{(14,15,0)}\vec{p})^2
\end{array}$$

The above sets are then grouped in two circular constraints respectively:

$$C_{circ_{cyl}}(\vec{p}) = \sum_{k=1}^{6} C_{circ_{cyl_k}}(\vec{p}) = 0$$

$$C_{circ_{cone}}(\vec{p}) = \sum_{k=1}^{6} C_{circ_{cone_k}}(\vec{p}) = 0$$

Multi-quadric object 2

The orthogonality constraints (4 and 5) between planes are formulated as:

$$\vec{n_1}^T\vec{n_2} = \vec{n_1}^T\vec{n_5} = \vec{n_2}^T\vec{n_5} = 0$$

which can be written the following vector formulation.

$$\begin{aligned}
C_{angl_1}(\vec{p}) &= (\vec{p}^T L_{(1,6,2)}\vec{p})^2 = 0 \\
C_{angl_2}(\vec{p}) &= (\vec{p}^T L_{(1,11,2)}\vec{p})^2 = 0 \\
C_{angl_3}(\vec{p}) &= (\vec{p}^T L_{(6,11,2)}\vec{p})^2 = 0
\end{aligned}$$

and grouped into a single angle constraint function

$$C_{angl}(\vec{p}) = \sum_i^3 C_{angl_i}(\vec{p}) = 0$$

The 6^{th} constraint can be expressed according to

$$d_1 + d_3 = d_2 + d_4$$

and then formulated by:

$$C_{dist}(\vec{p}) = (\vec{i}^T_{(4,5,9,10)}\vec{p})^2 = 0$$

Since we assume that the cylinder axis is parallel to the planes S_1, S_2, S_3, S_4, the distance from the cylinder axis to one of these planes could be defined as the distance from one particular point X_o of the axis and the given plane. The 8^{th} constraint can be formulated by:

$$d(X_o, S1) = d(X_o, S3), \qquad d(X_o, S2) = d(X_o, S4)$$

Taking into account that S_1, S_3 have opposite orientation as well as S_2, S_4, these equations can be written as:

$$X_o^T\vec{n_1} + d_1 = -X_o^T\vec{n_1} + d_3, \qquad X_o^T\vec{n_2} + d_2 = -X_o^T\vec{n_2} + d_4$$

leading to:

$$2X_o^T\vec{n_1} + d_1 - d_3 = 0, \qquad 2X_o^T\vec{n_2} + d_2 - d_4 = 0$$

By considering X_o as defined in Section "The cylinder" and by using the set of equations (9) and (15) the last equations are written as:

$$-2[u,v,w]^T\vec{n_1} + d_1 - d_3 = 0, \qquad -2[u,v,w]^T\vec{n_2} + d_2 - d_4 = 0$$

where u, v, w are the cross coefficients of the cylinder equation. A vector formulation of these equations is then given by:

$$C_{axe\_pos_1}(\vec{p}) = (-2\vec{p}^T L_{(1,22,2)}\vec{p} + i^T_{4,-5}\vec{p})^2 = 0$$
$$C_{axe\_pos_2}(\vec{p}) = (-2\vec{p}^T L_{(6,22,2)}\vec{p} + i^T_{9,-10}\vec{p})^2 = 0$$

The cylinder axis position constraint is then:

$$C_{axe\_pos}(\vec{p}) = C_{axe\_pos_1}(\vec{p}) + C_{axe\_pos_2}(\vec{p}) = 0$$

The cylinder circularity constraint (9^{th} constraint) is implicitly defined by the equations (15) and taking into account the constraint 7 which assumes that the cylinder axis is the same as normal $\vec{n_5}$ these equations are written as:

$$\begin{aligned}
C_{circ_{cyl_1}}(\vec{p}) &= (\vec{i_{16}}^T\vec{p} + \vec{p}^T U_{(11,11)}\vec{p} - 1)^2 & C_{circ_{cyl_4}}(\vec{p}) &= (\vec{i_{19}}^T\vec{p} + \vec{p}^T L_{(11,12,0)}\vec{p})^2 \\
C_{circ_{cyl_2}}(\vec{p}) &= (\vec{i_{17}}^T\vec{p} + \vec{p}^T U_{(12,12)}\vec{p} - 1)^2 & C_{circ_{cyl_5}}(\vec{p}) &= (\vec{i_{20}}^T\vec{p} + \vec{p}^T L_{(11,13,0)}\vec{p})^2 \\
C_{circ_{cyl_3}}(\vec{p}) &= (\vec{i_{18}}^T\vec{p} + \vec{p}^T U_{(13,13)}\vec{p} - 1)^2 & C_{circ_{cyl_6}}(\vec{p}) &= (\vec{i_{21}}^T\vec{p} + \vec{p}^T L_{(12,13,0)}\vec{p})^2
\end{aligned}$$

grouped then into a single constraint:

$$C_{circ}(\vec{p}) = \sum_{k=1}^{6} C_{circ_k}(\vec{p}) = 0$$

The 10^{th} constraint is satisfied if the center of the sphere (21) satisfies the cylinder axis equation (11). We can show easily that by assuming the constraint 7, by using the set of equations (9) and (15) and by requiring the coefficient a of the sphere to be unit that the equation (11) leads to the following equations:

$$\begin{aligned}
n_{x5}(v_s - v_c) &= n_{y5}(u_s - u_c) \\
n_{x5}(w_s - w_c) &= n_{z5}(u_s - u_c) \\
n_{y5}(w_s - w_c) &= n_{y5}(v_s - v_c)
\end{aligned}$$

where u_s means the coefficient u related to the sphere equation, etc. Thus, these equations can be written using the vector form:

$$\begin{aligned}
(\vec{p}^T T_{(11,12,22,23)}\vec{p})^2 &= 0 \\
(\vec{p}^T T_{(11,13,22,24)}\vec{p})^2 &= 0 \\
(\vec{p}^T T_{(12,13,23,24)}\vec{p})^2 &= 0
\end{aligned}$$

The constraint 10 can then be stated as:

$$C_{sph_{center}}(\vec{p}) = (\vec{p}^T T_{(11,12,22,23)}\vec{p})^2 + (\vec{p}^T T_{(11,13,22,24)}\vec{p})^2 + (\vec{p}^T T_{(12,13,23,24)}\vec{p})^2 = 0$$

The 11^{th} constraint is imposed by equating the sphere radius equation (22) to the cylinder radius equation (13). Requiring again the coefficient a of the sphere to be unit and by using the set of equations (9) and (15) this equality can be written using the vector form:

$$C_{equ_{radius}}(\vec{p}) = (\vec{i}^T_{(25,30)}\vec{p} + \vec{p}^T U_{(27,29,22,24)}\vec{p})^2 = 0$$

The 12^{th} constraint imposes a fixed position of the four plane surfaces S_1, S_2, S_3, S_4 with respect to the cylinder axis. It is formulated as:

$$\sqrt{2}(d_1 + d_3) = 2r_{cylinder}$$

By squaring this equation and by using the set of equations (9) and (15) it can be written as:

$$(d_1 + d_3)^2 = 2(u_c^2 + v_c^2 + w_c^2 - d_c^2)$$

Thus this constraint can be put using the vector form:

$$C_{median}(\vec{p}) = (\vec{p}^T(I_{(4,1)} - 2U_{(22,24)})\vec{p} + 2\vec{i}_{25}^T\vec{p})^2 = 0$$

Appendix 3: Levenberg-Marquadt algorithm

Here are the main steps of the Levenberg-Marquardt algorithm applied to a simple optimization function:

$E(\vec{p}) = F(\vec{p}) + C(\vec{p})$

$\alpha = \alpha_0$ % initialization
$E_{decrease} =$ big value

while $E_{decrease} > \epsilon$ % a threshold

Do G_E=Grad$(E(\vec{p}) = \frac{\partial}{\partial \vec{p}}(E(\vec{p}))$

Loop: H_E=Hessian$(E(\vec{p})) = \frac{\partial^2}{\partial^2 \vec{p}}(E(\vec{p}))$
$H_E = H_E + \alpha(diag(H_E))$
solve $H_E \vec{\delta p} = -G_E$
$\vec{p}_{updated} = \vec{p} + \vec{\delta p}$
$E_{decrease} = E(\vec{p}_{updated}) - E(\vec{p})$

if $E_{decrease} > 0$
increase α
go to Loop
else
$\vec{p} = \vec{p}_{updated}$
decrease α

end if
end while

Here a simple example of an optimization function and its derivatives:
$E(\vec{p}) = F(\vec{p}) + C(\vec{p})$
$F(\vec{p}) = \vec{p}^T \mathcal{H} \vec{p}$ the least squares function
$C(\vec{p}) = \lambda(\vec{p}^T A \vec{p} - 1)^2$ the weighted constraint function

$$
\begin{aligned}
G_E &= 2\mathcal{H}\vec{p} + 4\lambda A\vec{p}(\vec{p}^T A \vec{p} - 1) \\
H_E &= 2\mathcal{H} + \lambda[4(\vec{p}^T A \vec{p} - 1)A^T + 8(A\vec{p})(A\vec{p})^T]
\end{aligned}
$$

From this example we can notice the usefulness of the matrix formulation: the optimization function is compact, its derivatives are easy to compute using elementary matrix algebra rules and all the data terms are encapsulated into $\mathcal{H}$ (which needs to be calculated only once).

Appendix 4

Solving the linear system $H_E\vec{\delta p} = -G_E$ in the Levenberg-Marquardt algorithm has numerical perturbations due to the ill-conditioned matrix H_E for large values of λ_k. The key of the solution proposed to overcome this problem consist in splitting the system in two subsystems. The matrix associated with one of the subsystems will hold the matrix components which are sensitive to λ_k variations, the other matrix will hold the components which are not. Thus, both of the matrices will be well-conditioned. The two systems will be then solved consecutively and separately.

Let set the coefficient α in Levenberg-Marquardt algorithm to zero without loose of generality. The system $H_E\vec{\delta p} = -G_E$ could be written more explicitly as:

$$(L + R^T DR)\vec{\delta p} = -G_E \tag{62}$$

where

$$\begin{aligned}
L &= 2\mathcal{H} + 4\sum_{k=1}^{M} \lambda_k C_k(\vec{p}) A_k \\
R^T &= \left[\frac{\partial C_1}{\partial \vec{p}} \; \frac{\partial C_2}{\partial \vec{p}} \cdots \frac{\partial C_M}{\partial \vec{p}}\right] \\
D &= \begin{bmatrix} 2\lambda_1 & 0 & \cdots & 0 \\ 0 & 2\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 2\lambda_M \end{bmatrix} \\
G_E &= 2\mathcal{H}\vec{p} + R^T \nabla_C \\
\nabla_C &= [2\lambda_1 C_1(\vec{p}), 2\lambda_2 C_2(\vec{p}), \ldots 2\lambda_M C_M(\vec{p})]^T
\end{aligned} \tag{63}$$

As mentioned, the matrix L is well behaved since its condition number remains stable when the values of λ_k increase, whereas the condition number of $R^T DR$ increases with λ_k.

Consider the matrix $S = D^{-1}(RR^T)^{-1}R$. By multiplying equation (62) on both sides by S we get a system of M equations:

$$(SL + R)\vec{\delta p} = -SG_E \tag{64}$$

when λ_k values increase and become large $\|S\|$ tends towards zero whereas $\|R\|$ remain stable since it is independent of λ_k so we get $\|SL\| \ll \|R\|$ and thus the system (64) can be approximated by

$$R\vec{\delta p} = -SG_E \tag{65}$$

So now with this system of M equations and N (size of $\vec{\delta p}$) unknowns we can extract M components of $\vec{\delta p}$. The rank of R is equal to M so we can find an orthogonal matrix Q such:

$$QR^T = \begin{bmatrix} U \\ [0] \end{bmatrix} \tag{66}$$

where U is an (M, M) upper triangular non-singular matrix.

Since $QQ^T = I$, (65) can be written as

$$RQ^T Q\delta\vec{p} = -SG_E \tag{67}$$

By splitting $Q\delta\vec{p}$ into $[\delta\vec{z}_1, \delta\vec{z}_2]$ where $\delta\vec{z}_1$ and $\delta\vec{z}_2$ have a size of respectively M and $N - M$ we get from (67)

$$U^T \delta\vec{z}_1 = -SG_E \tag{68}$$

and then $\delta\vec{z}_1$ could be deduced from this equation. Now it remains to compute $\delta\vec{z}_2$.

Consider the matrix V whose columns are the basis of the null space of R. We have $RV = [0]$. By multiplying (62) by V^T we get:

$$V^T L\delta\vec{p} = -V^T G_E \tag{69}$$

Now since $RV = RQ^T QV = [0]$ by using (66) and splitting QV into $[J_1^T, J_2^T]^T$, where J_1 and J_2 are respectively (M, M) and $(N - M, M)$ matrices we get

$$[U^T, [0]^T] \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} = [0]$$

This implies that $J_1 = [0]$ since U is non singular and J_2 could be set to an arbitrarily value say I. Then we can set $QV = [[0]^T, I^T]^T$

The system (69) can be written:

$$\begin{aligned} V^T Q^T QLQ^T Q\delta\vec{p} &= -V^T G_E \\ (QV)^T (QLQ^T) Q\delta\vec{p} &= -V^T G_E \\ \left[[0]^T, I^T \right] QLQ^T \begin{bmatrix} \delta\vec{z}_1 \\ \delta\vec{z}_2 \end{bmatrix} &= -V^T G_E \end{aligned}$$

If we denote the matrix QLQ^T by W such as: $W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$ we get:

$$[[0]^T, I^T] \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} \delta\vec{z}_1 \\ \delta\vec{z}_2 \end{bmatrix} = -V^T G_E$$

from which we extract the system

$$W_{22}\delta\vec{z}_2 = -V^T G_E - W_{21}\delta\vec{z}_1 \tag{70}$$

and δz_2 can be then computed.

The computation of the term SG_E in (68) is expensive. Practically it is faster to use a simplified expression. From (63) we get

$$\begin{aligned} SG_E &= D^{-1}(RR^T)^{-1}R(2\mathcal{H}\vec{p} + R^T\nabla_C) \\ &= D^{-1}(2(RR^T)^{-1}R\mathcal{H}\vec{p} + \nabla_C) \\ &= D^{-1}(2(RQ^TQR^T)^{-1}RQ^TQ\mathcal{H}\vec{p} + \nabla_C) \\ &= D^{-1}(2[U^{-1},[0]]Q\mathcal{H}\vec{p} + \nabla_C) \end{aligned} \tag{71}$$

By splitting $Q\mathcal{H}\vec{p}$ into $[\vec{l_1}^T, \vec{l_2}^T]^T$ where $\vec{l_1}$ and $\vec{l_2}$ have respectively sizes of M and $N-M$ we get

$$SG_E = D^{-1}(2U^{-1}\vec{l_1} + \nabla_C) \tag{72}$$

and $\delta\vec{z_1}$ can be then computed with:

$$U^T\delta\vec{z_1} = -D^{-1}(2U^{-1}\vec{l_1} + \nabla_C) \tag{73}$$

Similarly the expression of V^TG_E in (70) can be simplified :

$$\begin{aligned} V^TG_E &= V^T(2\mathcal{H}\vec{p} + R^T\nabla_C) \\ &= 2V^T\mathcal{H}\vec{p}, \quad \text{since } RV = [0] \\ &= 2V^TQ^TQ\mathcal{H}\vec{p} \\ &= 2[[0]^T, I^T]\begin{bmatrix} \vec{l_1} \\ \vec{l_2} \end{bmatrix} \\ &= 2\vec{l_2} \end{aligned} \tag{74}$$

and the computation of δz_2 is the performed with the following system

$$W_{22}\delta\vec{z_2} = -2\vec{l_2} - W_{21}\delta\vec{z_1} \tag{75}$$

Once $\delta\vec{z_2}$ and $\delta\vec{z_1}$ are computed the $\vec{\delta p}$ vector is deduced with

$$\vec{\delta p} = Q^T\delta z \tag{76}$$

To recapitulate, the resolution of the equation $H_E\vec{\delta p} = -G_E$ in the Levenberg-Marquardt algorithm has to be performed through the following steps :

1) Compute D, ∇_C, R.

2) Compute Q and U from R using elementary geometric transformation (e.g. Householder transformation [29](p.224)).

3) Compute $Q\mathcal{H}\vec{p}$ and extract $\vec{l_1}^T$ and $\vec{l_2}$.

4) Compute $\delta\vec{z_1}$ from (73).

5) Compute $W = QLQ^T$ and extract W_{22} and W_{21}.

6) Compute $\delta\vec{z_2}$ from (75).

7) Compute $\vec{\delta p}$ from (76).