

LARGE SCALE REVERSE ENGINEERING WITH CONSTRAINTS

Yuri Belsky, Arie Karniel, Yoram Reich¹
Department of Solid Mechanics, Materials and Systems
Faculty of Engineering
Tel Aviv University
Tel Aviv 69978
Israel
{yuri, ariek, yoram}@eng.tau.ac.il

This paper presents a solution method for large-scale reverse engineering problems with geometry constraints. A large problem is decomposed into sequential manageable sub problems. The result of each solution step is an optimal trade-off between the set of constraints and the fitting of the surfaces to the measured points. This overall solution process trades off solution quality with manageable complexity of the problem. The presented approach will enable solving practical problems with hundreds of surfaces and constraints.

INTRODUCTION

Reverse engineering (RE) is typically the transformation process of a real object to a 3D surface or solid model. It is often necessary when the original documentation and drawings are not available or when the design process relies on real 3D objects. Traditional RE processes create geometric models based on Cloud of Points (COP), obtained by means of a 3D scanning device (e.g. 3D laser scanners, coordinate measure machines, etc.) (Varady *et al.*, 1997). Various commercial CAD related tools provide specific capabilities for supporting RE. However, knowing the design intent or even good engineering practice often allows imposing constraints on the sought solution (parallelism, co-linearity, etc.).

Constraints could be extracted automatically from the scanned data (Langbein, *et al.*, 2001; Benko, *et al.*, 2002); constraints could be predefined for a set of feature types (Thompson *et al.*, 1999); or the user could interactively set them.

Solving three-dimensional (3D) constrained geometric problems is complicated as such problem could have multiple solutions or be unsolvable. Solving these problems numerically, e.g., by optimization, is sensitive to the problem parameters and especially to the initial guess of the solution. Many techniques can be applied for solving the constrained RE problem, but most approaches described in the literature refer to quite small examples (Langbein, *et al.*, 2001; Benko, *et al.*, 2002; Thompson *et al.*, 1999; Werghi *et al.*, 1998, 2002). The techniques described would face difficulties when trying to scale up to handle practical problems. A common mechanical part is defined by dozens to hundreds geometry objects (e.g. surfaces) and may have hundreds to thousands constraints (order $O(n^2)$, where n is the number of geometry objects). Thus, solving a large-scale constraints problem is critical for industrial use of such processes.

¹ Corresponding author

In the method we present, a large problem is decomposed into simpler manageable problems. For large problems, this decomposition enables achieving a solution compared to no solution in other approaches; for medium size problems, we expect improved performance; while for small problems, we expect worse performance due to the overhead in the calculations.

Overview

The rest of this paper is structured as follows. First, we review previous work on exploiting geometry constraints. Then we present our proposed method of solving the problem of geometric fitting under a set of constraints. The novelty lies in the decomposition of the problem to subproblems while maintaining the constraints prioritization. An example is demonstrated to illustrate the methodology. Finally, the results are discussed and conclusions are drawn.

RELATED WORK

Identifying Constraints

In general, geometry constraints may be derived from various sources. The user can set them manually, or they can be extracted from the scanned model automatically. The latter approach was investigated in several studies. Miles *et al.* (2001) conducted a brief research for estimating frequencies of geometric regularities in simple mechanical parts. The results were used to restrict the number of the automatically extracted constraints from a model. Langbein *et al.* (2001) extended the above approach to finding shape regularities in terms of similarities between properties of B-rep model elements (e.g. approximately equal radii of cylinders or finding a special value, like an integer, which is approximately equal to the average radius). The conclusions were that the best result of such approach might be a figure of merit associated with each constraint, assessing its likely validity.

Typically, automatic constraints identification will result in redundant or contradictory constraints, while some constraints will be missed. The required time to check all the possible combinations may be unreasonable. Furthermore, automatically derived constraints may not be appropriate for a given part. For example, it might be considered that intersecting faces at an angle of about 90 (deg) should be orthogonal. However, small draft angles could have been added to a design to allow a cast component to be removed from a mould. Our approach is that constraints might be automatically extracted as reference, but the user should control the process and approve them.

Constraints in reverse engineering

Numerous studies consider constraints in the context of fitting free-form surfaces. A brief review is given in (Benko, 2002). Werghi *et al.* (1998) considered the simultaneous recovery of multiple quadric surfaces from range data under constraints, which may constrain a quadric to be of a particular surface type, or may relate the orientations of two surfaces. They assume that the constraints are known in advance. Their basic approach is to minimize a function containing a least-squares term and a penalty term associated with the constraints. Optimization is based on a sequential unconstrained programming technique, where a weight is allocated to each constraint; the weights are incremented sequentially to force the constraints to tend to zero. At each iteration, minimization is performed using Levenberg–

Marquardt technique. They noted problems with this approach, which include complex formulation of the constraint function, heavy reliance on the convexity of the constraint space, and the need for accurate initial estimates of the solution. To some extent, these are addressed by an alternative approach using genetic algorithms is reported in (Robertson et al., 1999), although at the expense of speed. The solution process was enhanced in Werghi *et al.* (2002) The basic idea is to attach different penalty functions to the objective function in such a way that the optimal solutions of successive unconstrained problems approach the optimal solution of the problem

PROPOSED METHOD

We assume a given set of geometry constraints (obtained from the user or other sources), a given set of fitted surface parameters and the cloud of points. The problem we solve is to optimally modify the surfaces (and auxiliary geometry objects) parameters, according to the constraints set, while keeping the solution close to the points. Solving the problem consist two sub-stages: creating a “Solution scheduling” plan, which is the decomposition of the problem into solution steps; and solving the problem, optimally modify the surfaces parameters according to measured data and the constraints.

Creating solution schedule

A constraint can be applied to surface parameters (self constraint, e.g. radius or surface type) or to geometry relations. Geometry relation constraints are set pairwise between the geometry objects (more complicated constraints can be separated to pairs using auxiliary geometry). They can be set automatically or interactively by the user. A relation constraint may have an assigned direction (i.e. a constraints from A to B implies that the solution of B is based on the solution of A). A constraint may have tolerance or it may be rigid (zero tolerance). A Priority Value based on the constraint tolerance is assigned to each constraint. The constraint directions in conjunction with the priority value are used in setting the solution plan.

The constraints set is modeled by a “semi directed graph” (see Figure 3). In this graph, geometry objects (e.g., surfaces) are considered as nodes, and the constraints are considered as edges. Directed arrows represent constraints with assigned direction. Non-directed constraints are represented by an edge without direction. There is no limit on the number of constraints between the geometry objects. The resulting graph representation is used for finding connected sub graphs. If separate sub graph are found, each sub graph becomes a distinct sub problem that can be solved independently. In some cases, if the constraints set is sparse, this primary stage can considerably reduce the solution complexity.

The graph representation of each sub graph is converted into a modified Design Structure Matrix (DSM). DSM based algorithms are utilized for further decomposition of the sub problems. The DSM approach facilitates mapping of interdependencies between elements (Eppinger, 1991. Eppinger et al. 1994), and has become a common modeling tools for systems (see review by Browning, 2001). In our work, DSM is applied to the geometry elements and the constraints interdependency set. Existing DSM algorithms can be divided into two categories: Clustering methods for grouping elements with undirected links, and Partitioning (or sequencing) methods for ordering and grouping directed links. The novel optimization algorithm in our work combines elements of both approaches for clustering the

geometry elements based on the “semi directed” constraints set. Each cluster is solution step and the clusters order defined precedence order, thus the resulting outcome is a “solution scheduling” plan.

An initial solution is computed based on Partitioning algorithm (Steward, 1981), which is applied to the directed constraints. Additional clustering is based on minimization of the following cost function:

$$\sum_{ic} A_{ic,ic} + \sum_{id} A_{id,id} D_{id}^P + \sum_c N_c (F \sum_{ic,jc} A_{ic,jc} + C \sum_{ic,jc} A_{ic,jc}) + \sum_{id,jd} (F \sum_{id,jd} A_{id,jd} + C \sum_{id,jd} A_{id,jd}) D_{id,jd}^P \quad (1)$$

whose elements explanation follow. The cost function is summed over the DSM with the following rules:

An element has a self-priority value A_{ii}

- If an element appears only once in the clusters its self-priority value is summed
- If the element has multiple entries, it is multiplied by the distance D_i raised to the power P , D_i^P . D_i is the sum of clusters between the first and last appearance of the element.

A constraint between elements i and j has the priority value A_{ij} .

- If elements i and j are in the same cluster, A_{ij} is multiplied by: F – Forward constant, in case $i > j$, or by C – Close loop constant if $i < j$, and by the cluster size N_c .
- If elements i, j are not in the same cluster, in addition to F and C constants, the priority value is multiplied by D_{ij} raised to the power P . D_{ij} is the size of block i and j create ($D_{ij} = \text{abs}(i - j) + 1$).

P is a size-limiting factor. In the case of no close loops chains, the size of clusters is limited by 2^P , (i.e. if $N_c > 2^P$ the cost function is lower for non-cluster solution than for a clustered solution. In this work, P was arbitrary selected as $P=2.4$, ($2^{2.4} \approx 5.3$), thus limiting non close loop chains clusters to the size of $N_c < 5.3$)

The optimization of the function is carried out with simulated annealing (Kirkpatrick, 1983).

Optimal constraints solution

At each solution step, according to the “solution schedule”, a cluster of surfaces is solved. (i.e. the surfaces parameters are optimized according to the constrains applied and the COP). The solution incorporates the constraints between surfaces in the cluster and constraints from previously solved clusters. Each solution step is modeled as a non-linear constrained optimization problem. The algorithm used is similar to the one described by Werghi *et al.* (2002). The objective function includes least square terms of surfaces distances from measured points and penalty functions terms controlling the constraints satisfaction. A novel addition to the solution process is the use of “rigid body” concept. Solved clusters are considered as one “rigid body” with 6 degrees of freedom (DOF). If several solved clusters are linked they are all considered as one “rigid body.” These DOF enable additional limited changes of the existing solution in the new solution step. This helps improve the overall fitting of the model surfaces by redistributing the error, accumulated during the sequential fitting process.

We wish to find a minimum of the objective function G by sequentially attempting to satisfy the constraints in priority order according to the solution-scheduling plan.

The objective function G for a model consisting of N surfaces, constrained by M positive constraints $C(k)$ is defined as:

$$G(k, p) = \sum_N d(p, k) + \mu \sum_M C(k) \quad (2)$$

where μ is a penalty term.

This function includes two terms. One term is a sum of distance functions between each surface in a cluster (described by a vector of surface parameters k) and its corresponding points $p(x, y, z)$. Each distance function is a dot product of two vector valued functions $S(k)$ and $P(p)$ (of the same dimension):

$$d(p, k) = S^T(k) \cdot P(p) = P^T(p) \cdot S(k) \quad (3)$$

For plane with normal $[k_1, k_2, k_3]$ and distance from origin k_4 :

$$S(k) = (k_1, k_2, k_3, k_4) \quad (4)$$

and

$$P(p) = (x, y, z, 1) \quad (5)$$

For a cylinder with axes defined by the unit vector $[k_1, k_2, k_3]$, nearest to the origin point on this axes $[k_4, k_5, k_6]$, and radius $[k_7]$:

$$S(k) = \left(\frac{(1-k_1^2)}{2k_7}, \frac{(1-k_2^2)}{2k_7}, \frac{(1-k_3^2)}{2k_7}, -\frac{k_1k_2}{k_7}, -\frac{k_1k_3}{k_7}, -\frac{k_2k_3}{k_7}, -\frac{k_4}{k_7}, -\frac{k_5}{k_7}, -\frac{k_6}{k_7}, \frac{k_4^2 + k_5^2 + k_6^2 - k_7^2}{2k_7} \right) \quad (6)$$

and

$$P(p) = (x^2, y^2, z^2, xy, xz, yz, x, y, z, 1) \quad (7)$$

The program used for evaluating our approach was implemented in MATLAB and employed its standard optimization routine *fminunc* for unconstrained optimization.

EXAMPLE

The example we use is a hinge part with 17 surfaces (planes and cylinders), Figure 1. Constraint set (17 constraints) is applied to the part, than 8 constraints are added (the last eight in Table 1). The constraints set includes concentric, parallel, perpendicular and distance constraints, which are applicable in this example. The part was duplicated in various transformations to get a complex object shown in Figure 2. The basic set of 25 constraints is duplicated, and additional constraints were added between parts, resulting in 195 constraints (Table 6). For testing, we used simulated point data, created by commercial CAD software. For checking the implication of applying constraints, the simulated points were deviated from the original. Each of the COP representing a surface was rotated around coordinate system axis randomly in the interval $[-1, 1]$ degrees and given displacement with direction vector having random coordinates $[-0.1, 0.1]$ mm. We added normal noise to the points coordinates of the whole COP with standard deviation 0.025 mm and trimmed by absolute maximum value 0.1 mm.

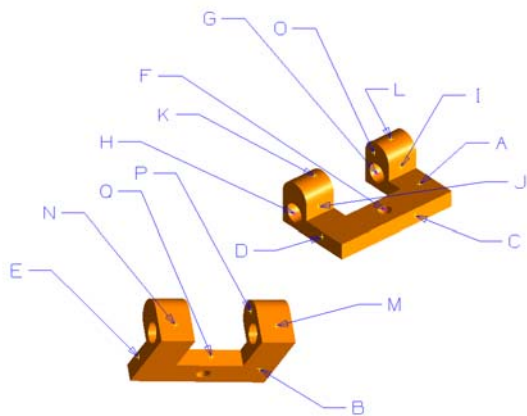


Figure 1: The basic hinge part

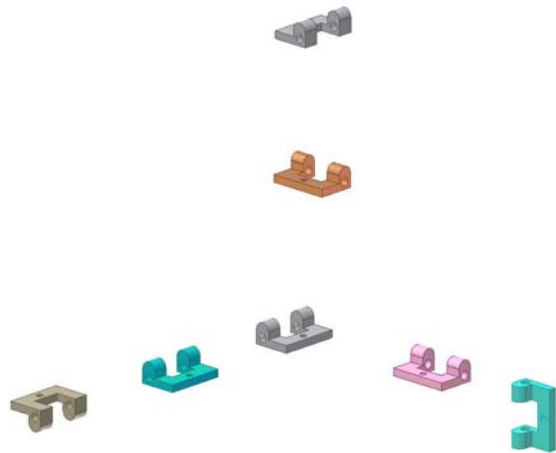


Figure 2: The complex geometric object

Table 1: 17 and 25 constraints set for one part

Constraint type	Element type A	Element type B	Surface A	Surface B	Direction	Constraint value	Units
Concentric	Cylinder	Cylinder	H	G	Don't care	0	mm
Parallel	Cylinder	Plane	H	A	A-B	0	Deg
Parallel	Cylinder-Axis	Plane	H	C	A-B	0	Deg
Perpendicular	Plane	Cylinder-Axis	A	F	A-B	0	Deg
Parallel	Plane	Plane	D	E	Don't care	0	Deg
Parallel	Plane	Plane	E	O	Don't care	0	Deg
Parallel	Plane	Plane	O	P	Don't care	0	Deg
Parallel	Plane	Plane	P	D	Don't care	0	Deg
Perpendicular	Plane	Plane	Q	P	B-A	90	Deg
Perpendicular	Plane	Plane	C	D	Don't care	90	Deg
Distance	Plane	Plane	E	O	Don't care	10	mm
Distance	Plane	Plane	P	D	Don't care	10	mm
Perpendicular	Plane	Plane	B	N	Don't care	90	Deg
Perpendicular	Plane	Plane	B	M	Don't care	90	Deg
Parallel	Plane	Plane	J	M	B-A	0	Deg
Parallel	Plane	Plane	I	N	B-A	0	Deg
Parallel	Cylinder-Axis	Cylinder-Axis	L	K	Don't care	0	Deg

Parallel	Cylinder-Axis	Plane	G	A	B-A	0	Deg
Perpendicular	Cylinder-Axis	Cylinder-Axis	G	F	A-B	90	Deg
Parallel	Cylinder-Axis	Plane	F	C	A-B	0	Deg
Parallel	Plane	Cylinder-Axis	C	G	A-B	0	Deg
Parallel	Plane	Plane	D	P	Don't care	0	Deg
Perpendicular	Plane	Plane	P	Q	Don't care	90	Deg
Perpendicular	Plane	Plane	E	Q	Don't care	90	Deg
Parallel	Plane	Plane	I	J	B-A	0	Deg

The constraints of the single part can be described by the following graph (Figure 3) that includes 3 connected sub graphs. The additional constraints (dotted lines) add connections within the sub graphs. This example illustrates a common engineering practice, assuming

manual setting of the constraints. The user is assumed to set a minimal set of requirement (thus most of the information is based on the measured points).

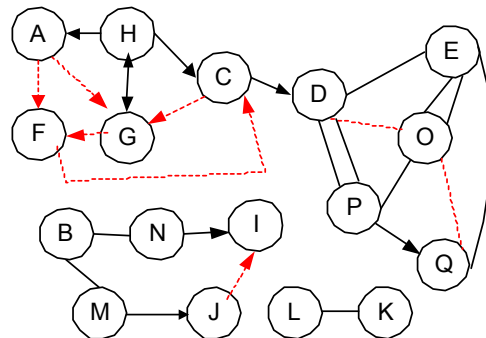


Figure 3: A constraint set semi-directed graph

Optimal clustering results for the two cases (17 and 25 constraints) are presented by DSM representation (Table 2 and Table 3) and by clusters list (Table 4 and Table 5) respectively. The additional constraints yield larger clusters, both in directed case (as H, A, G, F, C) as well as in a non-directed case (D, P, E, Q, O). A partitioning algorithm would result in a cluster of (H, G) in the case of the 17 constraints and the cluster (H, A, G, F, C) in the case of 25 constraints. All other elements would not be clustered (clusters of 1). Clustering algorithms cannot manage the direction of the directed constraints, or the order of clusters. Each constraint can be defined by some equations, depending on its type. For instance, coaxial constraint is defined by 5 equations.

Table 2: Optimal clustering 17 Constraints

	G	H	A	C	F	D	E	O	P	Q	M	J	B	N	I	K	L	
G	10	10																
H	10	10																
A			10	10														
C			10		10													
F				10		10												
D					10		10											
E						10		10										
O							20		10									
P						20		10		10								
Q										10	10							
M												10						
J												10	10					
B												10		10				
N													10		10			
I														10		10		
K																	10	
L																	10	10

Table 3: Optimal clustering 25 Constraints

	H	A	G	F	C	D	P	E	Q	O	B	N	M	J	I	K	L	
H	10		10															
A	10	10																
G	10	10	10		10													
F		10	10	10														
C	10			10	10													
D					10	10												
P						20	10											
E						10		10										
Q							10	10	10									
O						10	10	20	10	10								
B											10							
N												10	10					
M												10		10				
J													10	10				
I														10	10	10		
K																	10	
L																	10	10

Table 4: Clusters lists

Table 5: Clusters lists

Clusters of problem with 17 constraints		Clusters of problem with 25 constraints	
Cluster	Elements	Cluster	Elements
1	H, G	1	A, H, G, F, C
2	A, C, F	2	E, P, O, Q, D
3	D, E, O, P	3	B, M, N, J, I
4	Q	4	K, L
5	M, J, B		
6	N, I		
7	K, L		

We have checked the results of applying the constraints to the COP for one part and for the complex object (multiple parts). Two versions of multiple parts case were checked: without additional constraints between the parts and with constraints (Run 5,6 and 7,8 respectively), in both cases each part has 25 constraints. In each case a parametric surfaces model is created, subject to optimization and trade-off between the COP data and the applied constraints. Constraints satisfaction is measured by the deviation of its equation from zero. A constraint is considered as unsatisfied if its equation value is above a threshold. As described, a parametric model is always obtained; the differences between solutions are in the distribution of tolerances between the constraints. The solution process quality is therefore estimated by the following criteria: max constraint equation value; the number of unsatisfied constraints equations (i.e. that are above a threshold of: $1e-17$). We measure the run time on a Pentium III processor with 392 MB RAM. Comparison is done between solving the whole problem simultaneously (one cluster) versus the decomposition approach, in various cases. The results are summarized in table 6.

Number of iterations in tests 6-8 was smaller (2305 iterations for each cluster; there is an internal calculation that leads to this seemingly odd number) than in tests 1-5 (8355 iterations for each cluster) in order to let the larger problems converge in reasonable time. Penalty factor was set equal for all constraints. Using more intelligent, adaptive penalty functions, apparently would improve the results.

Table 6: Solution process comparison

Run #	No of clusters	No of surfaces	No of constraints	No of constraints equations	Overall runtime, sec	No of equations unsatisfied tolerance $1e-17$	Maximum constraint value
1.	7	17	17	47	671.82	13	9.790e-9
2.	1	17	17	47	378.23	8	4.517e-12
3.	4	17	25	55	436.48	11	8.269e-11
4.	1	17	25	55	415.42	3	1.016e-15
5.	46	119	175	385	4.78e+3	122	3.711e-5
6.	1	119	175	385	5.71e+4	24	3.134e-15
7.	50	119	195	425	1.27e+4	273	1.328e-4
8.	1	119	195	425	6.51e+4	269	2.229e-8

DISCUSSION AND CONCLUSIONS

Noticeably, simultaneous solution generates better satisfaction of constraints. However, for large-scale problems runtime may increase dramatically. While for small models applying the proposed approach could be unjustified, for larger ones this may reduce runtime, while preserving approximately the same degree of constraints satisfaction. For large problems, reduced runtime might mean the distinction between solvable and unsolvable problems. Fitting big clusters first improves the constraints satisfaction results (though the user defined order might not be preserved).

This work presents a method for solving the reconstruction of surfaces subjected to constraints. The scalability achieved is the main advantage of the proposed approach. Sub problems complexity can be bounded, by adding penalty term to the optimization target function. The penalty can be based on the number of surfaces per cluster, as implemented presently, or on the number of modified parameters. The algorithm always tries to satisfy the constraints. The algorithm tries to satisfy inconsistent constraints. In such a case, the constraints may not converge to the tolerance at the end of the optimization. Non-converging constraints should be presented to the user for validity and consistency checking. This implies the need for an iterative process.

The proposed approach allows great flexibility in applying various constraint types. The applied constraints may be geometry relations between the surfaces (or auxiliary geometry objects) or direct constraint on the surface parameters (e.g. radius, distance etc.).

BIBLIOGRAPHY

Benko P., Kos G., Varady T., Andor L., Martin R., Constrained fitting in reverse engineering, *Computer Aided Geometric Design*, 19(3):173–205, 2002.

Browning T. C., Applying the Design Structure Matrix to system decomposition in integrated problems - A review and new directions, *IEEE transaction on Engineering Management*, 48(3):292-306, 2001.

Eppinger S. D., Model-based Approaches to Managing Concurrent Engineering, *Journal of Engineering Design*, 2:283-290, 1991.

Eppinger S. D., Whitney D.E., Smith R. P., Gebala D. A., A model-based method for organizing tasks in product development, *Research in Engineering Design*, 6:1-13, 1994.

Kirkpatrick S., Gelatt C. D., Vecchi M. P., Optimization by Simulated Annealing, *Science*, 220(4598): 671-680, 1983.

Langbein F. C., Mills B. I., Marshall A. D., Martin R. R., Finding Approximate Shape Regularities in Reverse Engineered Solid Models Bounded By Simple Surfaces, In: D. C. Anderson, K. Lee (eds), *Proc. 6th ACM Symp. Solid Modelling and Applications*, pp. 206-215, ACM Press, New York, 2001.

Lukacs G., Marshall A. D., Martin R. R., Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation, in: *Burkhardt, H., Neumann, B. (Eds.), Proc. ECCV 98, Lecture Notes in Computer Science*, 1:671–686. Springer, Berlin, 1998

Mills B. I., Langbein F.C., Marshall A. D., Martin R. R., Estimate of frequencies of geometric regularities for use in RE of simple mechanical components, *Technical Report GVG 2001-1*, Geometry and Vision Group, Department of Computer Science, Cardiff University, 2001. <url: <http://ralph.cs.cf.ac.uk/papers/Geometry/survey.pdf>>

Robertson C., Fisher R.B., Corne D., Werghi N., Ashbrook A.P., Investigating Evolutionary Optimisation of Constrained Functions to Capture Shape Descriptions from Range Data, *Proc. 3rd On-line World Conference on Soft Computing (WSC3)*. Also in: *Advances in Soft Computing - Engineering Design and Manufacturing*, R. Roy, T. Furuhashi and P. K. Chawdhry (Eds), pp. 455—466, Springer, 1999

Steward D.V., *System Analysis and Management: Structure, Strategy and Design*, Petrocelli books, New York, 1981.

Thompson W. B., Owen J. C., de St. Germain H. J., Stark S. R. Jr., Henderson T. C., Feature-based reverse engineering of mechanical parts, *IEEE Transactions on Robotics and Automation*, 15(1): 57–66, 1999.

Varady T., Martin R. R., Cox J., Reverse engineering of geometric models, an introduction, *Computer Aided Design*, 29(4):255-268, 1997.

Werghi N., Fisher R., Ashbrook A., Robertson C., Modeling objects having quadric surfaces incorporating geometric constraint, In *Proceedings of the ECCV'98*, Freiburg, Germany, pp. 185–201, 1998.

Werghi N., Fisher R., Ashbrook A., Robertson C., Shape Reconstruction Incorporating Multiple Nonlinear Geometric Constraints, *Constraints*, 7:117–149, 2002.