Volume xx (200y), Number z, pp. 1–6

Perfecting 3D computer models reconstructed from measured data

I. Kovács, and T. Várady

Budapest University of Technology and Economics

Abstract

The goal of digital shape reconstruction is to create computer representations from measured data. Most of the 3D reconstruction methods give us a good approximation of an ideal model. However, inaccuracies occur due to noisy data and the numerical nature of the algorithms used for fitting the bounding surfaces. In this paper we try to eliminate these inaccuracies and create a "perfect" model for CAD/CAM systems. Our algorithms are based on a method published earlier[2]. This project expands the existing technology in two areas: it automatically sets up hypotheses and selects the most likely constraints instead of requesting this information from the user; it also tries to define global constraints related to the whole object. We determine the optimal axis orientation and an optimal grid division size for the coordinate system in which the object might have been defined; or determine the best - full or partial - axis of symmetries. We have been dealing with planar contours with constraints, however, it will be easy to extend this mathematical apparatus to 3D, as well. The related theoretical and numerical problems are illustrated by several test examples.

1. Introduction

Digital shape reconstruction (reverse engineering) is an expanding, challenging area of Computer Aided Geometric Design[16]. This technology is utilized in various applications where a given physical object is scanned in 3D, and a computer representation is produced in order to perform various complex computations. A wide range of applications emerges in engineering, medical sciences, and to preserve the cultural heritage of mankind[11]. Examples include redesigning and re-manufacturing old mechanical parts, creating surface geometries from clay models, or producing surfaces matching human body parts for hearing aids, dentures, prosthetics, etc.

1.1. Digital shape reconstruction

Digital shape reconstruction consists of the following technical phases: (1) 3D data acquisition (scanning), (2) filtering and merging point clouds, (3) creating triangular meshes, (4) simplifying and repairing meshes, (5) segmentation (partitioning into regions), (6) region classification, (7) fitting surfaces, (8) fitting connecting surfaces (e.g. fillets), *(9) perfecting surfaces (including constrained fitting and surface fairing)*, (10) exporting to CAD–CAM systems for downstream applications.

After the classification we want to determine the best suit surface. Usually we need to fit many surfaces simultaneously. If we fit the surfaces separately, there will be inaccuracies. Let be denote the surfaces by $\{s_i\}$, and the corresponding point clouds are $\{p_{ij}\}$. Our goal is minimize the average square distance between surfaces and the point clouds. The $\mathbf{x}$ consists the parameters of the surfaces. So the problem is

$$f_i(\mathbf{x}) = \sum_j d(p_{ij}, s_i)^2, \qquad f_i(\mathbf{x}) \to \min .$$

The fitting problem of simple surfaces are attributable to eigenvector problems[5], and for more complex surfaces also known effective numerical method[14].

1.2. Constrained fitting

The geometric constraints describes the connections between the objects. The geometric constraints one of the key issues of engineering design, because these describes the orthogonality, parallellability, tangential connections, symmetries, round values etc. We represent the constraints with equations.

(12/2013).

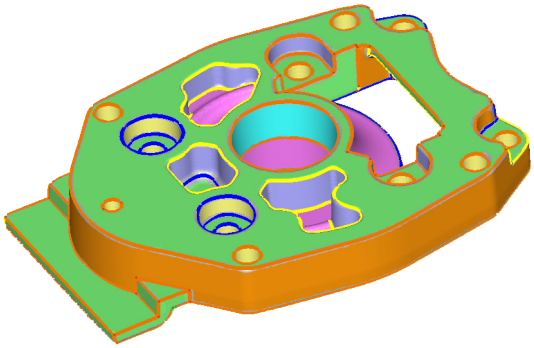

Figure 1: *An engineering object has many constraints.*

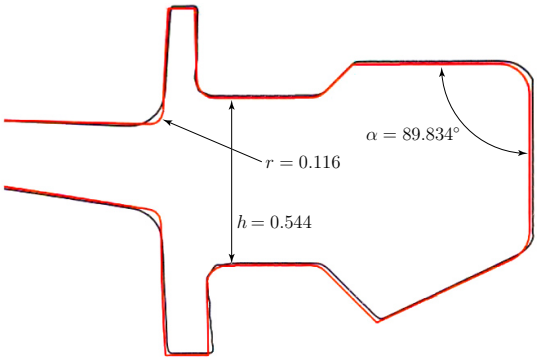

Figure 2: *Constraint inaccuracies due to noise.*

There are simple locally acting constraints (between surfaces, for example lines, circles, planes, cylinders, cones, extruded and rotational surfaces), and more complex globally acting constraints (between surface groups). The commonly occurring local constraints are:

- orthogonal/parallel curves and surfaces
- concentric curves and surfaces
- tangential curves and surfaces
- rounded numerical values
- fix numerical values

The commonly occurring global constraints are:

- common extrusion directions
- common rotational axis
- global grid
- global axis of symmetry
- global rotational symmetry

From the scanning we didn't have informations about the constraints, so we need to add that explicitly, or wee need to recognize it by some algorithm. Due to noise, and scanning inaccuracy, the constraints satisfying only in some tolerance level, see Figure 2. If we have a constraint system, we need to fit the surfaces besides the satisfying constraints. We called this process as *constrained fitting*.

Let $\{s_i\}$ and $\{p_{ij}\}$ as above. The goal is minimize

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

besides the constraint system satisfying. The α_i weights can depend for example on the area of surfaces. Our goal is minimize the average square distance between surfaces and the point clouds. There are numerical methods to solve this problem[18] [13], our methods based on an algorithm by P. Benkő, which uses modified newton method.[2]

1.3. Our project

In the most of reverse engineering systems, the constraints defined explicitly by a user. In fact, many of them can easy recognizable. In this paper we present an algorithm, how to state and evaluate hypothesizes for the possible local geometric constraints.

Furthermore we present methods to detect some full or partial global symmetries, including grids and axis of symmetries.

1.4. Previous work

There are several methods for constrained fitting. Our work based on P. Benkős algorithm[2]. Langbein et al. started to detect partial symmetries on point sets, the method based on algebraic considerations[9]. Another important publication is Mitra et. al work, where they detect partial global symmetries on 3D models by feature points[12].

2. Constrained fitting – basics

Now we present the base steps of P. Benkős algorithm, to help us understanding the new methods.

Consider the profile curve in Figure 3. If we fit the circles independently, the going to wrong by the tangential connections. Therefore the solution is the constrained fitting. In this example, let c_i be the circles with (A_i, B_i, C_i, D_i) parameters, where $A_i(x^2+y^2)+B_ix+C_iy+D_i = 0$ the equations of the circles. The minimalisable average square distance is

$$f(\mathbf{x}) = \sum_{i,j} d_{i,j}^2 = \frac{1}{n_i}\sum_j (A_i(x_{ij}^2+y_{ij}^2)+B_ix_{ij}+C_iy_{ij}+D_i)^2.$$

The constraint system includes the

- normalization constraints ($B_i^2+C_i^2-4A_iD_i = 1$),
- tangential constraints (for adjoint circles $2A_jD_i + 2A_iD_j - B_iB_j - C_iC_j \pm 1 = 0$).

2.1. The numerical method

Let be denote the surfaces by $\{s_i\}$, and the corresponding point clouds are $\{p_{ij}\}$. The goal is minimize the average

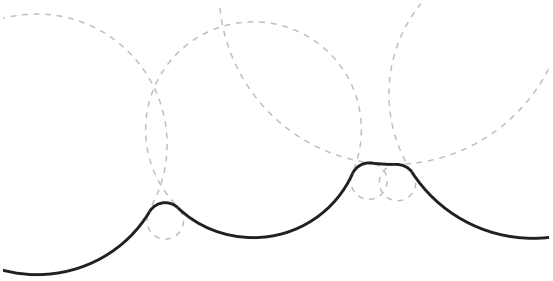

Figure 3: *Profile curve*

square distance between surfaces and the point clouds, such the constraints are satisfied.

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

Where $d(p_{ij}, s_i)$ denotes the distance between the s_i surface and the p point, and α_i are positive weights assigned to each surface. Let denote the constraint equations in form $c_k = 0$, so we can write the globally system of equations

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

So the goal is minimize f besides the condition $\mathbf{c} = 0$.

Let $\mathbf{c}(\mathbf{x}) = (c_1(\mathbf{x}), ..., c_k(\mathbf{x}))$ where the constraints are in priority queue. Suppose that $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$ smooth enough (at least C^2). The method is based on Newton iteration. We approximate $\mathbf{c}$ in first order, and f in second order. Let's see the Taylor approximation of $\mathbf{c}$ and f around $\mathbf{x}_0$.

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

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

By using this equations, the problem locally is in the form

$$C\tilde{\mathbf{d}} = 0 \tag{1}$$

$$\tilde{\mathbf{d}}^T A \tilde{\mathbf{d}} \to \min,$$

where $\tilde{\mathbf{d}} = (d_1, ..., d_n, 1)$, $C = [\mathbf{c}'(\mathbf{x}_0)|\mathbf{c}(\mathbf{x}_0)]$ and A is the following $(n+1) \times (n+1)$ size matrix:

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

To calculate $\tilde{\mathbf{d}}$ we reduce it to a lower dimensional vector $\mathbf{d}^*$ by (1), such that $\mathbf{d}^*$ has an independent coordinates. For this, we calculate M matrix, such $\mathbf{d} = M\mathbf{d}^*$, and $CM = 0$. Now the dimension of $\mathbf{d}^*$ give us, how many independent variables are there in the system. Finally we can solve ${\mathbf{d}^*}^T A^* \mathbf{d}^* \to \min$, where $A^* = M^T A M$, without constraints, which is solvable by a simple system of linear equations.

The way of calculate M is very similar to the Gauss elimination. During the elimination, we can check if some constraints contradict to each other, or if the system is overdetermined.

2.2. Auxiliary objects

For more complicated constraints we need use auxiliary objects. Let us consider the following constraint: three lines meets in one point. Formulate this constraint with equations is bit more complicated. The easier way, if we introduce a point object that each line passing through. Now we call that point as auxiliary object. We note that, to the auxiliary objects never belongs a point set, but its an important parts of the fitting.

3. Automatic detection of local constraints

Let $c(\mathbf{x}) = 0$ a simple constraint between two object. Let ε ba a tolerance level. The c constraint is within the tolerance if and only if $|c(\mathbf{x})| < \varepsilon$, and we want validate the constraint if it's hold. Fo this we introduce the following modified constraint equation:

$$s_\varepsilon(x) := \begin{cases} x & \text{if } |x| < \varepsilon \\ 0 & \text{otherwise.} \end{cases}$$

We observe that, if $c(x)$ is out of the tolerance level, then it vanishes, so it is the constant zero constraint, which causes nothing modification to the fitting. All other cases $s_\varepsilon(x)$ is equal to $c(x)$.

We note that, necessary condition for c, that $c(x)$ is a good representation for the deviation from the considered constraint. For example for the *line meets point* constraint, we must use the line-point distance function

$$c_{pl}(\mathbf{x}) = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}.$$

Also note that, s_ε is not continuous, but piecewise continuous function, so if we calculate numerically the derivative, we need make it piecewise.

To detect constraints automatically, we take the modified constraint for all object pair. The numerical method use only the constraints within the tolerance level. Let denote $S = \{s_i\}$ the set of objects, and $\{c_j\}$ are the constraint types, such $c_j(s_1, s_2)$ denote the concrete constraint between s_1 and s_2. So for all c_j, consider the following constraint set:

$$C_j = \{s_{\varepsilon_j}(c_j)(s_1, s_2) | s_1, s_2 \text{ suitable for } c_j\}.$$

So the global constraint system include the explicitly added constraints, and all C_j-s, the modified constraint sets.

We can handle the more complex locally constraints too.

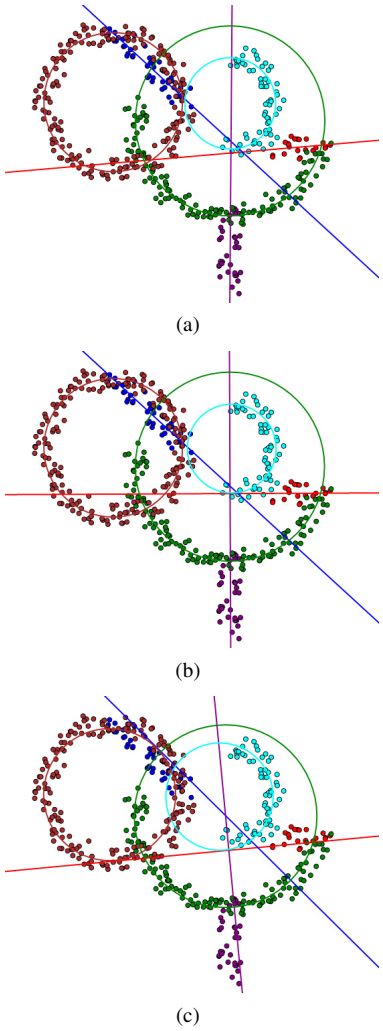

Figure 4: *Automatically detected constraints: (a) initial state; tolerance levels in the (b) and in the (c) cases, the tolerances:* 10/10 *degrees 'lines orthogonal'; 10/10 unit 'line pass the center of circle'; 15/25 unit for the 'line tangent circle' constraint.*

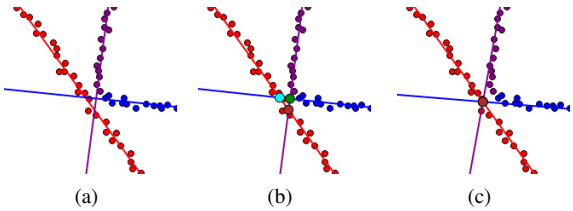

Figure 5: *Three lines meets a common point: (a) initial state; (b) the investigated auxiliary points; and, (c) detect the 'points are equal' constraint.*

We can investigate the corresponding auxiliary objects automatically, and then check the tolerances. Let see the following example: if we want to detect the *three line meets in one point* constraint, we create auxiliary points by the all intersections, with the corresponding *line meets point* constraints, and the algorithm can detect, if the intersection points are enough close to each other. Then we can conclude the *three line meets in one point* is satisfied.

4. Find the best fit global grid

In this section, we investigate the *'grid'* object, the corresponding constraints, and how to detect the best fit global grid to the data points.

The grid is an 5 dimensional auxiliary object, the parameters consists

- the orientation of grid (n),
- the origin of grid (p_0),
- and the positive grid constant (c).

We note that, the parameters are not well defined by the grid, because we can select all intersection points for the origin, and we have four ways to define the orientation.

We also can define constraints with the grid, with the similar way as earlier. For example the equation of the line ($Ax+By+C=0$) is orthogonal/paralell to the grid constraint, is in the form $c(\mathbf{x}) = \min(|An_1 - Bn_2|, |An_2 + Bn_1|) = 0$. We also can define the point meets grid constraint: $\langle p - p_0, n\rangle / c$ and $\langle p - p_0, n^{\perp}\rangle / c$ are integers. The most important constraints with the grid:

- some parameter (n, p_0, c) is fix,
- point pass to the grid,
- line orthogonal/parallel to the grid,
- line pass to the grid.

For detect the above constraints, we can use the automatic detection as wee see earlier, but it works only, if we have an almost perfect initialization of the grid. To detect the best fit global grid, our algorithm has four base steps (see Figure 6):

1. determine the best orientation,
2. fitting the corresponding lines,
3. determine the grid division size,
4. fitting objects to the grid.

4.1. Determine the orientation

Assume that, all lines belongs to a straight section. Let us denote the length of l_i is $h(l_i)$, an the argument of the normal vector of l_i in radian is $\arg(l_i)$. In the terms of grid α and $\alpha + \pi/2$ has the same orientation, so we work with the angles modulo $\pi/2$. Now the solution is given by clustering the points. The radius of clusters depends on the tolerance level. The weight $w(S) = \sum_{l \in S} h(l)$. We select the best cluster (where $w(S)$ is maximal), and the weighted average of angles gives the best orientation.

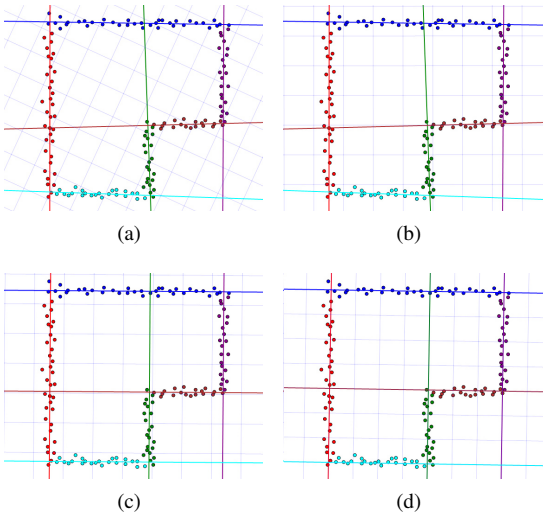

Figure 6: *Detect grid: (a) the initial state (independent fittings); (b) the optimal orientation; (c) re-fitted lines; and (d) the final fitting with the optimal grid constant.*

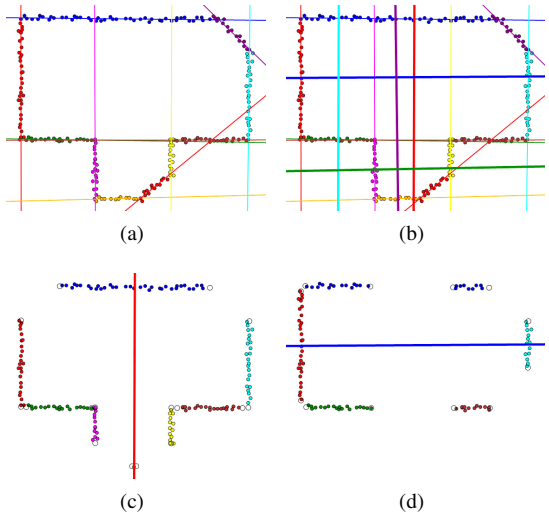

Figure 7: *Detecting axis of symmetry: (a) initial state; (b) axis candidates; (c) the best axis (70%); and (d) the second best axis (41%).*

4.2. Determine the grid division size

After setting up the best oriented grid, we fit the corresponding lines to the orientation, by automatic method, as we have seen in Section 3. Now the problem is find the best approximately common divisor to the distances of lines.

Let us denote the lines fitted to the orientation are $\{l_i\}$, and the all absolute distances between the parallel lines are $\{d_{ij}\} = \{n_l\}$. If d is a suitable grid division size, then the average residue from $\{n_l\}$ is small. The average residue is in the form

$$\delta(d) = \sum_l \min\left(\left\{\frac{n_l}{d}\right\}, 1 - \left\{\frac{n_l}{d}\right\}\right),$$

where $\{x\}$ denotes the fraction part of x.

Now the goal is find the minimum of $\delta(d)$ in the $[d_min, d_max]$ interval. Easy to see, δ function is piecewise monotone, and the monotonicity fails in the numbers in form n_l/k for some n_l, and k positive integer. Therefore we need to find the minimum only in that points, so we can find the minimum in $O(N^2 n_{\max}/d_{min})$ steps, where N is the number of distances, and $n_{\max} = \max_l n_l$.

After setting up the grid division size, we can automatically detect the line pass grid constraints.

5. Estimate global symmetries

Our another examined global constraint is the axis of symmetry. We present an algorithm for line-circle curves. To detect them, first we get all possibly axises (for some reason), evaluate them, and we select the best one. Let denote $P = \{p_i\}$ the endpoints of the sections and centers of circles, and $L = \{l_i\}$ the lines.

The main steps of the algorithm (see Figure 7):

1. Collect the perpendicular bisectors between the points of P, and the angle bisectors between the lines of L. That lines are the *auxiliary lines*.
2. Clustering between the auxiliary lines.
3. Evaluate the clusters (get the corresponding axises and grade the symmetry).
4. Select the best cluster, and the best axis. Constrained fitting.

The *auxiliary lines* A consists the perpendicular bisectors between the points of P: $A_1 = \{\text{PBisector}(p_i, p_j) : i < j\}$, and the angle bisectors between the lines of L: $A_2 = \{\text{ABisector}(l_i, l_j) : i < j\}$.

Now we clustering between the lines of A in two steps. First by the argument of normals (modulo π), then by the distance of the origin. The size of the clusters are not the best measure to the goodness of the symmetry, which shows a several examples. For each cluster C, l_C denote the average of C, that are the *axis candidates*.

By the clusters, we have information about which sections or circles are symmetric by the axis. For each pairs, we determine the arc of symmetric part. The sum of these arcs gives the weight of l_C, and we can select best one.

We also can define constraints for axis of symmetry by auxiliary objects. We use the

- *axis is perpendicular bisector of a section*,
- *axis is angle bisector of two lines*

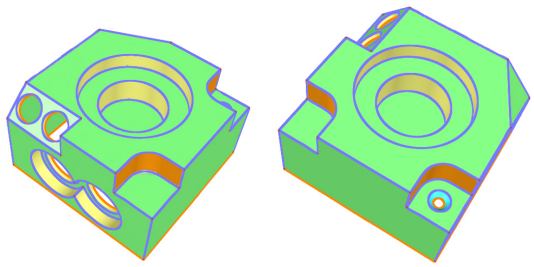

Figure 8: *Segmented object from measured data*

constraints. Finally we make the constrained fitting with the best axis.

6. Conclusion

In our project we worked with planar curves. The presented algorithms possibly extended to 3D. In future we want detect another global symmetries, such as rotational and translational symmetries. Also important way, extending the constrained fitting to free-form surfaces.

Acknowledgements

We would like to thank Péter Salvi, György Karikó and Pál Benkő. The research supported by OTKA (101845).

References

1. Geomagic, Inc., http://www.geomagic.com.
2. P. Benkő, G. Kós, T. Várady, L. Andor, and R. R. Martin. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19(3):173–205, 2002.
3. P. Benkő, R. R. Martin, and T. Várady. Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, 33(11):839–851, 2001.
4. P. Benkő and T. Várady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design*, 2004.
5. I. Coope. Circle fitting by linear and nonlinear least squares. *Journal of Optimization Theory and Applications*, 76(2):381–388, 1993.
6. J. Jiang, Z. Chen, and K. He. A feature-based method of rapidly detecting global exact symmetries in CAD models. *Computer-Aided Design*, 45(8-9):1081 – 1094, 2013.
7. Y. Ke, W. Zhu, F. Liu, and X. Shi. Constrained fitting for 2D profile-based reverse modeling. *Computer-Aided Design*, 38(2):101–114, 2006.
8. F. C. Langbein. *Beautification of reverse engineered geometric models*. PhD thesis, Cardiff University, 2003.
9. M. Li, F. C. Langbein, and R. R. Martin. Detecting approximate incomplete symmetries in discrete point sets. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pp. 335–340. ACM, 2007.
10. G. Lukács, R. R. Martin, and D. Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In *Computer Vision-ECCV'98*, pp. 671–686. Springer, 1998.
11. P. Marks. Capturing a Competitive Edge Through Digital Shape Sampling & Processing (DSSP). *SME Blue Book Series*, 2005.
12. N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006.
13. J. Porrill. Optimal combination and constraints for geometrical sensor data. *The International Journal of Robotics Research*, 7(6):66–77, 1988.
14. H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. In *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pp. 8–25. IEEE, 2002.
15. V. Schomaker, J. Waser, R. t. Marsh, and G. Bergman. To fit a plane or a line to a set of points by least squares. *Acta crystallographica*, 12(8):600–604, 1959.
16. T. Várady and R. R. Martin. Reverse engineering. *G. Farin, J. Hoschek, M. S. Kim, Handbook of Computer Aided Geometric Design, Chapter 26*, Elsevier, 2002.
17. T. Várady, P. Salvi, *3D-s számítógépes geometria és alakzatrekonstrukció tárgy diasorok*, BME IIT (2013)
18. N. Werghi, R. Fisher, C. Robertson, and A. Ashbrook. Modelling objects having quadric surfaces incorporating geometric constraints. In *Computer Vision-ECCV'98*, pp. 185–201. Springer, 1998.