Applying Engineering Constraints in Digital Shape Reconstruction

István Kovács*
Supervised by: Tamás Várady†

Budapest University of Technology and Economics

Abstract

The goal of digital shape reconstruction is to create computer models from point clouds; however, inaccuracies may occur due to the noise of measured data and the numerical nature of the algorithms used for fitting. As a consequence, faces will not be precisely parallel or orthogonal, smooth connections will be of poor quality, axes of concentric cylinders may be slightly tilted, and so on. In this paper we present algorithms to eliminate these inaccuracies and create perfect models, which are suitable for downstream CAD/CAM applications. We extend a formerly published technology [1] in two areas. We propose methods to (i) automatically set up hypotheses for likely geometric constraints and (ii) compute global constraints related to the whole object, such as, an optimal coordinate system and associated grid, or the best - full or partial - axes of symmetries. In this paper we investigate planar contours with constraints; nevertheless, extending this technology to 3D is in progress, as well. A few interesting examples will be presented to show how constrained fitting can improve the quality of reconstructed objects.

Keywords: Reverse engineering, Constrained fitting, Symmetry detection

1 Introduction

Digital shape reconstruction (reverse engineering) is an expanding, challenging area of Computer Aided Geometric Design [12]. This technology is utilized in various applications where a given physical object is scanned in 3D, and a computer representation is needed in order to perform various computations. A wide range of applications emerges in engineering, medical sciences, and to preserve the cultural heritage of mankind [6]. 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.

*kovacsi@math.bme.hu
†varady@iit.bme.hu

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 disjoint regions), (6) region classification, (7) fitting surfaces (i.e. approximating the data points), (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.

Assume segmentation has taken place, and classification produced a surface type for each region that will best approximate the related data points. The conventional approach is to fit surfaces individually. Let us denote the surfaces by $\{s_i\}$, and the corresponding point clouds by $\{p_{ij}\}$. Our goal is to minimize the average square distances between the surfaces and the point clouds. Let $\mathbf{x}$ contain the parameters of the surfaces. Then the problem is

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

Fitting simple surfaces is generally based on solving eigenvalue problems [2] [11]; for more complex surfaces efficient numerical methods exist [9]. Fitting surfaces *separately* is likely to produce inaccuracies; fortunately, the model quality can be perfected, if we recognize and enforce various geometric constraints and then fit groups of related surfaces *simultaneously*.

1.2 Constrained fitting

Geometric constraints define relationships amongst various entities. This is a key issue in engineering design; orthogonality, parallelism, tangency, symmetry, etc. can be best prescribed by means of constraints, which are expressed in the form of various algebraic equations.

We may distinguish between constraints that has *local* effect related to pairs of curves and surfaces, such as, lines, circles, planes, cylinders, cones, extruded and rotational surfaces, and more complex constraints that *globally* determine groups of surfaces. The most frequent local constraints include

- orthogonal/parallel curves and surfaces,

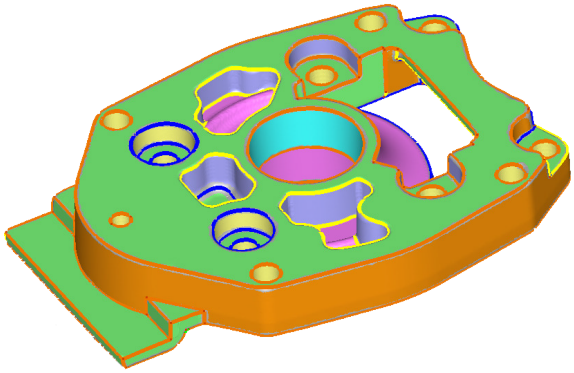

Figure 1: An engineering object with many self-contained constraints.

- concentric curves and surfaces,
- tangential curves and surfaces,
- rounded numerical values,
- fixed numerical values.

The most frequent global constraints include

- common direction for extrusions,
- common rotational axes,
- global grid,
- global axis of symmetry,
- global rotational symmetry.

The scanned data – in itself – do not carry information about the structure of the object and the constraints between its high-level geometric entities. These need to be set either explicitly by the user, or recognized and set by some "intelligent" algorithm. After individual fitting — due to noise and numerical inaccuracies — constraints will be satisfied only within some tolerances; an example with inaccurate values is shown in Figure 2. If we set a constraint system, we can enhance the model and *refit* the surfaces accordingly. This process is called *constrained fitting*.

Our goal is to minimize the average square distances between the point clouds and the surfaces while constraint equations are enforced.

1.3 Previous work

Numerical methods to solve this problem have been published earlier [14] [8]. An important paper on constrained fitting was published by Langbein et al. [3], where partial symmetries on point sets are detected based on an algebraic concept. In the paper of Mitra et al. [7] it is shown how partial global symmetries on 3D models by feature points can be detected. Our paper expands a numerical technique originally suggested by Benkő et al. [1], that can handle under- and over-constrained systems with priorities, applying a special extension of Newton's method. An interesting approach was recently published in [4], that discovers certain primitives, such as, planes, cylinders, cones and spheres using RANSAC method [10] and then sequentially enforces constraints amongst them.

1.4 Outline

In this paper we focus on algorithms, that substitute user driven, manual constrained fitting by automatic techniques. After presenting the basic algorithm of Benkő et al. [1] in Section 2, we present how to detect and enforce various hypotheses for likely *local* geometric constraints in Section 3. Then we continue with methods to detect *global* constraints, including best fit grids and optimal axes of symmetries - see Section 4 and 5, respectively. Finally, results will be illustrated by a few examples using 2D point sets and related constraints for planar curves.

2 Constrained fitting – basics

2.1 A simple example

Consider the profile curve in Figure 2(b) [13]. If we fitted these circles independently, the tangential constraints would not be satisfied, however, constrained fitting provides an appropriate solution. In this example, let c_i denote the circles to be constrained with parameters (A_i, B_i, C_i, D_i) , and the corresponding equations are $A_i(x^2+y^2)+B_ix+C_iy+D_i=0$. The average squared distance to be minimized is

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

The constraint system includes

- normalization constraints $(B_i^2 + C_i^2 - 4A_iD_i = 1)$; these assure that the distances from the points closely approximate the Euclidean distances, and
- tangential constraints $(2A_jD_i + 2A_iD_j - B_iB_j - C_iC_j \pm 1 = 0)$; these assure that each pair of adjacent circular arcs shares a common endpoint and tangent.

2.2 The numerical method

Let us continue with presenting the method of Benkő et al. [1], this is necessary to understand the rest of the paper. We use the previous notations, where $d(p_{ij}, s_i)$ denotes the distance between surface s_i and point p_{ij}. Let α_i be the positive weights assigned to the i-th surface. Let $\mathbf{x}$ contain the parameters of all surfaces, and let us define the

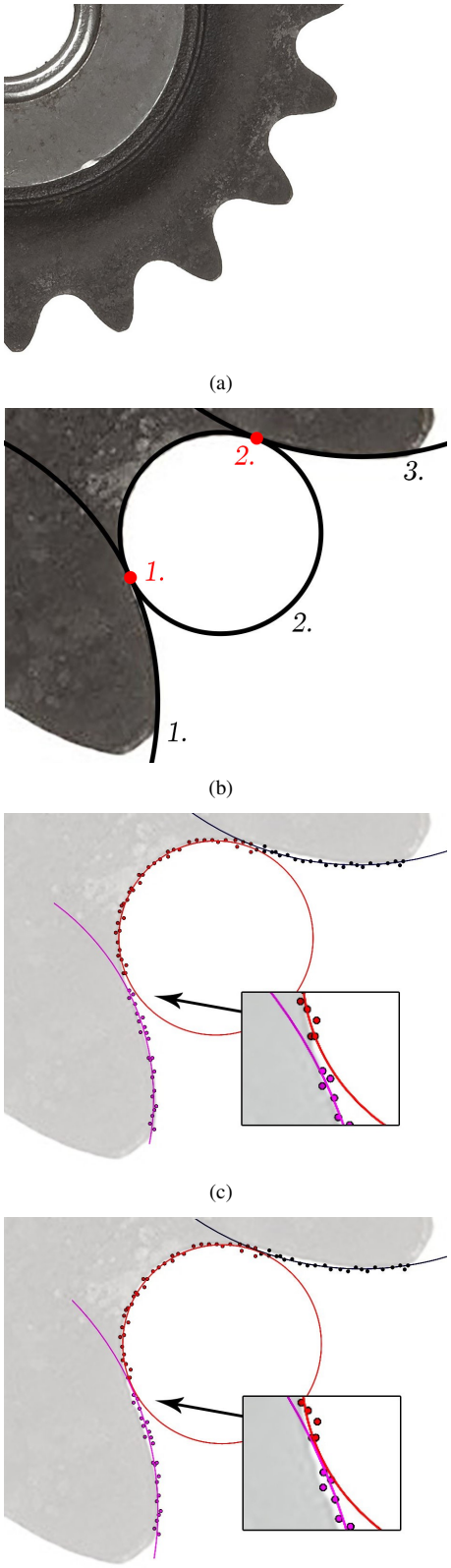

Figure 2: Constrained fitting: (a) profile of a gear wheel; (b) three circles with prescribed tangency; (c) independent fitting yields discontinuity; (d) fitting with constraints guarantees smooth connections.

constraint equations in the form of $c_k = 0$. Then we can write the global system of equations as

$$\mathbf{c}(\mathbf{x}) = 0. \tag{1}$$

We minimize the average square distance while the constraints are satisfied:

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

Let $\mathbf{c}(\mathbf{x}) = (c_1(\mathbf{x}), ..., c_k(\mathbf{x}))$, where the constraints are ordered by priority and suppose that $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$ are smooth enough (at least C^2). Here we have a highly nonlinear system of equations, that we are going to solve using a special Newton iteration. We approximate $\mathbf{c}$ in first order, and f in second order. The Taylor approximations of $\mathbf{c}$ and f around $\mathbf{x}_0$ are the following:

$$\mathbf{c}(\mathbf{x}_0 + \mathbf{d}) \approx \mathbf{c}(\mathbf{x}_0) + \mathbf{c}'(\mathbf{x}_0)\mathbf{d} \tag{3}$$

$$f(\mathbf{x}_0 + \mathbf{d}) \approx f(\mathbf{x}_0) + f'(\mathbf{x}_0)\mathbf{d} + \frac{1}{2}\mathbf{d}^T f''(\mathbf{x}_0)\mathbf{d} \tag{4}$$

In each step we want to determine a small difference vector $\mathbf{d}$. Using the above equations, the problem can be written locally in the form

$$C\tilde{\mathbf{d}} = 0 \tag{5}$$

$$\tilde{\mathbf{d}}^T A \tilde{\mathbf{d}} \to \min, \tag{6}$$

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

$$A = \left[\begin{array}{c|c} f''(\mathbf{x}_0) & f'(\mathbf{x}_0) \\ \hline f'(\mathbf{x}_0)^T & 0 \end{array}\right]. \tag{7}$$

In order to calculate $\tilde{\mathbf{d}}$ we have to reduce it to a lower dimensional vector $\mathbf{d}^*$ by (5), such that $\mathbf{d}^*$ has only *independent* coordinates. We calculate a matrix M, such that $\mathbf{d} = M\mathbf{d}^*$, and $CM = 0$. Now the dimension of $\mathbf{d}^*$ gives us, how many independent variables exist in the system. Finally, we can solve $\mathbf{d}^{*T} A^* \mathbf{d}^* \to \min$ without constraints, where $A^* = M^T A M$, and this can be solved as a simple system of linear equations. We note that, this minimization is always solvable, the proof is based on the fact, that $f''(\mathbf{x}_0)$ is symmetric positive definite in this case.

The way of calculating M is very similar to Gauss elimination. During elimination, we can check if some of the constraints contradict to each other, or if the system is over-determined (see details in the original paper [1]).

2.3 Auxiliary objects

The use of the so-called *auxiliary objects* is an important idea in constrained fitting. We illustrate this through a simple example. Take three lines that are supposed to meet in a common point (see Figure 4(a)). We can formulate the related constraints by taking line 1 and 2, compute their intersection and constrain this intersection point to lie on

line 3; then we take line 2 and 3, and line 3 and 1 with similar constraints. This set of equations defines a relatively simple problem in a very complicated way.

An alternative solution is to introduce an auxiliary point p. This is also an unknown entity, but now we can define our constraints by three simple equations: i.e. all three lines must pass through p. Clearly, we have increased the number of unknowns in the parameter vector $\mathbf{x}$, but the system of equations – and all related Taylor approximants – have become much simpler. Note, the unknown surface parameters are generally associated with a corresponding point sets, but for auxiliary objects such a data point has no meaning. Typical auxiliary entities include a point, a point and a normal, a distance, etc.; their exclusive role is to simplify the system of equations and thus our computations.

3 Automatic detection of local constraints

Let us start with a simple example. We wonder whether pairs of lines are perpendicular or not, and we wish to incorporate additional constraints into our system, if the likelihood of being perpendicular is high. This can clearly be controlled by a user defined angular tolerance, and extra constraints will be added automatically, if two lines span an angle between $90 \pm \varepsilon$.

Formally: let $c(\mathbf{x}) = 0$ a simple constraint between two objects. Let ε be a tolerance level. The c constraint is within tolerance if and only if $|c(\mathbf{x})| < \varepsilon$, and we want to validate whether the constraint holds. For this, we introduce the following function:

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

We observe that, if $c(\mathbf{x})$ is out of tolerance, then $s_\varepsilon(c(\mathbf{x}))$ vanishes, and the constant zero constraint will not modify our system, otherwise $s_\varepsilon(c(\mathbf{x}))$ reproduces $c(\mathbf{x})$.

A necessary condition for c is, that $c(\mathbf{x})$ represents a so-called *faithful* representation for the distance used for a given entity. (Faithful means that the true Euclidian distance or a close approximation is computed in the vicinity of the curve/surface to be fitted, see details [1]). For example, for the *line meets point* constraint, we must use a normalized 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 a piecewise continuous function, so if we calculate the derivative numerically, we need to make it piecewise, as well.

To detect constraints automatically, we take the modified constraints for all object pairs. The numerical method will enable constraints only that are within the related tolerance level. The user typically defines different tolerances for different – parallel, perpendicular, tangential,

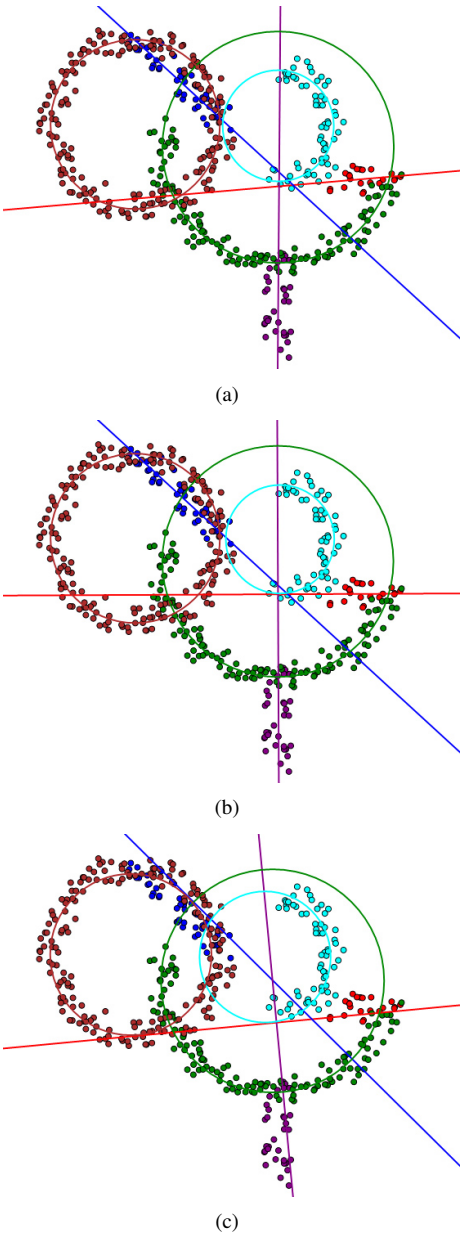

Figure 3: Automatically detected constraints: (a) — initial state; (b) and (c) — different configurations created by different tolerance levels

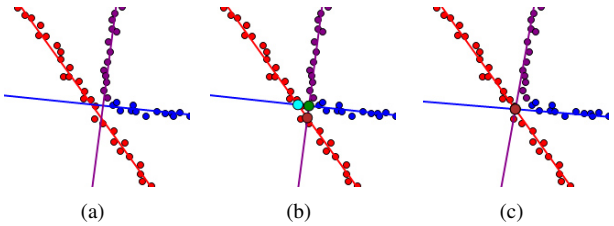

Figure 4: Three lines meet at a common point: (a) initial state; (b) pairwise intersection (auxiliary points); and (c) enforce the 'three points are equal' constraint.

concentric, etc. – constraints. Let us denote the set of objects by $S = \{s_i\}$, and the constraint types by $\{c_j\}$, such that $c_j(s_1, s_2)$ denotes an actual constraint between s_1 and s_2. Then for all c_j, consider the following constraint set:

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

Thus the global constraint system includes the explicitly defined constraints, and the C_j-s, i.e. the 'likely' constraint set.

A somewhat artificial example with three circles and three lines can be seen in Figure 3 that shows different configurations created by different constraint tolerances. Compare cases (b) and (c). The angular tolerances of 'lines orthogonal' are (b):10, (c):10 degrees. The distance tolerances of 'line passes through the center of a circle' are (b):10, (c):10 units, and 'line is tangent to circle' are (b):15, (c):25 units, respectively.

We can also handle more complex local (i.e. not pairwise) constraints. For example, take the previously mentioned *three lines meet in a single point* constraint in Figure 4. We may create auxiliary intersection points for all three pairs of lines, and by means of a corresponding *line close to point* constraints the algorithm can detect, whether the three intersection points are "likely to be" coincident or not. In the former case the three lines will be fitted simultaneously, enforcing a common point of intersection.

4 The best fit global grid

In this section, we investigate how to detect and create a best fit *'grid'* object and set the corresponding constraints.

The grid is represented as a 5 dimensional auxiliary object with the following parameters:

- the orientation of the grid (n),
- the origin of the grid (p_0),
- and a positive constant, the width of the cells (d).

Note that, the above parameters are not uniquely defined, since we can select all intersection points of the grid as origin, and we have four ways to define the orientation.

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

- certain parameters (n, p_0, d) are fixed,
- points are contained in the grid,
- lines are orthogonal/parallel to the grid,
- lines lie on the grid lines.

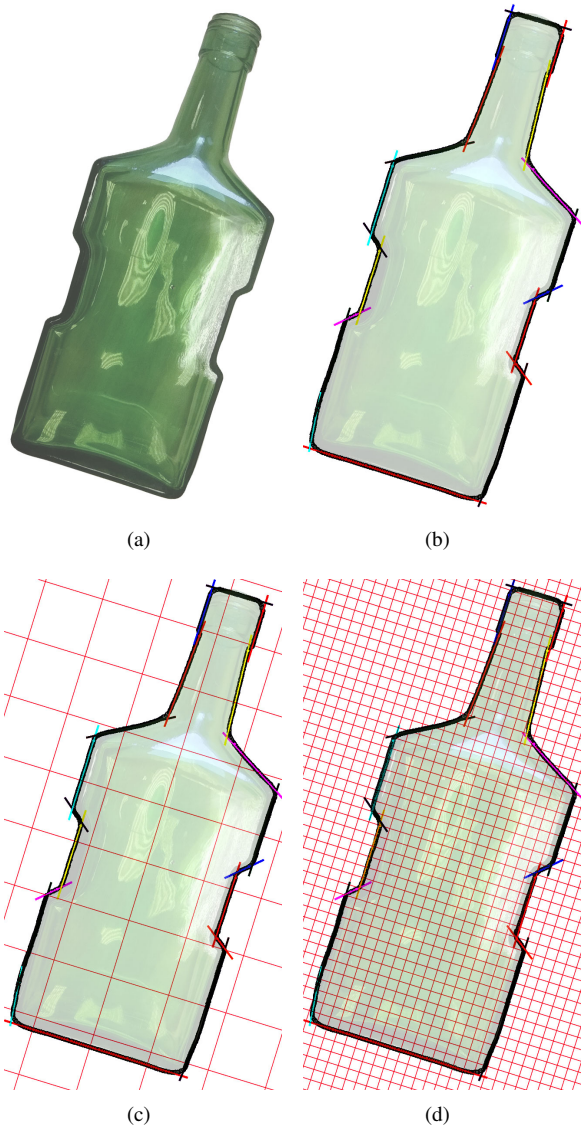

Figure 5: Detect grid: (a) original glass object; (b) segmented profile with straight segments; (c) optimal orientation; (d) final fit with optimal cell size.

For detecting the above constraints, we can use the automatic methods shown earlier, but this will work only, if the grid has been initialized 'almost perfectly'. Alternatively, we suggest an algorithm based on the following four basic steps (a related example can be depicted in Figure 5):

1. determine the best orientation,
2. fit the corresponding lines,
3. determine the best width parameter,
4. refit the objects matching the enhanced grid.

4.1 Determine the best orientation

Assume that, each line belongs to a straight section. Let us denote the length of l_i as $h(l_i)$, and the angle from x-

axis of its normal vector l_i in radian as $\angle(l_i)$. With respect to the grid, α and $\alpha+\pi/2$ have the same orientation, so we work with angles modulo $\pi/2$. The solution is given by clustering the angular values. We associate a radius with each cluster, that depends on a tolerance level and a weight by $w(S)=\sum_{l\in S}h(l)$. We select the best cluster (where $w(S)$ is maximal), and the weighted average of the angles will yield the best orientation.

4.2 Determine the best cell size

After setting up the best oriented grid, we fit the corresponding lines by the automatic method presented in Section 3. The next problem is determine the best common divisor of the distances between the parallel lines.

Let us denote the lines fitted according to the optimal orientation by $\{l_i\}$, and the absolute distances between the parallel lines by $\{d_{ij}\}=\{n_l\}$. If d is a suitable width of the grid cells, then the average remainder by $\{n_l\}$ is small. The sum of remainders can be written 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 to find the minimum of $\delta(d)$ in the $[d_{min},d_{max}]$ interval. It is easy to see, that the function δ is piecewise monotone, and the monotonicity drops at numbers being in the form of n_l/k, for certain n_l-s, where k is a positive integer. Therefore, we need to search for the minimum only at these points, and we can find this 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 optimal grid size, we can automatically detect the lines that satisfy all the grid constraints.

5 Estimate global symmetries

The second area of setting global constraints is the computation of axes of symmetry. We investigate algorithms for curves consisting of straight segments and circular arcs. First we determine all potential axes that may occur, then evaluate and prioritize them, and finally select the best one(s). Let $P=\{p_i\}$ denote the endpoints of the segments and the centers of the circles, and $L=\{l_i\}$ the lines.

The main steps of the algorithm are the following (see also Figure 6):

1. Collect all perpendicular bisectors between the points of P, and all angular bisectors between the lines of L. These bisector lines are called *auxiliary lines*.
2. Determine clusters of the auxiliary lines.
3. Evaluate the clusters (i.e. compute the corresponding axes and evaluate their 'measure' of symmetry).
4. Select the best axis (axes), and apply constrained fitting accordingly.

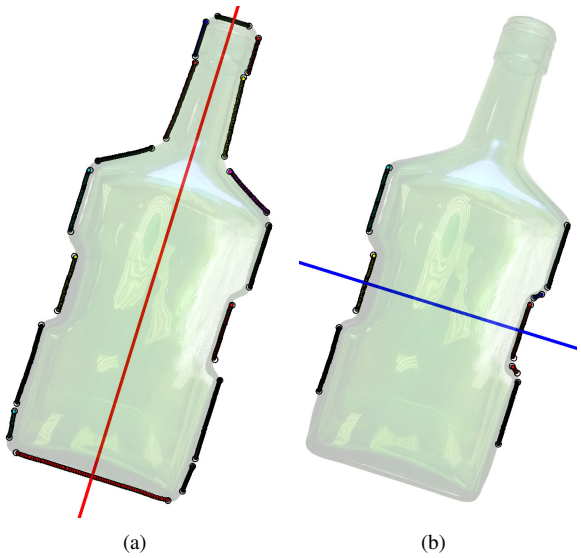

Figure 6: Detected axis of symmetry: (a) the best axis of symmetry (88.2%); (b) the second best axis of symmetry (35.9%).

The set of *auxiliary lines* A contains the perpendicular bisectors between the points of P: $A_1=\{\text{PBisector}(p_i,p_j):i<j\}$, and the angular bisectors between the lines of L: $A_2=\{\text{ABisector}(l_i,l_j):i<j\}$. Now we cluster these in two steps. First by the argument of the normal vectors (modulo π), then by the distances from the origin. For each cluster C, let l_C denote the average of C, these are the *axis candidates*. The number of elements in a given cluster is not necessarily the best quantity to measure the extent of symmetry; it is better to locate the corresponding symmetric parts by the related axis candidate and compute their arc lengths. With other words, symmetries amongst many small segments will be considered less important than those of a few large segments.

The clusters also provide information about symmetries of circular arcs, which help to enhance these computations. For each pair of arcs, we determine the corresponding arcs, or parts of arcs, that can really be considered as symmetric. The sum of these arcs yield additional weights to qualify the axis candidates.

We generally define constraints for axis of symmetry using auxiliary objects, as well. For example,

- *an axis is a perpendicular bisector of a segment*,
- *axis is an angular bisector of two lines*, etc.

Finally we perform constrained fitting according to the best axis of symmetry.

6 Examples

In this section, we present some examples using our algorithms of constrained fitting.

6.1 Case Study 1 - Gear wheel

As it was shown earlier in Figure 2, if we fit three circles independently, there will be small gaps between the adjacent arcs without tangential continuity, thus yielding a profile curve with poor quality. To avoid this, we must apply constrained fitting, as shown in Figure 2(d). In this case the least-squares deviation will be somewhat worse, but the prescribed constraints will be satisfied. This is illustrated numerically in Figure 7. The average least-squares errors increased by at most 15.5%, but this is still negligible compared to the magnitude of the circle radii. The radius of the first circle has been increased by almost 20%, actually this is the correct value. The reason for this is that the points lie on a relatively small segment of a large circle. In these cases the computed value of the best-fit radius is not so robust, and this may lead to relatively large changes once the constraints are enforced.

	Independent fit	**Constrained fit**	**Deviation**
1. error	1.015	1.201	15.50%
1. radius	125.762	155.810	19.28%
2. error	1.138	1.216	6.44%
2. radius	50.590	51.683	2.11%
3. error	0.822	0.855	3.77%
3. radius	145.962	156.278	6.60%
1. constraint	2.142	0	
2. constraint	1.010	0	

Figure 7: Numerical analysis of unconstrained vs. constrained fitting for the profile curve in Figure 2. (i) Least squares errors and (ii) radii of the three fitted circular arcs, (iii) estimated deviation errors at the connecting points of the circles.

6.2 Case Study 2 - Bottle

To demonstrate our algorithms for detecting global constraints we used the profile curve of a glass. Both algorithms (detecting grid and axis of symmetry) have produced satisfactory results.

Grid detection has already been shown in Figure 5; the algorithm located the best orientation and cell size for the grid, when performed constrained fitting. As explained earlier, the algorithm has a dynamic behaviour, i.e. it adjusts, which constraints are actually taken into consideration when the system of equations is finally solved. Close views of Figure 5 are shown in Figure 8. We can see the middle purple line in 8(b) is almost parallel to the grid, but the angle is out of the tolerance level. The same thing happens on the 8(e) with the first yellow line. Another interesting effect, is the third purple line in 8(e), which is fitted to the orientation, but not fitted to the grid. This illustrates that, the algorithm is sensitive to the local inconsistencies, and adaptively determines the constraints for the optimal global grid.

Symmetry detection was demonstrated in Figure 6. The algorithm determined two axes of symmetry with symmetry levels 88.2% and 35.9%, respectively. The first axis is the global axis of symmetry, while the other one indicates a local symmetry in the middle of the object.

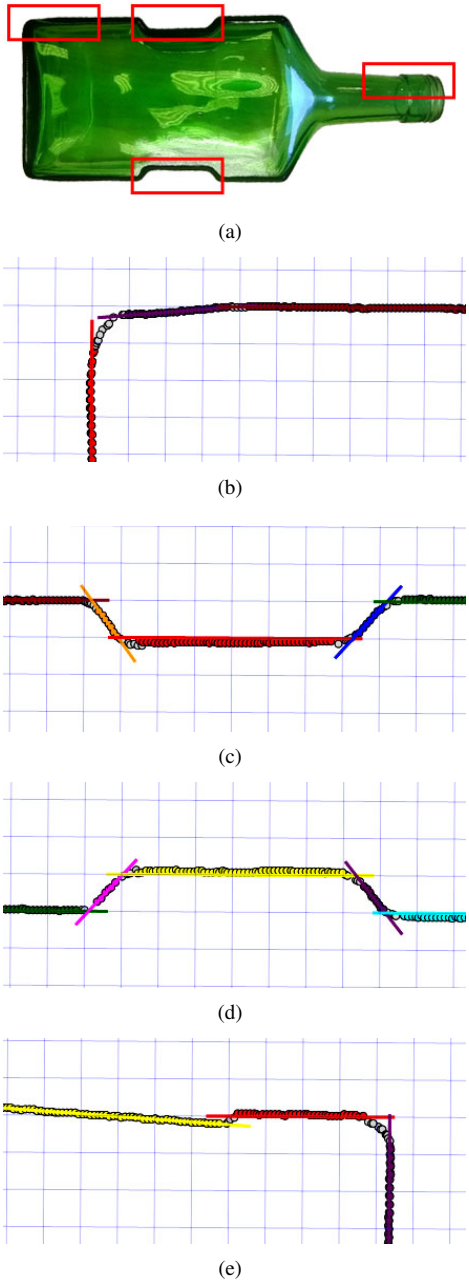

Figure 8: Close views of Figure 5.

7 Conclusion

It is a crucial issue to perfect engineering objects being reconstructed from measured data. Having only rough approximations for perpendicularity, parallelism, concentricity, etc, would not be acceptable for the majority of downstream CAD applications. In this project we have investigated *perfecting techniques* to automatically set up and enforce local and global geometric constraints. We have tested our methods for planar point data sets, represent-

ing straight and circular curve segments. These algorithms can be generalized to 3D objects in a reasonably straightforward way; this is subject of ongoing research. In the future, we plan to detect other types of global symmetries, such as, rotational and translational symmetries, then later apply these techniques to couple conventional and free-form curves and surfaces, as well.

Acknowledgements

I would like to thank my supervisor Dr. Tamás Várady for many constructive discussions, and guiding me to write this article. I would also like to thank the other members of our research team – Péter Salvi, György Karikó and Pál Benkő – for exchanging important technical ideas. This research is supported by the Hungarian Scientific Research Fund (OTKA No.101845).

References

[1] 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.

[2] I. Coope. Circle fitting by linear and nonlinear least squares. *Journal of Optimization Theory and Applications*, 76(2):381–388, 1993.

[3] 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.

[4] Y. Li, x. Wu, y. Chrysathou, A. Sharf, D. Cohen-Or, N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *In ACM Transactions on Graphics (TOG)* (Vol. 30, No. 4, p. 52). ACM, 2011.

[5] 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.

[6] P. Marks. Capturing a Competitive Edge Through Digital Shape Sampling & Processing (DSSP). *SME Blue Book Series*, 2005.

[7] 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.

[8] J. Porrill. Optimal combination and constraints for geometrical sensor data. *The International Journal of Robotics Research*, 7(6):66–77, 1988.

[9] 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.

[10] R. Schnabel, R. Wahl, R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. In *Computer Graphics Forum* (Vol. 26, No. 2, pp. 214-226). Blackwell Publishing Ltd. 2007.

[11] 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.

[12] 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.

[13] T. Várady, P. Salvi, *3D Geometric Modelling and Digital Shape Reconstruction, Lecture Notes*, Budapest University of technology and Economics, BME IIT, 2013.

[14] 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.