Better Surface Intersections by Constrained Evolution

C. Robertson and R. B. Fisher

Vision Group, Institute for Perception, Action and Behaviour
Division of Informatics, University of Edinburgh
Edinburgh, UK EH1 2QL
craigr@dai.ed.ac.uk

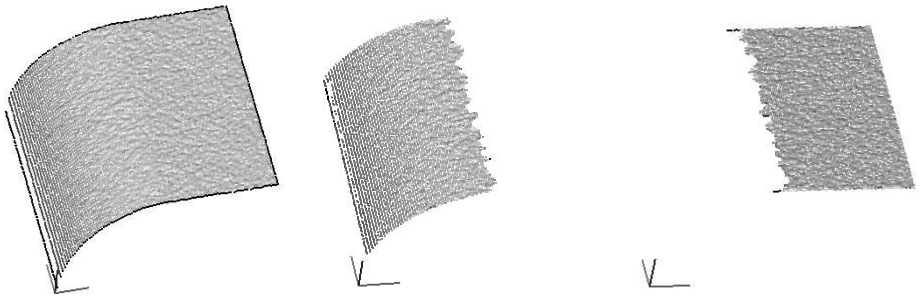
Abstract

Reverse-engineering a machined part to generate a CAD model requires range data to be collected and registered from many views then segmented into surface primitives. Correctly computing the intersections of these surface primitives is a critical part of building the CAD model. We describe a method for correcting the ragged boundaries often found with region-growing algorithms and show examples of its application. This method is useful for a large subset of the surface intersections that regularly appear in manufactured objects.

1 Introduction

Reverse-engineering a machined part to generate a CAD model requires range data to be collected and registered from many views then segmented into surface primitives. One method of performing this segmentation is to use some form of iterative surface-growing algorithm, for example [1, 4] and [8], in either $2\frac{1}{2}$D images or complete 3D data clouds. Behaviour at patch boundaries is rarely discussed though, except in terms of over-segmentation and under segmentation. Even comparisons of these types of algorithms, using several other objective tests, do not discuss surface intersections (for example [2]) directly. In our experience, although surface parameterization may be good, the intersection boundary is often very ragged and rarely very accurate (that is in the correct place) or at the correct angle. This makes its usefulness limited for the generation of CAD models. For example, the images in figure 1 shows the segmentation of noisy synthetic data representing a part where a cylinder meets a plane with tangent continuity. It can be seen that the segmentation boundary is poor and cannot be used to generate the CAD model, which in this instance would require the line of intersection.

One reason for the poorness of the boundary line is that region growing based segmentation schemes often have poor or unstable surface parameterizations at run-time. This may be due to either due to the growing algorithm itself or the stability of convergence of the fitting algorithms when using low numbers of points. This means that an accurate intersection line cannot be found



(a) (b) (c)

Figure 1: Example cylinder/plane segmentation showing resultant ragged boundary *(a)* Noisy data set *(b)* Segmented cylinder *(c)* Segmented planar part

analytically unless the parameterization is improved substantially or more constraints on the surfaces can be employed. In this paper we have developed a new method based substantially on the evolutionary approaches we have sucessfully employed previously [8, 7]. This allows us to simultaneously segment, parameterise and accurately intersect two surfaces.

2 Algorithm

2.1 Surface Extraction by Region Growing

Surface segmentation by region growing is a well known and often used approach. The method, whether employed in $2\frac{1}{2}$D or 3D is generally composed of the following elements:

- Local curvature estimation
- Local shape classification, pixel or polygon characterization
- Generation of growth seed patches
- An iterative growth scheme
  - Estimate competing surface parameters
  - Assess suitability of surrounding elements (often by computed residual) for agglomeration
  - Grow surfaces into suitable elements
  - Re-fit surface
- Assessment of end criterion, for example stability occuring

There are, of course, many variations on this scheme [2]

2.2 Boundary Correction Algorithm

What we require is a better method for performing all of the following functions:

- Correct surface boundary identification
- Incorporation of *a priori* surface information, for example surface type, to get the best parameterisation
- Given that boundary and type, optimal re-parameterization of the surfaces
- Incorporation of *a priori* geometric constraints on surfaces and/or boundaries

To fulfill this set of requirements requires an algorithm that can perform constrained optimization over a large solution space. The boundary correction algorithm we have employed, a GENOCOP III hybrid, is a floating point evolutionary algorithm as outlined in Michalewicz [5]. We have previously employed this approach for constrained surface fitting, as described in [8]. In this case we have used a splitting plane that splits the two data into groups of points which then have the relevant error function evaluations performed on them. This evaluation gives the goodness of fit for the particular chromosome.

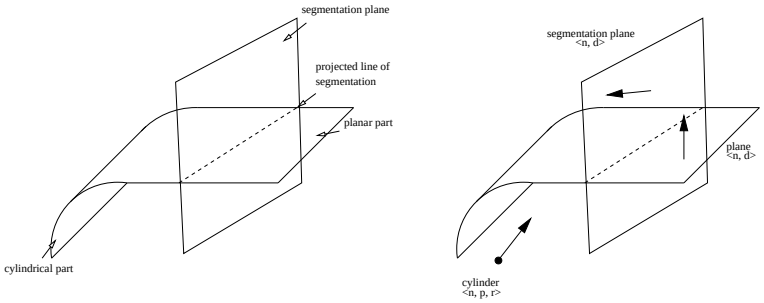


Figure 2: Boundary correction algorithm formulation

We assume that there are two surfaces present whose intersection can be represented by a segmenting plane. For our examples, we have used the following combinations:

- Convex cylinder meeting plane with tangent continuity
- Concave cylinder meeting convex cylinder with tangent continuity
- Cylinder meeting sphere with tangent continuity
- Concave cylinder meeting convex cylinder with tangent continuity, using different radii.

There are clearly many other examples which fit these constraints.

2.3 Shape Representation

In this version of the algorithm, models of degenerate quadric surfaces are used. This is a small subfamily of surfaces of the following types: spheres, cylinders, right circular cones etc. The machined surfaces that we address rarely contain whole pieces of these shapes so the surfaces are often fragmentary or partial.

In standard GAs binary encoding forms the chromosomes in the solution, however in an evolution program each gene is a floating point number. Genes for each of the present surfaces are then concatenated to form a chromosome.

In the case of **planes**, we used the 4 gene parametric representation $A :< \hat{n}, d >$ where $\hat{n}$ is the unit normal describing the plane and d is the constant defining its minimum distance from the origin. In the case of **spheres**, we used the 4 gene parametric representation $A :< \mathbf{P}, r >$ where $\mathbf{P}$ is the centre point and r is the sphere radius. In the case of **cylinders**, we used the 7 gene parametric representation : $A :< \mathbf{P}, \hat{n}, r >$ where $\mathbf{P}$ is the start point of the cylinder, $\hat{n}$ is the axis direction and r is the radius. For **cones**, the 7 gene representation used is : $A :< \mathbf{P}, \hat{n}, \alpha >$,where $\mathbf{P}$ is defined as the start point (or tip) of the cone, $\hat{n}$ is the axis direction and α is the half-angle between the axis and the slope of the cone.

A full chromosome, G, describing a given object, is then a set of concatenated part-chromosomes: $G = \{A_j\}$. The parametric representation of a set of degenerate quadrics as a chromosome is much shorter than a set of general quadrics as explored in [8]. This cuts down the complexity of constraint representation and makes it amenable to straighforward manipulation without the need to employ geometrical constraints on the surfaces' form, only on pairs of forms taken as systems.

The parameterisation chromosome is augmented with the parameterisation of the segmenting plane, S, to form the final optimisation chromosome: $G = \{A_j, S\}$.

2.4 Constraint Formulation

In virtually all cases, domain constraints on individual genes are used to narrow the search space for that gene. These are represented as one permanent part of the sequential quadratic penalty function matrix [6] used in the evaluation function. A good example of where domain constraints can reduce the search space is in the case of the three parameters describing a unit normal. Each of these parameters can never be outside the range $[-1, +1]$ so these make good domain constraints.

In-chromosome relational constraints are straightforward to formulate if a parametric form is used. For example, consider two planes:

$P_1 =< n_{x1}, n_{y1}, n_{z1}, d_1 >$ and

$P_2 =< n_{x2}, n_{y2}, n_{z2}, d_2 >$

which are known *a priori* to be orientated orthogonally. In this case, the chromosome would have the form :

$G = \{n_{x1}, n_{y1}, n_{z1}, d_1, n_{x2}, n_{y2}, n_{z2}, d_2\}$

and the orthogonality constraint would then appear as a non-linear inequality of the form: $|(n_{x1} \times n_{x2}) + (n_{y1} \times n_{y2}) + (n_{z1} \times n_{z2})| \leq \epsilon$ where ϵ is the constraint tolerance value. Similar formulae can be used to represent continuity, parallel axes, etc.

All surface normals are naturally subject to a unity constraint.

In order to perform the optimisation, Genocop requires a set of starting position on the constrained solution manifold. These are the seeds for the reference and search points which are then mutated around them. In our case this means designing a chromosome which is both close to being a concatenation of the individual least-squares results for the part-chromosomes as well as fulfilling the domain and relational constraints. Starting conditions for increasingly complex solutions with increasingly complex constraints have previously been found to be difficult [8] for the general quadric. However, when a parametric representation is used, start conditions become simple to generate, even when many constraints are used.

2.5 Algorithm

The internal workings of Genocop III are beyond the scope of this paper, suffice to say that it is an evolutionary optimisation scheme which can handle multiple constraints, in the form of inequalities, of both linear and non-linear types. It is described fully in [5] and [6].

The segmenting plane passes through the data and all points lying on one side are said to belong to *surface 1*, all points on the other side are said to belong to *surface 2*. These two data sets are then passed to the relevant distance evaluation function to compute the sum of all points to the surface represented in each chromosome. The sum of each of the distances then represents the fitness for that surface parameterisation. Populations fitting each of the *a priori* constraints are then generated in the usual way by Genocop until stability or some other stopping criteron is reached. The approach is illustrated in figure 2.

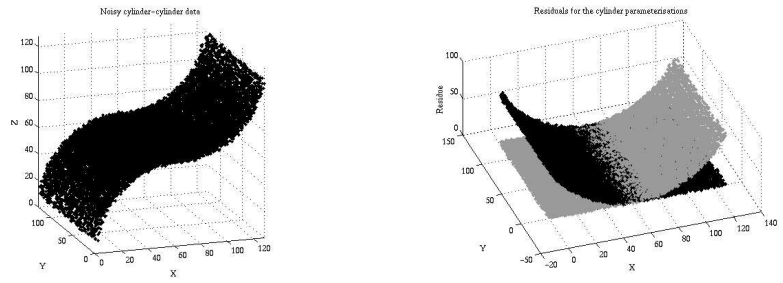
The evaluation function for the chromosomes is formulated as follows: Assume a segmentation plane parameterization, S and the two surface parameterizations, A_1 and A_2 the point set P is split by S into the two sets P_1 and P_2 of sizes m_1 and m_2. Given that we are able to use geometric distances for our error calculations we now wish to minimize the error function E:

$$E = \sum_{i=0}^{i=m_1} D(A_1, P_{1_i}) + \sum_{i=0}^{i=m_2} D(A_2, P_{2_i}) \tag{1}$$

D is the Euclidean distance function appropriate for each of the surfaces. It is also clear to see how this approach can be generalized to additional surfaces and constraints.

2.6 Boundary Uncertainty and Individual Error Contributions

Presuming that the surface intersection boundary is fixed and that points belonging to both surfaces will be scattered around it, it is interesting to examine how individual points contribute to the overall algorithm error. Shown in figure 3(a) are the points from a cylinder meeting another cylinder with tangent plane continuity. In figure 3(b) are the residuals generated by applying two perfect cylinder models to the same data, that is the distances from the points to the known cylinder models. Generating a segmentation plane that correctly assigns both sets of points is clearly very difficult. The conclusion drawn from this result is that even given a perfect surface parameterization, the assignment of points on the boundary will never be perfect and at best we can expect a reasonable approximation for the segmentation plane position and parameterization.



(a) (b)

Figure 3: **(a)** Noisy cylinder meets cylinder data, **(b)** Residuals of perfect cylinder model fitting (grey is surface 1, black is surface 2)

2.7 Data Preprocessing

Unusually for evolutionary approaches, the GENOCOP III algorithm needs a point on the constrained solution manifold from which to build its initial reference population. In order to efficiently generate this first solution we first preprocessed our data as follows:

- For each point we found its M nearest neighbours
- Through this point, we derived a surface normal by fitting a local plane through it and its M neighbours
- By back-projecting random groups of three normals we were able to approximate cylinder axes and sphere centres (the point which is closest to all three vectors is co-incident with the axis). We used three normals for extra robustness although only two are necessary.

- We then use a RANSAC based [10] 3-d line finder and 'node' finder to find the axis lines and the sphere centres. In our definition a node is a position in space which has points scattered about it. In our examples this is generally a sphere centre.

Once we have performed these relatively simple pre-processing operations we have topological information as well as an approximate set of parameterizations for the surfaces present.

3 Results

3.1 Trial Data

Trial data was generated based on three different figures:

- Cylinder meets plane with tangent continuity
- Cylinder meets cylinder with tangent continuity
- Cylinder meets sphere with tangent continuity

Each data set was subjected to Gaussian noise in the x,y and z components at different levels between 0 units and 2 units standard deviation in a total image size of $128 \times 128 \times 128$. All primary segmentations in this section were performed using our proprietry range segmentation algorithm *rangeseg* described in [4] which is generally considered to be the best of its kind (as documented [2]).

It should also be noted that these results are for artificially generated data with applied Gaussian noise in X,Y and Z. The noise level on our range scanner is around 3 orders of magnitude smaller than these levels in each dimension if the units are considered to be in millimetres.

Tests were conducted over 5 random data sets.

3.2 Constraints Applied

In all trial cases we constrained the normal vectors to be units and applied domain constraints on the parameter values. Also in some cases the segmenting plane is constrained to having a surface normal that is orthogonal to the axis of the cylinder and the plane.

3.3 Cylinder meets Plane with Surface Continuity

These results demonstrate that although the correction algorithm has problems with regions of high noise, the boundary is corrected from very ragged to straight and fulfils all of the contraints correctly. See table 1.

Table 1: Errors on Boundary Position After Correction - Cylinder/Plane Data

Percentage noise	Mean boundary position error(units)
0	0.0units
0.5	0.1units
1	0.1units
2	0.2units

3.4 Cylinder meets Cylinder with Surface Continuity

Again, the results show a major improvement over the optimized segmentation. All constraints are fulfilled. See table 2.

Table 2: Errors on Boundary Position After Correction - Cylinder/Cylinder Data

Percentage noise	Mean boundary position error(units)
0	0.0units
0.5	0.3units
1	0.8units
2	1.2units

3.5 Cylinder meets Sphere with Surface Continuity

These results show the ambiguity of points around the sphere-cylinder interface where the surface fit of points has a residual so low that they could belong to either surface. See table 3.

Table 3: Errors on Boundary Position After Correction - Cylinder/Sphere Data

Percentage noise	Mean boundary position error(units)
0	0.0units
0.5	0.4units
1	1.0units
2	1.4units

3.6 Notes on Robustness

The most notable influence on the boundary correction results is surface noise. As was shown in section 2.6, there are serious surface allocation problems in the boundary region which mean that any boundary surface that is applied will tend to either misclassify points or converge to an incorrect position in the presence of too much noise. This would be clearly true of any algorithm.

Correct boundary positioning is more difficult (takes longer to converge) in the case of cylinder meets cylinder and cylinder meets sphere due to the slow and ambiguous transition from one surface to another.

4 Conclusions and Further Work

4.1 Conclusions

The evolutionary approach is only one possible approach to boundary correction. The results are good, having a low mean deviation from the true values, but it is difficult to get high precision sub-pixel boundary estimates due to the individual error contributions of single points being very small. This causes even this simple, essentially linear, boundary to occasionally migrate some small distance from the *a priori* known best position. This is also a function of the quality of the surface parameterisations which, given second order surface estimations, are sometimes good locally but not globally.

4.2 Further Work

Since boundary correction is subject to errors from the segmentation scheme as well as the surface fitting the results are exceptionally good. One way to improve this scheme though might be to apply a run-time adaptive weighting that favours the contributions of points near the boundary. We also are seeking to extend the approach to other second-order meeting second-order blending for other degenerate surfaces.

Acknowledgements

The work presented in this paper was funded by a UK EPSRC grant number GR/H86905.

References

1. Besl, P.J. (1986) Surfaces in Early Range Image Understanding. PhD Dissertation, Electrical Engineering and Computer Science Department (RSD-TR-10-86), University of Michigan.
2. Hoover,A., Jean-Baptiste, G., Jiang, X., Flynn, P.J., Bunke, K., Goldgof, D., Bowyer, K., Eggert, D., Fitzgibbon, A., Fisher, R. (1996) "An Experimental Comparison of Range Segmentation Algorithms", IEEE Trans. Pat. Anal. and Mach. Intel., 18(7), pp 673–689.
3. Faux,I.D. and Pratt,M.J. (1985) *Computational Geometry for Design and Manufacture*, Ellis Horwood Limited, Chichester, UK.
4. Fisher, R.B.(1997), Fitzgibbon, A., Eggert, D., "Extracting Surface Patches from Complete Range Descriptions", Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada, pp 148-155.

5. Michalewicz, Z.,(1996) *Genetic Algorithms + Data Structures = Evolution Programs*, Third Edition, Springer.
6. Michalewicz, Z. (1996), Dasgupta D., Le Riche R.G., and Schoenauer, M., "Evolutionary algorithms for constrained engineering problems", special issue on Genetic Algorithms and Industrial Engineering, ed. M. Gen, G.S.Wasserman and A. E. Smith, International Journal of Computers and Industrial Engineering.
7. Robertson, C. (1999), Fisher, R.B., Werghi, N., Ashbrook, A., "An Improved Algorithm to Extract Surfaces from Complete Range Descriptions", Proc. World Manuf. Conf, WMC'99 (ISMT'99), pp 592-598, ICSC Academic Press, Durham.
8. Robertson, C. (1998), Corne, D., Fisher, R.B., Werghi, N., Ashbrook A., "Investigating Evolutionary Optimisation of Constrained Functions to Capture Shape Descriptions from Range Data", Proc. 3rd On-line World Conference on Soft Computing (WSC3) (see `http://www.cranfield.ac.uk/wsc3/` also in *Advances in Soft Computing - Engineering Design and Manufacturing*, eds. Roy, Furuhashi and Chawdhry, Springer-Verlag, 1998.
9. Robertson, C. (1999), Fisher, R.B., Werghi, N., Ashbrook, A., "An Evolutionary Approach to Fitting Constrained Degenerate Second Order Surfaces", to be published in the Proceedings of EvoIASP'99, Sweden (28th May 1999), eds. Poli, Voigt, Cagnoni, Corne, Smith and Fogarty, Springer-Verlag Berlin.
10. Bolles, R. C and Fischler, M. A (1980), "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Technical Note 213, Artificial Intelligence Center, SRI International, Menlo Park, California.