B-Spline Contour Representation and Symmetry Detection [1]

Philippe Saint-Marc and Gérard Medioni
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273
Email: medioni@usc.edu

The detection of edges is only one of the first steps in the understanding of images. Further processing necessarily involves grouping operations between contours. We present a representation of edge contours using approximating B-splines and show that such a representation facilitates the extraction of symmetries between contours. Our representation is rich, compact, stable, and does not critically depend on feature extraction whereas interpolating splines do. We turn our attention to the detection of two types of symmetries, parallel and skewed, which have proven to be of great importance to infer shape from contour, and show that our representation is computationally attractive. As an application, we show how parallel symmetries can be used to infer the 3-D orientation of a torus from its intensity image. Due to lack of space, the reader is referred to [4] for complete mathematical details, survey of previous work, and proper references.

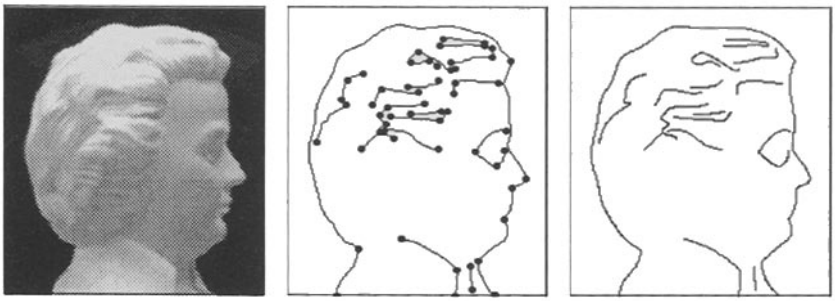
Contour Representation

A very promising idea to represent image contours is to use piecewise polynomials. The advantages are obvious: this representation is rich, compact, analytical and local in the sense that a small change in the original curve does not affect the representation entirely. The approach commonly used consists of first extracting a set of knots from the discrete curve and then to approximate the curve between each pair of knots by polynomials under continuity constraints at the knots. For example, Plass and Stone [2] propose to take the knots as the vertices of a polygonal approximation, then to use dynamic programming in order to select those knots which provide the best approximation by cubics. The main point that we formulate against these methods is that they rely heavily on the always critical segmentation step, which brings up the stability issue. Also, techniques such as dynamic programming can yield a complexity of $O(n^3)$ where n is the number of initial knots [2]. Instead, we propose to use the following B-spline least-squares fitting method which, as we shall see, does not require any knot selection and is relatively insensitive to noise.

A spline can be expressed as a linear combination of B-splines which are themselves piecewise polynomials [1], the coefficients being the vertices of the spline's *guiding polygon*. Thus, a spline can be easily manipulated by modifying its guiding polygon, hence its popularity in CAD/CAM systems. Furthermore, as B-splines are defined locally, modifying the position of a vertex does not affect the spline entirely. In the case of a planar curve, a spline $Q(u) = (X(u), Y(u))$ with $m+1$ vertices can be defined as $Q(u) = \sum_{j=0}^{m} V_j B_j(u) = \sum_{j=0}^{m} (X_j B_j(u), Y_j B_j(u))$, where (X_j, Y_j) are the *vertices* of the guiding polygon and $B_j(u)$ the B-splines.

Let C be an ordered set of $p+1$ points $P_i = (x_i, y_i)$, what is the spline which best approximates C? An approach proposed in [1] consists of minimizing the distance $R = \sum_{i=0}^{p} \|Q(u_i) - P_i\|^2 = \sum_{i=0}^{p} (\sum_{j=0}^{m} X_j B_j(u_i) - x_i)^2 + (\sum_{j=0}^{m} Y_j B_j(u_i) - y_i)^2$, where u_i is some parameter value associated with the i^{th} data point. Minimizing R is equivalent to setting all partial derivatives $\partial R/\partial X_l$ and $\partial R/\partial Y_l$ to 0, for $0 \leq l \leq m$, which yields two linear systems of equations. These are easily solved for all X_j and Y_j respectively using standard linear algebra, yielding the guiding polygon of the spline which best approximates the original curve. In the case of open curves, we have the option to force end-points to be interpolated. In this case, the first and last vertices are simply set to lie at the end-points so that the linear systems are reduced to $m-1$ equations of $m-1$ unknowns. In the case of closed curves, the linear systems are over constrained since some vertices are required to be identical. This method has proven to be relatively insensitive to noise [4]. The choice of m (the number of vertices)

[1]This research was supported by the Defense Advanced Research Projects Agency under contract F33615-87-C-1436 monitored by the Wright-Patterson Air Force Base.



(a) Intensity Image (b) Segmented Edges (c) Reconstruction

Figure 1: Eine Kleine Nachtmusik...

determines how close to the original data the approximation is, which is measured by R (see above). The automatic selection of the number of vertices is not trivial. Our approach is to first set a fitting tolerance r_0, then find the value of m which yields the normalized distance $r = R/(p+1)$ closer to r_0 using a binary search approach.

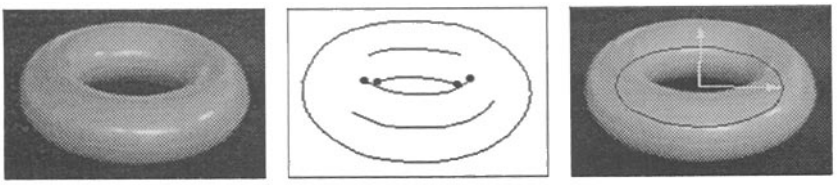
The input for our system is an edge map produced by an edge detector such as Canny's. Three stages are sequentially considered: linking, corner detection, and spline approximation. Linking of the edgels is done using a simple and fast algorithm which looks for 8-connected components [4]. No gap-bridging or other task is performed since it is our belief that point-wise surgery is too myopic, and that if grouping is needed, it has to be performed at a higher level. Corner detection is performed by detecting tangent discontinuities in connected components after application of the *adaptive smoothing* operator [4]. The final step consists of approximating each elementary curve by a spline. When a closed curve with no corners is considered, a global least-squares approximation is performed. In the case of an open curve or a closed curve with corners, each curve segment between pairs of corners is approximated with the constraint that the end-points be interpolated. This insures the reconstructed curves to be continuous at corner locations. Figure 1 shows the results obtained on a real example. The 167×222 intensity image of a Mozart bust is displayed in figure 1(a) and the contours obtained after edge detection and linking in figure 1(b), with detected corners overlaid. Finally, a quadratic B-spline approximation of each curve segment between corners is done using a fitting tolerance of 0.5 which leads to the reconstruction displayed in figure 1(c).

It is interesting to point out that the method is very tolerant of segmentation errors since, if a corner is missed, more vertices will be used, hence the reconstruction will still be satisfying [4]. In the following section, we show how this piecewise polynomial representation of image contours can be used to detect symmetries in the image plane.

Symmetry Detection

The detection of symmetries is an essential step when inferring shapes from contours. In [5], Ulupinar and Nevatia claim that there exist two kinds of symmetry which give significant information about the surface shape for a variety of 3-D objects: *parallel* and *skewed* symmetry. Most methods for detecting symmetries in edge maps use local properties in order to identify symmetric edge points. In this case, it is necessary to test every possible pair of edge points against the property which leads to an $O(n^2)$ algorithm where n is the number of points [3]. Instead, we propose to use the B-spline representation in order to identify symmetric edge segments. On one hand, the complexity is reduced since n now represents the number of edge segments. On the other hand, the computation is more global, hence less sensitive to noise. Let us go into more details in the case of parallel symmetries knowing that a similar approach is used to detect skewed symmetries [4].

Let $c_1(u)$ and $c_2(v)$ be two parametric planar curves, and $\theta_1(u)$ and $\theta_2(v)$ their respective tangent orientations. These curves are said [5] to be parallel symmetric if there exists a continuous monotonic function f such that $\theta_2(f(u)) = \theta_1(u)$. It is easy to show that when considering two conics, then $f(u)$



(a) Intensity Image (b) Parallel Symmetries (c) Positioning

Figure 2: Positioning of a Torus using Parallel Symmetries

is unique and is simply the ratio of two linear functions of u, with the result that two conics are always parallel symmetric [4]. Now supposing that $c_1(u)$ and $c_2(v)$ are only defined on the interval $[0,1]$, we will say that these two segments are parallel symmetric on $[u_0,u_1] \subseteq [0,1]$ iff $[f(u_0),f(u_1)] \subseteq [0,1]$. The detection of parallel symmetries between quadratic B-splines then follows: A quadratic B-spline can be expressed as a collection of connected conic segments $\mathcal{S} = \{c_i(u)\}$, for $i = 0,\cdots,m$, each defined on the interval $[0,1]$. Given another quadratic B-spline $\mathcal{S}' = \{c'_j(v)\}$, for $j = 0,\cdots,n$, each conic segment of $\mathcal{S}$ is compared against each conic segment of $\mathcal{S}'$ to eventually detect an elementary parallel symmetry between them. Given the simplicity of f and because of the usually small number of conic segments involved, the method is computationally very attractive. Grouping elementary symmetries can then be done by using simple connectivity criteria between segment pairs [4]. As an example, figure 2(b) shows the parallel symmetries detected in the edge map of the intensity image of figure 2(a).

The torus is an interesting example on which to demonstrate the application of parallel symmetry. Assuming that the object is far enough from the camera, and ignoring its actual size, it is reasonable to model the imaging process by an orthographic projection. The torus is a smooth solid of revolution, and the contours generated in its image correspond only to *limbs* or occluding contours, which are unfortunately viewer dependent. Even though it is possible, although complicated, to recover the position and orientation of a torus from its limbs, we propose instead to use the property of the torus that the axes of parallel symmetry in its image are the projection of its circular spine (3-D skeletal axis). This property allows us to recover the 3-D orientation quite simply: we fit an ellipse to the detected parallel symmetry axis, the orientation of the plane on which the torus is lying is given by the eccentricity of the ellipse and the angle of the major axis with the horizontal. Figure 2(c) shows the position of the torus recovered from the detected parallel symmetries of figure 2(b).

Conclusion

We have presented an approach to representing contours using approximating B-splines. It has attractive properties for use in Computer Vision: the representation is rich, compact, stable, local and segmented. We have shown how this representation can be used to extract parallel symmetries from edge maps. Similar ideas are used to extract skewed symmetries [4]. We are currently working on the selection of elementary symmetries, their grouping, and interpretation in order to generate higher level primitives. We also intend to apply these tools to the detection of local symmetries.

References

[1] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, Los Altos, CA 94022, 1987.

[2] M. Plass and M. Stone. Curve-Fitting with Piecewise Parametric Cubics. *ACM Transactions on Computer Graphics*, 17(3):229–239, 1983.

[3] J. Ponce. Ribbons, Symmetries, and Skew Symmetries. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1074–1079, Cambridge, Massachusetts, 1988.

[4] P. Saint-Marc and G. Medioni. B-Spline Contour Representation and Symmetry Detection. Technical report, Institute for Robotics and Intelligent Systems, USC, Los Angeles, California 90089-0273, 1990.

[5] F. Ulupinar and R. Nevatia. Using Symmetries for Analysis of Shape from Contour. In *Proceedings of the International Conference on Computer Vision*, pages 414–426, 1988.