Shape offsets via level sets

R Kimmel and A M Bruckstein*

An algorithm for shape offsetting is presented that is based on level-set propagation. This algorithm avoids the topological problems encountered in traditional offsetting algorithms, and it deals with curvature singularities by including an 'entropy condition' in its numerical implementation.

shape offsets, prairie fires, numerical algorithms, Huygens principle

Algorithms for shape offsetting are of great importance in computer-aided design (CAD), computer-aided manufacturing (CAM), the numerical control (NC) of machines, computer graphics and related fields. The need for offset shapes arises in applications such as the numerical control of sawing machines or milling machines in the car industry.

The problem of shape offsetting can be formulated as follows: given a simple, closed planar curve

$$\mathbf{X}_0(s)=[x(s),\ y(s)]^{\mathrm{T}} \tag{1}$$

where s is an arbitrary curve parameterization, find an offset curve which is simple and closed (or has simple and closed components) and is almost everywhere given by

$$\mathbf{X}_L(s)=\mathbf{X}_0(s)+\mathbf{N}(s,0)L \tag{2}$$

Equation 2 represents a curve running 'parallel' to $\mathbf{X}_0(s)$, where L is the displacement of the offset curve, and $\mathbf{N}(s,0)$ is the unit normal at the point $\mathbf{x}_0(s)$ given by

$$\mathbf{N}(s,0)=\frac{1}{(x_s^2(s)+y_s^2(s))^{1/2}}[-y_s(s),x_s(s)]^{\mathrm{T}} \tag{3}$$

Consider $\mathbf{X}(s,t)$ to be a curve continuously changing in time, so that, for all t, $\mathbf{X}(s,t)=\mathbf{X}_0(s)+t\mathbf{N}(s,0)$. This curve evolution can be described differentially by

$$\begin{cases}\dfrac{\partial \mathbf{X}(s,t)}{\partial t}=1\mathbf{N}(s,0)\\ \mathbf{X}(s,0)=\mathbf{X}_0(s)\end{cases} \tag{4}$$

Electrical Engineering Department, Technion, Haifa 32000, Israel
*Computer Science Department, Technion, Haifa 32000, Israel
Paper received: 9 March 1992. Revised: 6 July 1992

In Appendix A, it is proved that the evolution equation

$$\begin{cases}\dfrac{\partial \mathbf{X}(s,t)}{\partial t}=1\mathbf{N}(s,t)\\ \mathbf{X}(s,0)=\mathbf{X}_0(s)\end{cases} \tag{5}$$

postulating that each point on the curve moves in the direction of the *instantaneous* normal

$$\mathbf{N}(s,t)=[-y_s(s,t),x_s(s,t)]^{\mathrm{T}}\frac{1}{(x_s^2(s,t)+y_s^2(s,t))^{1/2}}$$

with velocity 1 provides the same flow as Equations 4 almost everywhere (where problems do not occur).

This propagation rule is the so-called 'prairie-fire' model for shape evolution (see, for example, References 1 and 2). If we could determine the solution of Equations 4 or Equations 5 for all $t>0$, we could generate not only the offset shape (at $t=L$), but a whole class of offset shapes. Hence, it is important to analyze these equations closely and obtain good numerical algorithms for their solution. This was indeed done in the context of flame-propagation models and shape analysis in computer vision, with an emphasis on the analysis of the possible singularities arising on the propagated curves. The following singularities (or 'shocks') may occur[3]:

- If there is a local maximum of the curvature such that $1/k<L$, a curvature-discontinuity 'shock' is formed after time $t=1/k$ (see Figure 1a), and propagating the curve beyond $t=1/k<L$ via Equation 4 forms a 'cusp' (see Figure 1b).
- At places where the original curve has breakpoints and the derivatives are not well defined, problems of determining the intersection of the propagated lines arise (see Figure 1c).
- Self intersections of the propagating curve may occur, giving rise to some difficult topological problems (see Figure 1d).

HISTORICAL REVIEW

The issue of generating offset curves has often been dealt with in the CAD literature. Several approaches to the problem have been proposed. In the CAD/CAM applications that we consider, there are often shape boundaries

 0010-4485/93/030154-09

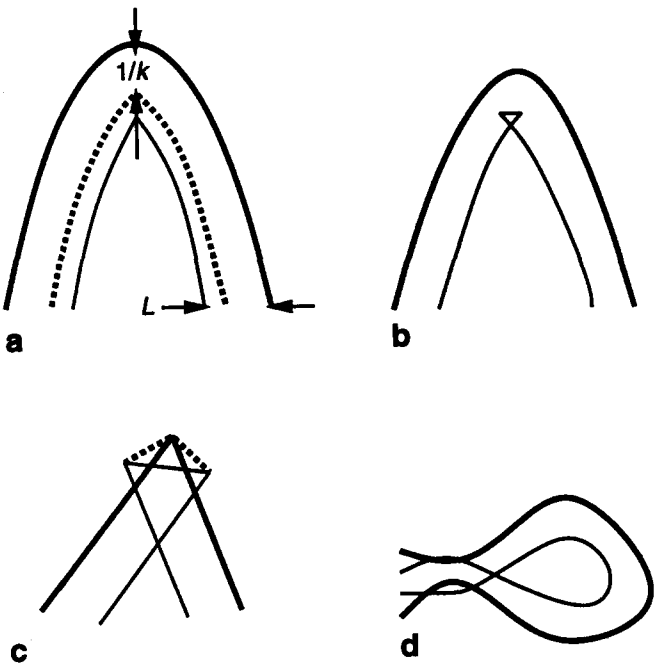


Figure 1. *Problems arise when singularities form in propagated curve*

that are defined by standard parameterized curves such as line segments, circular arcs or splines. In such cases, one could concentrate on determining evolution equations for the control points defining such curves. However, the offset curves of splines are not necessary splines themselves, and straightforward translations of spline control points only rarely produce correct offset curves. Klass[4] approximates the offset of a B-spline segment by moving the endpoints of the curve, and calculating the new tangents based on replacing the curvature k with a $\tilde{k}$ that obeys $1/k = 1/\tilde{k} - L$. He proposes an iterative procedure for calculating the tangents. A spline approximation of the offset curve is generated, and then the distance between the two curves is measured, and, if this distance deviates considerably from L, the spline is split into several spline segments and the process goes on. Tiller and Hanson[5] replace the B-spline parametric representation of the shape by a 'rational B-spline' representation with the control points located on the curve itself. The offset operation is then carried out by moving the control points a distance of L on the normal direction. These approaches require the development of some sophisticated procedures to deal with the problems of loops, shocks, cusps, self-intersection singularities etc. Hoschek[3] attacks the problem of finding self-intersection points on offset curves using an iterative geometrical algorithm. The algorithm is used to eliminate the tails and loops that arise in the offset curves. It requires as input two points that are close enough to each intersection point to guarantee convergence, and it does not solve the so-called island problem (see Figure 1d). Coquillart[6] suggests a new way of translating the rational-B-spline control points to preserve circles. The translation is controlled by the local curvature k and the distance D from the B-spline control points to the given curve. The translation of each control point in the normal direction is given by $\tilde{L} = L(1 + kD)$. Pham[7] uses the uniform cubic B-spline to represent curves. She also uses a version of B-splines with control points on the curve, and the offset curves are improved by adding control points ('knots'). Elber and Cohen[8] measure the distance error to offset curves (and surfaces), and use it to place new control points. Farin[9] provides a recursive procedure to offset Bézier segments that is similar to the one proposed by Klass. Meek and Walton[10] observe that, for clothoidal splines, the offset curve remains 'in the family' (i.e. the offset is also a clothoidal spline).

Wang and Jiang[11] use a vector representation of the shape comprising circular arcs and straight segments only. Clockwise directed vectors form the shape boundaries, and multiplication of consecutive vectors yields vectors indicating the offset direction. In his recent book[12], Held attacks the shape-offsetting problem using Voronoi diagrams. His aim is to design the course of a tool creating a hole 'pocket' in solid material. He restricts his methods to linear and circular segments whose representation can easily be fed into CNC machines. An alternative approach to the offset problem is provided by Saeed *et al.* in Reference 13. They propose the use of morphological methods to formulate the offset operation. Indeed, offset shapes are closely related to dilated or eroded shapes, as defined in mathematical morphology. Related problems are skeleton finders in computer vision and morphology[14], fat curves in computer graphics[15], and the calculation of Euclidean distance maps from plane curves[16].

The algorithms proposed so far for the offsetting problem deal with edge-intersection problems, shocks, cusps and self loops in complex, and rather unnatural, ways. In these algorithms, the curve-offsetting stage is followed by a procedure aimed at detecting the aforementioned problems and repairing the offset curve accordingly. In the sequel, we present a way to approximate the offset shape using level-set propagation on a rectangular grid designed to produce results of the desired accuracy. This is done by applying an algorithm devised by Osher and Sethian[17] for the stable and efficient propagation of wavefront curves in the plane. This numerical method generates offset curves according to Equation 5 and a physically motivated 'entropy condition', and it inherently avoids the topological problems that required special attention with previous algorithms.

NEW ALGORITHM

We propose to generate shape offsets via an ingenious algorithm invented in fluid dynamics for solving equations of the type of Equations 5. This algorithm translates the problem of curve evolution into a problem of 3D-surface evolution, so that the curves changing according to Equations 5 are zero (or level) sets of the time-varying surface. As the 3D surface evolves, it inherently avoids the generation of curve shocks by implementing a physically motivated entropy condition. The algorithm that produces the desired results works

on a square grid with a resolution determined by the desired accuracy of the results, and it is based on a recently discovered efficient numerical implementation of the surface-evolution equations[17].

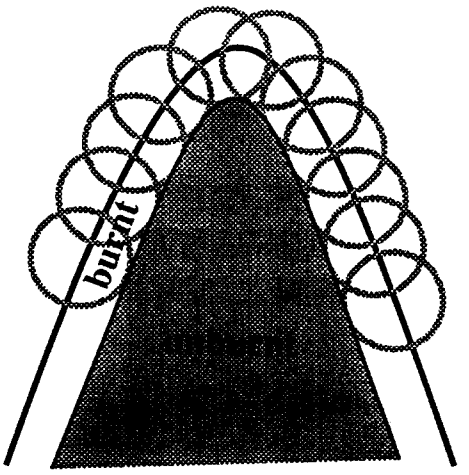
Huygens principle and entropy condition

According to the Huygens principle[18], the solution of the curve propagation of Equations 5 at time t, $\mathbf{X}(s,t)$, corresponds to the envelope generated by the set of all the disks of radius t centred on the initial curve $\mathbf{X}_0(s)$. Problems occur in curve evolution when the normals to the initial curve collide or cross and hence the curvature becomes singular. To obtain the solution according to Huygens' principle after a singularity develops, an 'entropy condition' should be enforced on the propagating curve. One can regard the curve as the wavefront of a propagating prairie fire separating two areas: the shape interior which is not burnt, and the already-burnt exterior. The flame propagates in the direction of the original curve normals (the so-called 'ignition curves'). If two ignition curves collide at some time t^*, neither one should have any effect on the propagating curve at $t > t^*$. The principle 'what was burnt until t cannot burn beyond t'[18] is the natural 'entropy condition' of this type of curve evolution. See Figure 2.

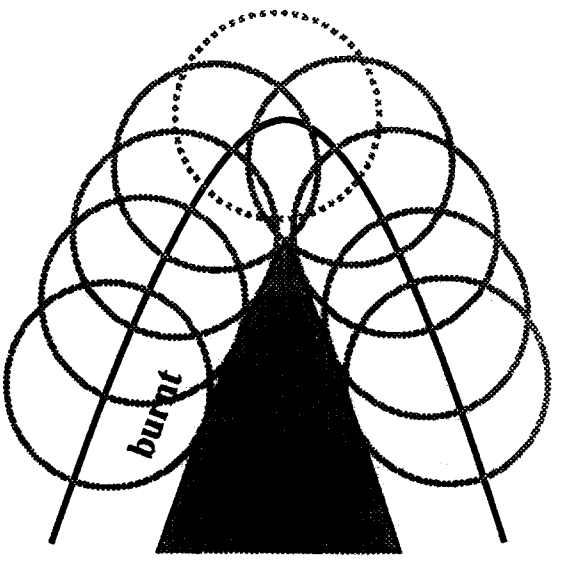
The direct approach to propagating the curve can be referred to as the 'Lagrangian' formulation, because the coordinates (s and t) are front-dependent[19]. The Lagrangian formulation is the direct numerical approximation of Equations 5:

$$\begin{cases} \dfrac{\partial x(s,t)}{\partial t} = -\dfrac{y_s(s,t)}{(x_s^2(s,t)+y_s^2(s,t))^{1/2}} \\ \dfrac{\partial y(s,t)}{\partial t} = \dfrac{x_s(s,t)}{(x_s^2(s,t)+y_s^2(s,t))^{1/2}} \end{cases} \tag{6}$$

Taking the discrete approximation of x_s and y_s as central derivatives in place (s) and a forward-derivative approximation in time (t) yields a numerical-propagation scheme. The direct numerical propagation of a curve according to Equations 6 is both numerically unstable and suffers from topological problems (see References 17 and 19). To avoid the various problems that occur in this approach, such as the need for reparameterization in order to keep numerical stability and to solve topological problems of self intersections by an external control procedure, the 'Eulerian formulation', described below, was developed.



a



b

Figure 2. *The principle 'what was burn until t cannot burn beyond t' is the natural 'entropy condition' of our curve evolution*

[(a) According to Huygens' principle, the front of the evolving curve is constructed by the front of all the disks of radius t centered on the initial curve. (b) If two ignition curves collide at some time t^*, neither one should have any effect on the propagating curve at $t > t^*$. Observe that the dotted circle does not have any effect on the front.]

Solution via Eulerian formulation

The Eulerian scheme is a recursive procedure which propagates the curve while inherently implementing the entropy condition. Introduce a smooth function $\phi(x,y,t)$ that is arbitrarily initialized so that $\phi(x,y,0)=0$ yields the curve $\mathbf{X}(s,0)$. Assume that $\mathbf{X}(s,0)$ is a closed curve, and restrict ϕ to be negative in the interior and positive in the exterior of the level set $\phi(x,y,0)=0$.

The basic idea is to determine an evolution of the surface $\phi(x,y,t)$ so that the level sets $\phi(x,y,t)=0$ provide the curves $\mathbf{X}(s,t)$ as if propagated by Equations 5, and also obey the entropy condition. If $\phi(x,y,t)=0$ along $\mathbf{X}(s,t)$, then, by the chain rule, we have

$$\frac{\partial}{\partial t}\phi(x,y,t)+\frac{\partial}{\partial x}\phi(x(s,t),y(s,t),t)x_t + \frac{\partial}{\partial y}\phi(x(s,t),y(s,t),t)y_t=0$$

or

$$\phi_t+\nabla\phi\mathbf{X}_t(s,t)=0 \tag{7}$$

where

$$\nabla\phi \equiv \left[\frac{\partial}{\partial x}\phi, \frac{\partial}{\partial y}\phi\right]$$

is the gradient of the function $\phi(x, y, t)$ at time t, at the point (x, y). The scalar velocity of each curve point in its normal direction is

$$v = \mathbf{N}(s,t)\cdot\mathbf{X}_t(s,t) \tag{8}$$

In our case, we need to 'impose' $v = 1$. The gradient $\nabla\phi$ is always normal to the curve given by $\phi(x, y, t) = 0$ so that

$$\mathbf{N}(s,t) = -\frac{\nabla\phi}{\|\nabla\phi\|}$$

the minus sign indicating the inward direction of propagation, and hence

$$v = \mathbf{N}\cdot\mathbf{X}_t = -\frac{\nabla\phi}{\|\nabla\phi\|}\mathbf{X}_t = 1 \tag{9}$$

Substituting this into Equation 7 yields the surface-evolution equation

$$\phi_t - \|\nabla\phi\| = 0 \tag{10}$$

Sethian called this approach Eulerian, since the coordinates here are the natural physical coordinates (x, y). Therefore, if we have a surface ϕ propagating according to Equation 10 with the level set $\phi(x, y, 0) = 0$ coinciding with $\mathbf{X}(s, 0)$, then $\phi(x, y, t) = 0$ produces $\mathbf{X}(s, t)$ propagated according to Equations 5, while inherently solving the topological problems. To drive a numerical scheme for the surface-propagation equation, we follow Reference 17, and show the connection with Hamilton–Jacobi methods, weak solutions and conservation laws.

Consider the 1D equation of the type of Equation 10:

$$\phi_t - \|\nabla\phi\| = \phi(x,t)_t - (\phi_x^2)^{1/2} = 0 \tag{11}$$

If we define $u \equiv \phi_x$ and $H[u] = -(u^2)^{1/2}$, the differentiation of the above with respect to x results in a so-called Hamilton–Jacobi equation, in a conservation-law form:

$$u_t + [H[u]]_x = 0 \tag{12}$$

The 'weak' solution of the above equation is defined as a function $u(x, t)$ that satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_a^b u(x,t)\,\mathrm{d}x = H[u(a,t)] - H[u(b,t)] \tag{13}$$

To devise a numerical scheme, define $u_i^n = u(i\Delta x, n\Delta t)$. A differential scheme of three points is said to be in conservation form if there is a 'flow' function $g(u_1, u_2)$ such that

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{g(u_i^n, u_{i+1}^n) - g(u_{i-1}^n, u_i^n)}{\Delta x} \tag{14}$$

where $g(u, u) = H(u)$ is the consistency condition. A scheme is said to be monotone if $u_i^{n+1} = F(u_{i-1}^n, u_i^n, u_{i+1}^n)$ is an increasing monotone function of its three variables. A basic result in numerical analysis is that a scheme which is monotone and can be represented in a conservation form automatically obeys the entropy condition[20].

Some schemes based on this idea, such as the Lax–Friedrichs and Godonov schemes, are presented in Reference 17. The simplest flow function from our implementation point of view is the so-called HJ flow, for which, for $H(u) = f(u^2)$, the numerical flow can be given by using in Equation 14 the function

$$g_{\mathrm{HJ}}(u_i^n, u_{i+1}^n) = f((\min(u_i^n, 0))^2 + ((\max(u_{i+1}^n, 0))^2) \tag{15}$$

and the appropriate (weak) entropy solution of ϕ can be written as

$$\phi_i^{n+1} = \phi_i^n - \Delta t\cdot g(D_-\phi_i^n, D_+\phi_i^n) \tag{16}$$

where

$$D_-\phi_i^n = \frac{\phi_i^n - \phi_{i-1}^n}{\Delta x}$$

and

$$D_+\phi_i^n = \frac{\phi_{i+1}^n - \phi_i^n}{\Delta x}$$

This is a so-called 1st-order scheme. More sophisticated higher-order schemes are introduced in Reference 17. The above scheme is readily extended to more than one dimension; for example, for $H(u, v) = f(u^2, v^2)$ (in our case $u = \phi_x$, $v = \phi_y$),

$$\phi_{ij}^{n+1} = \phi_{ij}^n - \Delta t\cdot g(D_-^x\phi_{ij}^n, D_+^x\phi_{ij}^n;\ D_-^y\phi_{ij}^n, D_+^y\phi_{ij}^n) \tag{17}$$

where

$$g_{\mathrm{HJ}} = f((\min(D_-^x\phi_{ij}^n, 0)^2 + (\max(D_+^x\phi_{ij}^n, 0))^2;\ (\min(D_-^y\phi_{ij}^n, 0))^2 + (\max(D_+^y\phi_{ij}^n, 0))^2) \tag{18}$$

The result is the following algorithm

Algorithm

(1) Choose a function $\phi(x, y, 0)$ such that

- $\phi(x, y, 0) = 0$ provides the initial curve $\mathbf{X}(s, 0)$,
- $\phi(x, y, 0) < 0$ in the interior of the initial curve,
- $\phi(x, y, 0) > 0$ in the exterior of the initial curve,
- $\phi(x, y, 0)$ is Lipschitz-continuous.

The next section discusses the possible ways of 'choosing' the initialization $\phi(x, y, 0)$.

(2) Propagate ϕ on an x–y grid of desired spatial resolution according to

$$\phi_t - \|\nabla\phi\| = 0$$

discretized using any conservation-form numerical scheme.

(3) Stop after $n = L/\Delta t$ time steps, and find the contour (level set) $\phi(x, y, L) = 0$ which is $\mathbf{X}_L(s)$.

See further below possible ways of implementing the level-set finder. The result is a weak solution of Equations 5 that obeys the entropy conditions.

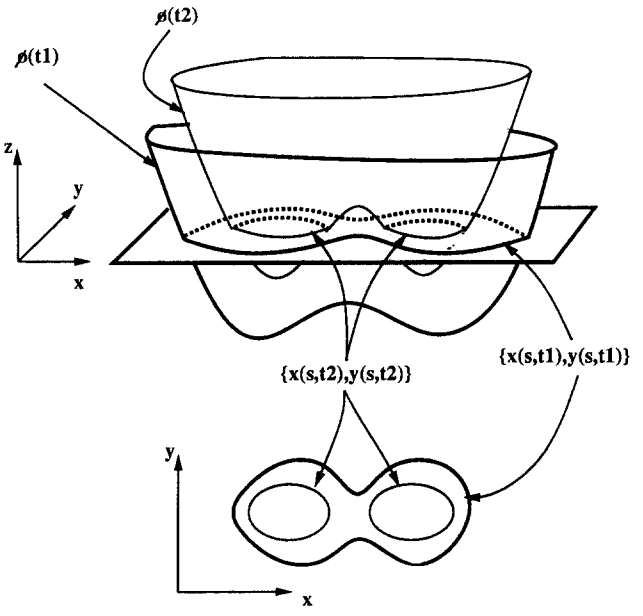


Figure 3. *When ϕ propagates in time, the function may stay continuous while the offset curves form two separate close curves which are no longer connected*

The algorithm automatically enforces the entropy condition, and frees us from the need to take care of possible topological changes (see Figure 3). The algorithm, by choosing forward derivatives, readily deals with shock formation and propagation within the numerical flow.

Initialization

It is obvious that an initialization of the type

$$\phi(x,y,0) = \begin{cases} +d((x,y),\mathbf{X}(s,0)) & (x,y) \in \text{exterior of } \mathbf{X}(s,0) \\ -d((x,y),\mathbf{X}(s,0)) & (x,y) \in \text{interior of } \mathbf{X}(s,0) \\ 0 & (x,y) \in \mathbf{X}(s,0) \end{cases} \tag{19}$$

where d is the (minimal) Euclidean distance of the point from the curve, would be a reasonable initialization, but, on the other hand, such an initialization is, in fact, equivalent to solving the problem of offsetting, since we could produce $\mathbf{X}(s,d)$ as the locus of all the points where $\phi(x,y,0)=d$.

However, to provide a proper solution, we can use the fact that $\phi(x,y,0)$ only needs to be continuous, and initialize $\phi(x,y,0)$ on the x–y grid as follows:

$$\phi(x,y,0) = \begin{cases} \min[+d((x,y),\mathbf{X}(s,0)),C] & (x,y) \in \text{exterior of } \mathbf{X}(s,0) \\ \max[-d((x,y),\mathbf{X}(s,0)),-C] & (x,y) \in \text{interior of } \mathbf{X}(s,0) \\ 0 & (x,y) \in \mathbf{X}(s,0) \end{cases} \tag{20}$$

where C is an arbitrary constant. If we choose $h=\Delta x=\Delta y=C=1$, then the values of the $\phi(x,y,0)$ function on the grid varies in the interval $[-1,1]$. The values of the open interval $(-1,1)$ are given only to grid points at a distance of less than the mesh size from the curve. This initialization problem is quite simple, and can be regarded as a problem of finding a very tight offset neighborhood to the initial curve before topological problems can even begin to affect the results. See Figure 4 for the results of the offsetting process with such an initialization.

Initializing $\phi(x,y,0)$ when the shape outline is a sequence of line segments and circular arcs (the common NC case (see Reference 12)) is quite simple, and it can be dealt with as follows:

- Find all the intersection points of the shape outline with the given grid.
- For each grid point (i,j), define a cell as $\mathcal{N}_{ij}=[\phi_{i,j}, \phi_{i+1,j}\phi_{i+1,j+1}, \phi_{i,j+1}]$. For each grid point (i,j), check whether the boundaries of the cell $\mathcal{N}_{ij}$ are intersected by the shape boundaries.

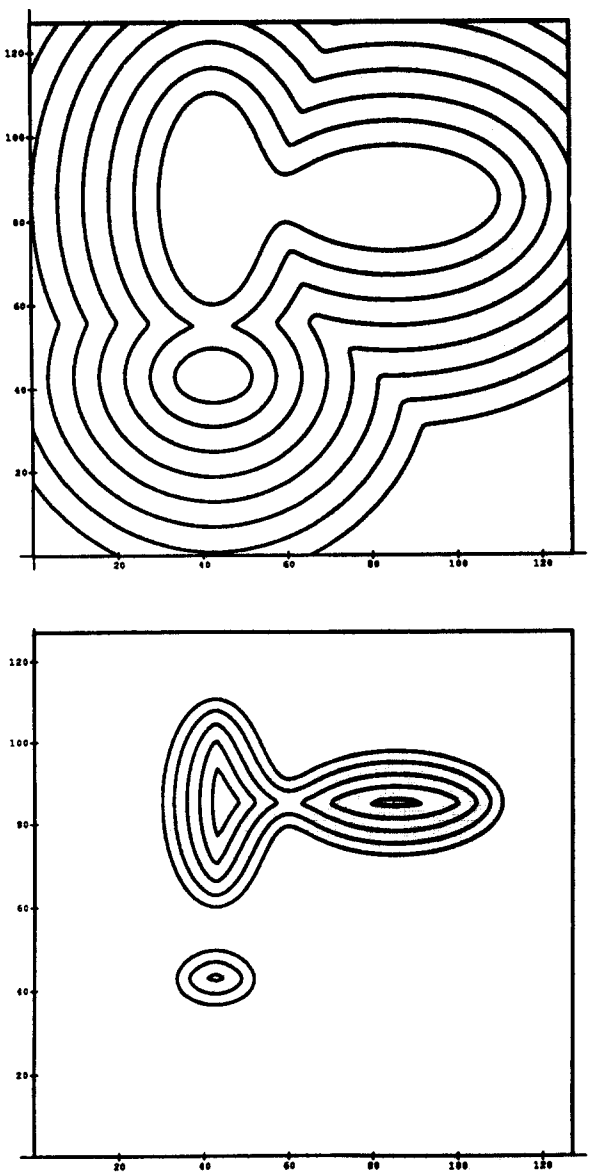


Figure 4. *Offsetting of simple shape on 128 × 128 grid*

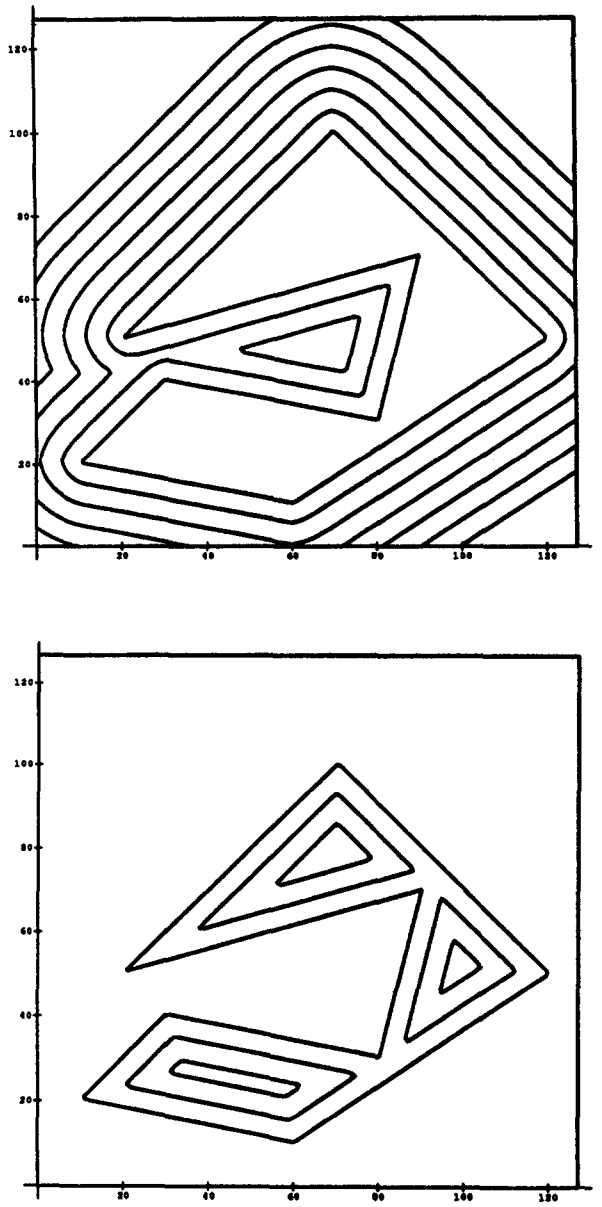


Figure 5. *Offsetting polygons*

- If there is an intersection, attach a value to the two grid points on the two sides of the intersection point so that the interior grid point is given a value within the interval $(-1, 0]$, and the exterior grid point a value within $[0, +1)$. These two values are samples of a linear approximation of the ϕ function whose zero is the intersection point.
- Set values for the rest of the grid points as follows: assign -1 to the interior points and $+1$ to the exterior points of the given shape. (Chains of 'dependent' points may occur. When this happens, attach values to the dependent points so that the intersection points are the zeros of the linear ϕ approximation). Figure 5 shows an example of polygon offsetting using this initialization process.

Note that, if we must offset a shape that is provided as a dark region on a light background in a given picture, *we can use the gray levels of the image in order to initialize* $\phi(x, y, 0)$. For example, if the gray level of the shape is black ($\simeq 0$), the background is white ($\simeq 1$), and the boundaries pass through the gray level gray ($\simeq 1/2$), then we can take $\phi(x, y, 0) = I(x, y) - 1/2$, as the required initialization, making direct use of the continuity of the gray levels in the picture, without any extra calculations (see Figure 6 for the Postscript halftoned version of a gray-level test picture, and Figure 7 for the offsets of this shape).

After initialization has been completed, the function ϕ is propagated according to the above-described algorithm for $n = L/\Delta t$ iteration steps. Finally, a contour finder must be invoked to produce the resulting contour $\mathbf{X}(s, L)$ from $\phi(x, y, L) = 0$. A description of the offsetting algorithm can be found in Appendix B.

Contour finder

Following Reference 21, a simple contour finder for $\mathbf{X}(s, L)$ can be generated in the following manner: for each grid point (i, j), use the same cell definition as in the previous section $\mathcal{N}_{ij}$. Now, if $\max[\mathcal{N}_{ij}] < 0$ or $\min[\mathcal{N}_{ij}] > 0$, then the contour $\mathbf{X}(s, L)$ does not pass through the cell. Otherwise, find the entrance and exit points of $\phi = 0$ by linear interpolation; this provides a line segment of $\mathbf{X}(s, L)$ belonging to the contour. The line segments need to be neither ordered nor directed in the same direction to *display* the desired contour (see Figure 8); however, using additional information such as the knowledge of interior and exterior points, one can produce any desired representation of the curve, such as a planar polygon, or a curve comprising cubic or any other polynomial arcs.

Figure 6. *Mickey Mouse in gray-level picture (128 × 128 pixels)*

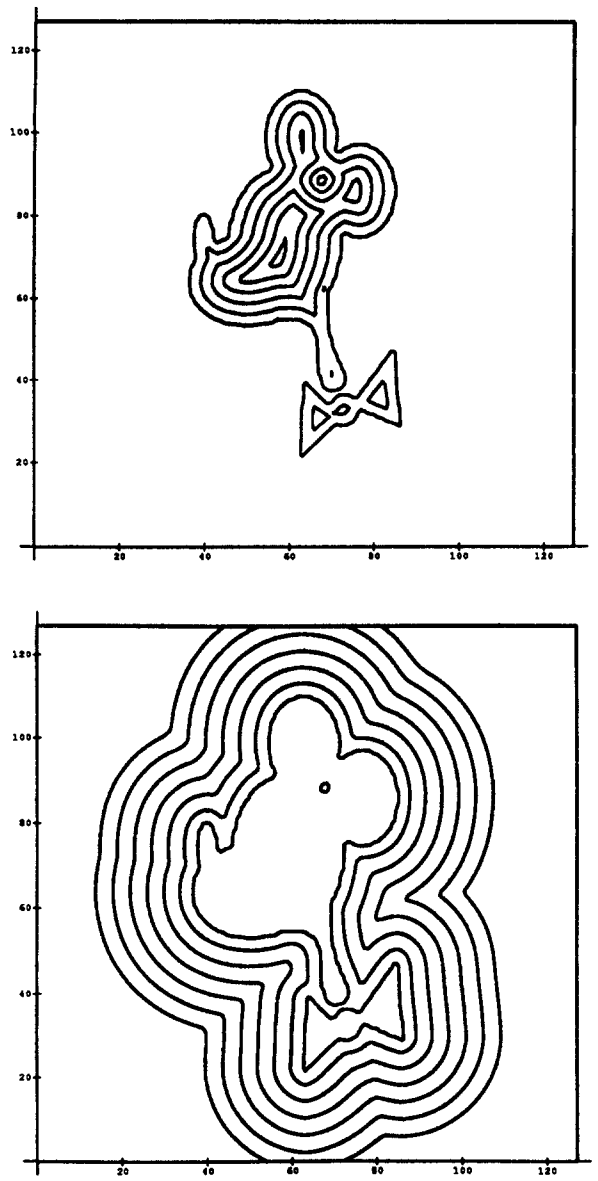


Figure 7. *Offsetting Mickey Mouse (grid size 128)*

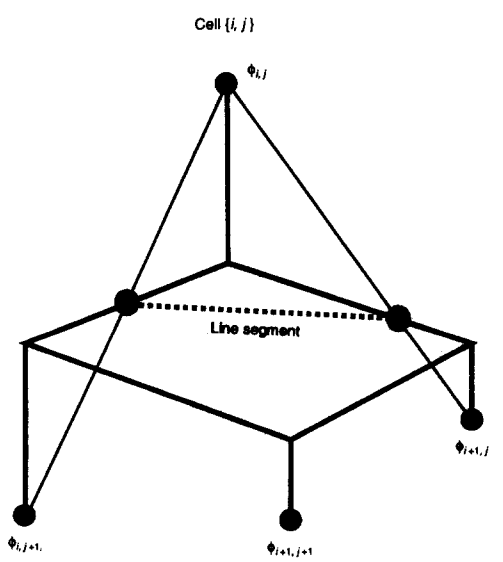


Figure 8. *Contour finder: find a line segment in* $\mathcal{N}_{ij}$

CONCLUDING REMARKS

We have described a new method of generating shape offsets so that topological problems are inherently avoided, and shocks, cusps and other singularities are also readily dealt with in an efficient numerical scheme. The algorithm works on an image grid with a resolution chosen according to the desired accuracy. It is easy to implement the algorithm in parallel using each mesh point as a small calculating device which communicates with its four close neighbors. In each iteration, we need to calculate the values of $\phi(x, y, t)$ for those grid points close to the current contour, and the rest of the grid points serve as sign holders. This can be exploited to reduce calculation effort.

In summary, we propose to introduce to the CAD/CAM field some recent advances in numerical methods for fluid dynamics. We have shown that wavefront-propagation methods in fluid dynamics also provide a new approach to the problem of shape offsetting.

ACKNOWLEDGEMENTS

We would like to thank Professor James Sethian for providing us with his reports on numerical methods of front evolution. We thank Professor Moshe Israeli, Professor Ben Kimia, Professor Allen Tannenbaum and Professor J Rubinstein for their help in introducing us to the world of numerical analysis and moving surfaces.

R Kimmel's work was partly supported by the Ollendorf Fund. A Bruckstein's work was supported in part by the Fund for the Promotion of Research at the Technion, Haifa, Israel.

REFERENCES

1 **Blum, H and Nagel, R N** 'Shape description using symmetric axis features' *Pattern Recognition* Vol 10 (1978) pp 167–180

2 **Calabi, L and Hartnett, W E** 'Shape recognition, prairie fires, convex deficiencies and skeletons' *Amer. Math. Mon.* Vol 75 (1968) pp 335–342

3 **Hoschek, J** 'Offset curves in the plane' *Comput.-Aided Des.* Vol 17 No 2 (1985)

4 **Klass, R** 'An offset spline approximation for plane cubic splines' *Comput.-Aided Des.* Vol 15 No 5 (1983) pp 297–299

5 **Tiller, W and Hanson, E G** 'Offset of two dimensional profiles' *IEEE Trans. Comput. Graph. & Applic.* Vol 4 No 9 (1984) pp 36–46

6 **Coquillart, S** 'Computing offsets of B-spline curves' *Comput.-Aided Des.* Vol 19 No 6 (1987) pp 305–309

7 **Pham, B** 'Offset approximation of uniform B-splines' *Comput.-Aided Des.* Vol 20 No 8 (1988) pp 471–474

8 **Elber, G and Cohen, E** 'Error bounded variable distance offset operator for free form curves and

surfaces' *Int. J. Comput. Geom. & Applic.* Vol 1 No 1 (1991) pp 67–78

9 **Farin, G** 'Curvature continuity and offsets of piecewise conics' *ACM Trans. Graph.* Vol 8 No 2 (1989) pp 89–99

10 **Meek, D S and Walton, D S** 'Offset curves of clothoidal splines' *Comput.-Aided Des.* Vol 22 No 4 (1990) pp 199–200

11 **Wang, R and Jiang, W** 'An algorithm of the offset shape' *Comput. Graph.* Vol 15 No 3 (1991) pp 435–439

12 **Held, M** *On the Computational Geometry of Pocket Machining* Springer-Verlag, Germany (1991)

13 **Saeed, S E O, de Pennington, A and Dodsworth, J R** 'Offsetting in geometric modeling' *Comput.-Aided Des.* Vol 20 No 1 (1988) pp 67–74

14 **Riazanoff, S, Cervelle, B and Chorowicz, J** 'Parametrisable skeletonization of binary and multi-level images' *Pattern Recognition Lett.* No 11 (1990) pp 25–33

15 **Yao, C and Rokne, J** 'Fat curves' *Comput. Graph. Forum* No 10 (1991) pp 237–248

16 **Vincent, L** 'Exact Euclidean distance function by chain propagation' *IEEE Trans. Comput. V. Pattern Recognition* (May 1991)

17 **Osher, S J and Sethian, J A** 'Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations' *J. Comput. Phys.* Vol 79 (1988) pp 12–49

18 **Sethian, J A** 'Curvature and the evolution of fronts' *Comm. Math. Phys.* Vol 101 (1985) pp 487–499

19 **Sethian, J A** 'A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed' *J. Differ. Geom.* No 33 (1989) pp 131–161

20 **Sod, G A** *Numerical Methods in Fluid Dynamics* Cambridge University Press (1985)

21 **Sethian, J A and Strain, J** 'Crystal growth and dendritic solidification' *J. Comput. Phys.* Vol 98 (1992)

22 **Kimia, B B, Tannenbaum, A and Zucker, S W** 'On the evolution of curves via a function of curvature. Part 1: The classical case' *J. Math. Anal. & Applic.* Vol 163 (1992) pp 438–458

23 **Kimia, B B** 'Toward a computational theory of shape' *PhD Thesis* Dep. Electrical Engineering, McGill University, Canada (1990)

BIBLIOGRAPHY

Mitchel, A R and Griffiths, D F *The Finite Difference Method in Partial Differential Equations* John Wiley, USA (1985)

Sethian, J A 'Numerical methods for propagating fronts' *in* **Concus, P and Finn, R (Eds.)** *Variational Methods for Free Surface Interfaces* Springer-Verlag, USA (1987)

APPENDIX A

Normal evolution

In this section, we show that, in a curve evolution described by

$$\frac{\partial}{\partial t}\mathbf{X}(s,t)=\mathbf{N}(s,t)$$

the direction of the normal does not change in time prior to shock formation (following References 17, 22 and 23).

Proof: Let us first define the metric along the curve and the arc-length parameter $\tilde{s}$:

$$g(s,t)\equiv\left|\frac{\partial}{\partial s}\mathbf{X}\right|=(x_s^2+y_s^2)^{1/2}$$

$$\tilde{s}(s,t)\equiv\int_0^s g(\xi,t)\,\mathrm{d}\xi$$

Now we can define the tangent, curvature and the normal in a standard way:

$$T\equiv\frac{\partial}{\partial\tilde{s}}\mathbf{X}=\frac{1}{g}\frac{\partial}{\partial s}\mathbf{X}$$

$$k\equiv\left|\frac{\partial}{\partial\tilde{s}}\mathbf{T}\right|=\frac{1}{g}\left|\frac{\partial}{\partial s}\mathbf{T}\right|$$

$$\mathbf{N}\equiv-\frac{\dfrac{\partial}{\partial\tilde{s}}\mathbf{T}}{\left|\dfrac{\partial}{\partial\tilde{s}}\mathbf{T}\right|}=-\frac{1}{kg}\frac{\partial}{\partial s}\mathbf{T}$$

It is easily established that the curve evolution induces

$$\frac{\partial}{\partial s}\mathbf{T}=-kg\mathbf{N}$$

$$\frac{\partial}{\partial s}\mathbf{N}=kg\mathbf{T}$$

The evolution of g can be computed as follows:

$$\begin{aligned}\frac{\partial g^2}{\partial t}&=\frac{\partial}{\partial t}\left\langle\frac{\partial}{\partial s}\mathbf{X},\frac{\partial}{\partial s}\mathbf{X}\right\rangle\\&=2\left\langle\frac{\partial}{\partial s}\mathbf{X},\frac{\partial}{\partial t}\frac{\partial}{\partial s}\mathbf{X}\right\rangle\\&=2\left\langle g\mathbf{T},\frac{\partial}{\partial s}\mathbf{N}\right\rangle\\&=2\langle g\mathbf{T},kg\mathbf{T}\rangle\\&=2g(kg)\end{aligned}$$

Hence,

$$\frac{\partial g}{\partial t} = kg$$

Then, the evolution of the tangent is as follows:

$$\begin{aligned}\frac{\partial}{\partial t}\mathbf{T} &= \frac{\partial}{\partial t}\left(\frac{1}{g}\frac{\partial}{\partial s}\mathbf{X}\right)\\ &= -\frac{g_t}{g^2}\frac{\partial}{\partial s}\mathbf{X} + \frac{1}{g}\frac{\partial}{\partial t}\frac{\partial}{\partial s}\mathbf{X}\\ &= -\frac{g_t}{g}\mathbf{T} + \frac{1}{g}\frac{\partial}{\partial s}\mathbf{N}\\ &= -\frac{kg}{g}\mathbf{T} + \frac{1}{g}kg\mathbf{T}\\ &= 0\end{aligned}$$

Thus, the normal (to the stationary tangent direction) is also stationary; in other words, there is no change in the normal direction while the curve evolves, and no shocks appear. Therefore, Equations 4 are equivalent to Equations 5. □

APPENDIX B

Offsetting algorithm

(1) Initialize $\phi_{ij}^0 \equiv \phi(ih, jh, 0)$, where $h = \Delta x = \Delta y$, according to the curve representation in hand. $\phi(x, y, 0) = 0$ is the implicit representation of the initial curve. See above for ways in which to determine $\phi(x, y, 0)$. Note that, if shape is defined by a dark region on a white background, set $\phi(x, y, 0)$ equal to the gray-level image of the object at the camera resolution.

(2) For $n = 1$ to $L/\Delta t$ do

$$\begin{aligned}\phi_{i,j}^{n+1} = \phi_{i,j}^n + \frac{\Delta t}{h}\{&(\min[(\phi_{i,j}^n - \phi_{i-1,j}^n), 0])^2\\ &+ (\max[(\phi_{i+1,j}^n - \phi_{i,j}^n), 0])^2\\ &+ (\min[(\phi_{i,j}^n - \phi_{i,j-1}^n), 0])^2\\ &+ (\max[(\phi_{i,j+1}^n - \phi_{i,j}^n), 0])^2\}^{1/2}\end{aligned}$$

(3) Find the level set $\phi(x, y, L) = 0$ by activating the contour-finder procedure as described above on $\phi_{i,j}^n$.