A Fast Algorithm for Active Contours

Donna J. Williams
Stetson University
DeLand, FL 32720

Mubarak Shah
University of Central Florida
Orlando, FL 32816

Abstract

A model for representing contours in an image in a form that allows interaction with higher level processes has been proposed by Kass *et al.* [2]. This active contour model is defined by an energy functional, and a solution is found using techniques of variational calculus. Amini *et al.* [1] have pointed out some of the problems with this approach and proposed an algorithm for the active contour model using dynamic programming. This approach is more stable and allows the inclusion of hard constraints in addition to the soft constraints inherent in the formulation of the functional; however it is slow, having complexity $O(nm^3)$.

In this paper we present a greedy algorithm which gives results comparable to the dynamic programming algorithm but is much faster, being $O(nm)$. Because the concept of curvature is basic to the formulation of the contour functional, formulas for approximation of curvature of discrete curves are presented and evaluated.

Introduction

The problem of how to represent a set of points which have been determined to lie on an edge is a challenging one. Kass, Witkin, and Terzopoulos [2] have proposed a model called active contours (snakes) which has the advantage that the final form of a contour can be influenced by feedback from a higher level process. The contour is initially placed near an edge under consideration, then image forces draw the contour to the edge in the image. As the algorithm iterates, the energy terms can be adjusted by higher level processes to obtain a local minimum that seems most useful to that process. However, there are some problems with the minimization procedure used. Amini *et al.* pointed out some of these, including instability and a tendency for points to bunch up on a strong portion of an edge. They have proposed a dynamic programming algorithm for minimizing the energy functional that allows addition of hard constraints to obtain more desirable behavior of the snakes.

In this paper we present some of the problems in both methods and propose a further algorithm which is stable, flexible, allows hard constraints, and runs much faster than the dynamic programming method. The energy functional being minimized has a continuity term and a curvature term in addition to the image energy and external energy terms. In the method presented here a formulation is used for the continuity term which spaces the points more evenly on the contour, rather than minimizing distance between points as in the previous methods. Discrete approximation of curvature in an accurate and efficient manner is a necessary for the curvature term, so a number of different approximations are examined and evaluated.

Minimum Energy Contours

The Active Contour Model of Kass *et al.* [2] was represented by a vector, $\mathbf{v}(s) = (x(s), y(s))$ having arc length, s, as parameter.[1] They defined an energy functional for the contour by:

$$E^*_{snake} = \int_0^1 E_{int}(\mathbf{v}(s)) + E_{im}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s))ds \quad (1)$$

E_{int} represents the internal energy of the contour due to bending or discontinuities, E_{im} is the image forces, and E_{con} is the external constraints. The image forces can be due to various events, e.g., lines, edges, and terminations. The internal spline energy is written:

$$E_{int} = (\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2)/2 \quad (2)$$

The first-order term will have larger values where there is a gap in the curve, and the second-order continuity term will be larger where the curve is bending rapidly. The values of α and β at a point determine the extent to which the contour is allowed to stretch or bend at that point. If α is 0 at a point, a discontinuity can occur, while if β is 0, a corner can develop. The minimum energy contour is determined using techniques of variational calculus.

Amini, Tehrani, and Weymouth [1] point out some of the problems involved in this approach and propose that the contour having minimum energy be determined using dynamic programming rather than variational calculus. This allows the introduction of *hard* constraints that cannot be violated, as well as the first- and second-order continuity constraints which are inherent in the problem formulation. These latter are known as *soft* constraints because they are not satisfied absolutely, only to a certain degree. Their approach also had the advantage of using points on the discrete grid and is numerically stable. However the method is very slow, $O(nm^3)$, where n is the contour size and

[1] Lower-case, bold letters like $\mathbf{v}$ will be used to denote vectors when they are interpreted as points, while lower-case, bold letters with an arrow above ($\vec{\mathbf{u}}$) will be used when the quantity represented is a vector from one point to another.

CH2934-8/90/0000/0592$01.00 © 1990 IEEE

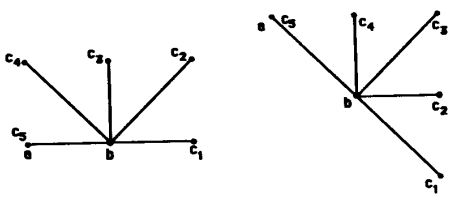

Figure 1: Arrangement of points for Table 1.

m is the neighborhood size. Neither author gives any guidelines for choosing the parameters involved, and use the same values of the parameters at every point.

Curvature Estimation

Both Kass *et al.* and Amini *et al.* approximate the derivatives in Equation 2 by finite differences. If $\mathbf{v}_i = (x_i, y_i)$ is a point on the contour, the following approximations are used:

$$\left|\frac{d\mathbf{v}_i}{ds}\right|^2 \approx |\mathbf{v}_i - \mathbf{v}_{i-1}|^2, \left|\frac{d^2\mathbf{v}_i}{ds^2}\right|^2 \approx |\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}|^2 \quad (3)$$

These formulas assume that the points on the contour are spaced at unit intervals and that the parameter is arc length. One effect of this is that unevenly spaced points will have higher curvature. The equation for curvature for a general parameter is given by

$$\kappa = \frac{|x'y'' - x''y'|}{(x'^2 + y'^2)^{3/2}} \quad (4)$$

It is not clear what measure of curvature is the best reflection of the geometric situation depicted by a contour. Table 1 gives a comparison of five possible curvature measures applied to the situations in Figure 1. Values in column one are reasonable and all fall in the range $[0, \pi]$ because digitization limits curvature, but the method is computationally expensive. Column two, using equation 4 is also slow. It allows values to become infinitely large. In column three, the bottom section shows a case where curvature decreases as the angle increases because of unevenly spaced points. A similar effect is seen in the top section of column four, which however has the advantage of being computationally very efficient. Column five depends solely on the *angle* between $\vec{\mathbf{u}}_i$ and $\vec{\mathbf{u}}_{i+1}$. When points are evenly spaced, column three multiplied by Δs gives column five; multiplying by Δs^2a gives column four.

Greedy Algorithm

In this section a greedy algorithm is presented which allows a contour with controlled first and second order continuity to converge on an area of high image energy, in this case edges. This algorithm allows the inclusion of hard constraints as described by Amini *et al.*, [1] but is much faster than their $O(nm^3)$ algorithm,

c	(1) $(d\theta/ds)^2$	(2) κ^2	(3) $\lvert\mathbf{v}_{ss}\rvert^2$	(4) $\lvert\Delta\vec{\mathbf{u}}_i\rvert^2$	(5) $\lvert\Delta\frac{\vec{\mathbf{u}}_i}{\lvert\vec{\mathbf{u}}_i\rvert}\rvert^2$
1	0.0	0.0	0.0	0.0	0.0
2	0.42	0.512	0.40	1.0	0.59
3	2.47	8.0	2.0	2.0	2.0
4	3.80	64.0	2.34	5.0	3.41
5	9.87	∞	4.0	4.0	4.0
1	0.0	0.0	0.0	0.0	0.0
2	0.42	0.512	0.40	1.0	0.59
3	1.23	4.0	1.0	4.0	2.0
4	3.80	64.0	2.34	5.0	3.41
5	4.93	∞	2.0	8.0	4.0

Table 1: Comparison of estimates of the square of the curvature using different methods. The first section of the table is the horizontal situation of Figure 1, the lower section is the diagonal case.

being $O(nm)$, for a contour having n points which are allowed to move to any point in a neighborhood of size m at each iteration. While the algorithm is not guaranteed to give a global minimum, experimental results were comparable to the other methods.

The quantity being minimized by this algorithm is

$$E = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})\, ds \quad (5)$$

The first and second terms are first- and second-order continuity constraints. They correspond to E_{int} in Equation 1. The last term measures some image quantity, in these examples edge strength, and is the same as the middle term of Equation 1. A term for external constraints could be included. The parameters α, β, and γ are used to balance the relative influence of the three terms.

The proposed algorithm is iterative, as are those of Kass and Amini. During each iteration, a neighborhood of each point is examined and the point in the neighborhood giving the smallest value for the energy term is chosen as the new location of the point.

Using $|\mathbf{v}_i - \mathbf{v}_{i-1}|^2$ for the first term causes the curve to shrink, as this is actually minimizing the distance between points. It also contributes to the problem of points bunching up on strong portions of the contour. It was decided that a term which encouraged even spacing of the points would satisfy the original goal of first-order continuity without the effect of shrinking. Thus the algorithm presented here uses the difference between the average distance between points, $\bar{d}$, and the distance between the two points under consideration: $\bar{d} - |\mathbf{v}_i - \mathbf{v}_{i-1}|$. Thus points having distance near the average will have the minimum value. At the end of each iteration a new value of $\bar{d}$ is computed.

The second term in Equation 5 is curvature. Since

the formulation of the continuity term causes the points to be relatively evenly spaced, $|\mathbf{v}_{i-1}-2\mathbf{v}_i+\mathbf{v}_{i+1}|^2$ gives a reasonable and quick estimate of curvature. This term, like the continuity term, is normalized by dividing by the largest value in the neighborhood, giving a number from 0 to 1.

The third term in Equation 5, E_{image}, is the image force which is gradient magnitude (mag). The maximum and minimum gradient in each neighborhood is determined, and $(min - mag)/(max - min)$ is used for the normalized edge strength term. This gradient magnitude term is negative so that points with large gradient will have small values. The minimum value allowed in the denominator was 5 when magnitude was in the range 0-255.

At the end of each iteration the curvature is determined at each point on the new contour, and if the value is a curvature maximum, β_i is set to 0 for the next iteration. This step functions as a primitive high level process giving feedback to the energy minimization step and is not an essential part of the algorithm.

Figure 2 demonstrates how the algorithm works. The energy function is computed for the current location of $\mathbf{v}_i$ and each of its neighbors. The location having the smallest value is chosen as the new position of $\mathbf{v}_i$.

Pseudo-Code for Greedy Algorithm:

n = number of points. m = size of neighborhood. Index arithmetic is modulo n.
Initialize α_i, β_i, and γ_i to 1 for all i.
do
  /* loop to move points to new locations */
  for $i = 0$ **to** n /*process point 0 first and last */
    E_{min} = BIG
    for $j = 0$ to $m - 1$
      $E_j = \alpha_i E_{cont,j} + \beta_i E_{curv,j} + \gamma_i E_{image,j}$
      if $E_j < E_{min}$ **then**
        $E_{min} = E_j$
        $jmin = j$
    Move point $\mathbf{v}_i$ to location $jmin$
    if $jmin$ **not** current location, $ptsmoved++$
  /* count points moved */
until $ptsmoved < threshold$

Experimental Results

In order to demonstrate the performance of the algorithm described in the previous section, results are given for the greedy algorithm developed above, for the original variational calculus solution, and for the dynamic programming algorithm. These programs were run on two real images: Bottle (Figure 3), and Cup (Figure 4). The Cup image tested the behavior of the algorithms when the contour spanned a region where the edge was weak or missing. In the image figures, the points on the contour having high curvature are marked with larger squares. At these points the second-order continuity restraint was relaxed. The neighborhood examined at each point consisted of the point itself and its eight neighbors. Thus the neighborhood size, m, was 9. In the image figures, (a) shows the beginning contours, all of which had 40-60 points spaced a distance of approximately 4-6 pixels apart. When corners of more than 30° formed β was set to 0.

Part (b) shows the final contour obtained using the variational calculus method proposed by Kass *et al.* Part (c) shows the result of the dynamic programming algorithm for the two pictures. In order to reduce the tendency to bunch up at strong points on the contour, a hard constraint prohibited movement perpendicular to the direction of maximum gradient. This constraint also reduced convergence time.

Part (d) in each figure shows the results of the greedy algorithm. The parameters were $\alpha = 1$, β is either 0 or 1, and $\gamma = 1.2$, so that the image gradient had more importance than either of the continuity terms in determining where points on the contour moved. The results achieved by all three of the methods presented are comparable, one giving slightly better results in one image, while a different method gives better results in another image. Corners are not set with the Kass method, so it gives contours that are more rounded at the corners. As expected, the contour does not follow the right side of the cup well with any of the methods. Where the cup edges are not strong, points belonging to the background appear to be the points converged to with the greedy algorithm. All three converged to the shadow edge at the right-hand bottom corner of the cup rather than to the cup itself, since that was the first edge encountered as the contour approached the cup.

Table 2 compares performance on four images. Values given are the user times in seconds and the number of iterations required to converge. The algorithms were implemented in C on a Harris HCX9 minicomputer. Using the greedy algorithm, the speedup over dynamic programming was significant in all cases, varying from a factor of 13 for the Box, to 48 for the Cup and Square. Neither method was significantly better in the number of iterations required. The results of the contours obtained with the greedy algorithm are at least as good as those of the dynamic programming algorithm, and the run times are much better. The variational calculus approach required time comparable to the greedy method for each iteration, but usually converged in fewer iterations. If values of β were allowed to change between iterations, the inverse of a pentadiagonal matrix would need to be computed, slowing its speed.

	Greedy		Dynamic Prog.		Var. Calc.	
image	secs.	iter.	secs.	iter.	secs.	iter.
square	0.25	2	12.16	3	.35	4
box	1.87	15	25.16	6	.47	4
bottle	1.22	11	29.47	8	1.50	12
cup	0.70	7	34.15	10	.30	4

Table 2: Comparison of runtime in seconds and number of iterations for the greedy, dynamic programming, and variational calculus methods.

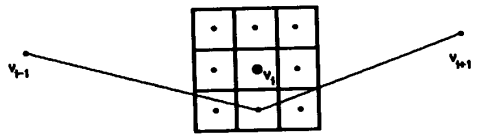

Figure 2: The energy function is computed at $\mathbf{v}_i$ and each of its eight neighbors. The point before and after it on the contour are used in computing the continuity constraints. The location having the smallest value is chosen as the new position of $\mathbf{v}_i$.

Conclusions A method of controlling snakes has been presented which combines speed, flexibility, and simplicity. It was compared to the original variational calculus method of Kass *et al.* and the dynamic programming method developed by Amini *et al.* and found to be comparable in final results, while being faster than dynamic programming and more stable and flexible for including hard constraints than the variational calculus approach. The introduction of the concept of curvature highlighted the problem of how to approximate curvture when a curve is represented by a set of discrete points. The advantages and disadvantages of a number of different approximations of curvature were pointed out.

References

[1] A. A. Amini, S. Tehrani, and T. E. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *Proc. Second International Conference on Computer Vision*, pages 95–99, 1988.

[2] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. of First International Conf. on Computer Vision*, pages 259–269, 1987.

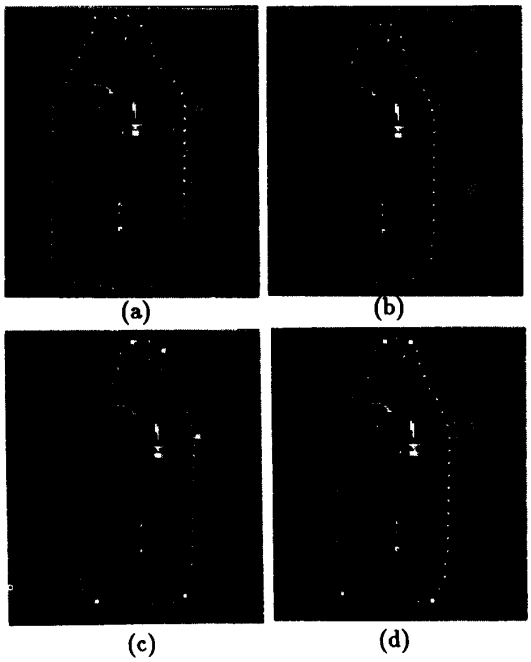

Figure 3: Bottle. (a) Original contour, (b) Kass method, (c) Dynamic programming algorithm, (d) Greedy algorithm.

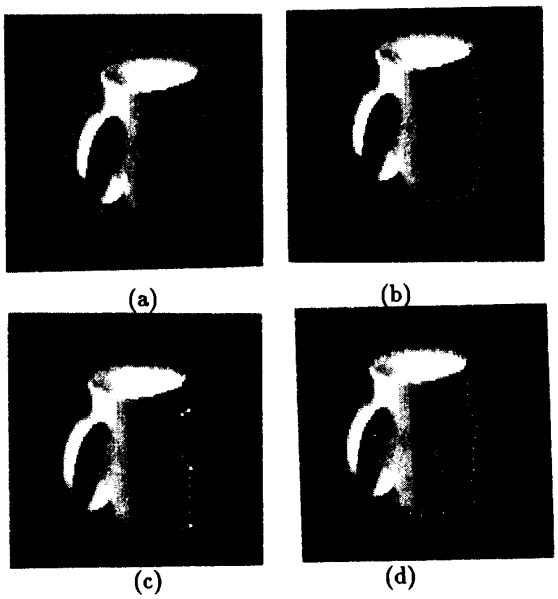

Figure 4: Cup. (a) Original contour, (b) Kass method, (c) Dynamic programming algorithm, (d) Greedy algorithm.