# Using Dynamic Programming for Minimizing the Energy of Active Contours in the Presence of Hard Constraints [1]

Amir A. Amini, Saeid Tehrani, and Terry E. Weymouth
Artificial Intelligence Laboratory
Electrical Engineering and Computer Science
University of Michigan - Ann Arbor 48109

## Abstract

Energy-Minimizing Active Contour Models (snakes) have recently been proposed by Kass *et al.* [8] as a top-down mechanism for locating features of interest in images. The Kass *et al.*'s algorithm involves four steps: setting up a variational integral on the continuous plane, deriving a pair of Euler equations, discretizing them, and solving the discrete equations iteratively until convergence. This algorithm suffers from a number of problems.

We discuss these problems and present an algorithm for active contours based on dynamic programming. The optimization problem is set up as a discrete multi-stage decision process and is solved by a "time-delayed" discrete dynamic programming algorithm. This formulation leads to a stable behavior for the active contours over iterations, in addition to allowing for hard constraints to be enforced on the behavior of the solution. Results of the application of the proposed algorithm to real images is presented.

## 1 Introduction

Energy-Minimizing Active Contour Models (snakes) have recently been proposed by Kass, Witkin, and Terzopolous as a top-down mechanism for locating features of interest in images [8]. Initially, the user places a snake near an image structure of some sort. The constraint forces that act on a snake then push the user-defined snake towards features of interest. The energy of a snake depends on where the snake is placed and how its shape changes locally in space. The idea is to have the snake lock on to features of an image structure by minimizing an integral measure which represents the snake's total energy. In the Kass *et al.*'s formalism this integral measure is minimized by solving a pair of Euler equations iteratively on the discrete grid. A number of difficulties arises when using the algorithm due to Kass *et al.*

A snake may try to minimize its energy by moving the points along the contour where there is a lower energy field. In fact, given a moderately noisy force field, it is not surprising to see points on a snake clustering into a dense structure at certain places. It is even possible for points to move on top of one another. This is because there is no constraint for the interdistance of the points on the contour. For the active contours, formalizing the problem as Kass *et al.*'s requires the energy to be a differentiable function. This requirement rules out the possibility of enforcing hard constraints on the outcome of the algorithm.

In addition, there is a need for estimates of high-order derivatives of the discrete data. This may lead the contours to have unpredictable behavior. In order to handle the large variations in the high order derivatives of the external forces, the image must be heavily filtered resulting in poor localization for boundaries. One may need to resort to scale space strategies in such cases, as discussed in [8]. However, tracking the snake in scale space could potentially be the source of new errors and problems.

Convergence of the iterative process to meaningful and physical features is not straightforward and requires selectively choosing the parameters. The iterative process attempts to solve for the global minimum and in general one can not say much about optimality after an iteration.

In this paper, we present a new algorithm for minimizing the energy of active contour models. We generalize the result of Kass *et al.* by allowing "hard constraints" to become an integral part of the minimization process. In addition, we take advantage of the discrete nature of the problem by carrying out the minimization on the discrete grid. Representing the problem in this way leads to a stable behavior for the active contours over iterations. For the minimization process, we employ a "time-delayed" discrete dynamic programming algorithm. [1]

Local geometry of the active contour models are constrained effectively by using hard constraints. Hard constraints are to be differentiated from "soft constraints," i.e., where the constraint is explicitly made a part of the overall energy measure. Soft constraints alone may seem adequate. However, with soft constraints alone, the behavior of the active contours can not be controlled effectively. In the variational formalism, one can not enforce hard constraints on the active contours. This statement stems from the fact that the energy terms which are external to the snake need be differentiable with respect to the coordinate axes.

In the discrete dynamic programming formulation, each iteration results in a locally optimum contour which has minimum possible energy. Moreover, the active contour is guaranteed to converge to a final solution in a finite number of iterations since the cost measure is monotonically decreasing with time. Our algorithm halts when there is no change in the cost. In addition, our problem representation only deals with the gradient of the image data as an external force field, and first and second derivatives of the active contour data. There are clear advantages to using lower order derivatives of data whenever possible.

This paper is organized as follows. First we review the variational approach to energy minimization for the active contours. The alternate discrete dynamic programming approach is presented next. In section 4 we present some experimental results. At the end we make conclusions about the approach, and recommend possible new directions.

[1] For review of dynamic programming see [1], [3].



CH2664-1/88/0000/0095$01.00 © 1988 IEEE

## 2 Energy Minimization for Active Contours

The behavior of snakes is controlled by internal and external forces. The internal forces serve as a smoothness constraint and the external forces guide the active contour towards image features. Following the notation of Kass *et al.*, given a parametric representation of an active contour $v(s) = (x(s), y(s))$, the energy functional can be written as:

$$\begin{aligned} E^*_{snake} &= \int_0^1 E_{snake}(v(s))ds \qquad (1) \\ &= \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds \end{aligned}$$

$E_{int}$ represents the internal energy of the active contour, $E_{image}$ represents the image forces, and $E_{con}$ represents the external constraint forces. With

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}, \qquad (2)$$

the energy term due to image forces is a linear combination of line, edge and termination energy terms, all computed from an image, $I(x,y)$. $E_{line} = I(x,y)$, $E_{edge} = -|\nabla I(x,y)|^2$, and $E_{term}$ is the curvature of the level contours in a Gaussian smoothed image. The internal energy is composed of a first order and a second order term forcing the active contour to act like a membrane or a thin plate.

$$E_{int}(s) = (\alpha(s)\,|v_s(s)|^2 + \beta(s)\,|v_{ss}(s)|^2)/2 \qquad (3)$$

The internal energy is the regularizing term. Letting $E_{ext} = E_{image} + E_{con}$, the energy integral becomes

$$\int_0^1 E_{ext}(v(s)) + \frac{1}{2}(\alpha(s)\,|v_s(s)|^2 + \beta(s)\,|v_{ss}(s)|^2)ds \qquad (4)$$

With the integrands having the form $F(s, v_s, v_{ss})$, a necessary condition for a function to minimize (4) is that it satisfy the following Euler equation:

$$F_v - \frac{\partial}{\partial s}F_{v_s} + \frac{\partial^2}{\partial s^2}F_{v_{ss}} = 0 \qquad (5)$$

Substituting the corresponding terms in the above equation, a pair of independent Euler equations is obtained,

$$-\alpha x_{ss} + \beta x_{ssss} + \frac{\partial E_{ext}}{\partial x} = 0 \qquad (6)$$

$$-\alpha y_{ss} + \beta y_{ssss} + \frac{\partial E_{ext}}{\partial y} = 0 \qquad (7)$$

Discretizing the Euler equations with $f_x(i) = \frac{\partial E_{ext}}{\partial x_i}$ and $f_y(i) = \frac{\partial E_{ext}}{\partial y_i}$,

$$\begin{aligned} & \alpha_i(v_i - v_{i-1}) - \alpha_{i+1}(v_{i+1} - v_i) \qquad (8) \\ + \;& \beta_{i-1}(v_{i-2} - 2v_{i-1} + v_i) - 2\beta_i(v_{i-1} - 2v_i + v_{i+1}) \\ + \;& \beta_{i+1}(v_i - 2v_{i+1} + v_{i+2}) + (f_x(i), f_y(i)) = 0 \end{aligned}$$

with $v(0) = v(n)$. Writing the equation in matrix forms, one for x and another for y yields,

$$\mathbf{Ax} + \mathbf{f_x(x,y)} = \mathbf{0} \qquad (9)$$

$$\mathbf{Ay} + \mathbf{f_y(x,y)} = \mathbf{0} \qquad (10)$$

The final step is to solve for position vectors iteratively,

$$\mathbf{x_t} = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\gamma\mathbf{x_{t-1}} - \mathbf{f_x}(\mathbf{x_{t-1}}, \mathbf{y_{t-1}})) \qquad (11)$$

$$\mathbf{y_t} = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\gamma\mathbf{y_{t-1}} - \mathbf{f_y}(\mathbf{x_{t-1}}, \mathbf{y_{t-1}})) \qquad (12)$$

## 3 Time-delayed Discrete Dynamic Programming for Energy Minimization

Consider the energy-minimization problem described in the previous section. Discretizing the internal energy term in equation (3),

$$E_{int}(i) = (\alpha_i\,|v_i - v_{i-1}|^2 + \beta_i\,|v_{i+1} - 2v_i + v_{i-1}|^2)/2 \qquad (13)$$

The total energy over all points is then,

$$\sum_{i=0}^{n-1} E_{int}(i) + E_{ext}(i) \qquad (14)$$

The problem of energy minimization can be viewed as a discrete multistage decision process. This is because not all variables in (14) are interrelated simultaneously. Starting from the initial point on the active contour, we can treat the minimization problem as one that at each of a finite set of stages $(i_0, i_1, ..., i_{n-1})$ a decision is chosen from a finite set of possible decisions.

At each stage, only a local neighborhood of a point on the active contour is considered. This characterizes the set of admissible paths in each iteration. Let us first consider only the first order term of the internal energy measure. This forces the active contour to act like a membrane. With $i$ representing the stage and $k$ the possible choices at each stage, an energy matrix for a given iteration can be computed to be,

$$\begin{aligned} E_t(i,k) &= \min_{0 \le j \le N}\{E_t(i-1, j) \qquad (15) \\ &+ \frac{1}{2}\alpha_i(v_i \oplus k - v_{(i-1+n)} \oplus j)^2 + E_{ext}(v_i \oplus k)\} \end{aligned}$$

In (15) $0 \le k \le N$ ($N$ being the number of possible directions at each stage), $v_m$ is the $m^{th}$ point on the active contour, and $\oplus$ is an operation by which all the points in a local neighborhood of a given point on the lattice are generated. Since in this case the active contour is assumed to be closed, all index addition and subtraction is done in modulo $n$ arithmetic. Figure 1 shows the basic idea in a simplified case where there are three possible choices at each stage. The arrows represent the second term in (15). The darker arrows correspond to the minimum at each stage.

In addition to the energy matrix, a position matrix is also needed. The $(i,k)$ entry of the position matrix stores the value of $j$ that minimizes equation (15). In order to find the path of minimum energy using the *backward* method of solution for discrete dynamic programming problems, $E_{min}(t) = \min_k E_t(n-1, k)$ is found. This is the energy of the optimum contour. Tracing back in the position matrix, we can find the optimal path. This process constitutes a single iteration. If there are $n$ points and $m$ directions at each point, each iteration is $O(n(m+1)^2)$. For finding the globally optimal contour, the iterative process continues until $E_{min}(t)$ does not change with t. An important feature of this algorithm is the fact that one is able to enforce hard constraints on the solution. Consider for example the case of an inequality constraint where it is desired that no two adjacent points on the active contour become closer than a distance $d$. In such a situation, when computing the energy matrix, the distance between points are also computed. If computing $E_t(i,k)$ yields a point that violates the distance constraint for some $j$, then, the best next

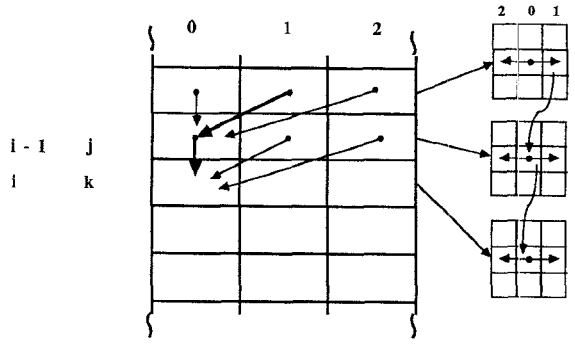


Figure 1: Computing the minimal-energy active contour in a single iteration with only the first order term in the internal energy measure. Each point on the active contour is only allowed to move to two other points or remain at its current position ($N = 2$). $j$ iterates over the $(i-1)^{st}$ row and $k$ iterates over the $i^{th}$ row. The arrows represent the second term in (15). The darker arrows correspond to the minimum at each stage.

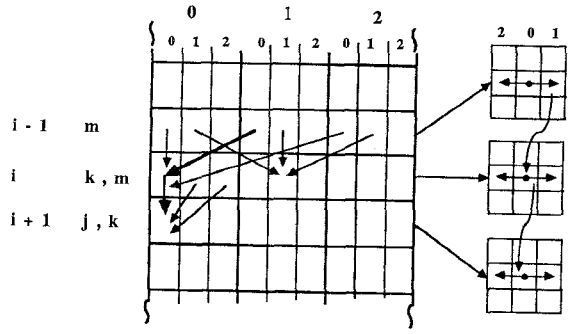


Figure 2: Computing the minimal-energy active contour in a single iteration with both terms of the internal energy measure. Each point on the active contour is only allowed to move to two other points in this example. In the $(i+1)^{st}$ row $j$ iterates over the larger cells, and $k$ iterates over the smaller sub-cells. In the $i^{th}$ row $k$ iterates over the larger cells, and $m$ iterates over the sub-cells. In the $(i-1)^{st}$ row we only need a single index; $m$ iterates over the larger cells. The arrows represent the internal energy terms in (16). The darker arrows correspond to the minimum at each stage.

minimum is considered and the value of the new $j$ is stored in the position matrix. If there does not exist a minimum satisfying the constraint, then the algorithm terminates. Another hard constraint which would be possible to enforce on an active contour, for example, might be when using a binary edge image. In section 4 we describe an example where the output of the Canny edge detector [2] is used as a hard constraint. If a point on an active contour gets attached to an edge element, a constraint is enforced to restrict the motion of the points only to neighboring edge elements. The constraint is enforced in the same spirit as the inequality constraint described above.

Let us now consider the case where the second order term is also included in the internal energy measure. When having a non-zero weight, this term forces the active contour to behave like a thin plate. When including this in the computation of $E_{min}(t)$, at stage $i+1$ one should consider all the possible points at stage $i$. However, it would be a pitfall to consider only the $(i,k)$ entry of the position matrix for the point at stage $i-1$. It would be incorrect to solve this "three-stage" optimization problem by making, sequentially, optimal two-stage decisions. We thus need to consider all the possible combinations for the $i$ and $i-1$ stages once at stage $i+1$.

Figure 2 shows what the time-delayed discrete dynamic programming algorithm involves for the case that there are two directions at each point. Description of the indexes in the figure follows. In the $(i+1)^{st}$ row $j$ iterates over the larger cells, and $k$ iterates over the smaller sub-cells. In the $i^{th}$ row $k$ iterates over the larger cells, and $m$ iterates over the sub-cells. In the $(i-1)^{st}$ row we only need a single index; $m$ iterates over the larger cells.

The basic idea is to delay the decision to be made at stage $i$ to stage $i+1$. One way to accomplish this is to use more storage. This allows for the delay in the decision. If there are $m$ possible directions at each point, each storage cell is subdivided further into $m+1$ sub-cells. The energy for each sub-cell is then computed to be:

$$\begin{aligned} E_t(i+1,j,k) &= \min_{0\le m\le N} E_t(i,k,m) + E_{ext}(v_i \oplus k) \quad (16) \\ &+ \frac{1}{2}(\alpha_i\,|v_i \oplus k - v_{i-1} \oplus m|^2 \\ &+ \beta_i\,|v_{i+1} \oplus j - 2v_i \oplus k + v_{i-1} \oplus m|^2) \end{aligned}$$

Again, in the above equation index arithmetic is done modulo $n$. Using this time-delayed algorithm, then, all the possible combinations for any three consecutive stages are considered. The time complexity for the algorithm then becomes $O(n(m+1)^3)$, where $n$ is the length of the active contour and $m$ is the number of directions at each point. The storage requirement also increases when considering the second order term in the internal energy measure. The storage increases from $n \times (m+1)$ to $n \times (m+1)^2$.

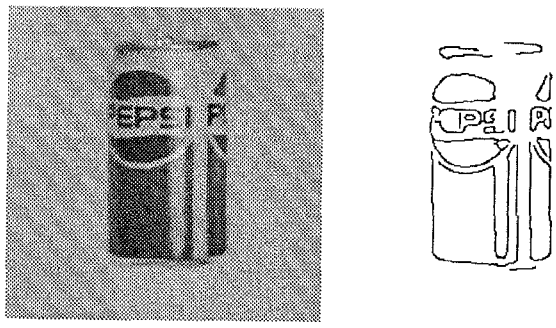

Figure 3: An image of a pepsi can, and its Canny edge detected image.

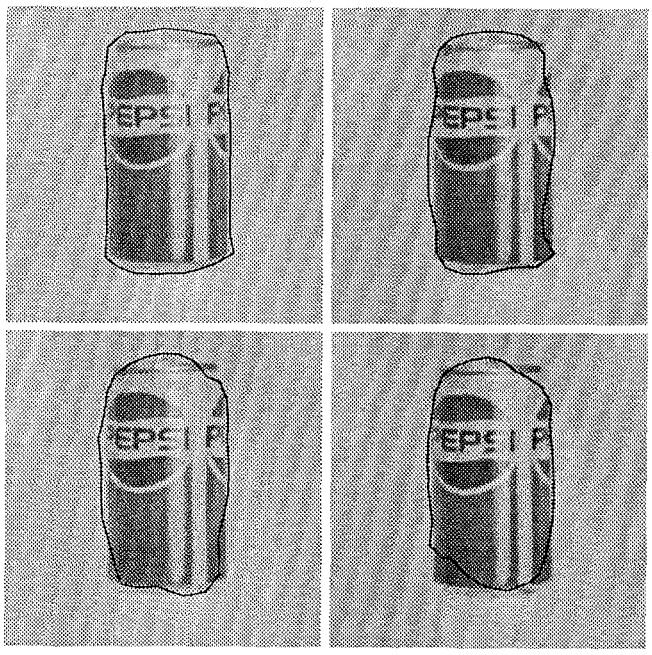

Figure 4: Iterations of (11) and (12) over the pepsi image. The first image on the left is the initial snake.

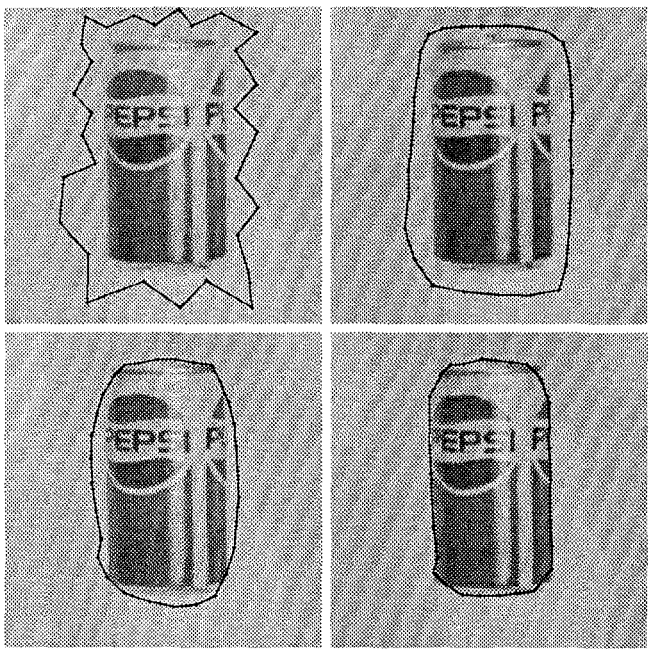

Figure 5: Iterations of (16) over the image. The first image on the left is with the initial active contour. A distance constraint is enforced.

## 4 Experimental Results

We have tested the time-delayed discrete dynamic programming algorithm on a number of real images. The user interactively specifies the position of the active contour and sets the weights for the internal and external active contour forces. In the current implementation, the multiplicative parameters for computing the internal and external forces are independent of position. In addition, the user specifies any hard constraints that are to be imposed on the contour. The active contour behavior is controlled by the combination of external and internal forces so long as all the hard constraints are satisfied. A hard constraint used in these experiments was that the distance between two adjacent points not be larger than some user specified number. Additional hard constraints were provided to the algorithm for the second set of experiments. The algorithm is insensitive to a large range of parameter settings for a number of parameters. Figure 3 shows an image of a pepsi can and its Canny edge-detected image. Figure 4 shows iterations of (11) and (12) over the pepsi image for a representative set of parameters. The initial contour has 30 points and is defined by the user. Figure 5 shows iterations of (16) with $N = 8$ over the image with the hard constraint that the distance between adjacent points in the solution contour must be greater than or equal to $d = 8$. In all figures iterations proceed from left to right and top to bottom. Each displayed contour represents every tenth iteration.

The second set of experiments involves running the algorithm over an image of leaves. Figure 6 shows an image of leaves and its corresponding Canny edge detected image. Figures 7 shows the result of the application of Kass *et al.* on the image. Figure 8 shows the result of the application of the discrete dynamic programming algorithm with a hard constraint on the inter-distance of points on the contour and an additional constraint in the form of a binary edge image. The initial contour has 30 points and $N = 8$.

## 5 Conclusions and Possible New Directions

Calculus of variations has been applied to a number of other optimization problems by researchers in computer vision [6]. As representative examples, it has been used in the problem of shape form shading [5], it has been applied to the problem of computing the optical flow from image sequences [7], and to the problem of surface approximation [9]. Such problems are all characterized by their ill-posed nature and the fact that there are usually more unknowns than equations. As was exemplified in the case of active contours, in order to

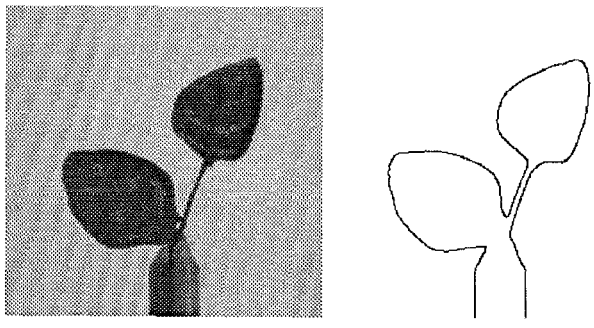

Figure 6: An image of leaves, and the corresponding Canny edge detected image.

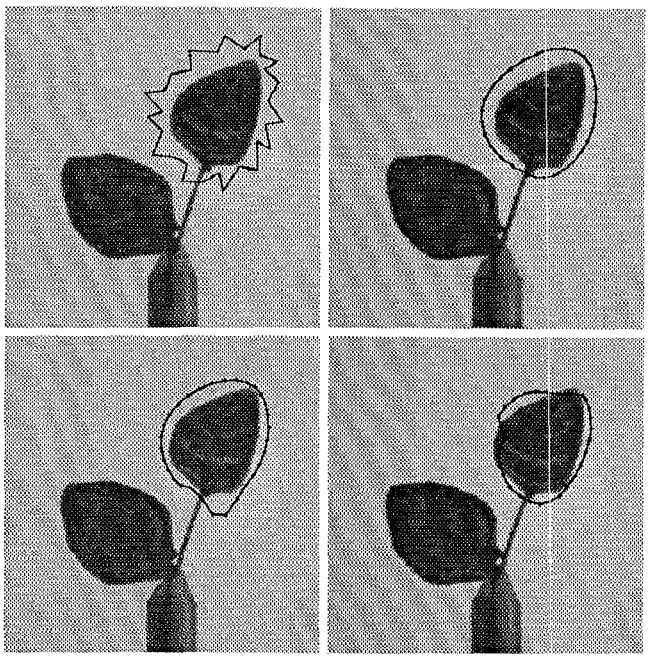

Figure 7: Iterations of (11) and (12) over an image of leaves. The first image on the left is the initial snake.

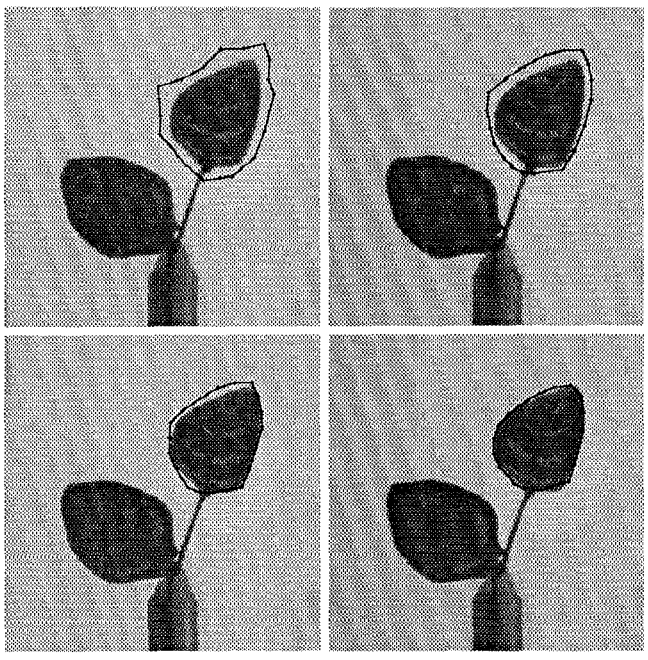

Figure 8: Iterations of (16) over the image. The first image on the left is the initial active contour. Both a distance inequality and a binary edge image are used as hard constraints.

solve such problems, a regularizing constraint is added to the original equation, a variational integral is derived, and the corresponding Euler equations are solved iteratively. Solving the Euler equations yields a possible solution to the variational problem. However, as was seen in the case of active contours, the successive approximation technique may be numerically instable. In addition, in the variational calculus formulation the optimization problem is formulated on the continuous plane and is solved approximately on the discrete grid. This also, may lead to numerical instabilities.

There are advantages to using dynamic programming. One is able to enforce hard constraints on the solution in a strict manner. In addition, only the lower order derivatives of the data are used in the dynamic programming formulation, as seen in the case of active contours. The "time-delayed" discrete dynamic programming algorithm has proven useful in practice. The approach is quite stable numerically. Also, although parameter settings do affect the results in general, but with the aid of the hard constraints, we have eliminated the sensitivity to a large degree. Work is under way to generalize the results discussed in this paper, we are investigating possible extensions of the work described here.

## Acknowledgements

This research was supported in part by NSF grant IRI-8711957 and a grant from NIH.

# References

[1] Bellman, R., and Dreyfus, S., *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, 1962.

[2] Canny, J., "A computational approach to edge detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, **8**, pp. 679-698, 1986.

[3] Dreyfus, S., and Law, A., *The Art and Theory of Dynamic Programming*, Academic Press, New York, 1977.

[4] Foulds, L., *Optimization Techniques*, Springer Verlag, New York, 1981.

[5] Horn, K., "Image Intensity Understanding," Artificial Intelligence, **8**, no. 2, pp. 201-231, 1977.

[6] Horn, K., *Robot Vision*, The MIT Press, 1986.

[7] Horn, K., and Shunck, B., "Determining Optical Flow," Artificial Intelligence, **17**, pp. 185-203, 1983.

[8] Kass, M., Witkin, A., and Terzopolous, D., "Snakes: Active Contour Models," Proceedings of International Conference on Computer Vision, pp. 259-268, London, 1987.

[9] Terzopolous, D., "Computing Visible Surface reconstruction," MIT AI Lab Memo 800, 1985.