Using Dynamic Programming for Solving Variational Problems in Vision

AMIR A. AMINI, STUDENT MEMBER, IEEE, TERRY E. WEYMOUTH, MEMBER, IEEE, AND RAMESH C. JAIN, SENIOR MEMBER, IEEE

Abstract—**Variational approaches have been proposed for solving many inverse problems in early vision, such as in the computation of optical flow, shape from shading, and energy-minimizing active contour models. In general however, variational approaches do not guarantee global optimality of the solution, require estimates of higher order derivatives of the discrete data, and do not allow direct and natural enforcement of constraints.**

In this paper we discuss dynamic programming as a novel approach to solving variational problems in vision. Dynamic programming ensures global optimality of the solution, it is numerically stable, and it allows for hard constraints to be enforced on the behavior of the solution within a natural and straightforward structure. As a specific example of the efficacy of the proposed approach, application of dynamic programming to the energy-minimizing active contours is described. The optimization problem is set up as a discrete multistage decision process and is solved by a "time-delayed" discrete dynamic programming algorithm. A parallel procedure is discussed that can result in savings in computational costs.

Index Terms—**Active contours, contour extraction, deformable models, dynamic programming, variational methods.**

I. Introduction

In many instances in computer vision research, the need arises to determine a surface or a contour having optimal properties amongst a large space of functions. For example, one might be interested in finding the smoothest surface which is close to the available data and which at the same time preserves the discontinuities in such data. Approaches to problems of this kind are mostly deterministic and involve solutions to variational principles [7], [27], [32], [34]. Stochastic formulations based on probabilistic models such as the Markov random fields and involving MAP estimation techniques have also been proposed [15], [16], [24].

Class of vision problems which have been formulated using smoothness models include as examples: optical flow [21], [22], visible surface reconstruction [8], [18], [24], [33], shape from shading [20], [22], edge detection [25], and energy-minimizing active contour models [23]. Invariably, an algorithmic solution involves derivation of an objective function and optimization of the derived equation for finding an appropriate solution. One deals with minimizing an objective function: using variational techniques [20], [21], direct techniques [1], [18], [31], or stochastic relaxation techniques [16], [24].

This paper discusses a novel optimization framework for vision problems where the derived equation may be nonconvex. In the past few years, vision researchers have reported the need for dealing with nonuniqueness and local minima. This problem is extremely important and commonly arises when there is a need to minimize an energy function of some form [38]. Dynamic programming is an optimization approach that simply stated, bypasses local minima. Application of dynamic programming to variational problems are focused upon in this paper, and the relationship between variational approaches and dynamic programming methods are discussed in detail. For the univariate variational problem, dynamic programming is used to optimize the continuous problem, and a discussion on dynamic programming treatment of the bivariate variational problems is included. The discrete form of dynamic programming is also used to optimize the active contours. For minimizing the energy of the active contours, we have devised a "time-delayed" discrete dynamic programming algorithm. The proposed approach provides necessary and sufficient conditions for optimality of solution functions. Dynamic programming has previously been applied to computer vision (see for example [14], [26], [37]). We address dynamic programming solutions of variational problems.

II. Variational Methods in Visual Optimization

The primary approach to solving problems involving functional optimization in computer vision has been calculus of variations [10], [19], [29]. In order to solve such problems with variational methods, a smoothness constraint is added to a physical equation, a variational integral is derived, and the corresponding Euler–Lagrange partial differential equation is solved iteratively.

In this section, we first state the Euler–Lagrange condition and some additional necessary conditions that classical variational theory has developed. We then discuss issues that are of concern when using variational formulations. These issues are related to optimality, stability, convergence, and constraints on variational problems.

Manuscript received August 29, 1988; revised March 31, 1989. Recommended for acceptance by W. E. L. Grimson. This work was supported in part by the National Science Foundation under Grant IRI-8711957.

A. A. Amini was with the Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109. He is now with the Division of Imaging Science, Department of Diagnostic Radiology and Department of Electrical Engineering, Yale University, New Haven, CT 06510.

T. E. Weymouth and R. C. Jain are with the Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 9036962.

A. Necessary Conditions

Let us review the variational techniques and the several optimality conditions that calculus of variations has developed [10], [11], [19], [29]. As will be seen, although these conditions are all necessary, they are usually not sufficient. Our arguments will be made for the univariate case, where one is interested in finding a curve minimizing a functional. The arguments carry over to the bivariate case.

Suppose that we can reduce the form of the visual optimization problem to the minimization of a functional of the form

$$J(y) = \int_{x_0}^{x_1} F(x, y, y')\, dx. \tag{1}$$

Assuming that the solution curve is unique, what we seek is the curve that yields a number J, by equation (1), that is smaller than the number yielded by any other *admissible curve*. The value of J corresponding to such a curve is called the *absolute (global) minimum* of the functional $J(y)$ and the associated curve is the *absolute (global) minimizer*.

Some additional definitions are in order. A *weak neighborhood* N_ϵ of a curve $y_*(x)$, $x_0 \le x \le x_1$, is the collection of all admissible curves $y(x)$ such that the expressions

$$\left| y(x) - y_*(x) \right| \le \epsilon \tag{2}$$

$$\left| y'(x) - y'_*(x) \right| \le \epsilon \tag{3}$$

hold for all $x \in [x_0, x_1]$ with $\epsilon > 0$. Notice that weak neighborhood is a function of ϵ. A curve $y_*(x)$ is said to yield a *weak relative minimum* of $J(y)$ if there exists a weak neighborhood of $y(x)$ such that $J(y_*) \le J(y)$ for all $y(x)$ in the neighborhood. Waiving the requirement imposed by (3), leads to the definition of a strong neighborhood and the corresponding concept of a *strong relative minimum*.

The sets associated with absolute, strong, and weak minimizers of an arbitrary functional create a monotonically increasing sequence of sets. The set of absolute minimizers is a subset of the strong minimizers and in turn the set of strong minimizers is a subset of the weak minimizers.

Suppose that we wish to test a curve $y(x)$ to see if it yields a weak relative minimum of the functional in (1). We can deform the curve by choosing an arbitrary $\eta(x)$ so that

$$\Omega(x) = y(x) + \epsilon\eta(x) \tag{4}$$

is admissible for all ϵ and $\eta(x_0) = \eta(x_1) = 0$. This class of deformations are *weak variations* of the curve since given $\delta > 0$, by choosing $|\epsilon|$ sufficiently small, we can force $\Omega(x)$ to lie in the weak neighborhood, N_δ of $y(x)$.

Euler–Lagrange Condition: Assuming the weak variation, Euler–Lagrange equation for (1) can easily be derived to be

$$\frac{d}{dx} F_{y'} - F_y = 0. \tag{5}$$

If a curve satisfies (5), we may have the correct answer, that is if one exists. However, we may have found a wide assortment of other curves as well.

Other necessary conditions exist for reducing the size of the set of curves that satisfy the Euler–Lagrange condition. Although we will not give the details of these other existing conditions, we will state two of them, namely, the Legendre and the Jacobi conditions. We will then state the method of Lagrange for dealing with the constrained variational problem.

Legendre Condition: It is shown in texts on variational calculus that if we add the condition that

$$F_{y'y'} \ge 0 \tag{6}$$

for all $x \in [x_0, x_1]$, then we eliminate a number of the weak relative maxima in the set of possible solutions.

Jacobi Condition: Jacobi in the early nineteenth century showed that the Legendre condition is not always effective in distinguishing relative minima. His studies led to a condition about the zeros of a solution $v(x)$ of the *Jacobi's equation*:

$$(F_{y'y'})v''(x) + \left[\frac{d}{dx} F_{y'y'}\right] v'(x) + \left[\frac{d}{dx} F_{yy'} - F_{yy}\right] v(x) = 0. \tag{7}$$

The condition is applied in the following manner: if $v(x)$ is found to satisfy the Jacobi's equation, and if it takes on the value of zero at x_1, yet is not identically zero, and $v(x)$ does not take on the value of zero anywhere else in $[x_0, x_1]$, then Legendre condition will assure weak relative minimality, given that the Euler–Lagrange equation is also satisfied.

There are other existing necessary conditions besides the ones that are stated here. For example Weierstrass also has a necessary condition for strong relative minimality.

Method of Lagrange for Constrained Variational Problems: Suppose that it is desired to minimize (1) subject to the constraints $h_i(x, y) = 0$ for $i = 1, 2, \cdots, m$. The approach in classical variational theory is to solve the modified Euler–Lagrange equation

$$\frac{d}{dx} \Phi_{y'} - \Phi_y = 0 \tag{8}$$

subject to the constraints. The functional Φ in this case is

$$\Phi = F + \sum_{i=1}^{m} \lambda_i(x) h_i(x, y). \tag{9}$$

The unknown functions $\lambda_i(x)$ are called the *Lagrange multipliers*. As can be seen from (8), the constraints h_i must be differentiable functions in order for the method to apply.

A second possible form of constraint surfaces arise in *isoperimetric* problems. In this class of problems, it is required that (1) be minimized while the following relations hold (for $i = 1, \cdots m$):

$$\int_{x0}^{x1} h_i(x, y, y')\, dx = c_i \tag{10}$$

where c_i are constants of specified value. The method of solution in this case is to solve (8) with

$$\Phi = F + \sum_{i=1}^{m} \lambda_i h_i(x, y, y'). \tag{11}$$

The important point here is that Lagrange multipliers are required for constraining the solution space. As will be seen, in using dynamic programming no such device is required.

B. Issues of Concern

There are certain issues that must be addressed in regards with the classical variational theory, and the computations that it requires to arrive at a solution.

In many vision problems, the usual approach of researchers has been to find a solution to the Euler–Lagrange equation. As stated in Section II-A, this equation is only necessary for optimality. The danger in using necessary conditions is that one can not guarantee absolute or relative optimality; it is possible for one to obtain maxima instead of minima and minima instead of maxima when using necessary conditions of variational theory. It is also possible for the solution to be a stationary point and satisfy the conditions stated previously. This problem is analogous to the situation in calculus where for a point to yield a minimum, it is necessary for the first derivative of the function to vanish there. However, if the derivative vanishes, it is not sufficient to conclude that the point yields a local minimum of the function. Although, one could use higher order necessary conditions (e.g., Jacobi, Weierstrass), these conditions are often difficult to test, and even if they could be tested, they do not guarantee sufficiency for the general case [12]. Classical variational theory says very little about properties of absolute minimizers. In practice, the best that it can offer is to ensure the solution to be a relative minimum of the weak type.

A second issue of concern is to enforce possibly nondifferentiable constraints on the solution. In many vision applications, the ability to enforce hard constraints on the solution is required. Grimson, for example, taking the direct approach to optimization has used a gradient projection algorithm for surface interpolation; enforcing as hard constraints the available stereo disparities [18]. In case of the variational approaches, Lagrangian-based methods could turn the constrained problem into an unconstrained problem. However, Lagrangian-based approaches require 1) higher dimensional spaces, since now there are more unknowns that must be dealt with, and 2) the constraints themselves must be differentiable. With dynamic programming, constraints simply limit the set of admissible solutions and in fact reduce the computational complexity.

Careful attention should be paid to a third issue of numerical stability and accuracy. In using variational approaches, there is a need for estimates of higher order derivatives of discrete data. Computing high order derivatives of discrete, noisy data leads to numerical instability due to amplification of high frequency noise content. In the variational calculus formulation, the optimization problem is formulated on the continuous plane and is solved using approximate processes. Using this approach however, solutions can become more accurate provided that derivatives of data never become unduly large. We will discuss the variational formulation for the active contours, where there are grounds for being concerned about numerical stability. As will be seen, dynamic programming can directly be applied to the discrete grid with no required approximations. In addition, order of derivatives are generally lower since functionals are directly optimized, and necessary conditions are not used.

Lastly, in finding numerical solutions when using variational techniques, except for some very simple cases, iterative techniques such as the Gauss–Seidel, or Jacobi methods must be employed [28]. A numerical analyst must pay careful attention to convergence issues. For the active contours, convergence of the iterative methods will be analyzed in Section IV.

III. Relationship Between Calculus of Variations and Dynamic Programming

There is a great deal in common between variational methods [10], [19], [29] and dynamic programming [4], [5], [6], [11], [13]. With calculus of variations, the function is sought which has associated with it, by a given functional, a numerical value less than that associated with any other function in a specified set of functions. Calculus of variations considers the extremal function to be a locus of points and attempts to determine this function by means of the Euler–Lagrange equation.

The approach of dynamic programming is to solve the optimization problem by studying a collection, or family of problems containing the particular problem as a member. This is known as embedding. Dynamic programming regards the extremal as an envelope of tangents, and attempts to determine the optimal direction at each point on an extremal.

A. The Univariate Case

Consider the functional

$$J(g) = \int_{x0}^{x1} F(x, g, g')\, dx. \tag{12}$$

The problem involving minimization of $J(g)$ with $y = g(x)$ is the so-called *simplest problem* in the calculus of variations. As we saw in Section II, a necessary condition for a function g_*, to be a solution to (12) is that it satisfy

the Euler–Lagrange equation

$$\frac{\partial F}{\partial g} - \frac{d}{dx}\frac{\partial F}{\partial g'} = 0. \tag{13}$$

We can derive the fundamental equation for dynamic programming for the variational problem stated above. We state the problem in a space with independent variable ξ for reasons that will become clear shortly

$$J(g) = \int_{\xi_0}^{\xi_1} F(\xi, g, g')\, d\xi. \tag{14}$$

Now, let ξ_0 alter with $x < \xi_1$, thus "embedding" the integral minimization problem in a family of related problems. In addition, let us introduce the optimal value function $S(x, y)$ by

$$S(x, y) = \min_{\{P\}} \left[\int_x^{\xi_1} F(\xi, g, g')\, d\xi \right]. \tag{15}$$

The set $\{P\}$ in (15) contains all admissible curves that connect the point $(x, g(x))$ with the end point $(\xi_1, g(\xi_1))$. By definition, $S(x, y)$ attains the minimal value of the right-hand side (RHS) of (15) for all values of (x, y) in its domain of definition.

The reasoning is similar to the discrete case of dynamic programming [4]. We pick an $x < \xi_1$ and then choose a small increment $\Delta\xi$ such that $x + \Delta\xi \leq \xi_1$. The candidates for the minimizing curve are only those admissible curves that are optimal on the interval $[x + \Delta\xi, \xi_1]$, and where over the interval of interest $[x, x + \Delta\xi]$ are arbitrary. The aim is to find the optimal infinitesimal curve over $[x, x + \Delta\xi]$. The principle of optimality [4] assures us that the best curve chosen as such will in fact be the absolutely minimizing curve for our problem.

Let us assume that g' is continuous in the interval $[x, x + \Delta\xi]$ and that F is continuous in its arguments over the same interval. We will then have the expression in (16) as an alternate form to the integral over $[x, x + \Delta\xi]$ and an optimized integral over $[x + \Delta\xi, \xi_1]$ in (15):

$$F(x, y, y')\,\Delta\xi + O(\Delta\xi) + S\big(x + \Delta\xi, y + y'\,\Delta\xi + O(\Delta\xi)\big). \tag{16}$$

In (16), a first order approximation is made to the integral over $[x, x + \Delta\xi]$, and a Taylor expansion is done on $g(x + \Delta\xi)$ in the neighborhood of x. Clearly then,

$$S(x, y) \leq F(x, y, y')\,\Delta\xi + O(\Delta\xi) + S\big(x + \Delta\xi, y + y'\,\Delta\xi + O(\Delta\xi)\big). \tag{17}$$

With equality achieved if the optimum y' is chosen over the interval $[x, x + \Delta\xi]$ and assuming an optimal curve over $[x + \Delta\xi, \xi_1]$; i.e.,

$$S(x, y) = \min_{y'} F(x, y, y')\,\Delta\xi + O(\Delta\xi) + S\big(x + \Delta\xi, y + y'\,\Delta\xi + O(\Delta\xi)\big). \tag{18}$$

In analogy with a discrete multistage decision process where the optimal sequence of decisions are desired, the choice of y' corresponds to the choice of decisions at each stage, and the term $y + y'\,\Delta\xi + O(\Delta\xi)$ corresponds to the new state that will be reached if y' is chosen. Assuming that S is differentiable in both its arguments, we can further expand the third term on RHS of (18) as

$$S(x, y) + S_x\,\Delta\xi + S_y y'\,\Delta\xi + O(\Delta\xi). \tag{19}$$

Substituting (19) in (18), subtracting $S(x, y)$ from both sides, dividing by $\Delta\xi$, and letting $\Delta\xi$ approach zero yields,

$$0 = \min_{y'} \left[F(x, y, y') + S_x + S_y y' \right]. \tag{20}$$

Equation (20) is a partial differential equation in S, and is known as the fundamental equation for dynamic programming [11]. This equation must be satisfied if the optimal value function is to attain the absolutely minimal value of the functional. The optimal policy function is the value of the derivative of the optimal curve at each point connecting (x, y) with the terminal point. We denote this function by $y'(x, y)$, for the present continuous case. As stated previously, dynamic programming seeks to determine an envelope of tangents.

We saw in Section II the kind of perturbation necessary for deriving the Euler–Lagrange equation. The perturbation function was an additive distortive function that perturbed the entire curve in an arbitrary manner. In dynamic programming the variations are quite different from this form. Over a small initial interval, the candidate curve is perturbed arbitrarily as long as it remains admissible. The remainder of the curve is defined to be optimal for the remaining problem so that each perturbation in the initial interval produces a dependent deformation of the remaining curve. Although the form of these deformations are unknown until the problem is solved completely, the variations are well defined.

An observation to be made about the form of (20) is that it is not a partial differential equation of classical types. In fact as can be seen, although it is well-behaved in the sense that it is a linear equation in partial derivaties of S, it involves a minimization operation. We can thus derive a necessary condition for the minimization in (20) by taking partial derivative of the expression in brackets with respect to y', rendering

$$F_{y'}(x, y, y') + S_y = 0. \tag{21}$$

The following equation must also be valid:

$$F(x, y, y') + S_x + S_y y' = 0. \tag{22}$$

Solving for S_y and S_x, and equating S_{xy} and S_{yx}, we can combine (21) and (22) into a quasi-linear partial differential equation in y', namely,

$$F_{y'y'}y'_x + F_{y'y'}y'y'_y + (F_{xy'} + F_{yy'}y' - F_y) = 0. \tag{23}$$

Finally, we should point out that the necessary conditions discussed in Section II: the Legendre, Weierstrass, and also the Euler–Lagrange conditions can all be derived from the fundamental equation. This is not surprising because these conditions are all necessary, and not sufficient.

B. Extending Dynamic Programming to Two Dimensions

A number of problems of early vision have been formulated as minimization of integral functionals of the form

$$J(z) = \int_0^{T_x} \int_0^{T_y} F(\bar{v})\, dy\, dx \tag{24}$$

where $\bar{v}$ denotes a vector of arguments $[x\ y\ z\ z_x\ z_y\ z_{xx} \cdots]$, and $z = z(x, y)$. So far, in this section we have seen the dynamic programming treatment of the univariate variational problems, and in particular we have seen the form that the optimal value function takes. In this section, we will see that the theory outlined for 1-D does not directly extend to the bivariate case. In a forthcoming paper, in the context of reconstruction of surfaces, we will discuss the necessary machinery for functional optimization of the bivariate problem with dynamic programming [3].

Consider the functional $J(z)$ defined over the domain in Fig. 1. With a change in notation, the form of the integral becomes

$$J(z) = \int_0^{T_x} \int_0^{T_y} F(\bar{v})\, d\kappa\, d\xi. \tag{25}$$

If we define the optimal value function $S(x, y, z)$ as

$$S(x, y, z) = \min_{\{p\}} \int_x^{T_x} \int_y^{T_y} F(\bar{v})\, d\kappa\, d\xi, \tag{26}$$

by additivity of integrals:

$$\begin{aligned} S(x, y, z) = \min_{\{p\}} & \int_x^{x+\Delta\xi} \int_y^{y+\Delta\kappa} F(\bar{v})\, d\kappa\, d\xi \\ & + \int_{x+\Delta\xi}^{T_x} \int_y^{T_y} F(\bar{v})\, d\kappa\, d\xi \\ & + \int_x^{T_x} \int_{y+\Delta\kappa}^{T_y} F(\bar{v})\, d\kappa\, d\xi \\ & - \int_{x+\Delta\xi}^{T_x} \int_{y+\Delta\kappa}^{T_y} F(\bar{v})\, d\kappa\, d\xi. \end{aligned} \tag{27}$$

One would be tempted to follow as in (17) for the 1-D case resulting in

$$\begin{aligned} S(x, y, z) \leq\ & F(\bar{v})\, \Delta\xi\, \Delta\kappa + O(\Delta\xi\, \Delta\kappa) \\ & + S(x + \Delta\xi, y, z(x + \Delta\xi, y)) \\ & + S(x, y + \Delta\kappa, z(x, y + \Delta\kappa)) \\ & - S(x + \Delta\xi, y + \Delta\kappa, \\ & \quad z(x + \Delta\xi, y + \Delta\kappa)). \end{aligned} \tag{28}$$

However, careful consideration of above equation convinces one that this step is not valid. The problem arises when the minimization operation is distributed over terms in the sum in (27). Hence, a recurrence relation for computing the optimal value function as defined in (26), as a direct extension of the one-dimensional case, cannot be derived.

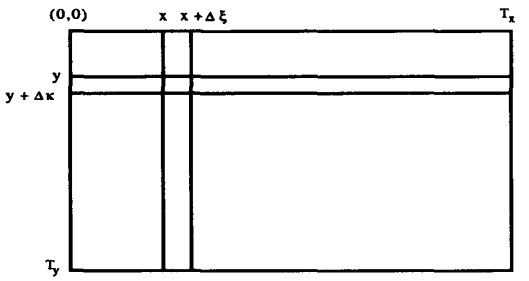


Fig. 1. The rectangular domain considered for finding the optimal solution for the 2-D functional optimization problem with dynamic programming.

C. Finding Numerical Solutions

The solution of the fundamental equation is clearly desirable because the fundamental equation guarantees absolute minimality of the solution within the region of the solution and, at one stroke, solves the problem for an entire range of possible initial conditions. However, analytical solutions are rare. Numerical treatment of the fundamental equation is certainly one method for finding solutions of functional optimization problems.

Alternatively, dynamic programming methods can directly treat the integral minimization problem. For the one-dimensional problem, consider

$$S_1(c) = \min_v \left[F(x, c, v) \cdot \delta\right] \tag{29}$$

where $c = g(0)$ and $v = g'(0)$. The integral minimization problem can thus be transformed to the problem of constructing the optimal value function by means of the recurrence relation:

$$S_N(c) = \min_v \left[F(x, c, v) \cdot \delta + S_{N-1}(c + v\delta)\right]. \tag{30}$$

The suggested computational process determines the minimum not by means of derivatives, but rather by a straightforward search technique. By evaluating the right side of (30) for a large set of v values, and then by directly comparing these, the absolute minimum is found.

Finally, one can perform optimization using discrete dynamic programming. In Section V, we will describe an application of this technique which exploits the discrete nature of the problem. The problem is that of minimization of the energy of active contours.

IV. The Variational Approach to Energy Minimization for Active Contours

The energy-minimizing active contour algorithm proposed in [23], and reformulated in [1], is a top-down mechanism for locating features of interest in images. The user or some other process places an active contour near an image structure of interest. The constraint forces that act on the active contour then push or pull the contour towards features of the image structure. The contour locks

on to features of an image structure by minimizing an integral measure which represents the active contour's total energy.

The forces acting on an active contour depend on where the contour is placed and how its shape changes locally in space. The behavior of energy-minimizing contours 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 which minimize the contour's total energy.

The total energy of an active contour with parametric representation $v(s) = (x(s), y(s))$, can be written as

$$E^*_{\text{total}} = \int_0^1 E(v(s))\, ds = \int_0^1 E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s))\, ds \tag{31}$$

where

$$E_{\text{int}}(v(s)) = \left(\alpha(s)\left|v_s(s)\right|^2 + \beta(s)\left|v_{ss}(s)\right|^2\right) \tag{32}$$

$$E_{\text{image}}(v(s)) = w_{\text{line}}E_{\text{line}}(v(s)) + w_{\text{edge}}E_{\text{edge}}(v(s)) + w_{\text{term}}E_{\text{term}}(v(s)) \tag{33}$$

$$E_{\text{con}}(v(s)) = -k(x_1 - x_2)^2. \tag{34}$$

The internal energy E_{int} represents the forces which constrain the curve to be smooth, E_{image} represents the forces derived from the image which constrain the curve to take the shape of features present in the image, and the constraint energy E_{con} represents the energy of a spring connected between a point on the contour and some point in the plane.

The internal energy results in $v(s)$ being a controlled continuity spline [32] with the first order membrane term in (33) favoring discrete points to become closer to one another and the second order thin-plate term favoring points to become equidistant.

The image energy is a linear combination of three terms all derived from the image: the line energy simply attracts the contour to lower or higher intensity values in the image depending on the sign of w_{line} with $E_{\text{line}} = I(x, y)$, the edge energy is calculated as $E_{\text{edge}} = -|\nabla I(x, y)|^2$ thus attracting the contour to image points with high gradient values, and E_{term} is the curvature of the level contours in a Gaussian smoothed image, attracting the contour towards line terminations.

The constraint energy attracts points on the contour to points in the plane. In (34), x_1 and x_2 represent such points on the contour and in the plane, respectively, and k is the "spring constant."

Letting $E_{\text{ext}} = E_{\text{image}} + E_{\text{con}}$, (31) becomes

$$\int_0^1 E_{\text{ext}}(v(s)) + \tfrac{1}{2}\left(\alpha(s)\left|v_s(s)\right|^2 + \beta(s)\left|v_{ss}(s)\right|^2\right) ds. \tag{35}$$

Representing the integrand by $F(s, v_s, v_{ss})$, the Euler–Lagrange necessary condition is derived as

$$F_v - \frac{\partial}{\partial s}F_{v_s} + \frac{\partial^2}{\partial s^2}F_{v_{ss}} = 0. \tag{36}$$

Substituting the terms in the above equation, we obtain a pair of independent Euler–Lagrange equations,

$$-\alpha x_{ss} + \beta x_{ssss} + \frac{\partial E_{\text{ext}}}{\partial x} = 0 \tag{37}$$

$$-\alpha y_{ss} + \beta y_{ssss} + \frac{\partial E_{\text{ext}}}{\partial y} = 0. \tag{38}$$

To solve numerically, the Euler equations with $f_x(i) = \partial E_{\text{ext}}/\partial x_i$ and $f_y(i) = \partial E_{\text{ext}}/\partial y_i$ are discretized, yielding

$$\begin{aligned}&\alpha_i(v_i - v_{i-1}) - \alpha_{i+1}(v_{i+1} - v_i)\\ &\quad + \beta_{i-1}(v_{i-2} - 2v_{i-1} + v_i)\\ &\quad - 2\beta_i(v_{i-1} - 2v_i + v_{i+1})\\ &\quad + \beta_{i+1}(v_i - 2v_{i+1} + v_{i+2}) + (f_x(i), f_y(i)) = 0.\end{aligned} \tag{39}$$

Writing the equation in matrix forms, one for x and another for y, yields

$$Ax + f_x(x, y) = 0 \tag{40}$$

$$Ay + f_y(x, y) = 0. \tag{41}$$

We can now solve for position vectors iteratively,

$$x_t = (A + \gamma I)^{-1}(\gamma x_{t-1} - f_x(x_{t-1}, y_{t-1})) \tag{42}$$

$$y_t = (A + \gamma I)^{-1}(\gamma y_{t-1} - f_y(x_{t-1}, y_{t-1})). \tag{43}$$

A. Discussion

In Section II-B, several points were raised in regard to optimality, numerical stability, convergence, and enforcement of hard constraints within the variational framework.

Let us look at these issues and some additional points, starting with convergence of the iterative method of solution. Arguing for the case of (42):

$$x_{t+1} = \gamma(A + \gamma I)^{-1}x_t + (A + \gamma I)^{-1}C_t. \tag{44}$$

With $C_t = -f_x(x_t, y_t)$ and $B = (A + \gamma I)^{-1}$, the equation can be reformulated to be:

$$x_{t+1} = (\gamma B)^{t+1}x_0 + \sum_{i=0}^{t} \gamma^i B^{i+1}C_{t-i}. \tag{45}$$

If the final solution is x_*, $x_* = \gamma B x_* + BC_\infty$. Then we have

$$\begin{aligned}x_{t+1} - x_* &= \gamma B(x_t - x_*) + B(C_t - C_\infty)\\ &= (\gamma B)^{t+1}(x_0 - x_*)\\ &\quad + \sum_{i=0}^{t} \gamma^i B^{i+1}(C_{t-i} - C_\infty).\end{aligned} \tag{46}$$

This is the error in successive approximations. For the simple case of $C_{t-i} = C_\infty$, for all i and t,

$$x_{t+1} - x_* = (\gamma B)^{t+1}(x_0 - x_*). \tag{47}$$

Letting $x_0 - x_* = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n$, where $\{u_i\}_{i=1}^n$ are the eigenvectors of γB with $\{\lambda_i\}_{i=1}^n$ being the eigenvalues, we have,

$$\alpha_1 \lambda_1^t u_1 + \alpha_2 \lambda_2^t u_2 + \cdots + \alpha_n \lambda_n^t u_n = 0. \tag{48}$$

If $\lambda_i > 1$, for any i, the process does not converge since the limit of $(\gamma B)^t$ does not exist when $t \to \infty$, and (48) does not yield an equality. If $C_{t-i} \neq C_\infty$, it is difficult to predict the convergence properties in general.

Optimality is not guaranteed within this formulation. Existence and uniqueness of the solution could be guaranteed (see [33]) if the external force field were convex or simply did not change with time. However, close attention to the form of iterative solution unveils that the external energy field changes with each iteration, and moreover for images of real scenes can not satisfy the convexity requirement. As we saw in Section II, although the Euler–Lagrange equation is a necessary condition for optimality in a local sense, it is not a sufficient condition.

In terms of possible constraints enforceable on the solution, within the variational formulation described here, constraints can be enforced on the solution if they can be added to the overall functional. Differentiability of constraints in the variational formulation however is necessary and strict enforcement of constraints is not accounted for in the variational approach to energy minimization. Although the weight associated with a desired constraint term may be increased to force more effect from it, the constraint will be satisfied at the cost of other constraints such as smoothness not being satisfied as closely.

There is a need for estimates of high order derivatives of the discrete data. The iterative solution requires estimate of the derivative of the gradient of the image data (arising from the edge energy functional). Unless the image is smoothed, this term may cause instabilities. Smoothing the image however results in poor localization for boundaries. One may need to resort to scale space strategies in such cases [23].

Finally, as characteristics of the formulation, if a contour is not subjected to any external forces, it will vanish to a line or a point, and furthermore, if it is not placed close to image boundaries, it will not get attracted.

In summary, although the computational requirements of the variational approach is linear, dynamic programming has important features, making the new formulation attractive.

V. Time-Delayed Discrete Dynamic Programming for Energy Minimization of Active Contours

Consider the energy-minimization problem described in the previous section. Discretizing the internal energy term in (32),

$$E_{\text{int}}(v_i) = \left(\alpha_i \left| v_i - v_{i-1} \right|^2 + \beta_i \left| v_{i+1} - 2v_i + v_{i-1} \right|^2\right)/2. \tag{49}$$

Discretizing the integral in (36),

$$E^*_{\text{total}} = \sum_{i=0}^{n-1} E_{\text{int}}(v_i) + E_{\text{ext}}(v_i). \tag{50}$$

In order to use dynamic programming, the observation is made that minimization of (50) can be viewed as a discrete multistage decision process. Starting from the initial point on the contour, we can treat the minimization problem as one that at each of a finite set of stages ($i_0, i_1, \cdots, i_{n-1}$), a decision is chosen from a finite set of possible decisions. One could solve for the optimal policy for the continuous multistage decision process [e.g., (30)]. However, with this approach, the discrete process introduced for numerical purposes has nothing to do with the original discrete grid upon which the contour points are initially placed. The algorithm described here takes advantage of the inherent discrete nature of the problem.

A correspondence can be made between minimization of the total energy measure (with only the first order term in the internal energy measure) and the problem of minimizing a function of the form

$$\begin{aligned} E(v_1, v_2, \cdots, v_n) &= E_1(v_1, v_2) + E_2(v_2, v_3) + \cdots \\ &\quad + E_{n-1}(v_{n-1}, v_n) \end{aligned} \tag{51}$$

where each variable is allowed to only take on m possible values. One way to find the minimum of the above function is by exhaustive enumeration. A more efficient method is via discrete dynamic programming, with v_i corresponding to the *state variable* in the ith decision stage.

The dynamic programming solution involves generating the sequence of functions of one variable, $\{s_i\}_{i=1}^{n-1}$ (the optimal value function), where for obtaining each s_i a minimization is performed over a single dimension. As an example, for a function having the form of (51), with $n = 5$,

$$\begin{aligned} s_1(v_2) &= \min_{v_1} E_1(v_1, v_2) \\ s_2(v_3) &= \min_{v_2} s_1(v_2) + E_2(v_2, v_3) \\ s_3(v_4) &= \min_{v_3} s_2(v_3) + E_3(v_3, v_4) \\ \min_{v_1, \cdots, v_5} E(v_1, v_2, v_3, v_4, v_5) &= \min_{v_4} s_3(v_4) + E_4(v_4, v_5). \end{aligned} \tag{52}$$

For the general case, $s_k(v_{k+1}) = \min_{v_k}\{s_{k-1}(v_k) + E_k(v_k, v_{k+1})\}$. In view of the form of (51), let us first consider only the first order term in E_{int}. With k representing the stage and v_k being the state variable, the recurrence relation for computing the optimal value func-

tion for contours is given by

$$s_k(v_{k+1}) = \min_{v_k} \left\{ s_{k-1}(v_k) + E_{\text{ext}}(v_k) + |v_{k+1} - v_k|^2 \right\}. \quad (53)$$

Fig. 2 shows the basic idea in a simplified case where there are only three possible states per stage. The cost associated with any given arc corresponds to the internal energies between two possible choices in decision sets.

In addition to the energy matrix corresponding to the optimal value function $\{s_k\}$, a position matrix is also needed. Each entry of the position matrix at stage k stores the value of v_k that minimizes (53). Fig. 3 illustrates the correspondence between the decision set and image pixels. In order to find the contour of minimum energy using the *backward* method of solution for discrete dynamic programming problems, $E_{\min}(t) = \min_{v_n} s_{n-1}(v_n)$ is found. This is the energy of the optimal contour. Tracing back in the position matrix, the optimal contour is found. This procedure constitutes a single iteration. If there are n points and m directions at each point, each iteration has complexity $O(nm^2)$. For finding the optimal contour, the iterative process continues until $E_{\min}(t)$ does not change with time. Convergence is guaranteed since the configuration of points on the contour will not change unless the total energy of the contour is reduced by the new configuration.

An important feature of this algorithm is 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 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 $s_k(v_{k+1})$ yields a point that violates the distance constraint for some value of v_k, then that choice is eliminated and the best next minimum satisfying the constraint is chosen. In this case the value of the new minimizer of v_k is stored in the position matrix. If no minimum exists which satisfies the constraint, the algorithm halts. Another useful example of a hard constraint might be a binary edge image. Use of such constraints will be illustrated in Section VI.

Let us now consider tne case when E_{int} includes the second order term. With the second order term in place,

$$E_{\text{total}}(v_1, v_2, \cdots, v_n) = E_1(v_1, v_2, v_3) + E_2(v_2, v_3, v_4) + \cdots + E_{n-2}(v_{n-2}, v_{n-1}, v_n) \quad (54)$$

where

$$E_{i-1}(v_{i-1}, v_i, v_{i+1}) = E_{\text{ext}}(v_i) + E_{\text{int}}(v_{i-1}, v_i, v_{i+1}). \quad (55)$$

In order to apply dynamic programming to (55), a two element vector of state variables, (v_{i+1}, v_i), is fixed. Now the optimal value function is a function of two adjacent

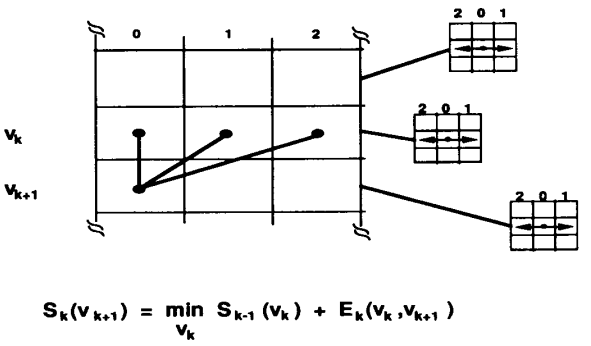


Fig. 2. One iteration of the algorithm for computation of the minimal-energy contour with only the first order term in E_{int}. In this example, each point on the contour is only allowed to move to two other points ($m = 3$). The darker arrows correspond to the minimum at each stage. Further description is given in the text.

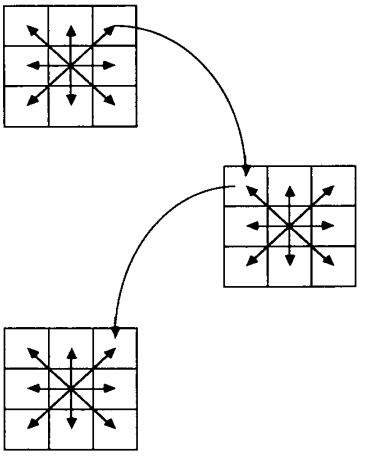


Fig. 3. The decision set for the discrete dynamic programming formulation (with $m = 9$). The two curved arrows depict the minimum energy configuration for the current iteration.

points on the contour and we can apply the standard form of dynamic programming in the usual way:

$$S_i(v_{i+1}, v_i) = \min_{v_{i-1}} S_{i-1}(v_i, v_{i-1}) + \alpha\left(|v_i - v_{i-1}|\right)^2 + \beta|v_{i+1} - 2v_i + v_{i-1}|^2 + E_{\text{ext}}(v_i). \quad (56)$$

Notice that the new dynamic programming table has m^2 entries for each stage. Each entry in the dynamic programming table in the $i + 1$ row represents fixed values for v_{i+1} and v_i and the minimization is done over possible values of v_{i-1}. Results are stored in the table entry at $i + 1$. Fig. 4 shows what the time-delayed discrete dynamic programming algorithm involves for a hypothetical case with $m = 3$.

The time complexity for the algorithm increases to $O(nm^3)$, where n is the length of the contour and m is the number of possible choices at each stage. The storage requirement also increases from $n \times m$ to $n \times m^2$ memory

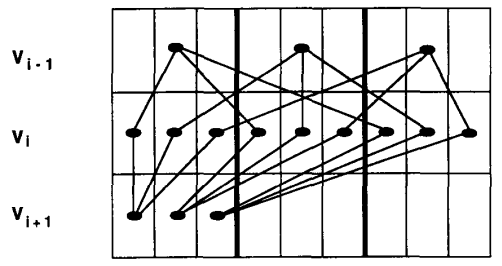


Fig. 4. Computing the minimal-energy contour in a single iteration with both terms of the internal energy measure. Each point on the contour is only allowed to move to two other points in this example ($m = 3$). Each entry in the dynamic programming table in the $i + 1$ row represents fixed values for v_{i+1} and v_i and the minimization is done over possible values of v_{i-1}. Results are stored in the table entry at $i + 1$.

elements when considering the second order term. In summary, the algorithm iteratively executes the steps in Fig. 5.

Examining the dynamic programming table, in order to make the algorithm independent of the number of choices at each stage and only linearly dependent on the number of points on the contour, consider the following strategy. We can compute the energy of each entry for a given row in the dynamic programming table in parallel. This is because there are no interactions among the entries in each row in Fig. 4. For full parallelism, the number of processors needed is m^2. This reduces the computational complexity to $O(n)$.

A. Discussion

In this section we list properties of the time-delayed discrete dynamic programming based approach.

- Relational constraints can be enforced in the dynamic programming formalism in a natural manner. The dynamic programming formalism provides the machinery for enforcing hard constraints on the distance of points on the contour.
- In the dynamic programming formulation lower order derivatives of data are used and the problem representation that is adopted only deals with the gradient of the image data, and first and second derivatives of the active contour data. There are clear advantages for using lower order derivatives of data whenever possible.
- The dynamic programming method used for optimization of active contours takes advantage of the discrete nature of the problem. Dynamic programming can directly treat the continuous version of the problem, however, the nature of the problem lends itself to the discrete dynamic programming approach for optimization.
- Each iteration results in an optimum contour within the resolution of the considered window. This is the case since in the dynamic programming approach, all possible choices are considered within an efficient parallelizable framework.
- In the discrete dynamic programming formulation, the active contour is guaranteed to converge to a final solution in a finite number of iterations since the energy measure is monotonically decreasing with time. The algorithm halts when there is no change in the total energy of the contour.

1. **Find total energy of the minimal energy contour for the set of pixels considered around each point.**
2. **If total energy has not changed from previous iteration, exit.**
3. **Move points to newly computed locations.**
4. **Go to 1.**

Fig. 5. Steps in the iterative algorithm.

VI. 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, all coefficients are set independent of position. In addition, the user can specify any hard constraints that are to be imposed on the contour.

These experiments illustrate two types of hard constraints. In all the experiments to follow the distance between adjacent points on the contour are not allowed to become smaller (or larger) than some user specified number. In one experiment that will be discussed, a hard constraint was provided to the algorithm in the form of a binary edge image.

Although the algorithm is not completely independent of parameter settings, we believe that it is insensitive to a large range of parameter settings for a number of parameters. In its current form, selection of parameter values is based on empirical observations. This seems to be the most widely used strategy (see for example [24]). Certain approaches to automatic selection of parameter values exist however. Gennert and Yuille [17] recently proposed the idea of determining the optimal weights in multiple objective function optimization using the min–max principle. Another approach proposed by Wahba [35], [36] and discussed in [7] and [30] is the method of cross validation which can be used to determine the optimal degree of smoothing.

Fig. 6 shows iterations of (42) and (43) over an image of a Pepsi can for a representative set of parameters with 30 points on the contour. Fig. 7 shows iterations of Fig. 5 over the same image and with the same number of points with the constraint that the distance between adjacent points be greater than $d = 8$. In all figures to follow iterations proceed from left to right and top to bottom. Each displayed contour represents every tenth iteration and points on the contour are represented by tick marks.

The second set of experiments involves running the algorithm over an image of leaves. Figs. 8 and 9 show the results.

The third experiment is done on an image of a computer mouse. Results are shown in Figs. 10 and 11.

The fourth experiment is on a cell image. Fig. 12 shows the cell image, and its corresponding Canny edges [9].

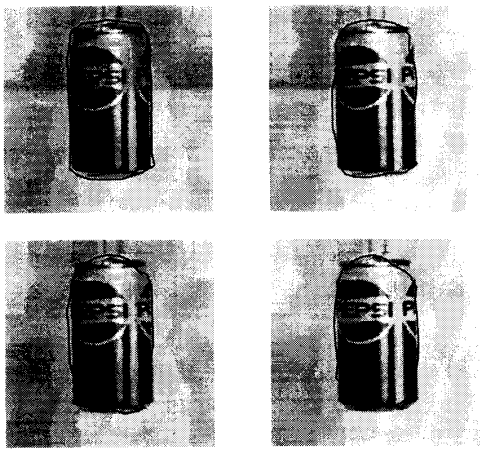


Fig. 6. Iterations of (42) and (43) over an image of a Pepsi can. The first image on the left is the initial active contour. The figure illustrates instabilities that may result.

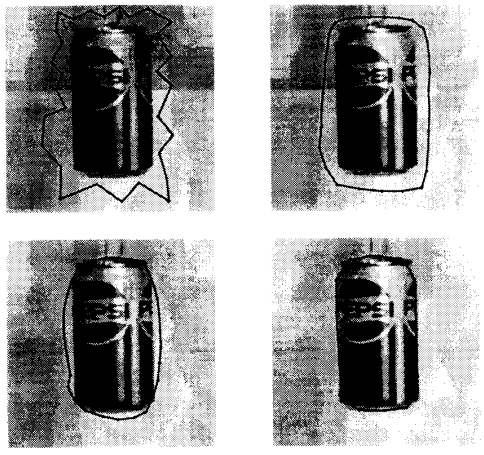


Fig. 7. Iterations of Fig. 5 over the image. The first image on the left is the initial active contour. An inequality constraint on the interdistance of points is enforced.

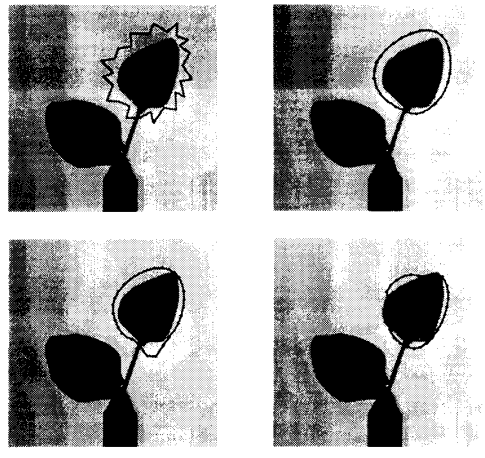


Fig. 8. Iterations of (42) and (43) over an image of leaves. The first image on the left is the initial active contour.

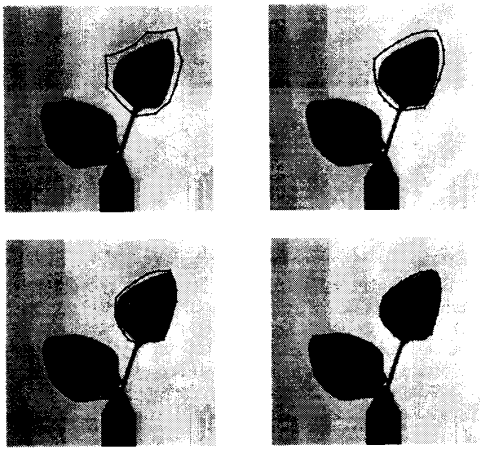


Fig. 9. Iterations of Fig. 5 over the image. The first image on the left is the initial active contour. An inequality constraint on the interdistance of points is enforced.

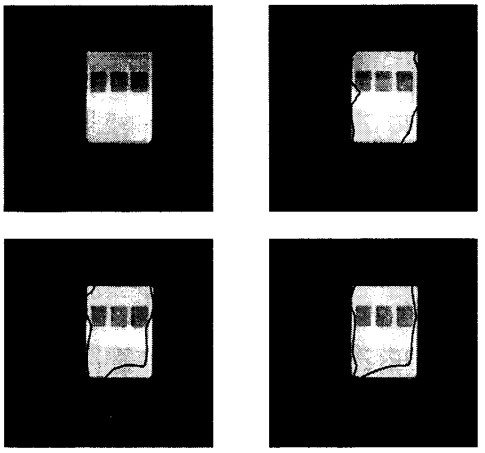


Fig. 10. Extracting the outline of a computer mouse with active contours using the variational approach to energy minimization.

Results of the application of the algorithm are shown in Fig. 13. As can be seen, in the four representative iterations, the contour expands rather than contract. In this experiment, use is also made of the binary edge image. The contour follows the boundary quite closely, and fills in the gaps where there is a null response from the edge detector. (These contours are referred to as the inflating/deflating contours [2].)

A final illustration (Table I) is given as a repeatable exercise which the reader can use to verify his or her own implementation of the algorithm given in this paper. This illustration is, if you will, a "repeatable experiment." This experiment involves the application of the time-delayed discrete dynamic programming algorithm on a 256×256 synthetic image. The synthetic image was generated by creating a 128×128 square of intensity 255 on a background of zero intensity in the center of the image. A normalized 7×7 box filter with unity entries was then

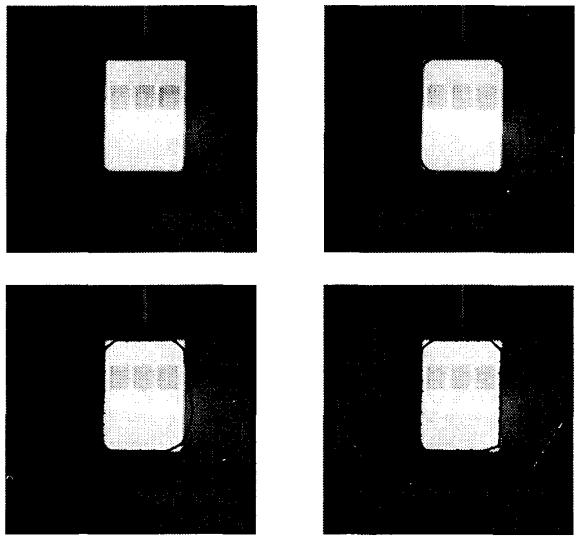


Fig. 11. Iterations of Fig. 5 over the image of a computer mouse with 30 points on the contour. Each picture represents every tenth iteration.

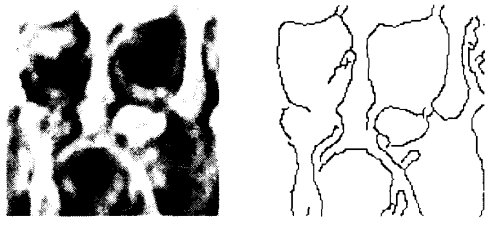


Fig. 12. A noisy image of cells, and its Canny edge-detected image.

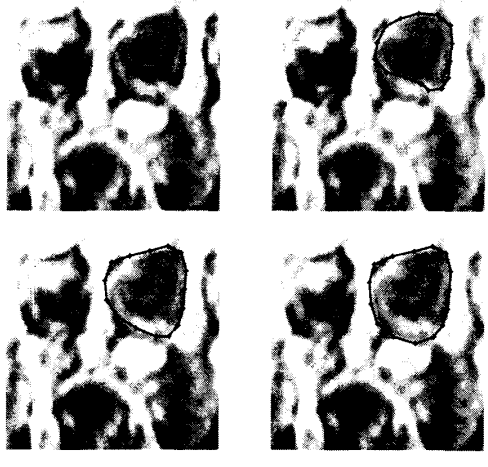


Fig. 13. The first image on the left is the initial contour, a circular contour. These contours are referred to as the inflating contours [2]. There are 15 points on the contour, and the edge image is used as a hard constraint in the optimization process.

convolved with the resulting image. The initial contour positions were: (206, 54), (217, 74), (206, 95), (220, 112), (206, 133), (222, 150), (208, 173), (221, 194), (199, 204), (180, 222), (166, 206), (137, 218), (118, 204), (96, 224), (84, 206), (60, 216), (50, 202), (48, 184), (34, 168), (47, 146), (32, 126), (46, 110), (32, 86), (46, 77), (42, 56), (57, 50), (73, 40), (94, 46), (108, 36), (126, 42).

TABLE I
PARAMETER VALUES FOR "REPEATABLE EXPERIMENT": SEE TEXT

α	β	w_{line}	w_{edge}	w_{term}
1.0	0.5	0.0	5.0	0.0

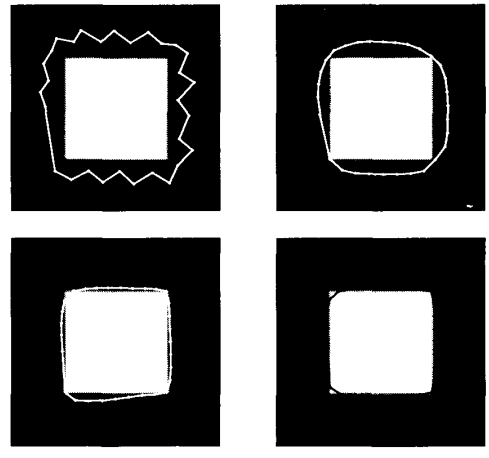


Fig. 14. A repeatable experiment with a synthetic box with ramp edges. There are 30 points on the contour, and the distance between consecutive points is restricted to remain greater than 8.

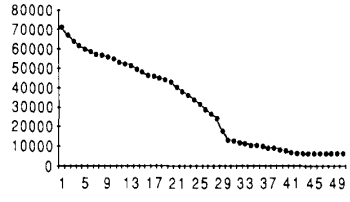


Fig. 15. A plot of energy (50) as a function of iterations.

Fig. 14 shows a sample of iterations of the algorithm over the image. Iterations proceed from left to right and top to bottom. Notice that in each iteration points on the contour are characterized by tick marks. In this experiment an inequality constraint was enforced on the minimum allowable distance between any two adjacent points on the contour. The distance between no two points was allowed to become less than 8 pixels. Fig. 15 shows a plot of energy as a function of iterations.

VII. CONCLUSIONS

Variational methods have been applied to a number of optimization problems by researchers in computer vision. As discussed in the paper, certain issues are of concern when using variational methods. Dynamic programming is an attractive methodology for optimization since it bypasses local minima and allows for enforcement of hard constraints on the solution within a natural and straightforward structure. For the active contours, we have de-

vised the time-delayed discrete dynamic programming algorithm for energy minimization. In the dynamic programming framework, hard constraints can be enforced on such quantities as the minimum allowable distance between adjacent points on the contour and position of the contour points. Constraints of this form result in more controlled behavior of the contours. Convergence of the algorithm is guaranteed, and so is the optimality of the solution.

REFERENCES

[1] A. Amini, S. Tehrani, and T. Weymouth, "Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints," in *Proc. Second Int. Conf. Computer Vision*, Tarpon Springs, FL, Dec. 1988.

[2] A. Amini, "Using dynamic programming for solving variational problems in vision: Applications involving deformable models for contours and surfaces," Ph.D. dissertation, Dep. Elec. Eng. Comput. Sci., The Univ. Michigan, Ann Arbor, 1990.

[3] A. Amini, T. Weymouth, and B. Schunck, "Surface weaving with deformable signals," submitted to the Third Int. Conf. Computer Vision, Osaka, Japan, 1990.

[4] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[5] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton University Press, 1961.

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

[7] M. Bertero, T. Poggio, and V. Torre, "Ill-posed problems in early vision," in *Proc. IEEE*, vol. 76, no. 8, pp. 869–889, Aug. 1988.

[8] A. Blake and A. Zisserman, *Visual Reconstruction*. Cambridge, MA: MIT Press, 1987.

[9] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 679–698, 1986.

[10] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, vol. I. London: Interscience, 1953.

[11] S. Dreyfus, *Dynamic Programming and the Calculus of Variations*. New York: Academic, 1965.

[12] ——, "The main results of optimal control theory made simple," *Population Dynamics*, 1972.

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

[14] M. Furst and P. Caines, "Edge detection with image enhancement via dynamic programming," *Comput. Vision, Graphics, Image Processing*, vol. 33, pp. 263–279, 1986.

[15] E. Gamble and T. Poggio, "Visual integration and detection of discontinuities: The key role of intensity edges," MIT AI Memo 970, 1987.

[16] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[17] M. Gennert and A. Yuille, "Determining the optimal weights in multiple objective function optimization," in *Proc. Second Int. Conf. Computer Vision*, Tarpon Springs, FL, Dec. 1988, pp. 87–89.

[18] W. E. L. Grimson, *From Images to Surfaces*. Cambridge, MA: MIT Press.

[19] F. Hildebrand, *Methods of Applied Mathematics*. Englewood Cliffs, NJ: Prentice-Hall, 1965.

[20] B. K. P. Horn, "Image intensity understanding," *Artificial Intell.*, vol. 8, no. 2, pp. 201–231, 1977.

[21] B. K. P. Horn and B. Schunck, "Determining optical flow," *Artificial Intell.*, vol. 17, nos. 1-3, pp. 185–203, 1981.

[22] B. K. P. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.

[23] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[24] J. Marroquin, S. Mitter, and T. Poggio, "Probabilistic solution of ill-posed problems in computational vision," *J. Amer. Statist. Assoc.*, vol. 82, no. 397, pp. 76–89, 1987.

[25] D. Mumford and J. Shah, "Boundary detection by minimizing functionals, I," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, CA, June 1985, pp. 22–26.

[26] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 139–154, 1985.

[27] T. Poggio and V. Torre, "Ill-posed problems and regularization analysis in early vision," in *Proc. AARPA Image Understanding Workshop*, New Orleans, LA, 1984, pp. 257–263.

[28] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge, MA: Cambridge University Press, 1988.

[29] A. Sage and C. White, *Optimum Systems Control*. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[30] B. Shahraray and D. Anderson, "Optimal estimation of contour properties by cross-validated regularization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 6, pp. 600–610, June 1989.

[31] T. Simchony, R. Chellapa, and Z. Lichtenstein, "Pyramid implementation of optimal step conjugate search algorithms for some computer vision problems," in *Proc. Second Int. Conf. Computer Vision*, Tarpon Springs, FL, Dec. 1988, pp. 580–590.

[32] D. Terzopoulos, "Regularization of inverse visual problems involving discontinuities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 4, pp. 413–424, July 1986.

[33] ——, "Computation of visible-surface representations," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 4, pp. 417–438, July 1988.

[34] A. Tikhonov and V. Arsenin, *Solution of Ill-posed Problems*. Washington, DC: Winston, 1977.

[35] G. Wahba and S. Wold, "Periodic splines for spectral density estimation: The use of cross validation for determining the degree of smoothing," *Commun. Statist.*, vol. 4, no. 2, pp. 125–141, 1975.

[36] G. Wahba and J. Wendelberger, "Some new mathematical methods for variational objective analysis using splines and cross validation," *Monthly Weather Rev.*, vol. 108, pp. 1122–1143, 1980.

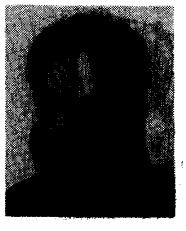
[37] H. Yamada, C. Merritt, and T. Kasvand, "Recognition of kidney glomerulus by dynamic programming matching method," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 5, pp. 731–737, Sept. 1988.

[38] A. Yuille, "Energy functions in early vision and analog networks," MIT AI Lab Memo 987, Nov. 1987.

Amir A. Amini (S'89) was born in Tehran, Iran, in 1965. He received the B.S. degree in electrical engineering with honors from the University of Massachusetts at Amherst in 1983, where he was the youngest graduate of the university, and the M.S.E. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, in 1984 and 1990, respectively.

Currently, he is with the Division of Imaging Science, Department of Diagnostic Radiology, and the Department of Electrical Engineering at Yale University, New Haven, CT. His current research interests are in three-dimensional modeling of biological shapes.

Dr. Amini is a member of Eta Kappa Nu and Phi Kappa Phi.



Terry E. Weymouth (S'83–M'86) received his dissertation in May of 1986 from the University of Massachusetts, where he was involved in computer vision research. His dissertation proposed one possible solution to the problem of natural outdoor scene interpretation, in which object description information, specialized procedures for object recognition, and object-centered control information are unified in a schema data structure. In the resulting system, the final interpretations of the scene resulted in a network describing the objects and their placement. Several scenes were interpreted as part of the dissertation.

He is now an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. His current work is in the application of blackboard architecture to the problems of machine vision. The goal of the research is to develop an understanding of the methods for combining image events (such as edges) into relational structures that describe objects in the scene. A system for machine vision is designed in a restricted domain, then the modules of that system are analyzed to characterize their general effectiveness and to understand when they can be applied. With this understanding, the collection

of modules can be combined into a toolbox for the design of new systems. The blackboard architecture provides a natural framework for the switching and recombining of experimental modules. This approach is being applied in medical image processing. In addition to the work based on blackboard architectures, he is also involved in research in sensor-based robot navigation. Navigation, even in known environments, requires sensory feedback. This feedback, however, must be tied to the prior knowledge of the scene wether derived from other views of the same sensor, from different sensors, or from world knowledge. He is working on two projects related to the assimilation of sensory information into an evolving description of the environment: dynamic stereo vision for capturing depth information, and knowledge-based landmark identification for reducing location uncertainty. Subsystems with these abilities, eventually, will become part of a robot navigation system.

Dr. Weymouth is a member of ACM and AAAI.

Ramesh Jain (M'79–SM'83) received the B.E. degree from Nagpur University in 1969 and the Ph.D. degree from the Indian Institute of Technology, Kharagpur, India, in 1975.

He is currently a Professor of Electrical Engineering and Computer Science, and the Director of the Artificial Intelligence Laboratory at the University of Michigan, Ann Arbor. Formerly he had been affiliated with Stanford University, IBM Almaden Research Labs, General Motors Research Labs, Wayne State University, University of Texas at Austin, University of Hamburg, West Germany, and Indian Institute of Technology, Kharagpur. His current research interests are in computer vision and artificial intelligence. He has published research papers addressing several aspects of the above areas. Some of the projects he has completed include: semiconductor wafer inspection using scanning electron microscopes and optical microscopes, automatic visual inspection of solder joints for IBM, laser range image processing for CAD-based vision systems, robotic navigation and rotorcraft navigation using machine vision, and applications of expert systems in inspection systems. He is a consultant to many companies in the areas of computer vision, artificial intelligence, and computer graphics.

Dr. Jain is a member of ACM, AAAI, the Pattern Recognition Society, the Cognitive Science Society, the Optical Society of America, the Society of Photo-Optical Instrumentation Engineers, and the Society of Manufacturing Engineers. He has been involved in organization of several professional conferences and workshops. Currently, he is on the editorial boards of *IEEE Expert*, *Machine Vision and Applications*, *Computer Vision, Graphics, and Image Processing*, the *Bulletin of Approximate Reasoning*, and *Image and Vision Computing*.