Energy Minimization of Contours Using Boundary Conditions

Sharat Chandran and A.K. Potty

Abstract—Reconstruction of objects from a scene may be viewed as a data fitting problem using energy minimizing splines as the basic shape. The *process* of obtaining the minimum to construct the "best" shape can sometimes be important. Some of the potential problems in the Euler-Lagrangian variational solution proposed in the original formulation [1], were brought to light in [2], and a dynamic programming (DP) method was also suggested. In this paper we further develop the DP solution. We show that in certain cases, the discrete form of the solution in [2], and adopted subsequently [3], [4], [5], [6] may also produce local minima, and develop a strategy to avoid this. We provide a stronger form of the conditions necessary to derive a solution when the energy depends on the second derivative, as in the case of "active contours."

Index Terms—Dynamic programming, energy minimization, deformable contours, optimal solutions, active contours.

1 Introduction

From [7] (see Table 1), we see that various early and intermediate level computer vision tasks are obtained by energy minimization of various functionals.

As a representative case, we consider obtaining contours designed to model shape, and we are particularly interested in applying the snake model of computing contours [1].

1.1 Minimizing Strategy

We turn our attention to the energy minimizing method, which, although does not affect the snakes paradigm, is *crucial* to the obtained behavior of the snake and is independent in its own right [2], [8]. These authors point out some problems in the original scheme, such as the correctness of the solution once a model is chosen, the running time of the algorithm, the robustness of the algorithm with respect to noise, etc. The basic objective of obtaining the minimum of the energy is achieved through DP (in the case of the former), and a greedy algorithm (in the case of the latter).

1.2 Related Research and This Paper

In this work, we further develop the non-variational solution in achieving global minimum. We remark that research, e.g., [3], [4], [5], [6] subsequent to [2] and implementing, for the purposes of this paper, virtually identical algorithms have also seemingly presupposed global minimum. Note that in, neither [1] nor [8] is there any attempt to obtain the global minimum energy of the snake. Further, our solution is not iterative and to this extent, it is similar to the spirit of the solution presented in [6].

The spline model implies a solution to a differential equation [1], [2]. The principal contribution in this paper, and the reason our algorithm differs from previously described algorithms is that we require the knowledge of exactly two end points of the contour. Unless such points are given, a boundary condition is not available, and hence the solution of any differential equation is suspect.

• *The authors are with the Computer Science and Engineering Department, IIT Bombay, India 400 076. E-mail: {sharat, apkp}@cse.iitb.ernet.in.*

Manuscript received 25 Feb. 1998. Recommended for acceptance by A.K. Jain. For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106432.

TABLE 1
Problems and the Corresponding Functionals

Problem	Regularization principle
Edge detection	$\int\left[(Sf-i)^2+\lambda(f_{xx})^2\right]dx$
Area based Optical flow	$\int\left[\left(i_x u+i_y v+i_t\right)^2+\lambda\left(u_x^2+u_y^2+v_x^2+v_y^2\right)\right]dxdy$
Contour based Optical flow	$\int\left[\left(V\cdot N-V^N\right)^2+\lambda\left(\frac{\delta V}{\delta x}\right)^2\right]$
Surface reconstruction	$\int\left[\left(S\cdot f-d^2+\lambda\left(f_{xx}+2f_{xy}^2+f_{yy}^2\right)\right)\right]dxdy$
Spatiotemporal approximation	$\int\left[(S\cdot f-i)^2+\lambda(\nabla f\cdot V+ft)^2\right]dxdydt$
Color	$\|I^y-Ax\|^2+\lambda\|Pz\|^2$
Shape from shading	$\int\left[\left(E-R(f,g)\right)^2+\lambda\left(f_x^2+f_y^2+g_x^2+g_y^2\right)\right]dxdy$
Stereo	$\int\left\{\left[\nabla^2 G*\left(L(x,y)-R\left(x+d(x,y),y\right)\right)\right]^2+\lambda(\nabla d)^2\right\}dxdy$
Contours	$\int E_{snake}(\mathbf{v}(s))ds$

2 The Basic Model

The basic model of the snake is described in this section, but we invite the interested reader to [1] for more details. Representing the contour by the position vector $\mathbf{v}(s) = (x(s), y(s))$ with the independent parameter s, we can write its energy functional as $E(\mathbf{v}(s)) = \int_0^1 E_{snake}(\mathbf{v}(s))ds$ where $E_{snake} = E_{int} + E_{ext}$.

Here, E_{ext} stands for the "external" energy of the curve and is due to image forces, and other external forces that the user may seek to impose. E_{int} represents the internal energy of the curve. Based upon theoretical and experimental considerations [1], the internal energy is modeled as

$$E_{int} = \frac{\left(\alpha(s)\left|\mathbf{v}_s(s)\right|^2+\beta(s)\left|\mathbf{v}_{ss}(s)\right|^2\right)}{2} \quad (1)$$

where α and β are parameters that are set by the user, and subscripts denote conventional derivatives as usual. We need to find $\mathbf{v}(s)$ such that $E(\mathbf{v}(s))$ is the minimum.

This is a classical problem from variational calculus and an iterative Euler-Lagrange equation can be set up for the discrete case. In [2], a few problems with this approach are pointed out. To their list, we would like to add

? The solution of any differential equation requires the specification of appropriate boundary conditions. A large class of unrelated functions can satisfy the differential equation, but not the differential equation *and* the boundary conditions. Our confidence in the described solution therefore wanes.

3 Prior DP Solution

First, consider the contour with the position vector $\mathbf{w} = v_1, v_2, \ldots, v_n$, $v_i = (x_i, y_i)$. The energy of the snake is

$$E_{snake} = \sum_{i=1}^{n} E_i$$

$$\text{where } E_i = \frac{\left(\alpha_i \left|v_i - v_{i-1}\right|^2 + \beta_i \left|v_{i+1} - 2vi + v_{i-1}\right|^2\right)}{2} + E_{ext}(v_i) \quad (2)$$

Note that we have produced a discrete version of (1). Further, we have assumed implicitly a step size of one, or alternatively, this restriction is absorbed in the α_i term. The approach consists of first supposing that the position vector v_i can have at most m degrees of freedom. The vector therefore can take only a finite set of values, and one could propose an exhaustive search algorithm. However such an algorithm is combinatorially implosive with a running time of $O(m^n)$. It would be hopeless to do such a search, even for small values of m, such as $m = 9$.

The total energy of the contour fortuitously works out to be sum of n individual energy terms $E_1, E_2, \ldots, E_n$, where each energy term E_i depend only on three points v_{i-1}, v_i, and v_{i+1}. Amini et al. [2] observed that due to this nature of the energy function (2), it is possible to do much better by organizing the energy terms as follows. We illustrate by considered only terms involving α, i.e., we have set $\beta_i = 0$ and ignored the external energy.

We set $S_0(v_1) = 0$ for each of the m positions of v_1, and

$$S_i(v_{i+1}) = \min_{v_i}\left\{S_{i-1}(v_i) + \alpha_i \left|v_i - v_{i+1}\right|^2\right\}$$

$$\therefore \quad \min E_{snake} = \min_m S_{n-1}(v_n)$$

For example, using superscripts to indicate the m degrees of freedom for the various components of the position vector,

$$S_1(v_2) = \min\left\{0 + \alpha\left|v_1^k - v_2\right|^2 : k = 1, 2, \ldots, m\right\}$$

where we have used $\alpha_i \equiv \alpha$, a common assumption, and fixed v_2 in some location within the m degrees of freedom. Then one iteration of the algorithm in [2] can be roughly summarized as in Fig. 1. After the minimum energy is obtained as $\min\left\{S_{n-1}(v_n^1), S_{n-1}(v_n^2), \ldots, S_{n-1}(v_n^m)\right\}$, a backward scan is used to find the actual vector **w**. The running time of the iteration is now $O(nm^2)$—a dramatic improvement.

The above is simply one possible position of the contour. It represents a local minimum energy position since we have allowed a degree of freedom of only m. The contour could of course be anywhere inside the image unbounded by m; thus, in order to obtain the final contour, Amini et. al. seemingly adopt an iterative algorithm as shown in Fig. 2 (compare Fig. 5, [2]) where m is set to a small value such as nine or 16.

3.1 Comments on Algorithm II

The problem with the implementation in [2] can be intuitively visualized if we consider a limiting case of a "point" snake, i.e., a snake made up of a single point. Such a snake is of course useless to the computer vision problem, but if we observe the iteration of the algorithm, and in particular Steps 3 and 4, we find that there is no application of the optimal cost function of the DP formulation. Instead, the algorithm reduces to an iterative greedy search in an $\sqrt{m} \times \sqrt{m}$ neighborhood. It comes as little surprise then that the authors of [8] state in their abstract "[our algorithm] … gives results comparable to the DP algorithm, but is much faster …."

In other words, Algorithm II achieves the best possible curve within one iteration (conceptually thought of as in one dimension). However, it does not apply the principle of dynamic programming moving from one iteration to another (conceptually in two dimensions), and therefore compromises on the promised attempt of the global minimum. Note that this is true for any bounded value of m, the degree of freedom.

```
0. for (i = 1; i < n ; i = i+ 1)
   for (k = 1; k ≤ m ; k = k + 1) S_i (k) = ∞
0. S_0 (k) = 0
1. for (i = 1; i < n ; i = i + 1)
2.   for (k = 1; k ≤  m ; k = k + 1) \>
3.     for (j = 1; j ≤  m ; j = j + 1) begin
4.       temp = 0.5 α|v_i − v_{i−1}^k|^2 + E_ext + S_{i−1}(j)
5.       if (temp < S_i(k)) begin
6.         S_i(k) = temp
7.         M(k) = j
         end
       end
```

Fig. 1. Algorithm I: One iteration in the DP algorithm.

1. Compute energy of the contour at some position using the above algorithm.
2. If the energy is higher than the energy of the previous position, exit with the previous position.
3. Perturb the next point locally.
4. Choose the lowest energy of the m possible ways of the above perturbation.
5. Go to Step 1.

Fig. 2. Algorithm II: Iterating the basic DP loop.

Another issue of concern in [2] is the optimization framework introduced (for example, (20) and (23) in [2]). It appears that this framework is not explicitly used in their algorithm. The parameter m is somewhat artificially introduced and has to be "hardcoded" in the algorithm. In comparison, the variational approach has no such parameter (the parameter γ is introduced for a different purpose, viz., the numerical solution). Yet, the Euler-Lagrange equation must follow from the continuous dynamic programming formulation.

An increasing evidence is that, in the method of [2], the algorithm halts when "there is no change in the optimal cost function." As defined above, this can very well happen in a local minimum, for instance, when there is no change in the image forces around the current location of the contour in a large image of size $1{,}028 \times 1{,}028$. This also implies that we cannot predict the running time of the overall algorithm even though we know the time for each iteration.

4 The DP Solution—Applying the Boundary Conditions

4.1 Admissibility

Before we present our ideas, we would like to start with some (fairly well known) clarifications. For obvious reasons, the minimum energy of the snake, a continuous function, is derived using techniques in numerical analysis. However, it is not clear what minimum is obtained. To amplify, provided a scene consisting of a set of objects of interest, what is not desired is to obtain one snake that has a minimum energy, but rather to tie a set of snakes to the set of objects of interests. In other words, we do not attempt the segmentation problem here. In the original formulation, the user or a higher level process positions the approximate snake near the object of interest. For us this implicitly defines a zone of *admissible* snakes What we do not desire is for the admissible snake to get trapped in a local minimum within the zone.

4.2 The Formulation of a Noniterative Solution

To set the stage for our solution, we first set (as in [2]) the energy functional of the snake to depend only on the first order derivative. We consider the general case later. Let

$$I = \int_{\alpha}^{\beta} F\big(\mathbf{v}(s), \mathbf{v}'(s), s\big)ds, \tag{3}$$

where $\mathbf{v}' = d\mathbf{v}/ds$.

Imposing the boundary conditions, $\mathbf{v}(\alpha) = \mathbf{v_a}$ and $\mathbf{v}(\beta) = \mathbf{v_b}$, the initial step is to divide the interval $\alpha \le s \le \beta$ into n subintervals of equal length Δs by the points $s_j = \alpha + j\Delta s$, $j = 0, \ldots, n$. For every admissible curve, let $\mathbf{v}_j = \mathbf{v}(s_j)$. If Δs is small, then $\mathbf{v}'(s_j)$ can be approximated by $\mathbf{v}'\big(s_j\big) \simeq \mathbf{v}_{j'} \approx \frac{\mathbf{v}_{j+1} - \mathbf{v}_j}{\Delta s}$.

In terms of the points $(s_j, \mathbf{v}_j)$, the integral can be represented approximately by the finite sum,

$$I \approx \sum_{j=0}^{n} F\left(\mathbf{v}_j, \frac{\mathbf{v}_{j+1} - \mathbf{v}_j}{\Delta s}, s_j\right)\Delta s \tag{4}$$

Hence, now the problem of finding the function $\mathbf{v}(s)$ which minimizes (3) can be replaced by the approximating problem of finding the $n-1$ values $\mathbf{v}_j$ which minimizes (4) when $\mathbf{v}_0$ and $\mathbf{v}_n$ are fixed at values $\mathbf{v_a}$ and $\mathbf{v_b}$, respectively.

Define the *state functions* [9] $\Lambda_k(\xi)$ as,

$$\Lambda_k(\xi) = \min_{\mathbf{v}_{k+1}, \ldots, \mathbf{v}_{n-1}} \sum_{j=k}^{n-1} F\left(\mathbf{v}_j, \frac{\mathbf{v}_{j+1} - \mathbf{v}_j}{\Delta s}, s_j\right)\Delta s, \; k = 0, 1, \ldots, n-1 \tag{5}$$

where $\mathbf{v}_k = \xi$. Then

$$\Lambda_k(\xi) = \min_{\mathbf{v}_{k+1}}\left\{F\left(\xi, \frac{\mathbf{v}_{k+1} - \xi}{\Delta s}, s_k\right)\Delta s + \min_{\mathbf{v}_{k+2}, \ldots, \mathbf{v}_{n-1}} \sum_{j=k+1}^{n-1} F\left(\mathbf{v}_j, \frac{\mathbf{v}_{j+1} - \mathbf{v}_j}{\Delta s}, s_j\right)\Delta s\right\}, \tag{6}$$

i.e.,

$$\Lambda_k(\xi) = \min_{\mathbf{v}_{k+1}}\left\{F\left(\xi, \frac{\mathbf{v}_{k+1} - \xi}{\Delta s}, s_k\right)\Delta s + \Lambda_{k+1}\big(\mathbf{v}_{k+1}\big)\right\} k = 0, 1, \ldots, n-2. \tag{7}$$

Also

$$\Lambda_{n-1}\big(\xi\big) = F\left(\xi, \frac{\mathbf{v_b} - \xi}{\Delta s}, s_{n-1}\right)\Delta s \tag{8}$$

Note that this formulation can be done only because (5) is separable. Here it should be mentioned that the DP method can be applied only to separable functions.

Due to this formulation, the minimum value of I is $_0$(a).

In the above formulation, ξ, and therefore $\mathbf{v}_k$ are continuous variables. To solve the problem numerically, we treat both ξ and $\mathbf{v}_k$ as discrete, that is, for each k, we determine a discrete set of values which ξ and so $\mathbf{v}_k$ can assume. Then $\Lambda_k(\xi)$ is tabulated only for these values. To compute $\Lambda_k(\xi)$, $\mathbf{v}_{k+1}$ is only allowed to take values in the discrete set of values appropriate to ξ.

In addition to tabulating the values of $\Lambda_k(\xi)$, we also tabulate $\hat{\mathbf{v}}_{k+1}\big(\xi\big)$, the value of $\mathbf{v}_{k+1}$ corresponding to ξ at the $k+1$st stage that minimizes (7). At the final stage, we determine $\Lambda_0(a)$, as well as $\mathbf{v}_1^* = \mathbf{v}_1(a)$. The optimal values of other $\mathbf{v}_k$ would be obtained by means of

$$\mathbf{v}_{k+1}^* = \hat{\mathbf{v}}_{k+1}\big(\mathbf{v}_k^*\big), \qquad k = 1, 2, \ldots, n-2. \tag{9}$$

4.3 Remarks About Our Formulation

By this method, we actually determine a whole family of extremal curves as we move away from the point $\mathbf{v_b}$. At the last step, one determines the particular extremal curve emanating from $\mathbf{v_b}$, which also passes through $\mathbf{v_a}$. This is illustrated in Fig. 3. This ensures that the contour obtained by the DP process is indeed the globally minimal contour passing through $\mathbf{v_a}$ and $\mathbf{v_b}$.

Fig. 3. Various curves emanate from point $\mathbf{v_b}$. The algorithm picks the one that passes through $\mathbf{v_a}$ and which has least cost. The figure also shows a stage in the DP process.

Also note that the "point snake" of Section 3.1 poses no problem since the boundary conditions will be satisfied. Finally, the method is intrinsically noniterative, and we can therefore bound the overall worst case running time to be $O(nP)$ where P is the size of the admissible region, and n is the desired number of points on the snake.

4.4 Incorporating Higher-Order Differential

Having considered the basic strategy, in this section we see how to incorporate higher order differentials which are essential to model the snake accurately. We set

$$I = \int_0^1 F\big(\mathbf{v}(s), \mathbf{v}'(s), \mathbf{v}''(s), s\big)ds \tag{10}$$

where $\mathbf{v}'(s) = d\mathbf{v}(s)/ds$ and $\mathbf{v}''(s) = d\mathbf{v}'(s)/ds$.

This can be approximated by the following finite sum,

$$I \approx \sum_{j=1}^{n-1} \mathcal{F}\big(\mathbf{v}_{j+1}, \mathbf{v}_j, \mathbf{v}_{j-1}\big)\Delta s \tag{11}$$

where

$$\mathcal{F}(x, y, z) = F\left(y, \frac{x-y}{\Delta s}, \frac{x - 2y + z}{\Delta s^2}, s\right)$$

Now, since (11) is *separable*, the *state functions* [9] associated with (11) become

$$\Lambda_k\big(\xi_1, \xi_2\big) = \min_{\mathbf{v}_{k+1}}\big\{\mathcal{F}\big(\mathbf{v}_{k+1}, \xi_1, \xi_2\big)\Delta s + \Lambda_{k+1}\big(\mathbf{v}_{k+1}, \mathbf{v}_k\big)\big\}, \quad k = 1, \ldots, n-2 \tag{12}$$

and

$$\Lambda_{n-1}(\xi_1, \xi_2) = \mathcal{F}(\mathbf{v_b}, \xi_1, \xi_2)\Delta s \tag{13}$$

Here, $\mathbf{v_b}$ is one of the two boundary positions, the other being $\mathbf{v_a}$ used later below.

To solve the problem numerically, we treat ξ and $\mathbf{v}_k$ as discrete, that is, for each k, we have a discrete set of values which ξ and, hence, $\mathbf{v}_k$ can assume.

The solution now proceeds with the *tabulation* of $\Lambda_k(\xi_1, \xi_2)$ (As a refinement, to compute $\Lambda_k(\xi_1, \xi_2)$, $\mathbf{v}_{k+1}$ is allowed to take only those values appropriate to ξ_1 and ξ_2). The progress is, intuitively, *backwards*, starting first with the use of (13) to tabulate the values of Λ_{n-1} obtained at different values of ξ_1 and ξ_2. At later stages we vary $\mathbf{v}_{k+1}$ over the possible values it can take subject to ξ_1 and ξ_2 and tabulate the values of $\Lambda_k(\xi_1, \xi_2)$. The corresponding values of $\mathbf{v}_{k+1}$ at which the minimum is attained for every pair (ξ_1, ξ_2), namely, $\hat{\mathbf{v}}_{k+1}\big(\xi_1, \xi_2\big)$ is also tabulated.

When all the iterations in (12) are completed, the algorithm proceeds, intuitively, forward. First, define $\mathbf{v}_0^* = \mathbf{v_a}$ and $\mathbf{v}_1^* = \mathbf{v_a} + \epsilon$, where ϵ is an error term introduced due to the discretization. Optimal positions $\mathbf{v}_{k+1}^*$ can be obtained from the tabulated information using (14)

$$\mathbf{v}_{k+1}^{*} = \hat{\mathbf{v}}_{k+1}\left(\mathbf{v}_{k}^{*}, \mathbf{v}_{k-1}^{*}\right),$$

$$k = 1, \ldots, n-2. \tag{14}$$

Note the use of the boundary values $\mathbf{v_a}$ and $\mathbf{v_b}$ as contrasted with [2], where the convergence of the solution depends on reduction in energy values.

5 Conclusion

A controlled continuity spline can be used to determine the shape of an object in an image by subjecting it to image forces, and determining its final equilibrium position. In this paper we have addressed the problem of computing the minimum energy using DP principles.

In particular, we have built upon both the continuous and discrete forms of the solution outlined in [2]. There, the dynamic programming principle to avoid local minima is used within a single iteration (i.e., in one dimension), but is not applied from one iteration to another (i.e., in two dimensions). By introducing the idea of a goal node using known locations of the final contour, we have provided a two dimensional form of the DP method. An application of our method is in tracking contours in multiple frames.

The framework of the solution provided in the continuous case in [2] fails when the energy functional depends on the second derivative. We have also derived a solution in this case again using principles of DP.

In order to demonstrate the correctness of the algorithm developed in Section 4, it was run on various images. We can conclude that our method matches the boundary that the human eye would have picked out whereas other methods sometime settle down in a nonglobal minimum.

References

[1] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active Contour Models,” *Int’l J. Computer Vision*, vol. 1, pp. 321-331, 1988.

[2] A. Amini, T. Weymouth, and R. Jain, “Using Dynamic Programming for Solving Variational Problems in Vision,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1990.

[3] Y. Ohnishi and Y. Ohta, “Extracting Energy Minimized Contours Using Dynamic Programming.” bachelor’s thesis, Tsukuba Univ., Mar. 1990. In Japanese.

[4] N. Ueda and K. Mase, “Minimization of Active Contour Energy Using Dynamic Programming,” *Spring Nat’l Convention Record, Inst. Electronics, Information, and Comm. Engineers*, 1991. In Japanese.

[5] K. Fujimura, N. Yokoya, and K. Yamamoto, “Motion Tracking of Deformable Objects by Active Contour Models Using Multi-Scale Dynamic Programming,” *J. Visual Comm. Image Representation*, Dec. 1993.

[6] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, “Dynamic Programming for Detecting, Tracking and Matching Deformable Contours,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 294-302, Mar. 1995.

[7] T. Poggio, V. Torre, and C. Koch, “Computational Vision & Regularisation Theory,” *Nature*, vol. 317, pp. 314-319, 1985.

[8] D.J. Williams and M. Shah, “A Fast Algorithm for Active Contours,” *Third Int’l Conf. Computer Vision*, pp. 592-595, 1990.

[9] R. Bellman, *Dynamic Programming*. Princeton, N.J.: Princeton Univ. Press, 1957.