Accelerating active contour algorithms with the Gradient Diffusion Field

Chris Kiser [*†] Chris Musial [†] Pradeep Sen [*]
Advanced Graphics Lab [*]
University of New Mexico
Boeing-SVS [†]
Albuquerque, NM

Abstract

Active contours were proposed by Kass et al. as a way to represent the contours of an image. Although the method is simple, one of its shortcomings is its inability to converge into concave structures. The Gradient Vector Flow (GVF) algorithm was put forth by Xu and Prince to succesfully address the concave structure problem. Although there has been much research into GVF, little has been done to reduce its computation time, which makes it unsuitable for applications requiring real-time processing of images. In this paper, we propose a method for computing an approximation of the GVF, called the Gradient Diffusion Field (GDF), which exhibits the same useful properties of the GVF but converges faster and requires less resources for implementation. Our proposed method is also more amenable for real-time hardware and we outline a method for implementing an active contour algorithm in FPGA hardware using the GDF.

1. Introduction

An active contour (also known as a "snake") is a collection of points that lie along the contour edge of an image that is found through an iterative solution [4]. As its name implies, the contour is active and dynamic in behavior; the model is constantly trying to find a minimum energy state and will therefore continually adjust itself to find it. However, the active contour algorithm exhibits problems with local minima when the salient feature of the image creates a concave edge. Xu and Prince have proposed a method called the Gradient Vector Field (GVF) to help resolve the local minima problem by generating a 2-D vector field that guides the contour inside the problematic concave regions [8].

Although the GVF improves the robustness of the active contour algorithm, it is impractical to compute for real-time systems. In this paper, we present a novel method called the Gradient Diffusion Field (GDF) which emulates the behavior of the GVF but is faster and easier to compute. We begin with a review of the active contour as well as its GVF extension. We then describe our proposed algorithm and present convergence results to compare GVF and GDF approaches. Finally, we describe an active contour algorithm based on GDF implementated on a Field Programmable Gate Array (FPGA) for operation at real-time rates.

2. Active Contours

The seminal paper in the field of active contours is by Kass et al., where they define the active contour to be "a controlled continuity spline under the influence of image forces and external constraint forces" [4]. Here, the image forces are a function of the image itself and include gradients, lines, edges and contrasts. The external forces are constraints applied by either controlling mechanisms (e.g. smoothness, roundness, "springiness") or human interface requirements. The goal is to minimize the energy of the active contour given by

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

where E_{int} is the internal energy of the contour due to mechanical forces, E_{im} is the energy added by the image, E_{con} is the forces given by external constraints, and $\mathbf{v}(s) = (x(s), y(s))$ is the parametric representation of the contour. The internal contour energy, E_{int}, of Eq. 1 is given by

$$E_{int} = \frac{1}{2}(\alpha(s)|\mathbf{v}'(s)|^2 + \beta(s)|\mathbf{v}''(s)|^2) \quad (2)$$

Here, the first order term $\mathbf{v}'(s)$ controls the spacing between points while the second order term $\mathbf{v}''(s)$ controls the amount of bend centered at each point, with $\alpha(s)$ and $\beta(s)$ as weighting coefficients. The second term of Eq. 1, E_{im}, expands to

$$E_{im} = w_{line}E_{line} + w_{edge}E_{edge} \quad (3)$$

The term E_{line} causes attraction to light or dark lines according to w_{line}. The term E_{edge} causes attraction the edges in the image $\mathbf{I}$ and can be written as

$$E_{edge} = -|\nabla \mathbf{I}(x, y)|^2 \quad (4)$$

where the ∇ operator is the 2-D gradient of image $\mathbf{I}$.

In their work, Kass et al. find the solution to the minimization of Eq. 1 through a semi-implicit, iterative method for solving a set of simultaneous equations [4]. The approach uses variational calculus to derive the minimizing equations in the Euler-Lagrange format. The coefficients of these equations are then formed as a matrix which is inverted to find the minimum energy of the contour equation. Other methods have been proposed for finding the active contour, such as dynamic programming [1], gradient descent [3], and greedy search [6]. The greedy method minimizes Eq. 1 over a small neighborhood for each contour point and then moves it to the lowest energy location. This method is susceptible to local minima but is more suitable for high-performance, real-time systems. Therefore, we will use the greedy method as the active contour iteration algorithm for the remainder of the paper.

Since their introduction, active contours have been used for a wide variety of applications. However, the failure of the algorithm to converge correctly for concave features led researchers to develop modifications to the active contour, such as the GVF.

3. Gradient Vector Flow

The Gradient Vector Flow (GVF) algorithm was proposed by Xu et al. to aid the convergence of the active contour into concave regions [8]. The traditional active contour algorithm uses the gradient in its minimization formula given in Eqs. 1 and 3 to pull the contour toward its minimum value. As Fig. 1c shows, the greedy search gets stuck in a local minimum and fails to converge to the concave region of the image. However, the GVF algorithm provides the magnitude of the vector flow field (Fig. 1d) for the minimization algorithm, allowing the active contour to converge correctly (Fig. 1e). Formally, the GVF proposed by Xu et al. can be written as

$$\begin{aligned} b(x,y) &= f_x(x,y)^2 + f_y(x,y)^2 \\ c^1(x,y) &= b(x,y) f_x(x,y) \\ c^2(x,y) &= b(x,y) f_y(x,y) \\ u^{n+1}_{i,j} &= (1 - b_{i,j}\Delta t)u^n_{i,j} + r(u^n_{i+1,j} + u^n_{i,j+1} \\ &\quad + u^n_{i-1,j} + u^n_{i,j-1} - 4u^n_{i,j}) + c^1_{i,j}\Delta t \\ v^{n+1}_{i,j} &= (1 - b_{i,j}\Delta t)v^n_{i,j} + r(v^n_{i+1,j} + v^n_{i,j+1} \\ &\quad + v^n_{i-1,j} + v^n_{i,j-1} - 4v^n_{i,j}) + c^2_{i,j}\Delta t \end{aligned} \tag{5}$$

The terms $f_x(x,y)^2$ and $f_y(x,y)^2$ are gradient edge maps in the x and y directions. The term Δt is a time constant and r is a scaling constant. These equations present an iterative method for calculating the GVF. Although the GVF makes active contours more robust, a large number of iterations of Eq. 5 are required to spread the magnitude of the GVF enough so that the active contour can converge into concave regions.

We define the *reach* of the GVF to be the distance from the edge to the first point where the magnitude of the GVF has zero value. For the example of Fig. 2, the reach of the GVF is 41 pixels after 200 iterations. This limits the maximum distance from the original edge that the GVF can affect the active contour. While more iterations of Eq. 5 wwould increase the reach of the GVF, the large number of iterations typically required is what makes it impractical for real-time applications.

Applications of the GVF are as diverse as the active contour itself. Wu in [7] used the GVF and active contours for tracking lip movements in human speech to aid in MPEG-4 compression. Paragios in [5] used the GVF and a form of the active contour for boundary extraction in medical and satellite images. Yu-Tai in [9] even expanded the GVF to 3 dimensions to form a 3-D model of a human urethra from a series of images. These are only a sample of many applications of the GVF, however to date no work has been uncovered which addresses reducing the computation time of the GVF.

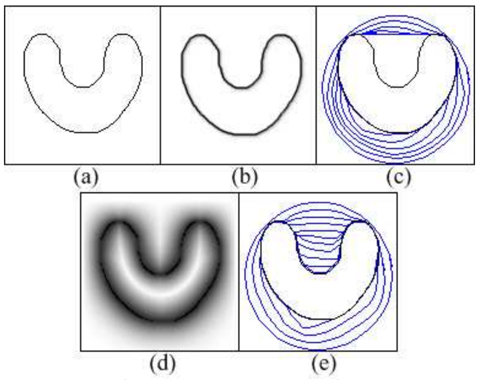

Figure 1. Comparison of GVF to traditional active contour using greedy search. (a) Test image and (b) the magnitude of its gradient. (c) 30 iterations of greedy algorithm using gradient magnitude. (d) Magnitude of GVF after 200 iterations. (e) 30 iterations of greedy algorithm using GVF.

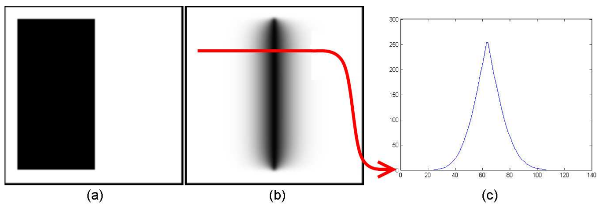

Figure 2. Visualization of the GVF spread. (a) Test image. (b) Magnitude of GVF after 200 iterations. (c) Cross section of GVF.

4. The Gradient Diffusion Field

The address the problem of the large number of iterations required by the GVF algorithm, we introduce an approximation of the magnitude of the GVF we call the Gradient Diffusion Field (GDF). The GDF is computed by applying a Gaussian of Gradient operator [2] recursively to diffuse or “smear” the gradient away from its corresponding edge. The formulation for the GDF is

$$GDF_{i+1} = \max(||\alpha(i) \cdot GoG * GDF_i||, GDF_i) \tag{6}$$

where

$$\alpha(i) = \alpha(i-1) - \frac{1}{k}, \qquad \alpha(0) = 1 \tag{7}$$

and

$$GDF_0 = ||GoG * \mathbf{I}|| \tag{8}$$

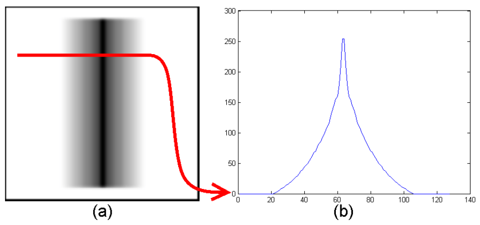

Figure 3. Visualization of spread of GDF. (a) Magnitude of GDF. (b) Cross section of the GDF at the line indicated.

The last equation shows that the starting point GDF_0 is the Gaussian Gradient of the image **I**. We then apply Eq. 6 recursively to find the pixel-by-pixel maximum between the previous GDF iteration and the current GDF iteration. Effectively, this equation is a recursive gradient that finds derivatives to the i^{th} order. This term will tend to add noise into the GDF, however the Gaussian portion of the Gaussian Gradient will help reduce noise effects. The term $\alpha(i)$ in Eq. 6 is added to further reduce the contribution of the higher order derivatives and thus help to reduce the noise effects. The term k is the number of iterations to perform.

5. Results

In order to compare our GDF algorithm with the GVF, we first measure the increase in capture range. Fig. 3 shows the computation of the GDF after 14 iterations. The peak of the GDF is maintained at the position of the edge in the original image and the reach was set to be 41 pixels, which allows us to compare it with the GVF approach of Fig. 2. The overall shape is similar to the GVF, but the number of iterations required to compute it was much less (14 vs. 200).

We show a plot of the reach vs. number of iterations for both the GVF and GDF in Fig. 4. We note that the GDF grows at a much larger rate, increasing its reach in a smaller number of iterations than the GVF. The result of applying the GDF algorithm to the test image of Fig. 1 is shown in Fig. 5.

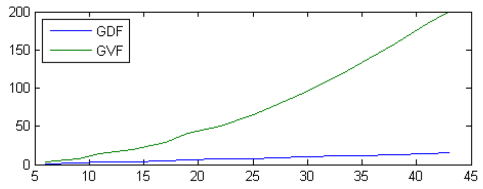

Figure 4. Increase in GDF capture range compared to GVF as a function of the number of iterations.

Our measure for successful approximation of the GVF is not accurately expressed with mathematical measurements such as Mean Square Error (MSE). Since

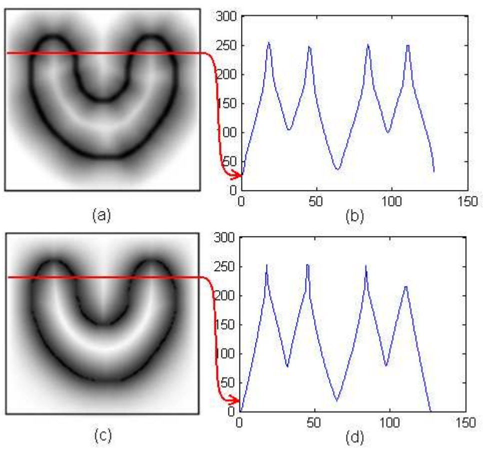

Figure 5. The cross sections of the magnitude of GDF and GVF for the test image. The shapes and peaks of the curves match.

our method is trying to approximate the impact of the GVF on the contour, the measure of success is whether the GDF can obtain the same increase in capture range and have the same effect on the active contour as the GVF. Fig. 6 shows successive stages of an active contour algorithm applied to our test image using the GDF. The active contour required 30 iterations to converge using the GVF in Fig. 1e and a comparable 33 iterations to converge in Fig. 6. We also tested the GDF with a realistic image that might be used with an active contour algorithm (Fig. 7). The GDF achieved the same quality as the GVF in a smaller number of iterations.

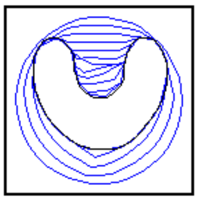

Figure 6. Convergence of active contour in 33 steps using the GDF.

6. FPGA Implementation

The GDF is also more efficient than the GVF in its implementation in Field Programmable Gate Arrays (FPGAs). Hardware implementations of the GVF or GDF can be realized by implementing each iteration of Eq. 5 or Eq. 6, respectively, in separate pipelined stages. Fig. 8 shows one of the stages of the GVF implementation, while Fig. 9 shows a stage of the GDF. The resource requirements for each of the GDF and GVF is shown in Table 1. We assume that inputs to each stage have been normalized to fit within a single hardware multiplier 18 bits wide.

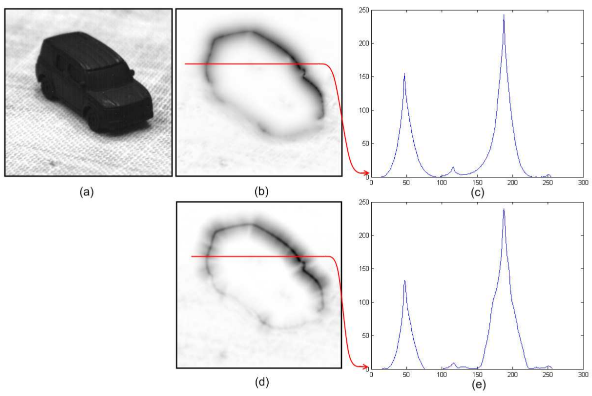

Figure 7. GDF applied to real-world test image (a). (b) result of GVF with cross-section (c). (d) result of GDF with cross-section (e). The GDF algorithm took 33 iterations, compared to 30 for the GVF for a comparable reach.

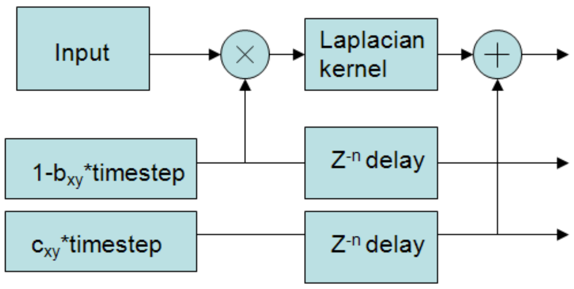

Figure 8. First stage of GVF

Although the GDF can be realized in real time hardware through a series of stages, the active contour algorithm is still an iterative algorithm. We can implement the Greedy active contour algorithm in a real time system using a discrete filter kernel method, which implements Eq. 1 on each cell of the kernel filter. This finds the energy at each pixel surrounding the contour point then moves the contour point to the location of minimum energy. The kernel size determines the maximum distance in pixels per iteration that each active contour point can move. For example, a 5×5 kernel can move each contour point a maximum of 2 pixels per iteration while a 7×7 kernel can move it by 3 pixels.

7. Conclusions

We have presented a novel method called the Gradient Diffusion Field that efficiently approximates the magnitude of the Gradient Vector Flow of an image and can be used to guide active contour algorithms when the image has concave regions. Although the method is practical for only for small movements in the active contour, the target application is for tracking objects using a high frame-rate video stream where the object has small shifts from image to image. Further study of this method is left for later work.

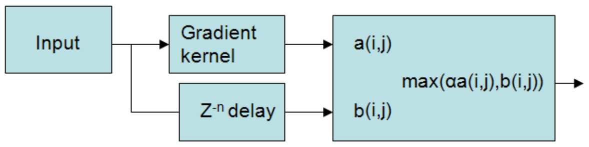

Figure 9. First stage of GDF

	1 GVF	1 GDF	200 GVF	14 GDF
Kernel Line Buffers	4	4	800	56
Kernel Multipliers	0	25	0	350
Kernel Adders	8	25	1600	350
Term Multipliers	4	0	800	0
Term Adders	6	0	1200	0
Term Line Buffers	2	1	400	14

Table 1. Resources required in the FPGA implementation of the GDF and GVF. The first two columns show resources for a single stage, the others for the full pipeline required for sufficient reach. Recall that 200 GVF iterations are approximately equivalent to 14 GDF iterations.

References

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

[2] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[3] J. Ivins and J. Porrill. Everthing you always wanted to know about snakes. *AIVRU Technical Memo 86, July 1993 (Revised June 1995; March 2000)*, 2000.

[4] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. Journal of Computer Vision*, 1(4):321–331, 1987.

[5] N. Paragios, O. Mellina-Gottardo, and R. Visvanathan. Gradient vector flow fast geodesic active contours. *ICCV 2001. Proceedings. Eighth IEEE International Conference on Computer Vision*, 1:67–73, 2001.

[6] D. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, 55(1):14–26, 1992.

[7] Z. Wu, P. Aleksic, and A. Katsaggelos. Lip tracking for mpeg-4 facial animation. *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*, pages 293–298, 2002.

[8] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.

[9] C. Yu-Tai, C. Lin, K. Tan, and K. Lau. Rectangular meshes construction of the human urethra using 3-d gvf snakes. *Proceedings of SPIE*, 5369:539–549, 2004.