In *Proceedings of IEEE International Conference on Image Processing (ICIP'01)*, volume 3, pages 1103–1106, 2001.

LEVEL SET SEGMENTATION IN GRAPHICS HARDWARE

M. Rumpf, R. Strzodka

University of Duisburg, Applied Mathematics, Lotharstr. 65, D-47048 Duisburg

ABSTRACT

Implicit active contours are a very flexible technique in the segmentation of digital images. A novel type of hardware implementation is presented here to approach real time applications. We propose to exploit the high performance of modern graphics cards for numerical computations. Vectors are regarded as images and linear algebraic operations on vectors are realized by the graphics operations of image blending. Thus, the performance benefits from the high memory bandwidth and the economy of command transfers, while the restricted precision does not infect the qualitative behavior of the level set propagation. Here we pick up a first order solver for the basic implicit level set model and present an implementation performing at 2ms for an explicit timestep on a 128^2 image.

1. INTRODUCTION

Robust and efficient segmentation algorithms on digital images are a key task in many computer vision applications. Morphological techniques such as the watershed transform and its successors [1, 2, 3] based on the simulated rainfall on the topographically interpreted image, and propagation techniques such as the active contour models [4, 5, 6, 7, 8] based on a controlled curve or surface evolution, have been presented recently. Especially the latter have proven very flexible in incorporating diverse criteria for segmentation. In early works the curves were explicitly parameterized, which thus prohibited the approximation of segments topologically different from the initial set in the front propagation, because the curve could not split. Implicit models [8, 6], in contrast, represent the curve as the zero level curve of a function $\phi(t,\cdot) : \Omega \to \mathbb{R}$ defined on the whole domain Ω. The evolution of the function is controlled by a PDE, where an external term includes information about the initial image. Upwind methods [9, 5] allow a stable implementation of this front propagation problem. The level set then is able to split and merge during the evolution.

Our concern is the efficient implementation of the basic implicit model in graphics hardware. As with other data transfer dominated problems such as volume rendering [10] and filtering [11], where implementations in graphics hardware have proven very efficient, the level set models could also benefit from the higher memory bandwidth and the vector-like image operations of graphics cards. The actual task is the reformulation of the numerical schemes so that operations can be performed on entire images while respecting the graphics restrictions of number formats and operations. Thereby the reduced number precision is of no further concern, as long as the algorithm can visibly reproduce the segmentation results, since this is the relevant criterion in applications.

2. REVIEWING THE LEVEL SET MODEL

We briefly review the analytical model and then consider a discretization of the basic level set model for image segmentation as presented in [12]. Here we mainly focus on the set of the required arithmetic operations for later conversion into image operations supported by graphics hardware.

We consider the domain $\Omega := [0,1]^d$, $d = 2, 3$ and ask for a solution of the following nonlinear initial value problem: Find $\phi : \mathbb{R}^+ \times \Omega \to \mathbb{R}$ such that

$$\begin{aligned} \partial_t \phi + f\|\nabla\phi\| &= 0 \quad , \quad \text{in } \mathbb{R}^+ \times \Omega\,, \\ \phi(0) &= \phi_0\,, \quad \text{on } \Omega\,, \\ \partial_\nu \phi &= 0 \quad , \quad \text{on } \mathbb{R}^+ \times \partial\Omega, \end{aligned}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is the normal velocity of the level set, which incorporates the external forces derived from the original image $p : \Omega \to \mathbb{R}$. We confine to propagation speed which depends solely on the underlying image. A common choice for f in this context is a characteristic function χ_S, where S is a set containing the image intensities chosen for segmentation. A dependence on the evolving level set function itself for example via curvature type regularization terms can in principle also be incorporated. However, we skip this expansion beyond the basic model here. It will be an issue for future considerations.

We discretize the problem with finite differences on a uniform quadrilateral, respectively hexahedral grid. We enumerate the grid elements with a d-dimensional multi-index α and denote the element centers by the d-dimensional vectors x^α. The discrete functions over this grid will be put in capital letters (Φ) and the corresponding element vectors consisting of all the constant values on grid-elements will be marked by a bar: $\bar{\Phi} = (\bar{\Phi}_\alpha)_\alpha = (\Phi(x^\alpha))_\alpha$.

Setting $u := \nabla\phi$ and $H(u) := f\|u\|$ we ask for a stable numerical flux approximating $H(u)$. Here we can pick up the upwinding methodology used in finite volume computations. Thus, the approximation of H with the numerical flux function $g : \mathbb{R}^{d\times d} \to \mathbb{R}$ gives us the upwind level set scheme

$$\Phi_\alpha^{k+1} = \Phi_\alpha^k - \tau_k \cdot g(D_\alpha^- \Phi^k, D_\alpha^+ \Phi^k) \quad (1)$$

$$D_\alpha^+ \Phi^k := \left(\frac{\Phi^k_{\alpha+e_i} - \Phi^k_\alpha}{\left| x_i^{\alpha+e_i} - x_i^\alpha \right|} \right)_{i=1\ldots d} \quad (2)$$

$$D_\alpha^- \Phi^k := \left(\frac{\Phi^k_\alpha - \Phi^k_{\alpha-e_i}}{\left| x_i^{\alpha} - x_i^{\alpha-e_i} \right|} \right)_{i=1\ldots d}, \quad (3)$$

where τ_k is the current timestep width. The discrete solution $\Phi^k(.)$ is expected to approximate $\phi(\sum_{j=0}^{k-1} \tau_j, .)$. To satisfy the natural boundary condition we set $D_\alpha^\pm \Phi^k := 0$ for all α enumerating border elements. For convex H a simple choice for g is the Enquist-Osher flux [9]

$$\begin{aligned} g(U,V) &:= H(0) + \int_0^U H'(s)^+ ds + \int_0^V H'(s)^- ds \\ &= F^+ \sqrt{\|U^+\|^2 + \|V^-\|^2} \\ &\quad + F^- \sqrt{\|U^-\|^2 + \|V^+\|^2} \quad (4) \\ U^+ &:= \max(U, 0), \qquad U^- := \min(U, 0). \end{aligned}$$

Moreover to ensure convergence we have to satisfy a CFL condition $\max_\alpha |F_\alpha| \cdot \tau_k < \min_{\alpha,i} \left| x_i^{\alpha+e_i} - x_i^\alpha \right|$. The equation (1) together with the definitions (2-4) define the explicit formula for the next timestep solution. This formula must now be expressed in terms of graphics operations.

3. THE UPWIND SCHEME IN IMAGE OPERATIONS

The high performance of graphics cards depends on the processing of large data blocks, e.g. an operation on an entire image is incomparably much faster than the transformation of each image's pixel by this operation. Therefore, we must formulate the upwind level set scheme in operations which act on an entire element vector $\bar{\Phi}$ represented by an image. There are moreover two major restrictions we have to pay attention to. Firstly, all numbers in graphics hardware are limited to the range $[0, 1]$. Greater or smaller numbers resulting in calculations are automatically clipped to 1 or 0 respectively. Therefore, though we can subtract images, we will never obtain negative results. Secondly, the internal fixed point numbers consists of at most 12 bits per color component (usually 8), which makes it difficult to represent small numbers and nonlinear operations.

We are faced with these restrictions when trying to implement the differential operators from (2) and (3), because the differential differences may always take a different sign and the denominators $\left| x_i^{\alpha+e_i} - x_i^\alpha \right|$ are very small. On an equidistant grid, however, all the denominators in these differential differences equal the grid specific diameter h, so that we can factorize it out of g (cf. (1) and (4)). Then, with the definition of a shift operator $T_\gamma \bar{V} := (\bar{V}_{\alpha-\gamma})_\alpha$ we can express the new differential operators as operations on element vectors

$$\begin{aligned} \bar{D}^+ \bar{\Phi}^k &:= \left(T_{-e_i} \bar{\Phi}^k - \bar{\Phi}^k \right)_{i=1\ldots d}, \\ \bar{D}^- \bar{\Phi}^k &:= \left(\bar{\Phi}^k - T_{e_i} \bar{\Phi}^k \right)_{i=1\ldots d}. \end{aligned}$$

Although we may not evaluate $\bar{D}^\pm \bar{\Phi}^k$ directly, because they still may take positive and negative values, we can compute the positive results $\left(\bar{D}^\pm \bar{\Phi}^k\right)^+$ and $-\left(\bar{D}^\pm \bar{\Phi}^k\right)^-$, which are exactly those needed in the computation of the numerical flux g (cf. (4)). Again we set $\left(\bar{D}^\pm \bar{\Phi}^k\right)_\alpha := 0$ for all α enumerating border elements to ensure the natural boundary condition.

But the computation of g poses another difficulty related to the restricted precision. For small fixed-point values in $[0, 1]$ the computation of the Euclidean norm (4) is very erroneous. Therefore, we approximate the Euclidean norm by a convex combination of the 1-norm and the ∞-norm:

$$\begin{aligned} \bar{g}(\bar{U}, \bar{V}) &:= \bar{F}^+ \bullet \bar{N}(\bar{U}^+, \bar{V}^-) + \bar{F}^- \bullet \bar{N}(\bar{U}^-, \bar{V}^+) \\ \bar{N}(\bar{A}, \bar{B}) &:= \left(N(\bar{A}_\alpha, \bar{B}_\alpha) \right)_\alpha \\ N(\bar{A}_\alpha, \bar{B}_\alpha) &:= c(|\bar{A}_\alpha|_1 + |\bar{B}_\alpha|_1) \\ &\quad + (1-c)\max(|\bar{A}_\alpha|_\infty, |\bar{B}_\alpha|_\infty), \end{aligned}$$

where $c \in [0, 1]$, and '$\bullet$' denotes the component-wise multiplication of vectors.

The upwind level set scheme expressed in image operations now reads

$$\bar{\Phi}^{k+1} = \bar{\Phi}^k - \frac{\tau_k}{h} \cdot \bar{g}(\bar{D}^- \bar{\Phi}^k, \bar{D}^+ \bar{\Phi}^k)$$

and the CFL condition $\frac{\tau_k}{h} < \frac{1}{\max_\alpha |F_\alpha|}$ can be satisfied permanently by choosing $\frac{\tau_k}{h} \leq 1$, if we norm our normal velocity functions such that $|f| < 1$. Table 1 lists all the operations needed for the evaluation of the scheme together with their counterparts in graphics hardware.

4. IMPLEMENTATION

Graphics cards reassemble a computer. They comprise the graphics processor unit (GPU) responsible for the execution of commands and the graphics memory where the image operands are stored. Therefore the GPU processes data from the graphics memory very much like the CPU does from the main memory. Also where registers store intermediate data during calculations for the CPU, framebuffers serve the same purpose in numerical computations for the

Table 1. Basic operations in the upwind level set scheme.

operation	formula	graphics operation
addition	$\bar{V} + \bar{W}$	image blending
multiplication	$\bar{V} \bullet \bar{W}$	image blending
scalar factor	$a\bar{V}$	image blending
subtraction	$(\bar{V} - \bar{W})^+$	extended blending
maximum	$\max(\bar{V}, \bar{W})$	extended blending
index shift	$T_\gamma \bar{V}$	change of coordinates

GPU. The most significant difference is the unified processing of data blocks by the GPU. While the CPU needs to run over all elements of a grid to perform an addition of two element vectors, for example, the GPU takes only a few commands to add all pixels of two images representing these vectors. The unified processing of data blocks thus supersedes the command transfer over the memory bus and exploits the higher memory bandwidth of graphics cards by linear memory access. This makes graphics cards inherently faster in the processing of large amounts of data.

We shortly describe how the operations from Table 1 are performed in graphics hardware. Typically graphics cards support two framebuffers: the front buffer which is usually displayed on the screen and the back buffer where one can perform calculations invisibly. The calculations take place in the back buffer through the process of image blending. The first operand is displayed into the buffer. Then the setting of source and destination factors and the blending equation determine in which manner the following image will be combined with the source in the buffer. Copying the second operand into the buffer thus performs the desired operation. The result is either further processed by another operation or returned to the graphics memory. The last operation in Table 1, the index-shift, needed for the differential differences, is simply achieved by the change of the drawing position for the desired image. We may now outline the program in pseudo code notation.

level set segmentation {
  load the original image $\bar{P}$ and the initial function $\bar{\Phi}^0$;
  precalculate the normal velocities $\bar{F}^+$ and $\bar{F}^-$ from $\bar{P}$;
  initialize the graphics hardware with $\bar{F}^+$,$\bar{F}^-$ and $\bar{\Phi}^0$;
  for each timestep k {
    calculate the differential differences
      $\bar{U}^+ = (\bar{D}^-\bar{\Phi}^k)^+$ and $\bar{V}^- = (\bar{D}^+\bar{\Phi}^k)^-$;
    calculate the Euclidean norm approximation $\bar{N}(\bar{U}^+, \bar{V}^-)$;
    calculate the first flux addend $\bar{F}^+ \bullet \bar{N}(\bar{U}^+, \bar{V}^-)$;
    calculate the differential differences
      $\bar{U}^- = (\bar{D}^-\bar{\Phi}^k)^-$ and $\bar{V}^+ = (\bar{D}^+\bar{\Phi}^k)^+$;
    calculate the Euclidean norm approximation $\bar{N}(\bar{U}^-, \bar{V}^+)$;
    calculate the second flux addend $\bar{F}^- \bullet \bar{N}(\bar{U}^-, \bar{V}^+)$;
    update the level set func $\bar{\Phi}^{k+1} = \bar{\Phi}^k - \frac{\tau_k}{h} \cdot \bar{g}(\bar{U}, \bar{V})$;
  }
}

5. RESULTS

Apart from the precalculation of the speed function and the execution of the controlling program structure, all computations took place in the graphics system. For the normal velocity function we have used a parameterized polynomial function on the image intensities which takes positive and negative values.

Figure 1 shows the segmentation on a slice through the human brain computed on the ELSA Gladiac Ultra graphics card powered by NVIDIA's GeForce2 Ultra chip. The images closely reassemble software results. Moreover, the first two images of the sequence demonstrate that the allowance of negative values in the speed function enables the initially too large contours to withdraw from regions with unfitting intensities.

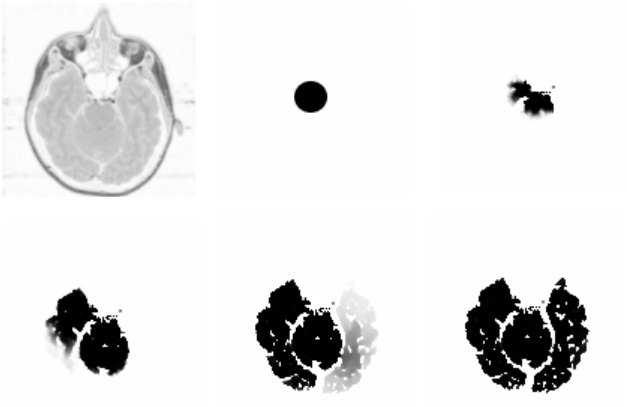

Fig. 1. Segmentation of a human brain computed in graphics hardware on a 128^2 image resolved by 8 bit. Besides the original image, the timesteps 0, 10, 50, 150 and 350 are depicted. The computation of one timestep took 2ms.

In Fig. 2 several differently colored seed points evolve independently to segment the pickets of a barbed wired fence. In this example all active contours use the same normal velocities, but in general the color components may evolve independently along different velocities while still being encoded in a single image. This example has been computed with the InfiniteReality2 graphics system of the SGI Onyx2 with 12 bit per color component and needed 30ms for one timestep on a 128^2 image.

The InfiniteReality2 system is significantly slower than the Gladiac Ultra because its operation of texture loading from the framebuffer, frequently needed in the algorithm, does not exploit the full bandwidth of the graphics memory. The higher precision of 12 bits would only account for a $\frac{3}{2}$ factor. To our experience 8 bits suffice in this application for the reproduction of the qualitative behavior, but higher precision may be desirable in the handling of curvature terms.

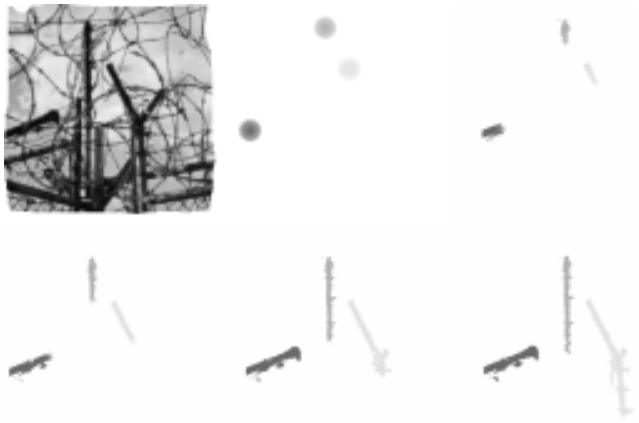

Fig. 2. Independent parallel segmentation of fence pickets in graphics hardware.

On the Gladiac Ultra texture loading from the framebuffer is much faster, but only when the texture and framebuffer formats match, so that we had to use color textures even when dealing with a single level set. Future graphics drivers will overcome this drawback gaining an additional factor of 3 to 4 in performance. The next generation of PC graphics cards will also include 3D texture support and more customizable operations on images, which will simplify the extension of our method to interactive 3D level set computations including internal forces.

6. CONCLUSIONS

We have demonstrated how the extended blending facilities of modern graphics cards can be used to rapidly evaluate the upwind level set scheme in graphics hardware. Since the blending equations together with the lookup table already offer most of the common arithmetic operations, this approach can be generalized to a wide range of numerical schemes, even implicit ones including linear equation solvers (cf. [13]).

We have explained how the use of graphics hardware in data transfer dominated problems can generally offer a superior performance than CPU based solutions, and for our schemes the computational results prove that the reduced number precision does not effect the qualitative outcome of the computations; and this should apply to many others. Furthermore, we have pointed out that the forthcoming generation of graphics cards and drivers promises further acceleration and an easy extension to 3D problems. Future research will therefore focus on this transition to 3D and more general models for the propagation speed.

The authors thank Matthias Hopf from Stuttgart and Michael Spielberg from Bonn for a lot of valuable information on graphics hardware programming.

7. REFERENCES

[1] F. Meyer and S. Beucher, "Morphological segmentation," *J. Vis. Commun. Image Represent.*, vol. 1, pp. 21–46, 1990.

[2] K. Haris, S. N. Efstratiadis, N. Maglaveras, and A. K. Katsaggelos, "Hybrid image segmentation using watersheds and fast region merging," *IEEE Transactions on Image Processing*, vol. 7, no. 12, 1998.

[3] J. M. Gauch, "Image segmentation and analysis via multiscale gradient watershed hierarchies," *IEEE Transactions on Image Processing*, vol. 8, no. 1, 1999.

[4] M. Bertalmio, G. Sapiro, and G. Randall, "Morphing active contours: A geometric approach to topology-independent image segmentation and tracking," in *Proc. IEEE-International Conference on Image Processing*, Chicago, October 1998.

[5] S. J. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton–Jacobi formulations," *J. of Comp. Physics*, vol. 79, pp. 12–49, 1988.

[6] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modelling with front propagation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, 1995.

[7] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, 1998.

[8] V. Caselles, F. Catté, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numer. Math.*, vol. 66, 1993.

[9] B. Engquist and S. Osher, "Stable and entropy-satisfying approximations for transonic flow calculations," *Math. Comp.*, vol. 34, no. 149, pp. 45–75, 1980.

[10] B. Cabral, N. Cam, and J. Foran, "Accelerated volume rendering and tomographic reconstruction using texture mapping hardware," in *ACM Symposium on Volume Visualization '94*, 1994, pp. 91–98.

[11] M. Hopf and T. Ertl, "Accelerating Morphological Analysis with Graphics Hardware," in *Workshop on Vision, Modelling, and Visualization VMV '00*, 2000, pp. 337–345.

[12] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 1999.

[13] M. Rumpf and R. Strzodka, "Using graphics cards for quantized FEM computations," in *Proceedings VIIP'01*, 2001, pp. 193–202.