Interactive Level-Set Smoothing for Photo Editing

Thomas C. Howard
tom.larisa@gmail.com

Bryan S. Morse
morse@byu.edu

INTERACTIVE LEVEL-SET SMOOTHING FOR PHOTO EDITING

Thomas C. Howard and Bryan S. Morse

Brigham Young University, Department of Computer Science
3361 TMCB, Provo, UT 84602

ABSTRACT

This paper presents an interactive image-smoothing tool based on properties and manipulation of image level sets. This tool uses PDE-based level-set smoothing to preserve edge sharpness while smoothing noise and jagged contours. Unlike existing approaches using PDEs, the duration and areas of application are controlled interactively with immediate feedback to the user. Interaction issues are addressed, and parameters for adjusting the PDE are automatically estimated based on image characteristics.

1. INTRODUCTION

In this paper, we present a tool for interactive edge-preserving image smoothing based on level-set smoothing techniques [1, 2, 3] with the following contributions:

- Interactive user control of the target area and the amount of smoothing performed,
- Automatic parameter estimation based on local image characteristics (no trial-and-error tweaking of the smoothing parameters), and
- Time-varying adjustment of smoothing parameters based on the duration of the user's application of the tool.

This combines the power of level-set techniques for nonlinear smoothing with a user's ability to direct and control the application of the locally-applied PDE.

2. BACKGROUND

Level-set methods have proven to be a powerful tool for image processing [1, 2, 3, and many others]. We begin by reviewing the basics of these methods, especially level-set smoothing.

2.1. Level-Set Properties

A *level set* L_k is the set of all points with the same value k:

$$L_k = \{\overline{x} : I(\overline{x}) = k\}$$

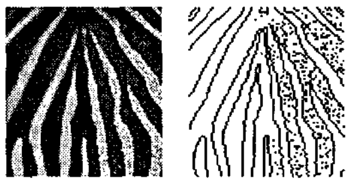

Fig. 1. Interactive level-set smoothing applied to one side of an image (zebra stripes). The noisy image [left] has been smoothed only on the left side using the interactive tool. The corresponding contours [right] show that the contours in the area where the tool was applied have been smoothed (or perhaps contracted entirely).

The image on the right side of Figure 1 shows an example of one particular level set for an image. Although a particular value may appear only infrequently (if at all) in an image, by assuming continuity of the image function we may infer the existence of intermediate values. Thus, level sets are continuous and form closed curves in the image plane.

In this way, *we can think of the image not as an array of pixels but as a topographic map.* Properties of the local contour can be used to drive or control other operations. Moreover, we can directly manipulate the contours themselves—smooth them, move them as desired, etc.

By again assuming a continuous, differentiable image, geometric properties of level curves can be calculated using differential geometry and do not require explicit extraction, interpolation, or representation of the level curve. For example, the level-set curvature κ can be computed directly from local first- and second-order derivatives (I_x, I_y, I_{xx}, I_{yy}, and I_{xy}) as follows:

$$\kappa = \operatorname{div}\frac{\nabla I}{\|\nabla I\|} = -\frac{I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy}}{I_{xx}^2 + I_{yy}^2} \tag{1}$$

2.2. Level-Set Manipulation

Similarly, level curves in an image can be moved without explicit representation. The link between movement of level curves and the underlying pixel representation is given by the following partial differential equation:

$$I_t = -F\|\nabla I\| \tag{2}$$

where I_t denotes the (instantaneous) change in pixel intensity and F the velocity of the curve in the direction of its normal [4, 1, 2].

2.3. Level-Set Smoothing

One can smooth level curves [1, 2] by defining the velocity F of the curve in the direction of its normal to be proportional to the negative of the level-set contour's curvature κ:

$$F = -\epsilon\kappa \tag{3}$$

By moving at a speed proportional to their curvature, level curves with higher curvature contract faster than smoother curves. This first smooths the curves, then causes them to contract, then eventually removing them. Noise, jagged-edges, and other small-scale artifacts are removed while preserving edge smoothness.

3. INTERACTIVE SMOOTHING TOOL

Although level-set smoothing is a powerful technique for nonlinear filtering of noise, jagged contours, and other image defects, the steady-state solution to this PDE is a constant image. If allowed it to run unrestrained, eventually each level curve devolves to a convex shape, then to a circle, then to a single point, then to nothing

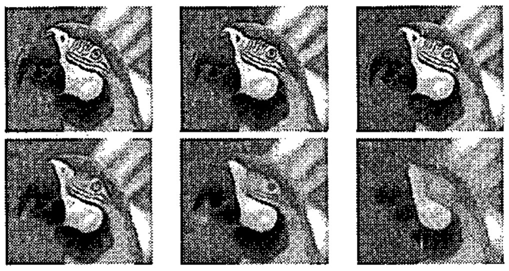

Fig. 2. Global level-set smoothing. Top-to-bottom, left-to-right: initial image; 1, 2, 16, 64, and 256 iterations.

(Figure 2). What initially provides desirable enhancement eventually produces an overly-smoothed, cartoonish image. This raises the following questions for this and similar PDE-based smoothing methods (e.g., anisotropic diffusion [5]):

- When do you stop? (How many iterations should be performed?)
- Should the smoothing occur everywhere at the same speed? If not, how do you determine how fast to smooth locally?
- Should the smoothing occur for the same duration everywhere? If not, how do you determine the appropriate local duration?

These questions have been addressed by previous researchers [6, 3, 7], though not yet answered fully.

Interactive user control provides both selective application and a stopping criterion. However, even for interative control, the other parameters must be determined. To provide true "point and apply" interaction, these other parameters must be automatically estimated without forcing the user to manually tweak them.

3.1. Speed and Responsiveness

Our implementation is based on Euler's method:

$$I(t+\Delta t) \approx I(t) + \Delta t\, I_t(t) \tag{4}$$

The time-step parameter Δt controls the speed of the movement on each iteration and must be chosen to maintain stability of the numerical implementation [2, 7].

Besides stability issues, the speed of the implementation is also crucial for effective user interaction. If the speed is too small, the tool is slow and unresponsive. If the speed is too high, the tool responds too quickly and becomes difficult to control. Substituting Eq. 3 into Eq. 4, we get

$$I(t+\Delta t) \approx I(t) - \Delta t\, \epsilon\, \kappa \tag{5}$$

where again ϵ is the proportionality between between the level-curve motion and the level-set curvature, and κ is the calculated level-set curvature at each point. Adjusting ϵ such that $0 \leq \epsilon \leq 1$ maintains stability while providing a control for interactive response. Furthermore, ϵ need not be uniform across the image.

Empirically, we found the tool to be most effective in the range $0.10 < \epsilon < 0.55$ when run on a 1 GHz Pentium 4 processor. (This range would obviously have to be tuned to various processor speeds.) Above 0.55, the method becomes too rapid to control well. Below 0.10, the tool is too slow and unresponsive.

One could obviously use a single responsiveness parameter ϵ across the entire image and tune this to the performance of the particular processor or to user preferences, but could we not automatically tune this parameter within this empirical range for each target image or even each neighborhood within each image?

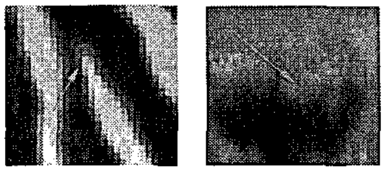

Fig. 3. Two sections of the zebra image ($\beta = 68.0$). The high-detail region (left, $\delta = 123.9$) should not be oversmoothed. The lower-detail region (right, $\delta = 56.3$) may be smoothed somewhat more aggressively.

3.2. Noise vs. Detail Estimation

The key to estimating the best speed parameter ϵ for each pixel is to recognize that we want to reduce noise and jagged contours while preserving as much as possible natural corners in the image. This is a classic trade-off, which we approach by using the differences between each pixel and its respective neighborhood: if a pixel is very different from all its neighbors (likely noise), we might want to smooth it to be more like its neighbors; if a pixel is very similar to some of its neighbors, yet very different from others (likely detail), we might want to be less aggressive in our smoothing.

We use a difference metric δ defined as the average difference between a pixel and its neighbors:

$$\delta = \frac{1}{8} \sum_{i=-1}^{1} \sum_{j=-1}^{1} |I(x,y) - I(x+i, y+j)|$$

The mean of δ over the entire image (denoted as μ_δ) gives a good indication of the type of image being processed. From anectodal data, $\mu_\delta < 30$ suggests a low-detail image, $30 < \mu_\delta < 120$ denotes a normal image, and $\mu_\delta > 120$ suggests a noisy image.

Moreover, μ_δ establishes a baseline from which to determine the amount of detail at each pixel: if a pixel's individual δ value is greater than the average (μ_δ), we might suspect local detail and smooth less aggressively; if a pixel's individual δ value is less than the average μ_δ, we might suspect less detail and smooth accordingly (Figure 3).

However, we might want to increase this baseline for extremely low-detail images in order to preserve greater detail. We define a baseline parameter $\beta = \mu_\delta + \frac{1}{1+e^{(\mu_\delta - 30)\lambda}} \sigma_\delta$ where σ_δ is the standard deviation of δ across the image. Notice that for low-detail images ($\mu_\delta < 30$), β approaches $\mu_\delta + \sigma_\delta$. For normal images, β is much nearer to μ_δ. The parameter λ controls the sharpness of this sigmoidal transition.

We can now automatically tune the responsiveness parameter ϵ for each pixel by comparing to this adjusted threshold β:

$$\epsilon = \begin{cases} 0.1 + \frac{|\delta - \mu_\delta|}{\mu_\delta} & \text{if } \delta < \beta \\ 0.1 & \text{otherwise} \end{cases}$$

3.3. Reducing the Responsiveness During Application

With interactive tools one generally likes to get a lot of result with little effort. Continued application generally means that the results need to be fine-tuned. For this reason, the interactive level-set smoothing tool gradually slows down the longer the user applies the tool to the area. (Of course, they can begin aggressive smoothing again by releasing and again pressing the mouse button.) Rather than using a global reduction in speed, we maintain a buffer of the ϵ values for each pixel. These values are initialized as described in the preceding section but are gradually reduced as each iteration is applied to the area under the cursor. As the user moves the curser to new regions, the smoothing is more aggressive. As they hold the curser over a region or return to a previously-visited region, the smoothing is more gradual (fine tuning). The reduction schedule used is $\Delta\epsilon = -\delta\mu_\delta c$ where c is a small constant. (We have found that 5×10^{-6} works well.)

3.4. Implementation

To determine the initial responsiveness of the smoothing tool based on our noise vs. detail estimates, we precompute the per-pixel differences δ, the mean difference μ_δ, the difference baseline β, and the per-pixel initial responsiveness ϵ. The per-pixel initial values of ϵ are then stored in a separate buffer. These initial values are then copied to another buffer which will then be updated (gradually reduced) during the smoothing process.

To avoid abrupt transitions at the cursor boundary, we use a weighting function w centered around the cursor position that is equal to 0 outside the cursor area, 1 inside an area just smaller than the cursor area, and transitions smoothly between 1 and 0 as one approaches the cursor boundary. Eq. 5 thus becomes

$$I(t + \Delta t) \approx I(t) - \Delta t\, \epsilon\, w\, \kappa \qquad (6)$$

As the image tool is applied (the mouse button is held down), the following operations happen for each pixel within the (user-selectable) cursor area:

1. Calculate derivatives up to second order using 3x3 masks.
2. Calculate the level-set curvature κ using Eq. 1.
3. Calculate the new pixel values using curve-shortening smoothing according to Eq. 6.
4. Write the new pixel values back to the image buffer.
5. Reduce the responsiveness values ϵ according to the reduction schedule until a minimum responsiveness of 0.1 is reached.

This process is repeated until the user releases the mouse button, at which time the responsiveness values are reset.

4. RESULTS

The effects of the interactive level-set smoothing tool may be seen in Figures 4–6.[1] Figure 4 was magnified using bicubic interpolation and suffers from the accompanying overshoot/undershoot ringing. The central portion of the figure has been cleaned up using the interactive level-set smoothing tool and only a few strokes of the cursor. Several areas in Figure 5 were corrupted in various ways (noise, jagged edges, etc.) then corrected using the smoothing tool in the areas marked on the figure. Figure 8 demonstrates the smoothing tool applied to a low-quality JPEG-compressed image. The original image suffers from blocking artifacts typical for such compression. These artifacts are removed while still preserving the sharpness of the image. Figure 6 shows how the tool may be used in one part of the image to remove artifacts (in this case, wrinkles) while leaving other parts of the image unchanged.

Although current commercial programs provide interactive blurring tools, which are useful for some tasks, such tools often not only eliminate the intended artifacts but blur desirable image content (Figure 7). The contour-smoothing and edge-preserving nature of the interactive level-set smoothing tool preserves sharpness by drawing on level-set smoothing.

[1]These figures were processed in color, but due to publishing limitations are reproduced here in greyscale. A version of this paper with the original color images is available from the second author's website.

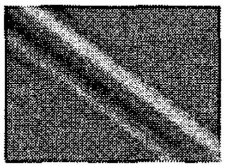
Fig. 4. Interactive level-set smoothing tool applied locally (central portion of the image) to an image to remove jagged contour and ringing overshoot/undershoot effects of bicubic interpolation

5. CONCLUSION

We have demonstrated here an interactive smoothing tool based on a locally-applied, level-curve-shortening PDE. As with existing level-set smoothing techniques, this tool preserves sharp edges while smoothing noise and level contours. The interactive nature of the tool allows the user to control the area and duration of application. A noise-estimation model attempts to separate noise from detail and adjusts the responsiveness (aggressiveness) of the tool accordingly. Gradual reduction of this responsiveness provides a stronger effect initially and a more careful, fine-tuning effect as the user continues to apply the tool to the same area.

The level-set smoothing PDE used here is admittedly among the oldest and simplest of such PDE-based approaches to image enhancement, and many others have since been developed. One could perhaps instead use Beltrami flow [8] to better couple the color channels during smoothing, or one could use other effects besides smoothing (sharpening, reaction-diffusion, etc.). (See [7] for an excellent survey.) We believe that many of these other PDE-based approaches might also be successfully incorporated into locally-applied interactive tools.

6. REFERENCES

[1] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel, "Image selective smoothing and edge detection by nonlinear diffusion," *SIAM J. of Numerical Analysis*, vol. 29, pp. 845–866, 1992.

[2] James A. Sethian, *Level Set Methods*, Cambridge University Press, 1996.

[3] Ravikanth Malladi and James A. Sethian, "Image processing:flows under min/max curvature and mean curvature," *Graphical Models and Image Processing*, vol. 58, no. 2, pp. 127–141, 1996.

[4] S. Osher and J. A. Sethian, "Fronts propogating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation," *J. Comput. Phys.*, vol. 79, pp. 12–49, 1988.

[5] Pietro Perona and Jalhandra Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629–639, 1990.

[6] Ross T. Whitaker and Stephen M. Pizer, "A multiscale approach to nonuniform diffusion," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 57, no. 1, pp. 99–110, Jan. 1993.

[7] Gillermo Sapiro, *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, 2001.

[8] Ron Kimmel, Ravi Malladi, and Nir Sochen, "Images as embedded maps and minimal surfaces: Movies, color, and volumetric medical images," in *Computer Vision and Pattern Recognition*, 1997.

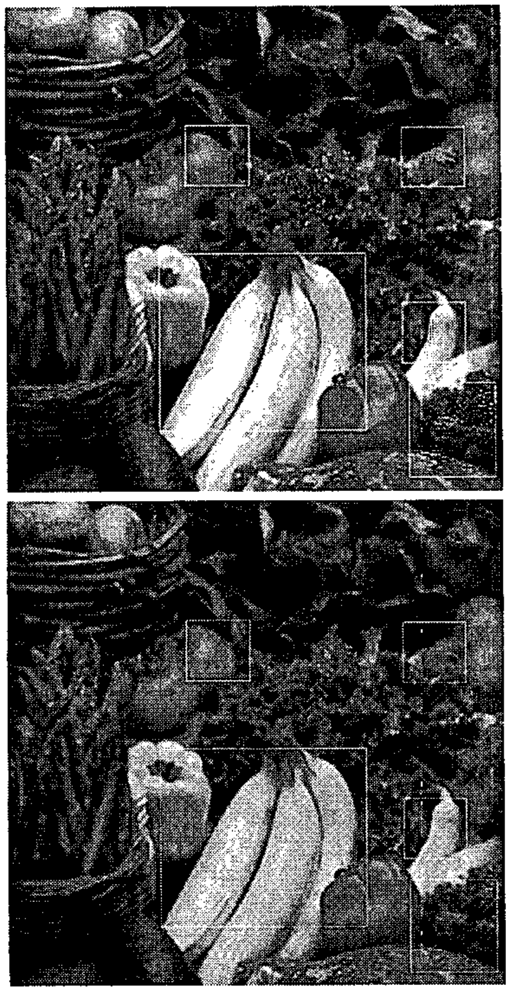

Fig. 5. Interactive level-set smoothing tool applied to an image with noise and jagged-edge corruption . The areas where the tool was applied are marked in both the before [top] and after [bottom] images.

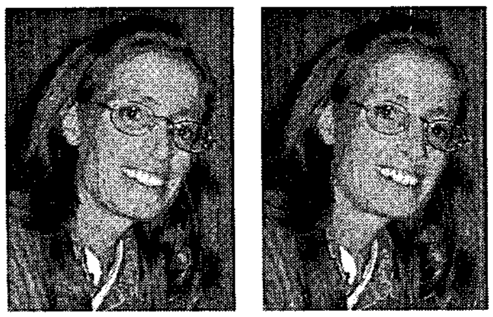

Fig. 6. The local level-set smoothing tool smooths wrinkles [left] while preserving sharpness of other facial features and leaving the rest of the image untouched [right]

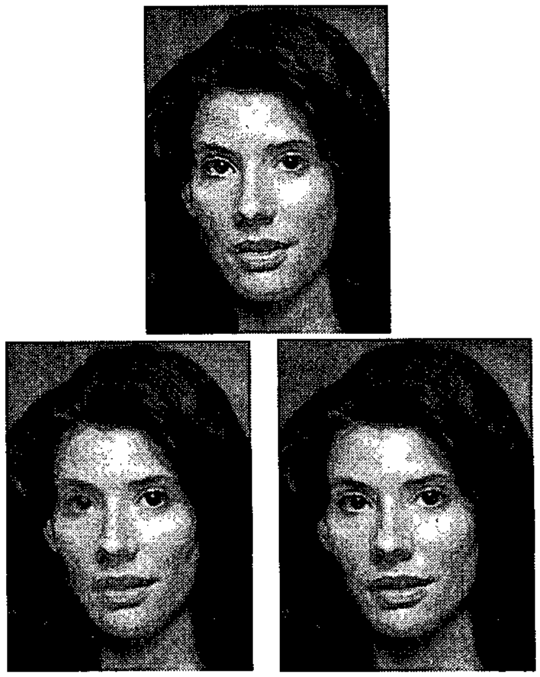

Fig. 7. The left side of the face in the image was corrupted using oversharpening to enhance the noise [top]. The effects were removed using both Photoshop's interactive blurring tool [bottom left] and the interactive level-set smoothing tool [bottom right].

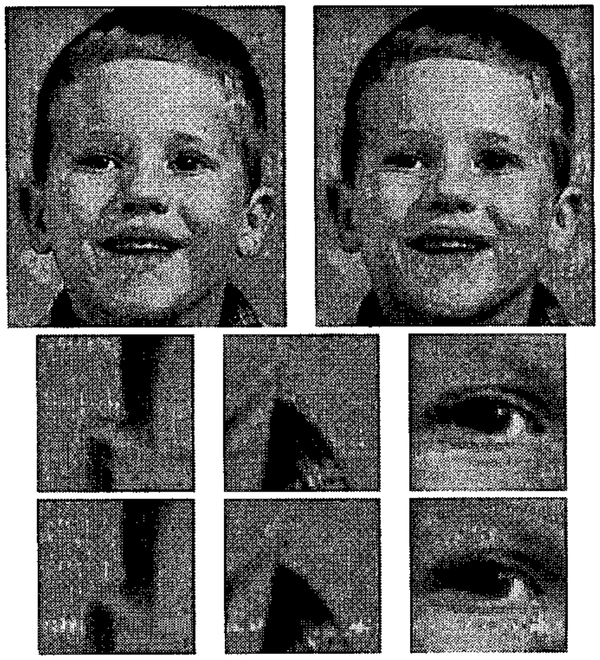

Fig. 8. Low-quality JPEG-compressed image with typical blocking artifacts [top left, middle row] smoothed using the interactive level-set smoothing tool [top right, bottom row]