Pacific Graphics 2008
T. Igarashi, N. Max, and F. Sillion
(Guest Editors)

Volume 27 (2008), Number 7

Shape-simplifying Image Abstraction

Henry Kang[†1] and Seungyong Lee[‡2]

[1]University of Missouri, St. Louis, USA
[2]POSTECH, Korea

Abstract

This paper presents a simple algorithm for producing stylistic abstraction of a photograph. Based on mean curvature flow in conjunction with shock filter, our method simplifies both shapes and colors simultaneously while preserving important features. In particular, we develop a constrained mean curvature flow, which outperforms the original mean curvature flow in conveying the directionality of features and shape boundaries. The proposed algorithm is iterative and incremental, and therefore the level of abstraction is intuitively controlled. Optionally, simple user masking can be incorporated into the algorithm to selectively control the abstraction speed and to protect particular regions. Experimental results show that our method effectively produces highly abstract yet feature-preserving illustrations from photographs.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Picture/Image Generation]: Display algorithms

1. Introduction

Image abstraction and stylization has been an active research topic in non-photorealistic rendering. It refers to a task of simplifying scene information in the image while retaining or emphasizing meaningful features to convey. Such reduction and clarification of visual information must be done in a stylistic fashion, so that the output of abstraction can serve the purpose of communicating the messages as well as pleasing the viewer's eyes. Herman and Duke [HD01] used the term "minimal graphics" to describe the benefit of such meaningful abstraction and stylization.

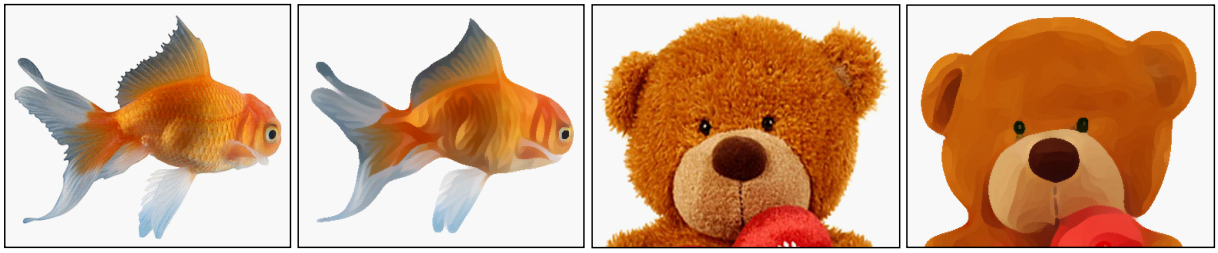
The central problem in image abstraction is how to decrease the complexity of the scene while protecting important structures. Conventional approaches often involve low-level vision techniques such as image segmentation, color quantization, or feature-preserving smoothing. They mainly focus on clustering/smoothing pixels within homogeneous regions and therefore simplifying the 'colors' that comprise the image, without hurting the boundaries between heterogeneous regions. However, such techniques in general do not simplify the region boundaries (i.e., 'shapes') themselves, and thus may require an additional process of curve fitting and editing for further abstraction. Also, the aggressive smoothing/merging of pixels often results in loss of important information, such as the directionality of features.

In this paper, we present a novel image abstraction algorithm that simplifies both shapes and colors in an integral fashion and therefore does not require any post-processing. It is based on constrained mean curvature flow in conjunction with shock filtering. The constrained mean curvature flow effectively protects and conveys directional characteristics of shapes, features, and textures. Also, since the curvature flow is iterative as well as incremental, one can intuitively control the level of abstraction; just stop the process when the desired level is reached. The proposed algorithm is straightforward and easy to implement. Fig. 1 shows some example abstraction results produced by our method.

2. Related work

Existing approaches for image abstraction often rely on image segmentation. DeCarlo and Santella [DS02] presented a system for abstracting and stylizing photographs, based on mean shift color image segmentation [CM02]. Collomosse et al. [CRH05] extended the mean shift segmentation to video, producing a temporally coherent cartoon-like image sequence. Wang et al. [WXSC04] used anisotropic mean

† kang@cs.umsl.edu
‡ leesy@postech.ac.kr



(a) Input (b) Output (c) Input (d) Output

Figure 1: *Image abstraction by our method*

shift filter for handling elongated structures often found along the temporal axis of a video. Wen et al. [WLL*06] also used the mean shift segmentation algorithm for generating a colored sketch from a photograph. Lecot and Levy [LL06] developed a triangle-based image segmentation algorithm for abstract and stylistic bitmap-to-vector image conversion.

Image segmentation is a natural choice of tool for the task of image abstraction as it aggressively reduces the complexity of the scene while protecting important structures. However, a crude segmentation often results in a rather incomplete abstraction as the segmented regions still retain rough and complex boundaries. Therefore, the region boundaries may have to go through some post-processing, which typically includes curve fitting, editing, smoothing, and stylizing [DS02, WLL*06, LL06, WXSC04]. In this paper, we present an abstraction algorithm that directly simplifies shapes and does not require any post-processing.

There are other abstraction approaches that are not based on image segmentation. Winnemöller et al. [WOG06] showed that bilateral filter [TM98] can be used (together with color quantization) to abstract color images as well as video. Orzan et al. [OBBT07] developed a multi-scale image abstraction system based on gradient reconstruction. These feature-preserving filters reduce the scene complexity by smoothing insignificant color variations while protecting important region boundaries. They do not, however, directly simplify those region boundaries (i.e., shapes), whereas our method enables simultaneous simplification of the region boundaries and the colors within.

3. Overview

To provide a simple and integrated scheme for shape/color simplification, we base our algorithm on *mean curvature flow* (a.k.a. mean curvature motion, geometric heat flow) [Gra86]. The mean curvature flow regularizes the given geometry at a speed proportional to the local mean curvature value. It is known as the quickest way to decrease the complexity of a geometric structure, and has proven to be a useful tool for removing noise from images, curves, or surfaces [ALM92, MSV95, DMSB99]. In this paper, we use it for the task of image abstraction and stylization.

The application of mean curvature flow alone, however, is problematic as the regularization process shrinks and blurs every shape boundary in the image, and eventually leaves no meaningful structures or edges in it. To prevent this, we interleave mean curvature flow with an edge enhancement step using *shock filter* [OR90]. By adjusting the frequency of inserting this shock filtering step, one can control the sharpness of edges.

Another limitation of mean curvature motion is that it does not properly protect the directionality of features or texture patterns. We thus propose an improved abstraction algorithm based on *constrained mean curvature flow*, which uses a feature direction field as a constraint. We will show that such constrained motion not only provides better protection of features, but also contributes to the stylistic look of the output illustration.

Our approach is based on an iterative image evolution process, and thus makes it easy to control the degree of abstraction as the user may simply stop the algorithm when the desired level is reached. Optionally, we allow for a selective user masking of important features to protect. Such user input can be easily incorporated into our algorithm, prompting the regularization speed of the masked area to slow down.

4. Methodology

4.1. Mean curvature flow

Suppose we are given an input image $I(\mathbf{x})$, where $\mathbf{x} = (x, y)$ denotes pixel location. If we view $I(\mathbf{x})$ as a height field, where height represents intensity (or luminance), we should be able to draw the iso-luminance contours (of the same height) on the image. These contours are often called *isophote curves* or *level sets*. Mean curvature flow (MCF), when applied on such a height field, regularizes the geometry of each of these isophote curves. The speed of regularization is proportional to the local isophote curvature. That is, a high-curvature portion of the curve is smoothed faster. The overall shape of the curve is then simplified and shrunk,

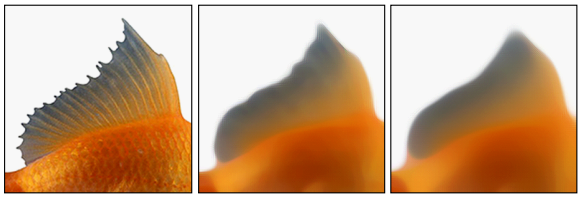
and eventually collapses to a circle. The evolution equation under mean curvature flow is defined as:

$$I_t = \kappa |\nabla I| \tag{1}$$

where I_t is the derivative of image I with respect to time t, and $\nabla I = (I_x, I_y)$ is an image gradient. κ denotes local isophote curvature, and can be computed from the image [ALM92]:

$$\kappa = \frac{I_x^2 I_{yy} - 2 I_x I_y I_{xy} + I_y^2 I_{xx}}{(I_x^2 + I_y^2)^{3/2}} \tag{2}$$

In case of a color image, RGB channels are processed separately. As the mean curvature flow regularizes all isophote curves, it simplifies every region boundary present in the image. Also, Eq. 1 indicates that an isophote curve is smoothed via the adjustment of local intensities (heights) along the curve. Specifically, the intensity increases where $\kappa > 0$ (convex), and decreases where $\kappa < 0$ (concave). This results in the simultaneous abstraction of both shapes (region boundaries) and colors (intensities). Fig. 2 shows how mean curvature flow simplifies an example image.



(a) Input (b) MCF (c) MCF

Figure 2: *Mean curvature flow: (b) 20 iterations (c) 60 iterations*

4.2. Shock filtering

The mean curvature flow aggressively contracts isophote curves by blurring pixel intensities along edge direction. Iteration of such shrinking and blurring eventually obliterates most of the height discontinuities (edges) in the image, some of which may be of interest to the viewers. For better protection of edges, we regularly perform edge enhancement by shock filtering.

Shock filter [OR90] is originally developed as a data deblurring tool, and its evolution equation is formulated as:

$$I_t = -sign(\triangle I) |\nabla I| \tag{3}$$

where $\triangle I = I_{xx} + I_{yy}$ denotes Laplacian, the second spatial derivative of the image. Eq. 3 is known to evolve image I in the following way. At each time step, I increases (or 'dilates') in *maximum influence zone*, where $\triangle I < 0$. On the other hand, I decreases (or 'erodes') in *minimum influence zone*, where $\triangle I > 0$. As a result, the evolution sharpens the edges at the zero-crossings of $\triangle I$, while in each influence zone the dilation/erosion process gradually reduces $|\nabla I|$, and in time, I becomes piecewise constant. Overall, shock filter sharpens the discontinuities between heterogeneous regions, whereas it flattens each homogeneous region.

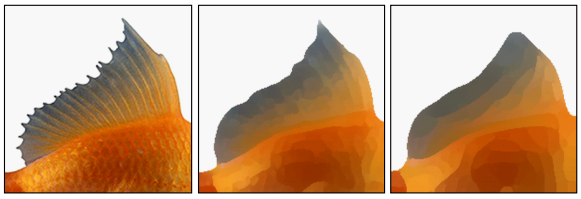
To determine the influence zone, we employ Laplacian-of-Gaussian (LoG) function, denoted $\triangle G_\sigma$, where G_σ is a bivariate Gaussian function of standard deviation σ. This is used to reduce the sensitivity to image noise and also to help control the size of the influence zone (and thus the style of abstraction). Our modified evolution equation for shock filtering thus reads:

$$I_t = -sign(\triangle G_\sigma * I) |\nabla I| \tag{4}$$

The convolution with LoG can be implemented by applying Gaussian blur before Laplacian operation. The dilation process is implemented by max filter in a 3×3 neighborhood (and the erosion by min filter). For a color image, we use the sum of RGB components to determine the max and min values. In practice, we perform only a single step of shock filtering (Eq. 4) after every k-th iteration of mean curvature flow (see Algorithm 1). Fig. 3 shows the effect of shock filtering, i.e., edge sharpening and region flattening.

Algorithm 1 Image Abstraction by MCF

```
loop
  for 1 to k do
    I ← MeanCurvatureFlow(I)
  end for
  I ← ShockFiltering(I)
end loop
```



(a) Input (b) MCF + Shock (c) MCF + Shock

Figure 3: *Algorithm 1: (b) 20 iterations (c) 60 iterations. The iteration number means how many times the system runs mean curvature flow. We set $k = 10$ in this example. Therefore, shock filter is called 2 times for (b) and 6 times for (c).*

4.3. Constrained mean curvature flow

From the standpoint of stylistic image abstraction, the mean curvature flow may be a bit too aggressive as it quickly dismantles meaningful structures, along with their features. While the use of shock filter slows down the fading of edges, it does not effectively protect directional features or texture patterns. To address this, we extract a smooth vector field that describes local feature directions, and use it as a constraint for the mean curvature flow.

We employ a nonlinear vector smoothing method in [KLC07] to construct the direction field from I: First get the gradient vectors via Sobel operator and rotate them counter-clockwise by $\pi/2$. The resulting vectors are aligned with tangent directions of edge curves, and thus called *tangent vectors*. These vectors are then bilateral-filtered to form a smooth and feature-preserving direction field, which we call tangent vector field (TVF) (see Fig. 4d). Let $\mathbf{t}(\mathbf{x})$ denote the smoothed tangent vector, which is interpreted as the 'desired' feature direction at $\mathbf{x}$.

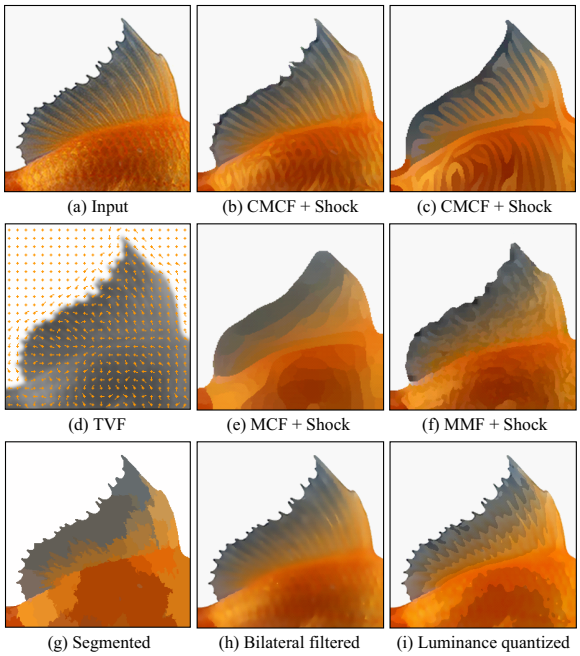


Figure 4: *Algorithm 2: (b) 20 iterations (c) 60 iterations. The iteration number means how many times the system runs constrained mean curvature flow. We set $k = 10$ in this example.*

Given the direction field $\mathbf{t}(\mathbf{x})$, the idea is to constrain the speed of isophote regularization as follows:

$$I_t = s \cdot \kappa ||\nabla I|| \tag{5}$$

The speed control function s in Eq. 5 is defined as:

$$s(\mathbf{x}) = (1-r) + r \cdot |\mathbf{t}(\mathbf{x}) \cdot \nabla I(\mathbf{x})^{\perp}| \tag{6}$$

where $\nabla I^{\perp}$ denotes the vector perpendicular to the current local gradient (both ∇I and $\mathbf{t}$ are assumed to be unit vectors). r is a control parameter between $[0,1]$ (default value $r = 1$). If $\nabla I^{\perp}$ is not aligned with $\mathbf{t}$, small s will result. That is, it discourages the regularization along anything but the (desired) feature directions, and therefore has the effect of protecting the directionality of features, shapes, and texture patterns (see Fig. 4b and c). Moreover, as it directs the diffusion of colors along feature direction, it adds to the stylistic look of the output. Algorithm 2 describes the modified image abstraction algorithm based on the constrained mean curvature flow (CMCF). Note if we set $r = 0$, Algorithm 2 is identical to Algorithm 1.

Fig. 4e shows the abstraction result via Algotirhm 1 (mean curvature flow + shock filtering) for comparison. It should also be noted that this constrained mean curvature flow differs from min/max curvature flow (MMF) [MS96], which is another curvature-driven denoising process. In the context of image abstraction and stylization, min/max flow does not preserve the feature directionality or edge smoothness as well as our constrained mean curvature flow (see Fig. 4f).

Algorithm 2 Image Abstraction by CMCF

```
loop
  for 1 to k do
    t ← TVF(I)
    I ← ConstrainedMeanCurvatureFlow(I, t)
  end for
  I ← ShockFiltering(I)
end loop
```

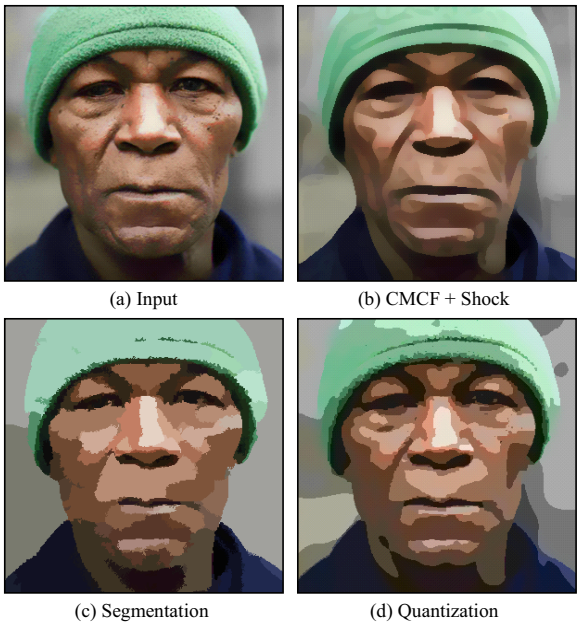


Figure 5: *Comparison with other techniques: (a) Input (courtesy paulbence [Fli]) (b) Algorithm 2 (50 iterations of CMCF, k = 10) (c) Mean shift segmentation (d) Bilateral filtering + Luminance quantization*

Fig. 4g, h, and i show how other abstraction techniques process the same input image. Fig. 4g is obtained by mean shift segmentation procedure. Note that it aggressively simplifies the image regions (and colors), but not the region boundaries. Also, important feature directionality is lost

(a) Input (b) 20 iterations (c) 40 iterations (d) 60 iterations

Figure 6: *Controlling the level of abstraction. The iteration number means how many times the system runs constrained mean curvature flow. We set $k = 10$ in this example.*

in the process. Fig. 4h is the output of bilateral filtering, in which the shape boundaries are similarly untouched as the filter removes insignificant height discontinuities only. Fig. 4i is a further simplification of Fig. 4h via luminance quantization (as suggest in [WOG06]). Although it is possible to further simplify the individual region boundaries of Fig. 4g and i, it may involve rather cumbersome procedures, such as low-resolution curve fitting of each boundary, eliminating tiny regions, possible hole filling after smoothing or removing of each region, protecting junction points as well as their geometric continuities, and coloring of interior regions that are defined by a network of curves.

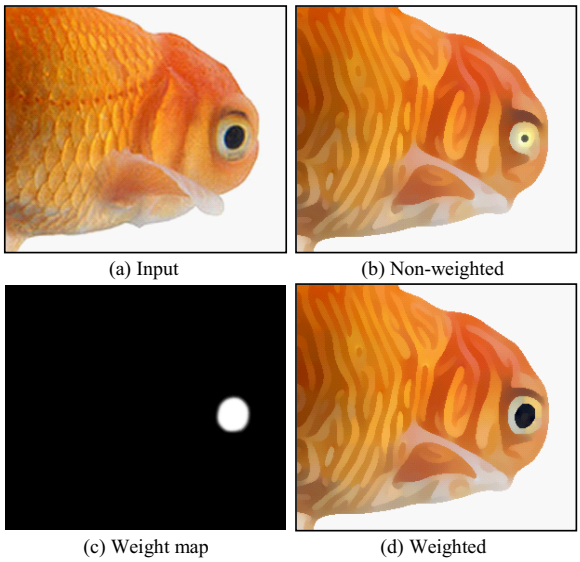
Fig. 5b is another abstraction result by our method, shown in comparison with conventional approaches (Fig. 5c and d). In this example, our method performs well not only in terms of shape simplification, but also in conveying important features. In our result (Fig. 5b), features like the seam on the hat, thick eyebrows, wrinkles under both eyes, cheek bones, structure of mouth and chin, are all well preserved and conveyed.

Fig. 6 shows the iterative nature of our algorithm and how it helps to control the level of abstraction. The system displays the continuous evolution of the image, and lets the user decide when to stop the abstraction.

4.4. Incorporating user input

Sometimes the user may want to protect some particular area in the image by slowing down the regularization speed there. We enable this by allowing interactive area masking (such as in [Her01]). The user may 'paint' a particular area covering an important feature to protect. We implement painting as plotting of circles along the path of a mouse cursor.

Let $w(\mathbf{x})$ denote a 'weight map' in which the user-painted pixels are marked with nonzero values. Before painting, $w(\mathbf{x})$ is initialized to zero. We emulate a smooth-brush effect, by using a circle whose interior is weighted to form a



(a) Input (b) Non-weighted (c) Weight map (d) Weighted

Figure 7: *Incorporating user input*

Gaussian-like volume (that is, the center of the circle has the biggest weight, which is 1.0). Adjacent circles on the path may overlap and the weights are summed in such area. The total weight at each pixel, denoted $w(\mathbf{x})$, is clipped to $[0, 1]$.

To slow down the regularization in a painted region, we modify the speed $s(\mathbf{x})$ if $\mathbf{x}$ is associated with a nonzero weight, that is, $w(\mathbf{x}) > 0$.

$$s(\mathbf{x}) = \alpha + (1-\alpha)\cdot[1 - w(\mathbf{x})], \quad \text{if } w(\mathbf{x}) > 0 \qquad (7)$$

where we set $\alpha = 0.1$ by default. Fig. 7 shows how a simple user input makes a difference in feature protection. As shown in this figure, it is particularly useful for conserving circular features as the curvature motion quickly shrinks any circular shapes. It may also be used to protect sharp corners.

5. Results

We use Algorithm 2 for producing abstraction results. Eq. 5 and 6 indicate that one can use parameter r to control the strength of constraint in Algorithm 2. In particular, Algorithm 2 degenerates to Algorithm 1 when $r = 0$. Fig. 8 shows different results obtained by adjusting parameters, such as r (strength of constraint), k (shock filtering frequency), σ (kernel size for shock filtering). The eye of the bird has been masked for protection. Note that large r conveys directional features better, small k emphasizes fine details, and big σ expands the influence zones.

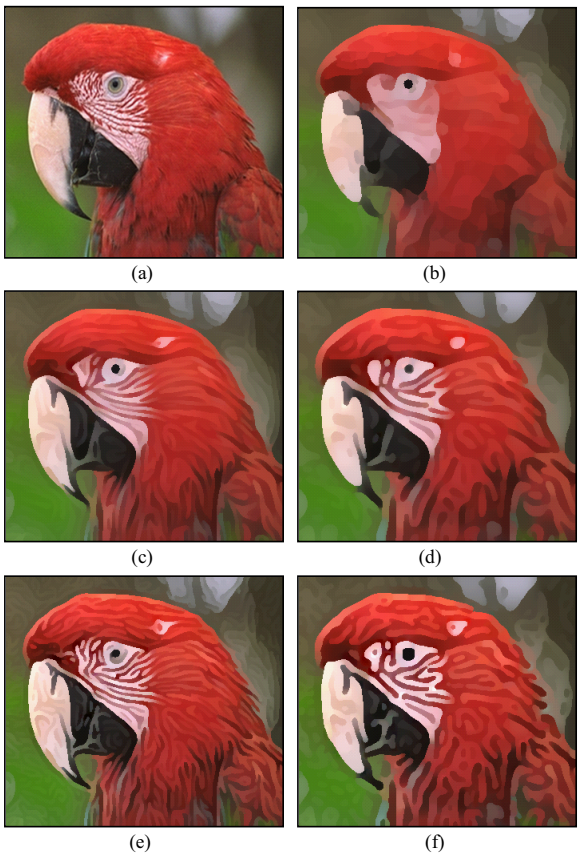


Figure 8: *Parameter control: (a) Input (b)* $r = 0, k = 10, \sigma = 1.0$ *(c)* $r = 1, k = 10, \sigma = 1.0$ *(d)* $r = 1, k = 10, \sigma = 2.0$ *(e)* $r = 1, k = 5, \sigma = 1.0$ *(f)* $r = 1, k = 5, \sigma = 2.0$

Fig. 9 shows some additional abstraction results obtained by Algorithm 2. The following parameters are used: $r \in [0.5, 1.0], k \in [8, 10], \sigma \in [1.0, 2.0]$, and $40 \sim 60$ iterations of CMCF. Note our method aggressively simplifies the shapes, yet successfully conveys visual information in a concise fashion. Fig. 9d, e, and f show that the texture directionality is well protected and stylized by the constrained mean curvature flow. Some circular features of high importance have been masked before applying the main algorithm (as described in Section 4.4). For example, eyes in Fig. 9a, e, and f were masked. In Fig. 9b, we left the eyes unmasked (because they are non-circular) but masked the beauty spot on the cheek to protect it. In the masked area, the abstraction process is slowed down considerably. As shown in Fig. 10, the user masking enables our algorithm to perform importance-adaptive abstraction [DS02, OBBT07] with relative ease (note that the masked areas remain sharp). Fig. 11 shows how our scheme handles a complex scene. Notice the significant reduction of scene complexity while retaining the essentials in a stylistic way.

We tested our algorithm on an Intel Xeon® 3GHz PC with 2GB memory. The processing time depends mainly on the image size and the number of iterations. For a 500×500 color image, Algorithm 2 took about 30 seconds to complete 50 iterations. For a large image, it is possible to achieve significant speed-up by adopting a downsample-then-process strategy, followed by upsampling and shock filtering to revive sharp edges. Our algorithm is based on local operations, and therefore ideal for GPU implementation, which would bring about dramatic acceleration.

6. Conclusions

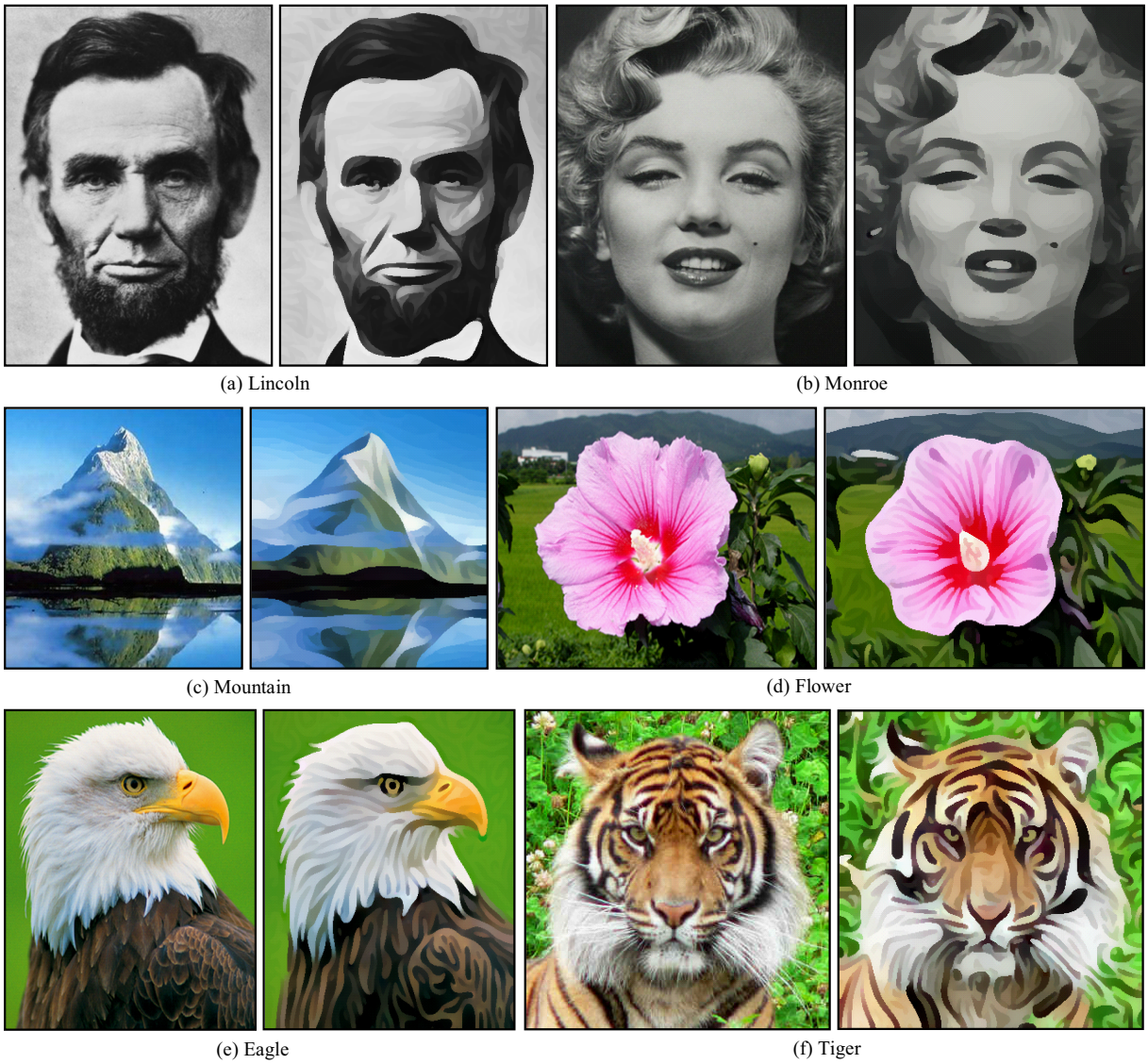
We have presented a novel approach to image abstraction and stylization. Based on an iterative constrained mean curvature flow, our algorithm provides merits such as integrated simplification of shapes and colors, protection of feature directionality, an intuitive control of the degree of abstraction, and the simplicity of implementation.

As shown in Fig. 7, an obvious limitation of our approach is that the curvature flow contracts small, circular shapes very quickly. In case the circular shape is of high importance (for example, an 'eye'), it needs to be masked before running the algorithm. Incorporating an automatic eye-detection technique [DLCD04] may be helpful in this regard.

In the current implementation, the weight map is mainly used to protect highly important area. We may further extend its use to enable *adaptive* shape regularization, in which a shape contracts with a different speed locally. This could be achieved by including prior knowledge on the object or a more sophisticated user input [DS02]. A related topic is how to enable not only 'shrinking' but also 'expanding' of certain shapes in the image, to produce a more exaggerated style of abstraction.

Acknowledgment

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the IT R&D program of MCST/IITA (2008-F-031-01).



(a) Lincoln (b) Monroe

(c) Mountain (d) Flower

(e) Eagle (f) Tiger

Figure 9: *Results*

References

[ALM92] ALVAREZ L., LIONS P. L., MOREL J. M.: Image selective smoothing and edge detection by nonlinear diffusion ii. *SIAM J. Numer. Anal. 29*, 3 (1992), 845–866.

[CM02] COMANICIU D., MEER P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24* (2002), 603–619.

[CRH05] COLLOMOSSE J. P., ROWNTREE D., HALL P. M.: Stroke surfaces: Temporally coherent non-photorealistic animations from video. *IEEE TVCG 11*, 5 (2005), 540–549.

[DeC] DECARLO D.: http://www.cs.rutgers.edu/∼decarlo/data/sg02.

[DLCD04] D'ORAZIO T., LEO M., CICIRELLI G., DISTANTE A.: An algorithm for real time eye detection in face images. In *Proc. ICPR* (2004), pp. 278 – 281.

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. ACM SIGGRAPH* (1999), pp. 317–324.

[DS02] DECARLO D., SANTELLA A.: Stylization and abstraction of photographs. In *Proc. ACM SIGGRAPH* (2002), pp. 769–776.

[Fli] FLICKR: http://www.flickr.com.

[Gra86] GRAYSON M.: The heat equation shrinks embedded plane curves to round points. *J. Differential Geometry 26* (1986), 285–314.

[Gre] GREENSPUN P.: http://philip.greenspun.com.

(a) (b) (c)

Figure 10: *Importance-adaptive abstraction: (a) Input (courtesy DeCarlo and Santella [DeC]) (b) Weight map (c) Abstracted*

(a) (b)

Figure 11: *Abstraction of a complex scene: (a) Input (courtesy Philip Greenspun [Gre]) (b) Abstracted (no user interaction)*

[HD01] HERMAN I., DUKE D.: Minimal graphics. *IEEE Computer Graphics and Applications 21*, 6 (2001), 18–21.

[Her01] HERTZMANN A.: Paint by relaxation. In *Proc. Computer Graphics International* (2001), pp. 47–54.

[KLC07] KANG H., LEE S., CHUI C. K.: Coherent line drawing. In *Proc. Non-photorealistic Animation and Rendering* (2007), pp. 43–50.

[LL06] LECOT G., LEVY B.: Ardeco: Automatic region detection and conversion. In *Proc. Eurogrphics Symposium on Rendering* (2006), pp. 349–360.

[MS96] MALLADI R., SETHIAN J.: Image processing: Flows under min/max curvature and mean curvature. *Geometric Models and Image Processing 58*, 2 (1996), 127–141.

[MSV95] MALLADI R., SETHIAN J., VEMURI B.: Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 2 (1995), 158 – 175.

[OBBT07] ORZAN A., BOUSSEAU A., BARLA P., THOLLOT J.: Structure-preserving manipulation of photographs. In *Proc. Non-photorealistic Animation and Rendering* (2007), pp. 103–110.

[OR90] OSHER S., RUDIN L.: Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis 27* (1990), 919–940.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proc. International Conference on Computer Vision* (1998), pp. 839–846.

[WLL*06] WEN F., LUAN Q., LIANG L., XU Y.-Q., SHUM H.-Y.: Color sketch generation. In *Proc. Non-photorealistic Animation and Rendering* (2006), pp. 47–54.

[WOG06] WINNEMÖLLER H., OLSEN S., GOOCH B.: Real-time video abstraction. In *Proc. ACM SIGGRAPH* (2006), pp. 1221–1226.

[WXSC04] WANG J., XU Y., SHUM H.-Y., COHEN M.: Video tooning. In *Proc. ACM SIGGRAPH* (2004), pp. 574–583.