The Visual Computer manuscript No.
(will be inserted by the editor)

Hyung W. Kang · Wenjie He · Charles K. Chui · Uday K. Chakraborty

Interactive Sketch Generation

Abstract In this paper, we propose an interactive system for generating artistic sketches from images, based on stylized multiresolution B-spline curve model and livewire contour tracing paradigm. Our multiresolution B-spline stroke model allows interactive and continuous control of style and shape of the stroke at any level of details. Especially, we introduce a novel mathematical paradigm called *wavelet frame* to provide essential properties for multiresolution stroke editing, such as feature point preservation, locality, time-efficiency, good approximation, etc. The livewire stroke map construction leads the user-guided stroke to automatically lock on to the target contour, allowing fast and accurate sketch drawing. We classify the target contours as *outlines* and *interior flow*, and develop two respective livewire techniques based on extended graph formulation and vector flow field. Experimental results show that the proposed system facilitates quick and easy generation of artistic sketches of various styles.

Keywords Non-photorealistic rendering · Interactive sketch · Wavelet frame · Multiresolution B-spline · Livewire contour tracing

1 Introduction

Sketch is a simple form of drawing which consists of the salient outlines of a target object. A well-drawn sketch typically contains a small number of curved lines (or *strokes*) and yet effectively conveys the identifying characteristics to the viewers, so that they can quickly recognize and appreciate the subject without being distracted by redundant or unimportant information. That is, generating a sketch requires the capabilities for both identifying the key characteristics and accurately drawing them with curved strokes, which makes it a difficult task for humans and computers alike.

(a) Input image (b) Sketch result

Fig. 1 A sketch generated with our system

In this paper, we propose an interactive system for quick and easy generation of artistic sketches from photographs. We model the stroke as a stylized multiresolution B-spline curve based on a novel mathematical paradigm called *wavelet frame*, providing continuous smoothing and editing of the stroke at multiple levels of details. The use of wavelet frame ensures efficient smoothing/reconsturction/editing process of the stroke and also preserves the feature points, which are essential requirements in interactive sketch applications. Also, to facilitate fast and accurate sketching of both the *outlines* and the *interior essentials* of the target objects, we develop two respective livewire contour tracing techniques based on extended graph formulation and direction vector flow field. Figure 1 shows an example sketch generated with our system from a photograph.

1.1 Related Work

For the past decade or so, a wide variety of techniques have been proposed in the field of non-photorealistic rendering (NPR), for automatically creating simplified and artistic ren-

This research was partly supported by NSF Grant #CCR-0098331 and ARO Grant #W911NF-04-1-0298. It was also supported by the MIC, Korea, under ITRC support program.

Department of Mathematics and Computer Science
University of Missouri - St. Louis
One University Blvd. St. Louis, MO 63121, USA
Tel.: +1-314-516-5841
Fax: +1-314-516-5400
E-mail: {kang,he,chui,uday}@arch.umsl.edu

(a) Input image (b) Edge detection

Fig. 2 Limitation of automatic edge detection

derings from arbitrary reference photographs [22,3,13,19, 24,5,12,10,6]. While these techniques all produce their own distinct rendering styles and simulate specific artistic media such as oil painting, pen-and-ink illustration, watercolor painting, pencil drawing, stippling, engraving, mosaics, stylization, etc., none of these directly addresses the problem of creating a sketch where the salient outlines of the subjects are clearly and accurately depicted. Instead, the existing techniques mainly use strokes to fill the interior of the objects, while their outlines are often obscured or implicitly formed by the aggregate of the interior strokes. In some systems, while outline sketching can be assisted by edge detection techniques [22,13,6], the resulting boundary curves are often incomplete or unclear. Part of the reason for this is that the result of automatic edge detection is seldom perfect (see Fig. 2).

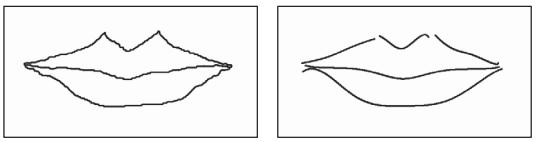
On the other hand, interactive 2D NPR systems such as in [11,21,3,24,5,4,23] provide flexibility to the users in that strokes can be interactively placed at arbitrary positions. However, it is a time-consuming task, even for a professional artist, to manually draw the strokes for accurately tracing the outlines of the objects or for adding tones or textures in their interior regions. While commercial painting programs [4,23] usually provide interactive boundary selection tools (such as 'magic wand' or 'lasso'), their results are often rough and inaccurate, thus requiring manual post-corrections. Also, the existing systems mainly focus on the stroke placing process, and little effort has been made for providing effective stroke smoothing/editing capability which is essential in producing creative, original artworks.

1.2 Overview

The next section discusses the mathematical details of our stroke model, including the B-spline wavelet frame theory. In Section 3, we show how the outlines are sketched, rendered, and edited based on our extended livewire technique and multiscale stroke model. In Section 4, we present another livewire technique that works on a vector flow field, to assist the sketching of the interior essentials of the target objects. Section 5 shows various experimental results and finally, Section 6 concludes this paper.

2 Stroke Model

A sketch is typically represented by a set of smooth stroke curves. In general, the use of smoother curves makes the resulting sketch look more abstracted, stylish, and artistic. Thus, for effective sketch generation, we need a stroke curve model which provides smoothing and editing in multiple levels of details. The representative model with this capability was introduced in [9], based on classical wavelet formulation of cubic B-spline curve. Although this model is an elegant and powerful tool for multiscale curve manipulation, it has some limitations to be used for sketch application. First, the endpoints of the curve are not preserved while smoothing. Fig. 3 shows an example of this phenomenon. Especially, note that discontinuities occur at the 4 feature points on the upper lip boundary, represented by the endpoints of 3 consecutive strokes. Another limitation is that the analysis filters do not have local support, and thus the changes made to the control points cannot be localized and spread all over the control points (and the detail coefficients) in the next level. The resulting analysis process may also slow down the interactive feedback of sketch process. To overcome these limitations, we propose a novel stroke model based on a more flexible multiresolution analysis scheme called *wavelet frame*. Since it is out of the scope of this paper to fully explore its mathematical background, we briefly summarize its formulation in this section. More interested readers are referred to [20,2].



(a) Original strokes (b) Smoothed strokes

Fig. 3 Smoothing with classical wavelet formulation

2.1 Multiresolution analysis

Let $\Phi^j(u) = [\phi_1^j(u), \ldots, \phi_{m_j}^j(u)], u \in [0,1], j = 0, \ldots, n$ be a sequence of vector-valued functions that satisfy the equations $\Phi^{j-1}(u) = \Phi^j(u)P^j$ for some refinement matrices P^j. Let V^j be the spaces of linear span of Φ^j, which satisfy $V^0 \subset V^1 \subset \cdots \subset V^n$. We call $\{V^j\}$ a *multiresolution analysis* (MRA). We define W^j as a complement subspace of V^j in V^{j+1}, that is, $V^{j+1} = V^j + W^j$. We further let W^j be the spaces spanned by $\Psi^j(u) = [\psi_1^j(u), \ldots, \psi_{\ell_j}^j(u)]$, where $\Psi^{j-1}(u) = \Phi^j(u)Q^j$. The basis functions of $\Psi^j(u)$ are called *wavelet frames*, in contrast to the classical wavelets for which $V^{j+1} = V^j \oplus W^j$ is required [1].

2.2 Classical approach on curves

Consider a sequence of curves $f^j(u) = \Phi^j(u)C^j, u \in [0,1]$, where $C^j = [c_1^j, \ldots, c_{m_j}^j]^T$ is a set of control points for the curve at resolution level j. The analysis phase of $f^j(u)$ is implemented through two analysis matrices A^j and B^j in the form of

$$C^{j-1} = A^j C^j, \quad D^{j-1} = B^j C^j, \tag{1}$$

where D^{j-1} gives the wavelet coefficients. The synthesis phase is given by

$$C^j = P^j C^{j-1} + Q^j D^{j-1}, \tag{2}$$

which relies on the *perfect reconstruction* condition,

$$P^j A^j + Q^j B^j = I^j, \tag{3}$$

or equivalently,

$$\begin{bmatrix} P^j & Q^j \end{bmatrix} \begin{bmatrix} A^j \\ B^j \end{bmatrix} = I^j. \tag{4}$$

where I^j is the identity matrix of corresponding size.

2.3 Wavelet frame approach

The classical approach, which includes the biorthogonal and semi-orthogonal cases, is established on an *exact* system, where $\dim(V^{j+1}) = \dim(V^j) + \dim(W^j)$. In this paper, we adopt a *redundant* system, where $\dim(V^{j+1}) < \dim(V^j) + \dim(W^j)$, in order to acquire the desirable properties we need. We rewrite (3) as

$$Q^j B^j = I^j - P^j A^j. \tag{5}$$

One can show that for any $m_{j-1} \times m_j$ matrix A^j, $\operatorname{rank}(I^j - P^j A^j) \geq m_j - m_{j-1}$. For an exact system, A^j must satisfy

$$\operatorname{rank}(I^j - P^j A^j) = m_j - m_{j-1} \tag{6}$$

due to the fact that $\operatorname{rank}(Q^j) = \operatorname{rank}(B^j) = m_j - m_{j-1}$. But for a redundant system, this restriction is removed, thus we have more freedom in selecting A^j.

2.4 Selection of analysis and synthesis matrices

We choose the popular endpoint-interpolating cubic B-splines for the $\Phi^j(u)$'s. (See [9] for the P^j's.) In designing A^j's, we would like to have the following properties: 1) Endpoint preservation; 2) Locality; 3) Good approximation; 4) Computational efficiency.

To have the property of endpoint preservation for all the levels of curves $f^j(u)$, we require that $f^j(0) = f^{j-1}(0)$ and $f^j(1) = f^{j-1}(1)$ for $j = 1, \ldots, n$. That is, $c_1^j = c_1^{j-1}$ and $c_{m_j}^j = c_{m_{j-1}}^{j-1}$, which means that A^j should have the following form by the first equation of (1),

$$A^j = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ * & \ldots & \ldots & * \\ 0 & \ldots & 0 & 1 \end{bmatrix}. \tag{7}$$

To have the locality property, we require that each control point in $\{c_1^{j-1}, \ldots, c_{m_{j-1}}^{j-1}\}$ is determined by only a fixed number of consecutive control points in $\{c_1^j, \ldots, c_{m_j}^j\}$. In other words, all except a fixed number (independent of j) of consecutive entries of each row of A^j are zeros.

For better approximation, A^j should be close to the pseudo inverse of P^j in view of

$$f^j(u) - f^{j-1}(u) = \Phi^j(u)(I - P^j A^j)C^j.$$

The procedure of customizing A^j is based on the above discussion. First, we take a shift-invariant approximation of a truncated version of the pseudo inverse of P^j. Here, all the entries of i-th row are truncated except those with consecutive indices from $2(i-1) - \tau$ to $2(i-1) + \tau$, where τ is a control parameter. Also, the shift-invariant property of a matrix means that the non-zero entries of the $(k+1)$-th row are derived by shifting the non-zero entries of the k-th row 2 positions to the right. We require the shift-invariant property for all rows except the first three and the last three rows of A^j. Then we change the first and the last rows of the resulting matrix to the form of (7) to complete our A^j. After that, by the matrix factorization of the right hand side of (5), Q^j's and B^j's can be derived with the locality property, so that both analysis and synthesis filters are linear and hence very efficient.

We note that there is no guarantee for an exact system to obtain all the desirable properties above. For example, in the method of [9], one cannot simply change the first row and the last row of A^j to force the form of (7) in order to achieve the endpoint preservation property, because the new A^j would not satisfy the condition (6) for an exact system any more.

2.5 Redundant system approach for curve editing

As shown in [9], in order to preserve the multilevel structure during curve editing, we need the following constraints:

$$A^j P^j = B^j Q^j = I, \quad A^j Q^j = B^j P^j = \mathbf{0}. \tag{8}$$

But for a redundant system, (8) is not true in general due to the fact that $[P^j \; Q^j]$ is not a square matrix, which means that we cannot use the same analysis phase as in (1) while preserving the multilevel structure during curve editing. Thus, we make an adjustment based on a matrix extension technique, that is, we extend $[P^j \; Q^j]$ to an invertible square matrix. Equation (4) ensures that there exist P_a^j, Q_a^j, A_a^j, and B_a^j, such that

$$\begin{bmatrix} P^j & Q^j \\ P_a^j & Q_a^j \end{bmatrix} \begin{bmatrix} A^j & A_a^j \\ B^j & B_a^j \end{bmatrix} = I_+^j, \tag{9}$$

where I_+^j is the identity matrix of appropriate size. We associate each C^j with an $(\ell_j + m_{j-1} - m_j)$-dimensional auxiliary vector C_a^j, which can be chosen zero initially. The analysis phase for the redundant system is modified from (1) by adding some correction term as follows,

$$C^{j-1} = A^j C^j + A_a^j C_a^j, \quad D^{j-1} = B^j C^j + B_a^j C_a^j, \quad (10)$$

as is illustrated in Fig. 4.

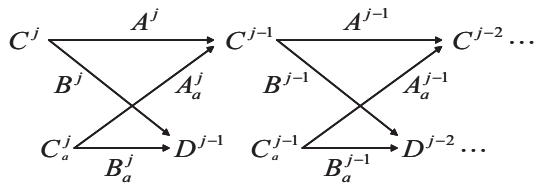


Fig. 4 Extended filterbank process

The role of the terms $A_a^j C_a^j$ and $B_a^j C_a^j$ in (10) is to preserve the multilevel structure between level j and level $j-1$. C_a^j is not used in rendering the curve $f^j(u)$.

When C^{j-1} is modified, C^j and C_a^j would change accordingly, and their changes satisfy $\Delta C^j = P^j \Delta C^{j-1}$ and $\Delta C_a^j = P_a^j \Delta C^{j-1}$. The synthesis phase is kept the same as in (2). The multilevel structure requires that $\Delta C^{j-1} = A^j \Delta C^j + A_a^j \Delta C_a^j$ and $\mathbf{0} = B^j \Delta C^j + B_a^j \Delta C_a^j$, or equivalently, $I = A^j P^j + A_a^j P_a^j$ and $\mathbf{0} = B^j P^j + B_a^j P_a^j$, which is guaranteed by (9).

2.6 Application to sketching

We apply the redundant system designed above, which contains the major analysis and synthesis matrices A^j, B^j, P^j, Q^j, together with the fine-tune compensation matrices A_a^j, B_a^j, P_a^j, Q_a^j, to our stroke model with the analysis (10) and the synthesis phase (2). Note that P_a^j and Q_a^j do not appear in (10) and (2). P^j and Q^j are of sizes $(2^j+3)\times(2^{j-1}+3)$ and $(2^j+3)\times(2^j+2)$, respectively. The sizes of A^j and B^j are the same as those of the transposes of P^j and Q^j, respectively. Our system enjoys all the desirable properties including endpoint preservation, local support, good approximation, and linear time complexity in both phases. Fig. 5 shows that our stroke model preserves the feature points while smoothing the original figure of Fig. 3.

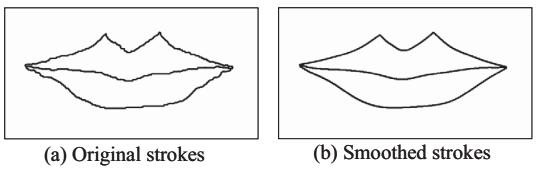


Fig. 5 Preserving feature points while smoothing

3 Sketching Outlines

We decompose the structure of each target object into two components: the *outlines* and the *interior flow*. Note that these two components represent the object *boundaries* and their interior *regions*, respectively. This section discusses the method for sketching the outlines of the target objects. Figure 6 shows our outline sketching process.

For effective outline tracing, we modify/extend the existing livewire algorithms [18, 7, 8, 16]. Livewire (also called Intelligent Scissors) is a highly interactive segmentation tool based on dynamic programming. Modeling the input image as a 2D graph, livewire constructs on the fly a path map in which for each node the minimum-cost path from the user-specified starting node is recorded. As the user steers the livewire curve around the target contour, the system displays the recorded minimum-cost path leading to the current cursor location, giving the impression that the livewire automatically locks on to the target contour. Among all the interactive segmentation techniques, livewire is most suited for our sketch application since it gives the tightest control to the user and does not require post-correction.

Although this paradigm greatly facilitates the boundary tracing process, there are some limitations in the existing livewire algorithms: First, since the algorithm is built on 2D graph and each node can be visited only once in a path, a self-intersecting contour cannot be accurately traced. Also, since the cost of a path is computed as a sum of the local costs of single edges in the path, it is not possible to incorporate high-order internal curve energy (such as curvature) which is useful for reducing the digression rate, especially around noisy or unclear edges. To overcome these limitations, we develop an improved livewire algorithm based on an extended cost function and a 3D graph.

3.1 Cost Function

Our livewire technique is used to find the optimal stroke (rather than path) among all the candidate strokes starting from the user-specified starting point (called *seed point*) $\mathbf{p}_0$, and thus we will use the term *stroke map* instead of path map. While each stroke is recorded in the stroke map as a sequence of pixels $([\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_n])$, it is displayed and edited in the form of multiresolution B-spline curve by using the pixel sequence as control points. Our local cost function for computing the cost of each candidate stroke in the stroke map is defined as follows:

$$c(\mathbf{p}_{k-1}, \mathbf{p}_k, \mathbf{p}_{k+1}) = \alpha|\mathbf{p}_{k+1} - \mathbf{p}_k|^2 + \beta|\mathbf{p}_{k+1} - 2\mathbf{p}_k + \mathbf{p}_{k-1}|^2 + \gamma c_{ext}(\mathbf{p}_k, \mathbf{p}_{k+1}) \quad (11)$$

where α, β, and γ are weight parameters. As in the formulation of active contour [17], $|\mathbf{p}_{k+1} - \mathbf{p}_k|^2$ denotes the first-order continuity of the stroke, and $|\mathbf{p}_{k+1} - 2\mathbf{p}_k + \mathbf{p}_{k-1}|^2$ models its curvature (bending energy). These two terms represent the internal curve energy which is newly added in

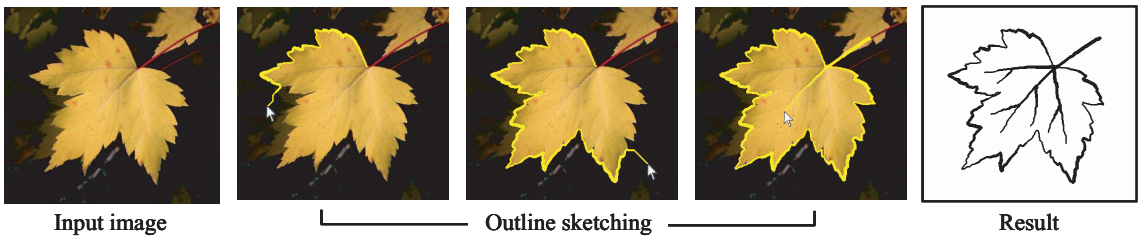


Fig. 6 Outline sketching

our livewire formulation. $c_{ext}(\mathbf{p}_k, \mathbf{p}_{k+1})$ denotes the external energy which is inversely proportional to the gradient magnitudes of the pixels involved. Note that our local cost is defined for three (instead of two) successive pixels $\mathbf{p}_{k-1}$, $\mathbf{p}_k$, $\mathbf{p}_{k+1}$. Thus, unlike previous livewire algorithms, we need to define a graph node as a combination of two neighboring pixels $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ since the internal energy spans two consecutive arcs. Then the recurrence relation for our livewire dynamic programming is written as:

$$cc(\mathbf{p}_k, \mathbf{p}_{k+1}) = \min_{\mathbf{p}_{k-1}} \{cc(\mathbf{p}_{k-1}, \mathbf{p}_k) + c(\mathbf{p}_{k-1}, \mathbf{p}_k, \mathbf{p}_{k+1})\} \tag{12}$$

where $cc(\mathbf{a}, \mathbf{b})$ denotes the cumulative cost of a stroke ending with an arc $(\mathbf{a}, \mathbf{b})$.

3.2 Graph Formulation

Since our recurrence relation (Eq. 12) depends on two consecutive pixels and a single pixel in an image only has a constant number of neighbors (at most 8), it requires a 3-dimensional graph where two pixels constitute a single node. Given an input image with N pixels, the corresponding graph will be constructed with $N \times 8$ nodes, based on the 8-neighbor system. Each node is represented by 3 numbers (x, y, z), where x and y denote the location in the image and z denotes the source neighboring pixel from which it is connected. That is, (x, y) denotes position and z denotes direction. Note a single pixel $\mathbf{p}$ can have 8 distinct nodes in the graph indicating that these nodes will be treated independently even though they all lead to the same pixel $\mathbf{p}$. Our livewire algorithm is essentially an extended version of dynamic programming running on this 3D graph, based on the recurrence relation in Eq. 12 and the cost function in Eq. 11.

Fig. 7 shows some advantages of our livewire formulation. Note that with the use of 3D graph as well as the extended cost function, the user is provided with more flexibility to customize the tracing operation according to the characteristics of the input image. For example, by adjusting the weight values (α, β, γ) of our cost function, various different tracing results can be obtained from the same outline. It is even possible to produce a self-intersecting stroke since a single pixel is now associated with multiple graph nodes (Fig. 7(c)). Also, the incorporation of high-order internal energy contributes to improving the accuracy and smoothness of tracing in the region with noisy or unclear edges, consuming less seed points (Fig. 7(b)).

Fig. 7 Flexibility of our scheme

3.3 Stroke Map Localization

One problem of the original livewire algorithm is that it is time-consuming to compute the optimal path for every pixel in the entire image, and thus it slows down the feedback at the seed point selection. To resolve this problem, we employ the stroke map localization strategy from our previous work [16]. Based on this strategy, at each time the stroke map is constructed only in a local window centered at the current cursor. As the cursor moves, the stroke map is expanded to the new local window and incrementally updated. This ensures real-time performance of the outline tracing task regardless of the image size, and also improves the accuracy of tracing in that it reduces the digression ratio by computing the optimal stroke only in the user-guided window sequence.

3.4 Stroke Rendering and Editing

Following the user's interactive cursor movement around the target outline, the system expands the stroke map and instantly displays the optimal stroke up to the current cursor location, giving the impression that the outline stroke is automatically and accurately drawn by the computer. This greatly eases the tracing process since the tedium of manual contour tracing is removed, and yet the user is allowed to have high-level control over the sketching process through operations like steering the cursor, selecting the desired stroke, placing a new seed point to extend the outline, or signaling to complete the current stroke, etc.

Fig. 8 Example stroke styles

During the interactive outline tracing process, the system renders strokes as stylized multiresolution B-spline curves. Each stroke selected in the stroke map is converted to an endpoint-interpolating multiresolution B-spline curve by sampling the control points from its original pixel sequence and applying our analysis filters. Including the two endpoints, $(2^j + 3)$ control points are sampled with maximum possible integer j. As shown in Fig. 8, the resulting curve is displayed with a specific user-defined style by employing existing stroke stylization techniques [15]. During interaction, the smoothness of the stroke can be controlled by adjusting its level of details (see Fig. 9). The linear interpolation between adjacent-level curves is supported as in [9] to allow continuous stroke smoothing and reconstruction. Also, each stroke in the stroke sequence can be displayed in its own distinct level of details (see Fig. 9(d)). Note that the stroke curve can be smoothed and reconstructed without deviating from the cursor (or the seed point) since the endpoints of the stroke are preserved as described in Section 2.

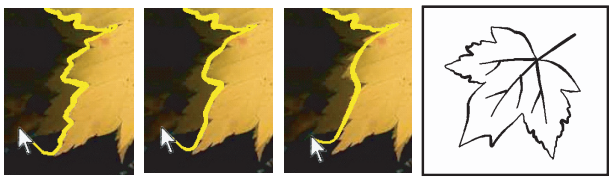


Fig. 9 Continuous level control while tracing

Each constructed outline is in general composed of a sequence of strokes, separated by the seed points. These seed points can be directly used as *feature points* which need preserving, and can be post-edited: a new seed can be added to split a stroke; a seed can be removed to merge strokes; a seed can be moved to change the end points of the two associated strokes simultaneously. Also, any stroke curve may be interactively post-edited in any level. The stroke can be locally edited by ordinary B-spline curve editing, and it can also be globally edited (to change the overall sweep of the curve) by editing the curve in a lower resolution and then return back to its original resolution (see Fig. 10). The latter operation is especially important in sketch application, and yet very difficult to achieve with a single-level B-spline model (since B-spline has a local support).

Fig. 10 Multiresolution stroke editing

4 Sketching Interior Flow

While outlines are the most essential components in sketch drawings, artists often put a collection of strokes in the interior of the objects also, in order to add depth or richness to the sketch. In general, artists draw (or paint) strokes to depict these interior elements of the object, which are usually a combination of the color, tone, or texture information of the corresponding local region. These elements, as a whole, are often perceived by artists to form a virtual 'flow' in the interior, as we can see in many traditional paintings or illustrations. In this section, we describe how to effectively generate this *interior flow*, and based on which how to guide the user's stroke placement (and alignment) in the interior of the objects.

4.1 Generating Tangent Vector Flow

The main factor which governs the direction of the interior flow is the neighboring outlines. That is, the flow vector at a pixel can be defined by its tangent vector (the perpendicular vector of the image gradient). Unlike the outline pixels, however, the interior pixels often form small textures or a homogeneous region and thus their tangent vectors may be unrelated to those of the nearby 'dominant' outlines, or their magnitudes may be so small that their directions are meaningless. Thus, to force the tangent vectors of all the interior pixels to follow the flow of the dominant outlines, we employ the gradient vector flow construction algorithm based on variational formulation [25].

The tangent vector flow field is defined by $\mathbf{v}(x, y) = [u(x, y), v(x, y)]$ that minimizes the following energy functional:

$$E(\mathbf{v}) = \iint |\nabla I^{\perp}|^2 |\mathbf{v} - \nabla I^{\perp}|^2 + \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) dx dy \tag{13}$$

where $\nabla I^{\perp} = (-\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x})$ and μ is a regularization parameter. Note that we use the image tangent vectors $\nabla I^{\perp}$ instead of the gradient vectors $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ used in [25]. For a pixel where $|\nabla I^{\perp}|$ is large, its flow vector $\mathbf{v}$ is more affected by the first term and thus tends to resemble the original tangent vector. That is, the dominant outlines are preserved. When $|\nabla I^{\perp}|$ is small, its $\mathbf{v}$ is more affected by the

second term and adjusts its direction according to the neighboring vectors to create a smoothly varying field around it. Based on calculus of variations, the above energy functional is minimized by solving the following Euler equations:

$$\mu\nabla^2 u - (u + \frac{\partial I}{\partial y})((\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2) = 0 \quad (14)$$

$$\mu\nabla^2 v - (v - \frac{\partial I}{\partial x})((\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2) = 0 \quad (15)$$

where ∇^2 is a Laplacian operator. These equations are solved numerically, starting from the initial $\mathbf{v}$ (set as $\nabla I^{\perp}$), and terminating when a steady state is reached. Fig. 11 illustrates tangent vector flow fields obtained from sample images. As shown in the figure, the tangent vector flow can also be generated from an outline sketch, in which case the vectors are aligned more strictly with the outlines.

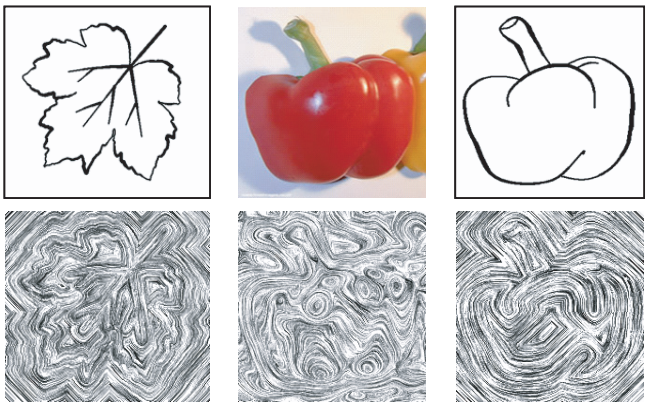


Fig. 11 Example tangent vector flow fields

4.2 Flow-based Livewire

The constructed tangent vector flow field can be used to suggest a direction for each stroke to be placed in the interior of the objects. As in the case of outline tracing, however, it is a difficult task to accurately place the interior strokes by manually tracing the flow of vectors. Thus, we propose a new type of livewire technique where the livewire follows the flow of the vector field. To our knowledge, all existing livewire algorithms have been designed on a scalar field where each node is associated with a scalar value (such as pixel intensity or gradient magnitude), rather than a (direction) vector value.

We construct a 2D graph from an image where the node is represented by each pixel and the arc by the link between neighboring pixels. Each node $\mathbf{p} = (x, y)$ is associated with a direction vector $\mathbf{v}(\mathbf{p}) = \mathbf{v}(x, y) = [u(x, y), v(x, y)]$. Given the starting node specified by the user, our goal is to construct a stroke map where a minimum-cost stroke (from the starting node) is recorded for each node in the graph. Here, the minimum-cost stroke is defined as the stroke which best follows the local flow of the vector field. Thus, the cost function for an arc $c(\mathbf{p}, \mathbf{q})$ between neighboring pixel $\mathbf{p}$ and $\mathbf{q}$ is defined as follows:

$$c(\mathbf{p}, \mathbf{q}) = 1 - \frac{|\mathbf{v}(\mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})|}{|\mathbf{v}(\mathbf{p})|\ |\mathbf{q} - \mathbf{p}|} \quad (16)$$

where $\mathbf{v}(\mathbf{p})$ denotes the flow vector at pixel $\mathbf{p}$ and $|\mathbf{q}-\mathbf{p}|$ denotes the displacement vector from $\mathbf{p}$ to $\mathbf{q}$. Also, $|\mathbf{v}(\mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})|/|\mathbf{v}(\mathbf{p})|\ |\mathbf{q}-\mathbf{p}|$ computes the cosine of the angle between $\mathbf{v}(\mathbf{p})$ and $|\mathbf{q}-\mathbf{p}|$. That is, the cost of an arc is proportional to the angle formed by the arc and the desired direction. Based on the new cost function and the graph structure, the dynamic programming is applied to construct the stroke map, and the user can easily trace the flow of the direction vectors by interactively steering the cursor and selecting from the candidate strokes in the map. Fig. 12 shows how our flow-based livewire algorithm works. Instead of snapping to the nearby outlines (Fig. 12(b)), the livewire now follows the given interior direction flow (Fig. 12(c)).

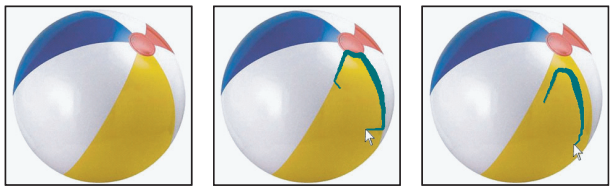


Fig. 12 Flow-based livewire

Our stylized multiresolution B-spline curves are once again used for modeling, rendering, and editing interior strokes. There are various ways to stylize the interior strokes (see Fig. 13). For example, the stylization technique in [15] can be applied to simulate ordinary pen strokes or brush strokes. The strokes can also be textured to produce realistic painting effects [14], and the strokes may be alpha-blended on the canvas to simulate watercolor painting [13]. In addition, each interior stroke can also be displayed as a collection of strokes as in [21], to expedite the sketch process and reduce the tedium of placing individual strokes for making collective tonal or textural effects.

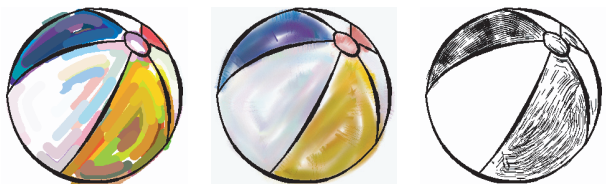


Fig. 13 Stylizations of interior strokes

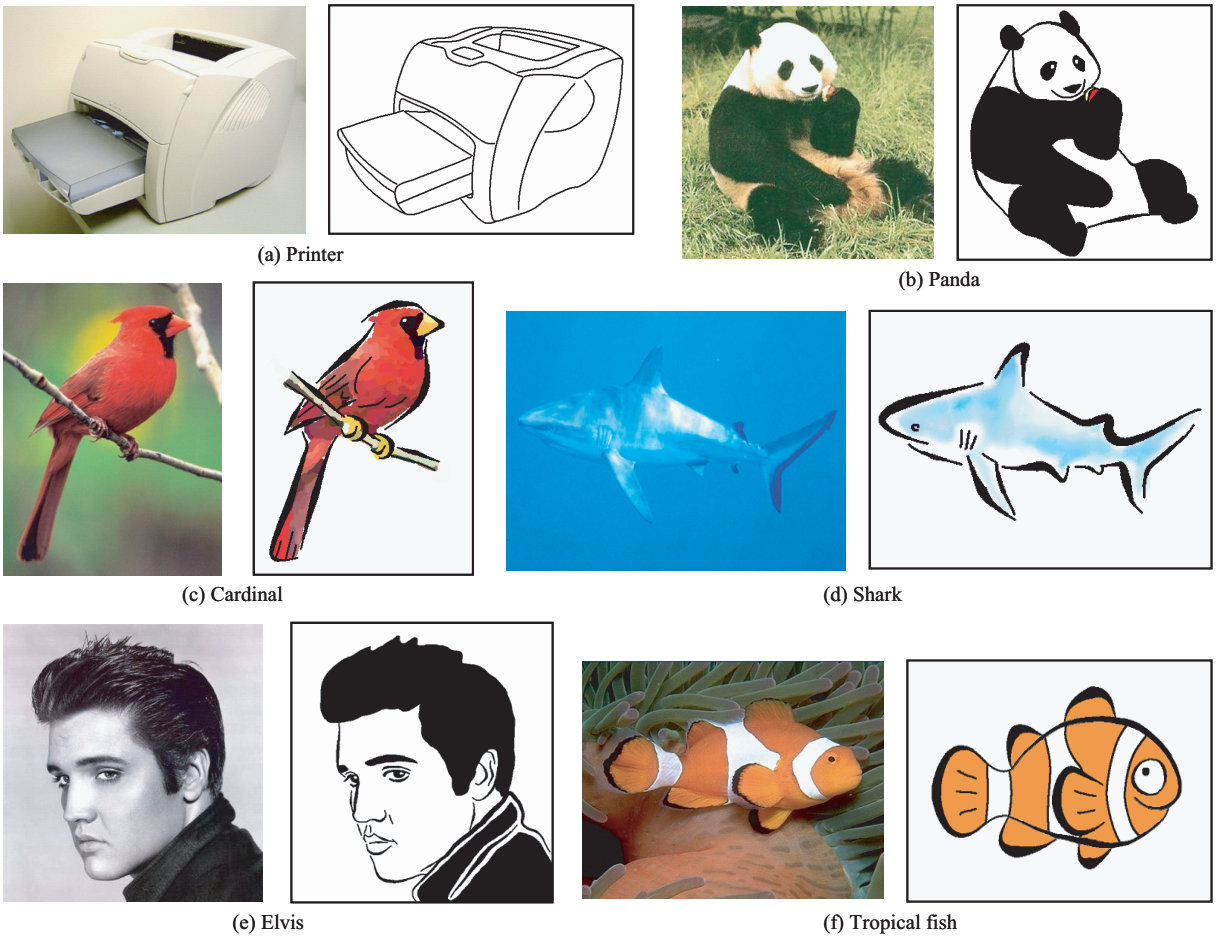


Fig. 14 Various sketch results

5 Experimental Results

Fig. 14 contains various sketch results generated with our interactive sketch system. As shown in Fig. 14(a), our system can be used to produce technical (rather than artistic) illustrations of commercial products, by reducing stylizations and focusing on accurate outline capture. This type of illustrations are often required in the product manuals or assembly instruction guides. Fig. 14(b) shows that our outline tracing scheme works well even for complex foreground outlines and background structure. The interiors of some regions are colored with the flood-filling method to produce the cartoon-style sketch. In Fig. 14(c) and Fig. 14(d), the interior strokes are stylized to simulate oil painting and watercolor blending effects, respectively. Fig. 14(e) and Fig. 1 show the sketches generated from complex portrait images. Note that in Fig. 1, the feathers hanging from the hat are sketched by tracing the interior flow. Also, in Fig. 14(e), interior strokes are used to generate double outlines (for the shirt). This is an often-used trick by illustrators to reveal the outlines more clearly. Fig. 14(f) shows that by tracing, smoothing, and editing the curves in multiresolution we can easily create an exaggerated, cartoon-like figure from a photograph. All the results in this paper have been generated in 3-20 minutes by non-professional users (including interaction time plus preprocessing time). Our system is implemented on Pentium 4 PC (3 GHz CPU, 1 GB Memory and NVIDIA® GeForce FX 5950 GPU).

6 Conclusions

We have presented an interactive system for generating artistic sketches from images, based on stylized multiresolution B-spline model and livewire contour tracing paradigm. As shown in our experimental results, the proposed system facilitates quick and easy generation of artistic sketches of various styles from photographs. We are currently extending our wavelet frame formulation to non-uniform B-spline to handle the strokes with more flexibility and efficiency. Also, we will incorporate various other stroke styles and devise a technique to let the system select optimal styles automatically from the style database.

References

1. Chui, C. K.: An Introduction to Wavelets, Academic Press (1992)
2. Chui, C. K., He, W.: Compactly supported tight frames associated with refinable functions, Appl. Comp. Harmonic Anal. **8**, 293–319 (2000)
3. Curtis, C., Anderson, S., Seims, J., Fleischer, K., Salesin, D.: Computer-generated watercolor, In Proc. ACM SIGGRAPH, pp. 421-430 (1997)
4. Davis, J.: The Adobe Photoshop 7 Wow Book, Peachpit Press (2003)
5. Deussen, O., Hiller, S., Van Overveld, K., Strothotte, T.: Floating points: A method for computing stipple drawings, In Computer Graphics Forum, Vol. 19, No. 3, pp. 40-51 (2000)
6. DeCarlo, D. and Santella, A.: Stylization and Abstraction of Photographs. In Proc. ACM SIGGRAPH, pp. 769-776 (2002)
7. Falcão, A., Udupa, J., Samarasekera, S., Sharma, S., Hirsch, B., Lotufo., R.: User-Steered Image Segmentation Paradigms: Live Wire and Live Lane. Graphical Models and Image Processing, Vol. 60, No. 5, pp. 233-260 (1998)
8. Falcão, A., Udupa, J., Miyazawa, F.: An Ultra-Fast User-Steered Image Segmentation Paradigm: Live Wire on the Fly. IEEE Transactions on Medical Imaging, Vol. 19, No. 1, pp. 55-62 (2000)
9. Finkelstein, A., Salesin, D.: Multiresolution Curves, In Proc. ACM SIGGRAPH, pp. 261-268 (1994)
10. Gooch, B., Coombe, G., Shirley, P.: Artistic Vision: Painterly Rendering Using Computer Vision Techniques. In Proc. Non-Photorealistic Animation and Rendering, pp. 83-90 (2002)
11. Haeberli, P.: Paint by numbers: Abstract image representations, In Proc. ACM SIGGRAPH, pp. 207-214 (1990)
12. Hausner, A.: Simulating decorative mosaic, In Proc. ACM SIGGRAPH, pp. 573-578 (2001)
13. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes, In Proc. ACM SIGGRAPH, pp. 453-460 (1998)
14. Hertzmann, A: Fast paint texture, In Proc. Non-Photorealistic Animation and Rendering, pp. 91-96 (2002)
15. Hsu, S., Lee, I.: Drawing and animation using skeletal strokes, In Proc. ACM SIGGRAPH, pp. 109-118 (1994)
16. Kang, H., Shin, S.: Enhanced lane: interactive image segmentation by incremental path map construction, In Graphical Models, Vol. 64, No. 5, pp. 282-303 (2002)
17. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active Contour Models. International Journal of Computer Vision, Vol. 1, No. 4, pp. 321-331 (1988)
18. Mortensen, E., Barrett, W.: Intelligent Scissors for Image Composition. In Proc. ACM SIGGRAPH, pp. 191-198 (1995)
19. Ostromoukhov, V: Digital facial engraving. In Proc. ACM SIGGRAPH, pp. 417-424 (1999)
20. Ron, A., Shen, Z. W.: Affine systems in $L_2(R^d)$: the analysis of the analysis operator, J. Funct. Anal. **148**(2), 408–447 (1997)
21. Salisbury, M., Anderson, S., Barzel, R., Salesin, D.: Interactive pen-and-ink illustration, In Proc. ACM SIGGRAPH, pp. 101-108 (1994)
22. Salisbury, M., Wong, M., Hughes, J., Salesin, D.: Orientable textures for image-based pen-and-ink illustration, In Proc. ACM SIGGRAPH, pp. 401-406 (1997)
23. Steuer, S.: The Adobe Illustrator CS Wow Book, Peachpit Press (2003)
24. Sousa, M., Buchanan, J.: Observational Models for Graphite Pencil Materials, In Computer Graphics Forum, Vol. 19, No. 1, pp. 27-49 (2000)
25. Xu, C., Prince, J. L.: Snakes, Shapes, and Gradient Vector Flow, IEEE Transactions on Image Processing, pp. 359-369 (1998)

Hyung W. Kang received his Ph.D. degree in computer science from KAIST (Korea Advanced Institute of Science and Technology) in 2002. He is currently an assistant professor of computer science at UMSL (University of Missouri, St. Louis). His main research interests include image-based modeling and rendering, non-photorealistic rendering, image and video processing, facial expression animation, curve and surface modeling, etc.

Wenjie He received the B.S. degree in mathematics from Peking University, Beijing, China in 1988 and the Ph.D. degree from the University of Georgia, Athens, Georgia in 1998. He is currently an Associate Professor of Computer Science at the University of Missouri-Saint Louis. His research interests include wavelets, wavelet tight frames, computer graphics, and image processing.

Charles K. Chui is currently Curators' Professor at the University of Missouri, St. Louis, and Consulting Professor of Statistics at Stanford University. He is co-editor-in-chief of Applied and Computational Harmonic Analysis, and serves on the editorial board of seven other journals. His research interests are in the areas of Approximation Theory, Computational Harmonic Analysis, Surface Subdivisions, and Mathematics of Imaging.

Uday K. Chakraborty is an associate professor of computer science at UMSL, and has previously held positions at CMC Limited (India and UK), Jadavpur University (India) and GMD (Germany). He is an editor of Computing and Information Technology, and has guest-edited special issues of several other journals. His research interests include evolutionary computation, fuzzy cognitive maps, and heuristics. He has (co-)authored one book and about sixty papers.