

ColorSketch: A Drawing Assistant for Generating Color Sketches from Photos

Guanbin Li and Sai Bi ■ *University of Hong Kong*

Jue Wang ■ *Adobe Research*

Yingqing Xu ■ *Tsinghua University*

Yizhou Yu ■ *Zhejiang University*

Color sketching is an abstract artistic style that attempts to augment sparse pencil strokes with casual colored brush strokes. The gouache paints used in these sketches generate less visible brush marks than oil paintings and are less fluid than watercolors. As an example, Elisha Cooper, a well-known writer and children's book author, has published multiple sketchbooks recording his visits to places such as California¹ and New York City.² As Figure 1 illustrates, the colored brush strokes in such sketches provide rich visual information and create a vivid impression of the objects and scenes they depict.

The interactive drawing system ColorSketch can help novice users generate color sketches from photos. To preserve artistic freedom and expressiveness, the proposed system gives users full control over pencil strokes, while automatically augmenting pencil sketches using color mapping, brush stroke rendering, and blank area creation.

Many systems have been proposed to help users draw pencil sketch lines using the underpainting technique, which consists of drawing on a semitransparent canvas over the reference photo.^{3,4} However, even with good sketch lines, novice users still encounter difficulties when creating a color sketch, such as deciding which gouache paints to use to approximate the pixel colors in a photo and which subregions to

intentionally leave blank and which to fill with colored brush strokes. To help novice users, we developed ColorSketch, a sketching interface that automatically resolves stylization issues related to gouache painting colors, brush layouts, and region filling styles given user-provided sparse pencil strokes. In a live sketch session, to preserve artistic freedom and expressiveness, our system gives the user complete control over pencil strokes, including their location, shape, drawing order, and level of abstraction. While the user is drawing, our system analyzes the existing sketch layout and automatically generates color-brushed regions that are compatible with the pencil sketches. In this way, color sketches created using our interface are a mix of user-provided abstraction and computer-generated stylization. (See the “Related Work in User-Assisted Drawing” sidebar for earlier work in this area.)

Given an input photo, producing a color sketch using our system consists of two stages: an offline automatic preprocessing stage and an online drawing stage carried out with the sketching interface. In the offline stage, a series of preprocessing steps are performed on the reference photo, including hierarchical image segmentation, occlusion contour detection, and depth-map generation. During the online stage, the user draws pencil strokes,

and our system provides real-time autocompletion suggestions, which are displayed as shadow strokes that the user can easily accept. It also analyzes pencil strokes in real time to infer image regions the user implicitly defines while sketching and automatically adds stylization effects to such regions. The stylization effects include region boundary smoothing, pixel to gouache color mapping, brush stroke rendering according to an automatically computed layout, and blank area creation according to automatic occlusion analysis results.

Technically, although portions of ColorSketch follow the basic framework of a painting system, our approaches for computing the center, orientation, color, and order of brush strokes are novel, allowing spatially coherent brush stroke placement and rendering. The resulting brush marks are noticeable, but not too obvious, mimicking the style of gouache paintings. In addition, blank area is an important feature of color sketches. Our blank area creation technique generates results resembling those drawn by artists. We tested our interface in a user study. Comparisons between sketches produced with and without our system indicate that users, especially novice ones, can generate much better color sketches more efficiently with our system than using traditional manual tools.

System Overview

When developing our computer-assisted color sketching system, we distilled a few principles from sketchbooks^{1,2} and artist sketches (see Figure 1):

- Pencil strokes delineate object contours and the boundaries of salient regions. They are sparse and are approximately aligned with such contours or boundaries. Artists usually draw contours of large regions first and then gradually add smaller ones.
- Regions are brushed with gouache paints instead of being filled with solid colors. The brush stroke sizes vary with the size of the region. For instance, smaller regions are filled with smaller brush strokes. Gouache paints are more opaque than watercolors, and they generate less visible brush marks than those in oil paintings.
- The color of gouache paints used for each region deviates significantly from the original pixel colors within the same region. It is typically brighter and less saturated than the original pixel colors, while its hue remains approximately the same.
- Many regions are not completely filled with brush strokes. Instead, some have border areas intentionally left blank. Blank areas increase

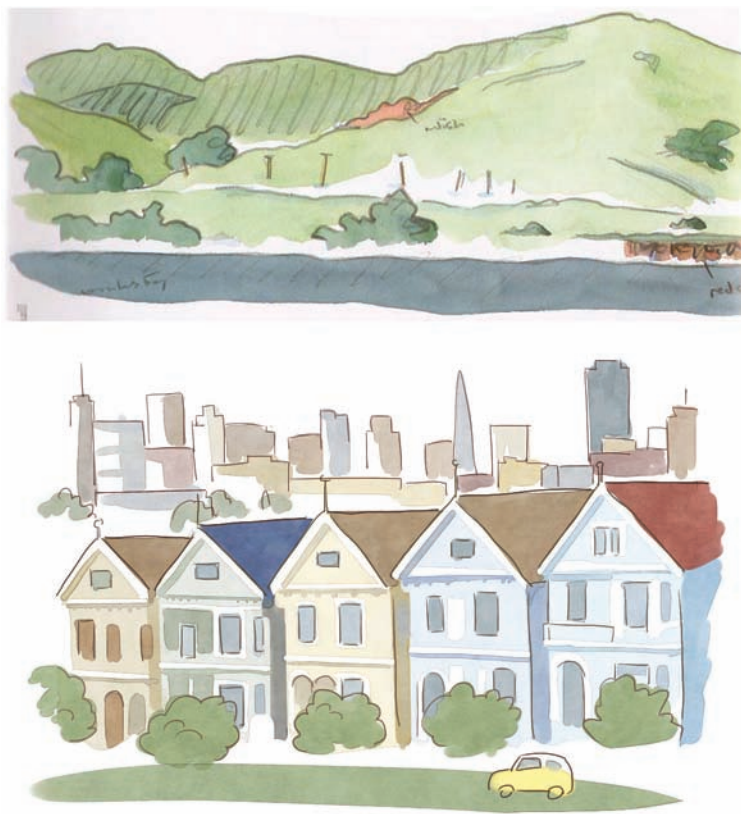


Figure 1. Example color sketches manually drawn by artists. Color sketching is an abstract artistic style that utilizes sparse details, outlining, and white space to highlight scene objects.

spatial nonuniformity and give a pleasant look. They are primarily used to highlight areas such as those near occlusion boundaries on partially occluded objects or regions.

The ColorSketch system workflow thus incorporates elements unique to the color sketching technique (see Figure 2). Figure 3 shows some sample color sketches generated with our ColorSketch interface.

As we explained earlier, producing a color sketch from a reference photo using our system consists of two stages: an offline preprocessing stage and an online interactive drawing stage. In the preprocessing stage, our system performs contour detection and hierarchical image segmentation to extract visual guides that emphasize the boundaries of different objects in the photo. It automatically generates these guides in the form of thin strokes.

During the online stage, as Figure 2b shows, the user draws sparse pencil strokes to delineate object shapes using the under-painting technique. This is the main user input to our system. We give the user sufficient freedom to choose appropriate levels of abstraction in different parts of the image. To facilitate contour sketching, our interface dynamically displays a stroke automatically extracted

Related Work in User-Assisted Drawing

Researchers working in nonphotorealistic rendering (NPR) have extensively studied automatically converting an image into a stylized rendering. The various systems proposed focus on different styles, such as pencil and color pencil drawing,¹ pen and ink illustration,² oil painting,³ and watercolor.⁴ These approaches aim at automatically generating high-quality rendering results, with little or no user interaction. In contrast, our system is a drawing assistance tool that supports user interaction.

Various interactive systems have been proposed to help users draw sketches or create digital art, either for general cases^{5,6} or specific objects such as human faces⁷ or 3D models.⁸ Our work was partially inspired by ShadowDraw, a system that guides freeform sketching.⁹ As the user draws, ShadowDraw dynamically updates a shadow image underlying the user's strokes as a guide for drawing object contours. The shadow image is generated by searching through a large image dataset. Other researchers have explored using crowdsourcing to generate sketch guidance.^{10,11} In contrast, our system uses a novel, autocompletion mechanism to help users sketch object contours, using a single reference image.

Recently, Emmanuel Iarussi and his colleagues presented a drawing tool that provides automated guidance with model photographs to help people practice traditional drawing-by-observation techniques.⁵ Qingkun Su and his colleagues proposed EZ-Sketch, a system that helps users draw accurate pencil sketches over an image.¹² It employs a multilevel optimization framework to adjust the positions of user strokes for improved accuracy. Our system adopts a different strategy, online autocompletion guidance, to help users draw more accurately and efficiently. Furthermore, our system focuses mainly on generating stylized color regions given the sketch lines.

Doug DeCarlo and Anthony Santella introduced a computational approach to stylizing and abstracting photographs.¹³ Their system uses mean shift to segment an image into coherent regions and transforms them into a style that features bold edges and constant-color regions. Jue Wang and his colleagues extended this approach to video by treating a video as a space-time volume and segmenting it into contiguous blobs using spatial-temporal mean shift segmentation.¹⁴ However, the advocated style in these approaches differs from color sketching. The constant color adopted for a region is simply the average pixel color in that region, and there are no blank areas within the regions. In contrast, our system performs color stylization, which maps pixel colors to more meaningful painting colors.

The style adopted by Fang Wen and his colleagues is perhaps most similar to color sketching.¹⁵ Nevertheless, they do not paint brush strokes within regions. We also introduce a more systematic approach that creates blank

areas near occlusion boundaries to highlight regions. More importantly, their system produces semiautomatic results, whereas our system is a drawing assistant that gives the user artistic freedom and control.

References

1. C. Lu, L. Xu, and J. Jia, "Combining Sketch and Tone for Pencil Drawing Production," *Proc. Int'l Symp. Non-photorealistic Animation and Rendering*, 2012, pp. 65–73.
2. M.P. Salisbury et al., "Orientable Textures For Image-Based Pen-and-Ink Illustration," *Proc. 24th Ann. Conf. Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1997, pp. 401–406.
3. K. Zeng et al., "From Image Parsing to Painterly Rendering," *ACM Trans. Graphics*, vol. 29, no. 1, 2009, article no. 2.
4. A. Bousseau et al., "Interactive Watercolor Rendering with Temporal Coherence and Abstraction," *Proc. 4th Int'l Symp. Non-photorealistic Animation and Rendering (NPAR)*, 2006, pp. 141–149.
5. E. Iarussi, A. Bousseau, and T. Tsandilas, "The Drawing Assistant: Automated Drawing Guidance and Feedback from Photographs," *Proc. 26th Ann. ACM Symp. User Interface Software and Technology (UIST)*, 2013, pp. 183–192.
6. L. Benedetti et al., "Painting with Bob: Assisted Creativity for Novices," *Proc. 27th Ann. ACM Symp. User Interface Software and Technology (UIST)*, 2014, pp. 419–428.
7. D. Dixon, M. Prasad, and T. Hammond, "iCanDraw: Using Sketch Recognition and Corrective Feedback to Assist a User in Drawing Human Faces," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI)*, 2010, pp. 897–906.
8. S. Grabli et al., "Programmable Rendering of Line Drawing from 3D Scenes," *ACM Trans. Graphics*, vol. 29, no. 2, 2010, article no. 18.
9. Y. Lee, C. Zitnick, and M. Cohen, "ShadowDraw: Real-Time User Guidance for Freehand Drawing," *ACM Trans. Graphics*, vol. 30, no. 4, 2011, article no. 27.
10. Y. Gingold et al., "Diamonds from the Rough: Improving Drawing, Painting, and Singing via Crowdsourcing," *Proc. AAAI Workshop on Human Computation (HCOMP)*, 2012.
11. A. Limpaecher et al., "Real-Time Drawing Assistance through Crowdsourcing," *ACM Trans. Graphics*, vol. 32, no. 4, 2013, article no. 54.
12. Q. Su et al., "EZ-Sketching: Three-Level Optimization for Error-Tolerant Image Tracing," *ACM Trans. Graphics*, vol. 33, no. 4, 2014, article no. 54.
13. D. DeCarlo and A. Santella, "Stylization and Abstraction of Photographs," *ACM Trans. Graphics*, vol. 21, no. 3, 2002, pp. 769–776.
14. J. Wang et al., "Video Tooning," *ACM Trans. Graphics*, vol. 23, no. 3, 2004, pp. 574–583.
15. F. Wen et al., "Color Sketch Generation," *Proc. 4th Int'l Symp. Non-photorealistic Animation and Rendering*, 2006, pp. 47–54.

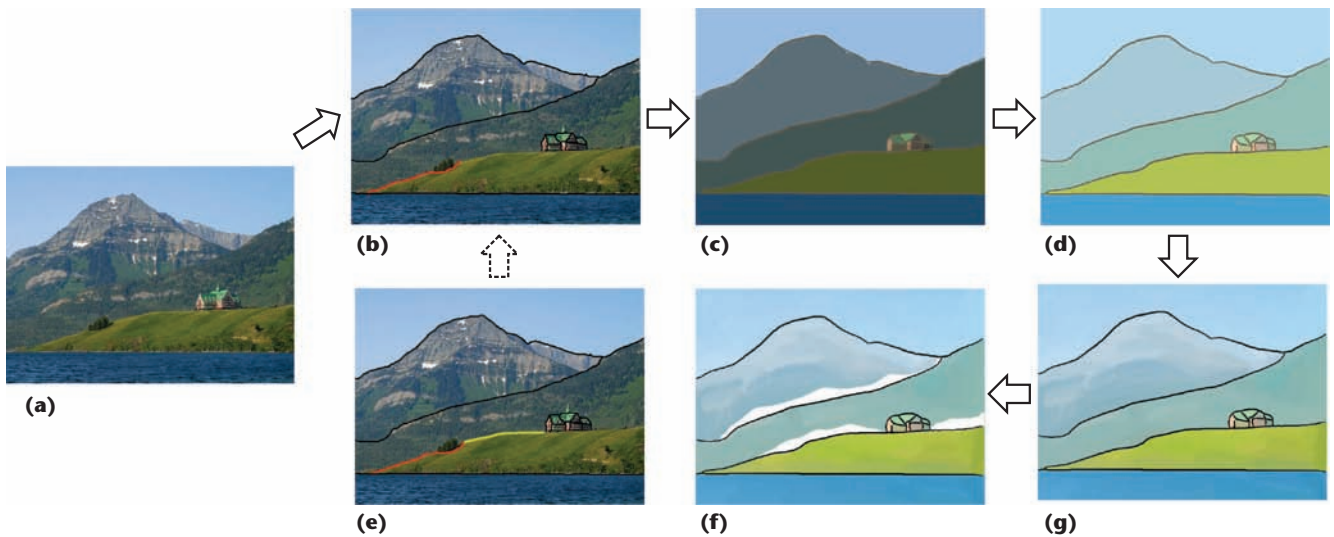


Figure 2. ColorSketch system workflow: (a) reference image, (b) contour drawing, (c) region detection, (d) color stylization, (e) system contour guides (preprocessing), (f) blank area insertion, and (g) brush placement and rendering.



Figure 3. Sample color sketches generated with our ColorSketch interface. The top row shows reference images. The first two sketches in the bottom row were drawn by novice painters, and the last two sketches were drawn by skilled painters.

during the preprocessing stage as a suggestion near the stroke the user is drawing (see Figure 2e). The user can choose to accept the stroke or ignore it and continue to draw. Whenever the user finishes a new pencil stroke or revises an existing stroke, the system automatically updates the final sketching result by adjusting and rendering color regions.

Given that the casual user strokes may not form closed regions, our system automatically infers closed regions given the existing pencil strokes (see Figure 2c). It further computes the color of the gouache paints that should be applied in each region (see Figure 2d) as well as the layout and the rendering order of the brush strokes that will be used for filling it (Figure 2g). Our system also determines the areas that should be left blank according to occlusion boundaries detected in the preprocessing stage (see Figure 2f).

Preprocessing

Given an input image, our system first scales it so that the length of the longer side equals 400 pixels and automatically segments it into coherent regions. The boundaries of these regions are used to generate autocompletion suggestions in the online drawing stage. The precomputed regions are obtained using a hierarchical segmentation algorithm.⁵ We chose this algorithm because of its high accuracy on the Berkeley Segmentation Benchmark⁶ with respect to manually produced ground truth. In addition, this algorithm can generate region boundaries hierarchically with different levels of details.

In our system, we run the source code provided in earlier work⁵ and extract region boundaries at three levels according to three different threshold (0.15, 0.30, and 0.50) in the generated ultrametric contour map (UCM), as Figure 4 shows. The extracted

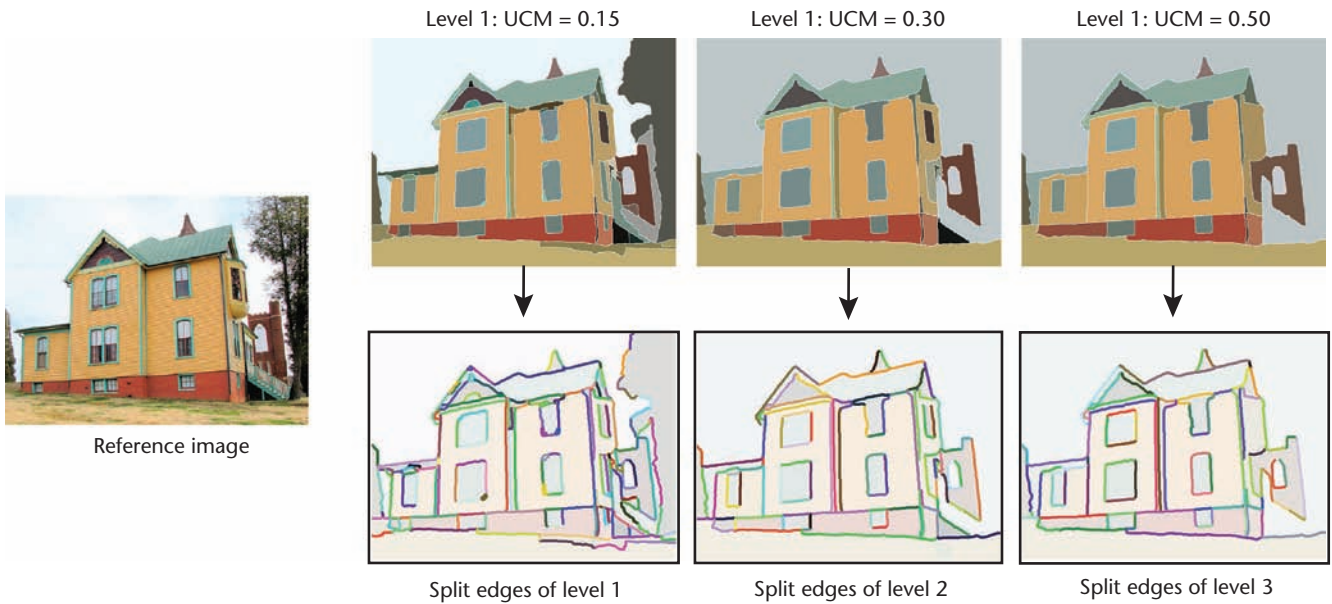


Figure 4. An example of hierarchical image segmentation and boundary segment (stroke) generation. Segmentations at three different levels of detail are extracted from the hierarchical segmentation result, and region boundaries therein are further divided into segments at T-junctions and local maxima of curvature.

boundaries are further divided into segments at T-junctions and local maxima of curvature.⁷

Color sketch artists tend to leave blank areas near boundaries of occluded areas. To quickly handle occlusion boundaries during an interactive drawing session, we precompute them and a depth map from the input image using a scene-based occlusion reasoning method proposed in earlier work,⁸ using the provided source code and parameters setting. When the user adds a new pencil stroke during an interactive session, if it is part of a precomputed occlusion boundary or if there is a large difference in depth values along the shared boundary between two adjacent regions, our system automatically leaves a blank area in the occluded region. This rule applies to all user-defined regions except for the sky and the ground plane, both of which can be detected using the model proposed in earlier work.⁸

Interactive Color Sketching

The ColorSketch's user interface consists of a tool bar and two side-by-side windows. The left window is used for drawing; it shows the reference photo overlaid with a translucent canvas. The right output window displays the automatically completed color sketching result and updates it in real time. The user can also fine-tune the result directly in the output window.

To start drawing a color sketch, the user chooses the pencil tool and draws line strokes on the canvas, using the underneath reference photo as a guide. When the user is drawing a stroke, our system continuously analyzes it in the background,

searching through a set of strokes automatically extracted during preprocessing to find the best candidate stroke that the user is most likely meant to draw. ColorSketch then shows this candidate stroke as a shadow on the canvas. The user can directly accept it using a keyboard shortcut. The accepted shadow strokes are usually more accurately positioned than the user's unfinished strokes. In this way, our system can greatly improve the user's drawing efficiency.

When there are multiple candidate strokes near the unfinished stroke, the system chooses one according to the following rules:

1. The degree of consistency between the tangential directions of the candidate stroke and the unfinished stroke is less than or equal to 0.5.
2. If more than one candidate stroke exists under rule 1, the number of overlapping pixels between the two strokes is considered. If all the candidate strokes in this stage have overlapping pixels less than or equal to 50, the system chooses the one with the largest overlap. Otherwise, it chooses the one according to rule 3.
3. The total length of the candidate stroke is considered, and longer strokes are preferred.

Once a new stroke is finished, our system checks whether it can be joined together with the existing ones to illustrate a closed region. Given that casual strokes may never form a perfect closed shape, we implicitly join nearby strokes by connecting the endpoints of strokes that are very close. Specifically, we run a nearest-neighbor search on the

two endpoints of the new stroke. The new stroke is joined to an existing stroke if the distance between their closest endpoints is below a prescribed threshold (eight pixels in our system). If multiple candidates satisfy this threshold, we simply calculate the mean position of their related endpoints and deform all strokes so that their endpoints can reach this mean position.

At any time, these joined strokes on the canvas divide the image space into one or more regions. Every region is surrounded by a sequence of joined strokes forming a closed loop. Once all regions have been detected, our system completes the stylization of these regions automatically in real time by adding colored brush strokes and blank areas and updates the result in the output window.

Our interface provides additional tools to help users fine-tune the sketching results. These tools include a virtual pencil to insert regions with boundaries that do not need to be emphasized with visible pencil strokes, a highlighting tool that lets the user manually add white highlight areas inside a region, and a brush tool useful for fine-tuning automatically placed brush strokes and blank areas.

Region Boundary Smoothing

We have observed that pencil strokes drawn by artists are relatively smooth, but strokes drawn by average users are full of zigzags. To improve their quality, we propose a smoothing scheme that processes the complete boundary of one region at a time rather than smoothing individual strokes.

Once all regions have been detected, our system treats all the strokes lying on region boundaries as a network of boundary curves. These boundary curves seamlessly join together at T-junctions surrounded by three or more regions. T-junctions divide the boundary of every region into segments. Each boundary segment is smoothed with the Gaussian filter, which requires a supporting neighborhood. The supporting neighborhood at a boundary segment's endpoints extends into its neighboring segments.

Note that a boundary segment is typically shared by two adjacent regions, and its neighboring segments are different along the boundary of these two regions. Therefore, the smoothing result of a boundary segment depends on its region membership. To fix this ambiguity, we assign the boundary segment to the larger region of the two. At the beginning of the region-based boundary smoothing step, all the regions are sorted into a decreasing order of their area, and smoothing is applied sequentially to these regions following this order. When this process reaches one region,

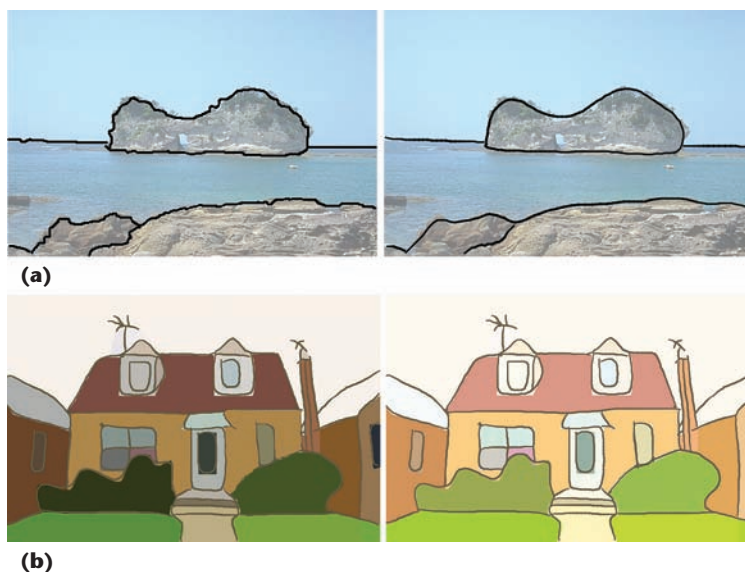


Figure 5. Region boundary smoothing and color stylization: (a) sample pencil strokes with (right) and without (left) smoothing and (b) region color before (left) and after (right) stylization.

we only need to smooth its remaining boundary segments that have not been smoothed earlier. All points on image borders are fixed during smoothing. Figure 5a demonstrates the result before and after boundary smoothing.

Color Stylization

As we discussed earlier, the color of gouache paints used for each region is decided by a base color as well as brush effect rendering, and it deviates significantly from the original pixel colors within the same region. To determine the correct base painting color for every region during interactive sketching, we use a machine-learning approach. For each region, the average pixel color of the pixels inside it and the corresponding painting color determined by artists form a pair.

Given a sufficient number of color pairs collected in this way, we can create a mapping function that converts the original average pixel color to its corresponding painting color. For this mapping function, we collected 35 images and their corresponding color sketches drawn by a few artists. We manually marked 500 pairs of regions in them. We calculated an average color for each region in the HSV color space, resulting in 500 pairs of average colors. By analyzing these color pairs using linear regression, we found that a linear transformation can effectively describe the relationship between the original color and the drawn color.

Following this discovery, we fit a global affine transform connecting an original color R and its corresponding painting color R' : $R' = MR$, where M is the transformation matrix. Noting that the hue channel is periodic on a circle, to make the model

more accurate, given the hue value of a pair training sample, denoted as (h_a, h_b) , if $|h_a - h_b| > 180$, we modified one of the hue value to ensure that the absolute distance is less than 180:

$$\begin{cases} h_a = 360 + h_a & \text{if } h_a \leq h_b \\ h_b = 360 + h_b & \text{if } h_a > h_b. \end{cases} \quad (1)$$

The training dataset follows this matrix:

$$\begin{bmatrix} H(R') \\ S(R') \\ V(R') \\ 1 \end{bmatrix} = \begin{bmatrix} 0.96 & -0.51 & 0.175 & -4.11 \\ -0.02 & 0.53 & -0.11 & 19.09 \\ -0.01 & 0.031 & 0.82 & 41.5 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} H(R) \\ S(R) \\ V(R) \\ 1 \end{bmatrix}.$$

When calculating the transformed color using the learned matrix, if the result of hue value is out of range, it should be moduled by 360 because it is periodic. The example in Figure 5b demonstrates the effectiveness of our color stylization method.

Brush Stroke Placement and Rendering

In a color sketch, every region needs to be completely or partially filled with colored brush strokes. To do so automatically, we developed a two-step method. In the first step, our system determines the center and orientation of every brush stroke. It then determines the actual color of every brush and renders these brushes in the region.

In the first step, we compute a smooth orientation field inside the region and perform anisotropic vector quantization to finalize the center and orientation of all brush strokes within the region. Specifically, we calculate the unit tangent vector at every pixel along the boundary of the region and propagate these tangent vectors toward the interior of the region using a distance field of the boundary and the fast marching method.⁹ That is, each interior pixel receives a unit vector from the nearest boundary pixel. Afterward, we smooth the propagated vector field with a large Gaussian kernel to make the pixel-wise orientations spatially coherent. We noticed that, in some special cases, the direction field of some pixels may become $(0, 0)$ after smoothing. In this case, we simply set them as an initial direction—that is, $(x = 0.57, y = 0.81)$, as suggested by some skilled painters.

Because every straight brush stroke can be approximated as a thin ellipse, to obtain a brush stroke placement scheme within the region, we perform anisotropic vector quantization on top of the orientation field to decompose the region into a number of elongated cells. Our anisotropic vector quantization is based on an anisotropic

distance, which also tries to make the pixel colors within each cell as uniform as possible. Given the location (x_c, y_c) , orientation (u, v) , and color h_c at a pixel C , the anisotropic distance from another pixel $P = (x, y)$ with orientation (u', v') and color h is defined as follows:

$$d^a(P, C) = 0.5 \times \left(\sqrt{\left(\frac{x'}{a}\right)^2 + \left(\frac{y'}{b}\right)^2 + \left(\frac{\|h - h_c\|}{w}\right)^2} + \sqrt{\left(\frac{x''}{a}\right)^2 + \left(\frac{y''}{b}\right)^2 + \left(\frac{\|h - h_c\|}{w}\right)^2} \right), \quad (2)$$

where (x', y') are the coordinates of P in the local frame at C and (x'', y'') are the coordinates of C in the local frame at P . In these two local frames, the x axis is aligned with the direction of (u, v) and (u', v') , respectively; a and b are respectively the length of the major and minor axes of the ellipse approximating the brush strokes; and w is a constant balancing the relative importance between the location and color differences between the two pixels.

In our experiments, w is always set to $10a$. In practice, anisotropic vector quantization is implemented as an iterative process similar to K -means clustering, except that the Euclidean distance in K -means is replaced with the anisotropic distance and the orientation at the center of a cell is taken from the above orientation field. This iterative process gives rise to nonoverlapping elongated cells in the end. The location and orientation at the center of these elongated cells are taken as the center and orientation of the brush strokes (see Figure 6a). The shapes of the brush strokes in our system are taken from a database of 450 colored brush strokes painted by artists, not from the cells in the anisotropic vector quantization.

We can compare our brush placement strategy with one that is commonly adopted in synthetically generated oil paintings, where brush centers are sampled either randomly or over a coarse grid, and orientations orthogonal to image gradients are taken as brush orientations.¹⁰ As Figure 7 shows, our strategy generates more spatially coherent results that are consistent with gouache painting styles.

Our system further determines the actual painting color of every brush according to the color mapping we discussed earlier. To make the brush marks less visible, we need to control the amount of permissible color variation within a region. Let ΔH , ΔS , and ΔV be the maximum variation permitted for the three channels of the HSV

color space respectively. We empirically found that $\Delta H = 8$, $\Delta S = 20$, and $\Delta V = 12$ (out of the largest possible range of 100) yield realistic color variations in colored sketches. Let $(H_{ave}, S_{ave}, V_{ave})$ be the average of the original pixel colors in the entire region and (H_i, S_i, V_i) be the average color of pixels covered by the i th brush stroke. If (H_i, S_i, V_i) is outside the maximum permissible range, we perform additional scaling to suppress the dynamic range of brush colors as follows:

$$H_i^* = \begin{cases} H_{ave} + \frac{H_i - H_{ave}}{H_{ave} - H_{min}} \Delta H & \text{if } H_i < H_{ave} \\ H_{ave} + \frac{H_i - H_{ave}}{H_{max} - H_{ave}} \Delta H & \text{if } H_i > H_{ave} \end{cases} \quad (3)$$

where $H_{min} = \min_i H_i$ and $H_{max} = \max_i H_i$. The same operations are applied to the S and V channels. Because the hue channel is periodic on a circle, the difference between two hue values should be calculated in a recurring mode. Specifically, if $|H_i - H_{ave}| > 180$, they should be updated according to Equation 1 before operating on Equation 3, and if H_i^* is out of range, it should be modulated by 360. After such color adjustment, each new color is mapped to a painting color using the mapping we described earlier. We then search our brush database for a brush stroke with the most similar color (see Figure 6b).

Gouache paints are mostly opaque, so different rendering orders of the same set of brush strokes could give rise to different sketching results. Thus, the final rendering order of previously computed brush strokes becomes important. When artists draw a set of brush strokes to color a region, instead of following a random order, they typically brush one subregion first before moving to the next. Inspired by this observation, we order the brush strokes as follows. We first compute the average orientation of all computed brush strokes within a region. Again, if in some extreme cases, the average orientation becomes $(0, 0)$, we simply set it as an initial direction $(x = 0.57, y = 0.81)$. We then project the centers of all strokes onto the line defined by the average orientation and sort all strokes according to the position of their projections on the line. Finally, all computed brush strokes are rendered following this order (from bottom left to top right projections), and feathering is performed within a narrow band at the boundary of every stroke to make the transition less noticeable. When two strokes overlap each other, the later added stroke simply covers the previous ones.



(a)



(b)

Figure 6. Brush stroke placement and rendering. (a) In the sample anisotropic clustering result in the sky region, the white circles are cluster centers, and the white lines represent their orientations. Cluster centers and orientations are directly taken as brush centers and orientations. (b) These sample brush strokes are from our brush database of 450 colored brush strokes painted by artists.



(a)

(b)

Figure 7. Comparison of brush-placement strategies: (a) brush placement with our anisotropic clustering and (b) brush placement using random locations as brush centers and directions orthogonal to image gradients as brush orientations. Our brush placement strategy generates more spatially coherent results.

Blank Area Creation

Some regions in a color sketch are intentionally left blank to indicate occlusion or highlight particular regions. Blank areas can either appear inside a region or next to a portion of the region boundary. Our system can automatically detect occlusion boundaries and create blank areas next to them. It also provides an interactive tool to let users create a blank area around a highlight in the reference photo. In both scenarios, the actual

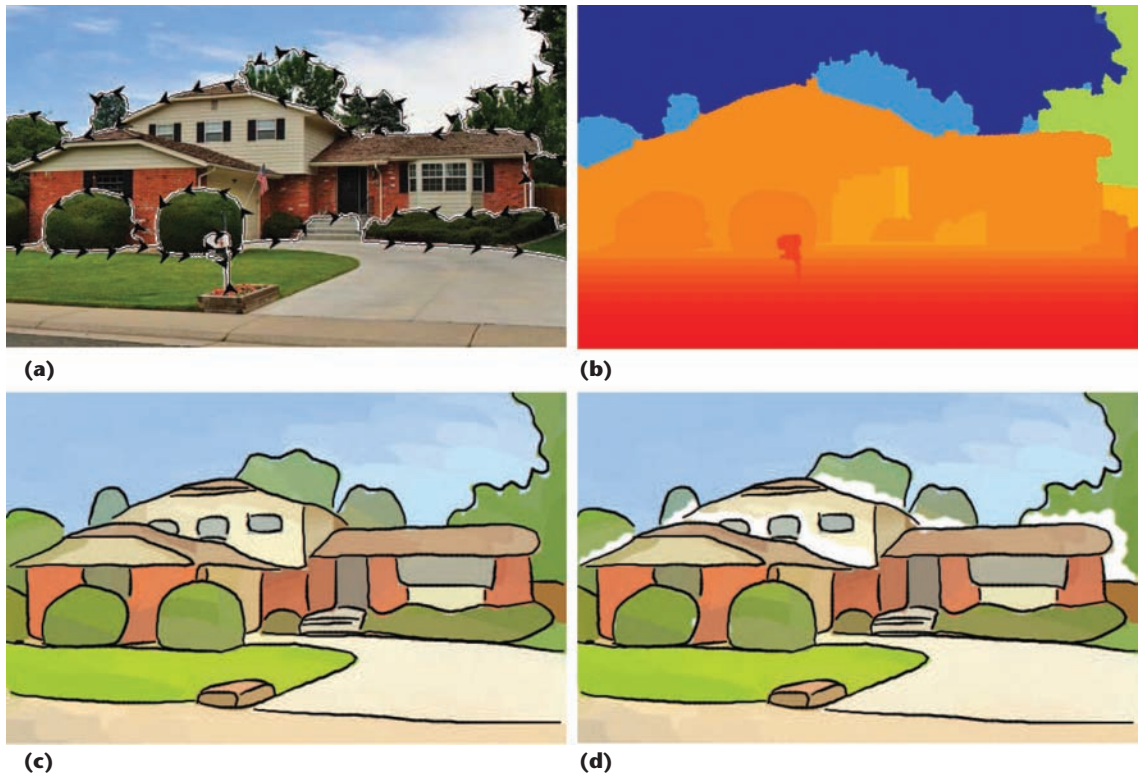


Figure 8. Flow chart for blank area creation: (a) occlusion boundary, (b) depth map, (c) without blank areas, and (d) with blank areas. Blank areas are created in partially occluded regions near depth discontinuities.

shape of the blank areas is automatically determined by the solution of a Poisson equation.

As we discussed earlier, for two adjacent regions, if one region is occluded by the other, artists tend to leave a blank area near the boundary of the occluded region, except for large background regions such as the sky and ocean. To simultaneously create blank areas near occlusion boundaries as the user draws, we use the precalculated depth map to determine occlusion relationships between adjacent regions. When a new pencil stroke is determined to be part of a precomputed occlusion boundary or if there is a depth discontinuity along the shared boundary between two adjacent regions, our system automatically creates a blank area in the partially occluded region near the occlusion boundary. Figure 8 shows an example of this effect produced by our system.

Let S be a region inside the reference photo I . Suppose we are to create a blank area next to a portion of the boundary of S denoted as $\partial S'$. We define a scalar function f over S such that it takes large values along $\partial S'$ and supports a smooth transition to smaller values along $\partial S - \partial S'$. Given f and a threshold τ , S can be easily divided into two subregions. In one of the subregions, f is greater than τ , and it is smaller than τ in the other. The former subregion should be left blank because it is next to $\partial S'$. In practice, we obtain the function f by solving the following equation:

$$\begin{aligned} \Delta f &= 0 \\ \text{s.t. } f(c) &= c & p \in \partial S' \\ f(p) &= I(p) & p \in \partial S - \partial S' \end{aligned} \quad (4)$$

where pixels in $\partial S'$ are set to a large constant c and pixels in $\partial S - \partial S'$ are fixed to their intensity values from the reference photo I . In our experiments, c is always set to five and the threshold τ is typically set to four if pixel intensities $\in [0, 1]$. Equation 4 is actually a special case of the more generic Poisson equation.¹¹ It can be discretized into a sparse linear system that can be solved efficiently.¹²

To create blank areas inside a region to indicate highlights, the process is slightly different. The user can first draw line segments inside the region. Pixels on these line segments are set to a large constant, while all pixels on the region boundary are fixed to their original intensity values. Then an equation similar to Equation 4 can be solved to obtain a scalar function f defined over the region. Finally, all the pixels where f is larger than a threshold τ are left blank, and τ is again set to four in our experiment.

Evaluation

To evaluate the effectiveness and performance of our color sketching system, we conducted a two-stage user study. In the first stage, we asked subjects to produce two color sketches of a given reference image using Photoshop and our Color-Sketch interface. In the second stage, we invited a



Figure 9. Comparison of color sketches generated with Photoshop and our sketching interface during the user study. In each vertical pair, the upper one was created with Photoshop, and the bottom one was created with ColorSketch. From left to right, the first two columns were drawn by skilled painters, and the last two were drawn by novices.

separate group of subjects to evaluate the drawings collected in the first stage.

Drawing Stage

We invited 20 people to participate in the first stage. Ten of them had previously received formal training in drawing and painting, and the remaining subjects were novices who had little drawing experience. For each subject, we selected a reference image and asked him/her to draw two color sketches, one with Photoshop tools such as brushes, pencils, color pickers, the lasso tool, and the quick selection tool and the other one with our ColorSketch interface.

Because the order of the two tools used could influence the user experience and results, we had half of the participants use Photoshop first, and the rest began with ColorSketch. The users were encouraged to do their best, and there were no restrictions on the Photoshop and ColorSketch tools they were allowed to use.

Our study utilized 10 reference images, each of which was used by a skilled painter and a painting layman. The content of the reference images includes human figures, buildings, and natural scenes.

Before the user study, we gave the subjects a tutorial on the two systems and allowed them to practice on a simple image so they could become familiar with both systems. After the training, each participant was given 20 minutes to generate a color sketch with Photoshop and another 20 minutes to generate a second sketch with our system.

Figures 3 and 9 shows sample color sketches drawn with our sketching system. As a comparison, Figure 9 also shows sample color sketches drawn with Photoshop.

Evaluation Stage

The evaluation consisted of two parts. In the first part, we asked the participants from the drawing stage to complete the following short questionnaire:

- Rate the usefulness of our system from 1 to 5, where 1 indicates useless and 5 indicates quite useful.
- Rate our system's ease of use from 1 to 5, where 1 indicates very difficult to use and 5 indicates very easy to use.
- Compare the sketch generated using our system with the one generated in Photoshop, and rate it from 1 to 5, which 1 is much worse and 5 is much better.
- Please provide additional comments.

In the second part, we invited 30 additional subjects to evaluate the color sketches created in the drawing stage. There were 20 pairs of color sketches in total. For each pair (in random order), we placed the two sketches side by side and asked the participants to choose the one that is more visually appealing. We allowed the participants to indicate a tie if they could not decide. Every participant was asked to evaluate all 20 pairs of sketches.

Results

The goal of our participant survey was to evaluate our system in term of its usefulness, usability, and helpfulness. Regarding our system's usefulness and usability, the novices gave a mean usability rating of 4.6/5.0, and the skilled painters gave it a 4.2/5.0. For usefulness, the novices gave a mean rating of 4.1/5.0, and the skilled painters gave it a 3.4/5.0. These ratings indicate all users agreed that

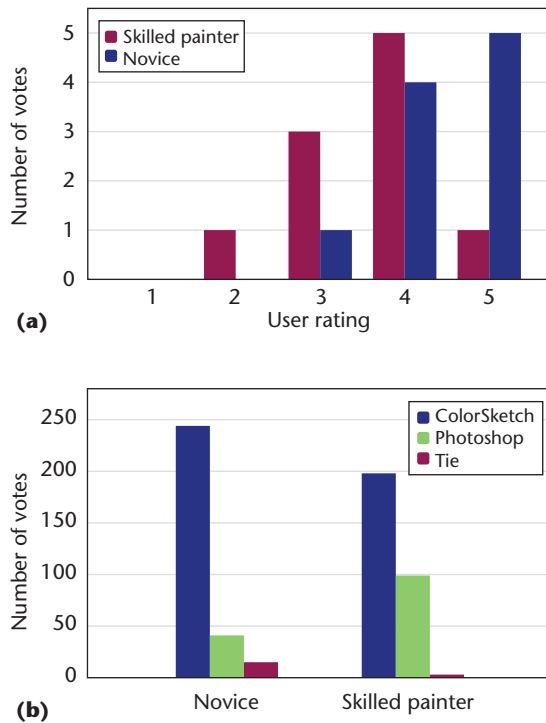


Figure 10. User study results. (a) Users' self-rating of the sketches produced with our system and Photoshop. (b) Total number of votes received by all sketches produced with ColorSketch and Photoshop, respectively.

our system is useful in producing color sketches and it is also easy to use. Novice painters found our system more useful than skilled painters.


Figure 10a presents the participants' self ratings of the sketches produced with our system compared with those produced using Photoshop. We can see that most people, regardless of their painting experience, felt that our system helped them generate the color sketches. In addition, six out of the 10 skilled painters found that they could produce much better results with our system, and 90 percent of the novices claimed that they achieved better results with our system.

In the second part of the evaluation, the participants considered the sketch generated with ColorSketch better than the one generated with Photoshop for seven out of 10 pairs of sketches created by the skilled painters. The percentage is even higher for the sketches produced by the novices, which shows that all users generated better results with our system. Figure 10b shows the total number of supporting votes received by our system. In the group of skilled painters, sketches produced with our system received 65 percent of the votes, and 80 percent of the votes went to our system in the group of novices.

To make a fairer comparison, we also asked eight of the participants (including four novice users

and four skilled painters) to draw a color sketch with CoreDRAW (a vector-based tool). None of the novices thought CoreDRAW was easier to use than Photoshop for color sketch drawing because Photoshop includes tools like the lasso and quick selection, which can help them quickly select a region for editing. Photoshop also has a pen tool to help them efficiently delineate the contour, unlike CoreDRAW. Most of the skilled painters thought both CoreDRAW and Photoshop could be used to create color sketches conveniently because they were skilled in using the Wacom device. Although CoreDRAW can be used to create vectorized images, the comparison is not relevant here because ColorSketch could only create bitmap images.

Overall, our results show that ColorSketch can help users create better color sketches and is considered more helpful for novice users. During the user study, we found that users had difficulty selecting proper colors and brush sizes in Photoshop, which resulted in poor sketches. This was especially true for reference images with rich colors or complex structural details. Choosing the right colors and brush sizes for such images requires much time and painting knowledge. In contrast, with our system, the user only needs to focus on placing pencil strokes along certain region boundaries, without worrying about brush strokes and color stylization. Of course, experienced painters can generate better results in Photoshop if given more time. Nevertheless, all users can save time with our system. According to our observation, it usually takes a novice user about six minutes to generate a color sketch with our system, whereas it takes an experienced painter more than 15 minutes to draw a color sketch with a similar quality.

The ColorSketch system mainly focuses on generating color sketches that imitate the gouache painting style. Utilizing the data-driven methods to generate other styles of paintings is worth exploring in the future. What's more, generalizing this system to give users more artistic freedom (such as more color selections and brush styles, more freedom to draw parts of the objects in an image, and the option to synthesize different parts from multiple images) is another direction in our future research. 

Acknowledgments

Guanbin Li was supported by a Hong Kong Post-graduate Fellowship. This work was also partially

supported by the Hong Kong Research Grants Council under General Research Funds (HKU17209714). Yingqing Xu was supported in part by the National Key Research and Development Program of China under grant 2016YFB1001402 and in part by the National Nature Science Foundation of China (NSFC) under grant 61373072.

References

1. E. Cooper, *California: A Sketchbook*, Chronicle Books, 2000.
2. E. Cooper, *A Year in New York*, City & Co, 1995.
3. Y. Lee, C. Zitnick, and M. Cohen, "ShadowDraw: Real-Time User Guidance for Freehand Drawing," *ACM Trans. Graphics*, vol. 30, no. 4, 2011, article no. 27.
4. Q. Su et al., "EZ-Sketching: Three-Level Optimization for Error-Tolerant Image Tracing," *ACM Trans. Graphics*, vol. 33, no. 4, 2014, article no. 54.
5. P. Arbelaez et al., "Contour Detection and Hierarchical Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, 2011, pp. 898–916.
6. D. Martin et al., "A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," *Proc. Int'l Conf. Computer Vision (ICCV)*, 2001, pp. 416–423.
7. D.G. Lowe, "Organization of Smooth Image Curves at Multiple Scales," *Int'l J. Computer Vision*, vol. 3, no. 2, 1989, pp. 119–130.
8. D. Hoiem et al., "Recovering Occlusion Boundaries from a Single Image," *Proc. 11th Int'l Conf. Computer Vision (ICCV)*, 2007, pp. 1–8.
9. J. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge Univ. Press, 1999.
10. K. Zeng et al., "From Image Parsing to Painterly Rendering," *ACM Trans. Graphics*, vol. 29, no. 1, 2009, article no. 2.
11. P. Pérez, M. Gangnet, and A. Blake, "Poisson Image Editing," *ACM Trans. Graphics*, vol. 22, 2003, pp. 313–318.
12. S. Toledo, V. Rotkin, and D. Chen, "Taucs: A Library of Sparse Linear Solvers, Version 2.2," tech report, Tel-Aviv Univ., 2003.

Guanbin Li is a PhD candidate in the Department of Computer Science at the University of Hong Kong. His research interests include computer vision, image processing, and deep machine learning. Li has an MS in computer science from Sun-Yat Sen University. He is a recipient of a Hong Kong Postgraduate Fellowship. Contact him at gbli@cs.hku.hk.

Sai Bi is a research assistant at the University of Hong Kong. His research interests include computer graphics and computer vision. Bi has a BE in computer science from the University of Hong Kong. He is a recipient of the HK-SAR Government Scholarship, HKU Foundation Scholarship, and HKU Undergraduate Research Fellowship. Contact him at fsbi@cs.hku.hk.

Jue Wang is a principal scientist at Adobe Research. His research interests include image and video processing and computational photography. Wang has a PhD in electrical engineering from the University of Washington. He received the Microsoft Research Fellowship and the Yang Research Award from the University of Washington. He is a senior member of IEEE and a member of the ACM. Contact him at juewang@adobe.com.

Yingqing Xu is a Cheung Kong Scholar Chair Professor at Tsinghua University. His research interests include human-computer interaction, computer graphics, and e-heritage. Xu has a PhD in computer graphics from the Chinese Academy of Sciences. He is a distinguished member of the China Computer Federation and a member of the ACM, Siggraph, and Chinese Artists Association. Contact him at yqxu@tsinghua.edu.cn.

Yizhou Yu is a professor in the College of Computer Science and Technology at Zhejiang University. His research interests include computer graphics, computer vision, digital geometry processing, video analytics, and biomedical data analysis. Yu has a PhD from the University of California, Berkeley. He is a recipient of the US National Science Foundation Career Award, Microsoft Graduate Fellowship, and ACM SIGGRAPH/EG Symposium on Computer Animation Best Paper Award. He is on the editorial board of International Journal of Software and Informatics and a senior member of IEEE. Contact him at yizhouy@acm.org.

myCS

Read your subscriptions through the myCS publications portal at <http://mycs.computer.org>.