ColorSketch: A Drawing Assistant for Generating Color Sketches from Photos

Guanbin Li, Sai Bi, Jue Wang, Yingqing Xu, Yizhou Yu

***Abstract*—A color sketch creates a vivid depiction of a scene using sparse pencil strokes and casual colored brush strokes. In this paper, we introduce an interactive drawing system, called ColorSketch, for helping novice users generate color sketches from photos. Our system is motivated by the fact that novice users are often capable of tracing object boundaries using pencil strokes, but have difficulties to choose proper colors and brush over an image region in a visually pleasing way. To preserve artistic freedom and expressiveness, our system lets users have full control over pencil strokes for depicting object shapes and geometric details at an appropriate level of abstraction, and automatically augment pencil sketches using color brushes, such as color mapping, brush stroke rendering as well as blank area creation. Experimental and user study results demonstrate that users, especially novice ones, can generate much better color sketches more efficiently with our system than using traditional manual tools.**

***Index Terms*—Region Detection, Color Stylization, Brush Layout, Blank Area Creation**

I. Introduction

Color sketching is a widely adopted sketching style among artists, which attempts to augment sparse pencil strokes with casual colored brush strokes using gouache paints. For example, Elisha Cooper, a well-known writer and children's book author, has published multiple sketchbooks recording his visits to places such as California [1] and New York City [2]. As shown in Fig. 2, the colored brush strokes in such sketches provide rich visual information and create a vivid impression of the objects and scenes they depict. Gouache paints generate less visible brush marks than oil paintings, and are also less fluid than watercolors. We call this sketching style *color sketching*. Serving a similar purpose as pencil sketching, color sketching typically abstracts away many details while preserving the painter's own style and artistic choices.

In this paper, we study the problem of generating a color sketch by abstracting from a real photograph. Many systems have been proposed to help users draw pencil sketch lines using the under painting technique [3], [4]. However, even with good sketch lines, novice users still face many issues when creating a color sketch, such as which gouache paints should be used for approximating the pixel colors in the photo, and which sub-regions should be intentionally left blank while the others are filled with colored brush strokes.

Guanbin Li and Sai Bi are with the Department of Computer Science, The University of Hong Kong. E-mail: gbli@cs.hku.hk, fsbi@cs.hku.hk. Jue Wang is with Adobe Research. E-mail: juewang@adobe.com. Yingqing Xu is with the Department of Information Art and Design, Tsinghua University. E-mail: yqxu@tsinghua.edu.cn. Yizhou Yu is with Zhejiang University. E-mail: yizhouy@acm.org.

To help novice users overcome the above difficulties, we present 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 the user's artistic freedom and expressiveness, our system let the user have complete control over pencil strokes, including their location, shape, drawing order and the 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 existing pencil sketches. In this way, color sketches created through our interface is a mix of both user-provided abstraction and computer-generated stylization.

Given an input photo, producing a color sketch using our system consists of two stages, an offline automatic pre-processing stage, and an online drawing stage carried out through 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 wherever desirable, and our system provides realtime "auto-completion" suggestions 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 through sketching, and automatically add 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.

Our main contribution is an intelligent drawing interface tailored for color sketches. It provides two types of user assistance during a drawing session: (1) "auto-completion" suggestions for drawing pencil strokes; and (2) automatic color region stylization that is guided by the pencil strokes. 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, based on the "membrane"

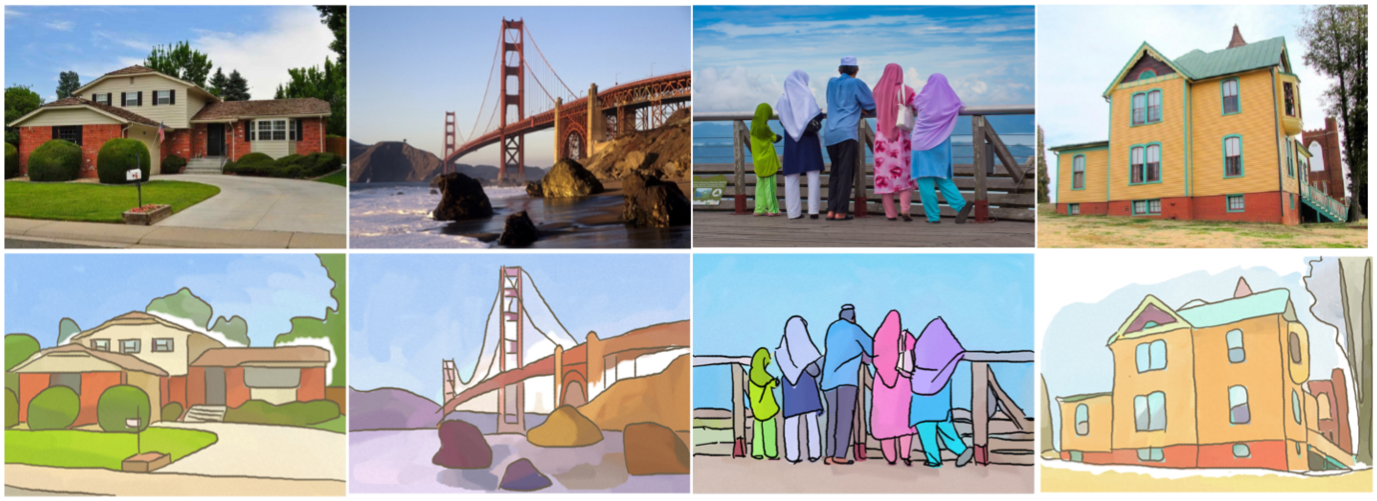


Fig. 1: Sample color sketches generated with our ColorSketch interface. The top row shows reference images. The first two sketches in the bottom row were drawn by novice painters while the last two sketches were drawn by skilled painters.

equation, generates results resembling those drawn by artists.

We have 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.

II. Background and Related Work

A. Related Work

1) Automatic image stylization: Automatically converting an image into a stylized rendering has been extensively studied in Non-Photorealistic Rendering (NPR). Various systems have been proposed which aim at different styles, such as pencil and color pencil drawing [5], pen and ink illustration [6], oil painting [7], and watercolor [8]. These approaches aim at automatically generating high quality rendering results, with little or no user interaction. In contrast, our system is designed to be a drawing assistance tool that supports immense user interaction.

2) Sketching interfaces: Various interactive systems have been proposed to help users draw sketches or create digital art, either for specific objects such as human faces [9] or 3D models [10], or more general cases [11], [12]. Our work was partially inspired by ShadowDraw [3], a system for guiding freeform sketching. As the user draws, their system dynamically updates a shadow image underlying the user's strokes as a guidance for drawing object contours. The shadow image is generated by searching through a large image dataset. Gingold *et al.* [13] and Limpaecher *et al.* [14] explored using crowdsourcing for generating sketch guidance. In contrast, our system employs a novel, auto-completion mechanism for assisting sketching object contours, using a single reference image.

Recently, Iarussi *et al.* [11] presented a drawing tool which provided automated guidance over model photographs to help people practice traditional drawing-by-observation techniques. Su *et al.* [4] proposed EZ-Sketch, a system for helping users draw accurate pencil sketches over an image. It employs a multi-level optimization framework to adjust the positions of user strokes for improved accuracy. Our system adopts a different strategy for helping users draw more accurately and efficiently, by providing online auto-completion guidance. Furthermore, our system focuses mainly on a different problem of generating stylized color regions given the sketch lines.

3) Region-based stylization: DeCarlo and Santella [15] introduced a computational approach to stylizing and abstracting photographs. Their system uses mean shift to segment the image into coherent regions, and transforms them into a style that features bold edges and constant-color regions. Wang *et al.* [16] 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. However, the advocated style in these approaches is different 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 regions. In comparison, our system performs color stylization which maps pixel colors to more meaningful painting colors.

The style adopted by Wen *et al.* [17] is perhaps most similar to the color sketching style proposed in this paper. Nevertheless, one obvious difference is that they do not paint brush strokes within regions. We also introduce a more systematic approach that creates blank areas near occlusion boundaries and highlight regions. More importantly, their system produces results in a semi-automatic way, while our system is a drawing assistant that gives the user artistic freedom and control.

B. Background

We distilled a few principles from sketchbooks [1], [2] and color sketches drawn by artists (Fig. 2). These principles are

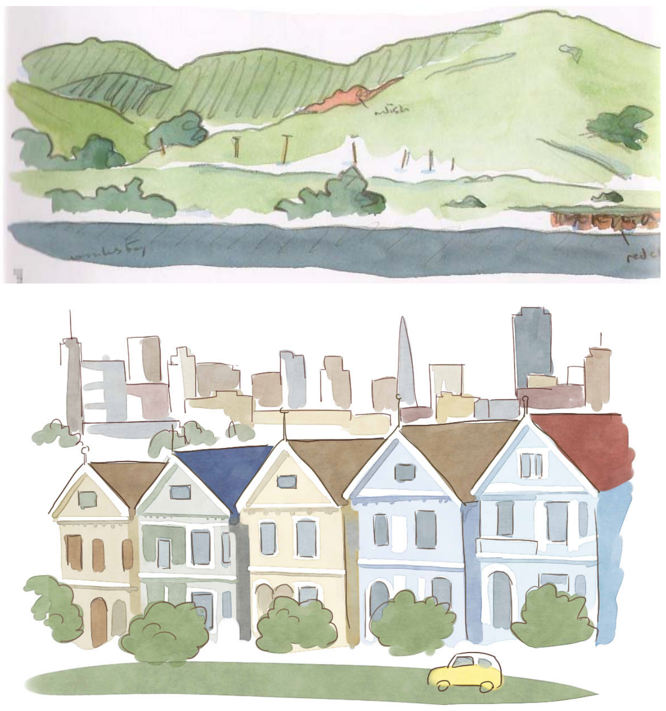


Fig. 2: Examples of color sketches manually drawn by artists.

suited for adoption in our computer-assisted color sketching system.

1) Pencil strokes delineate object contours as well as 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.
2) Regions are brushed with gouache paints instead of being filled with solid colors. The size of brush strokes varies with the size of the region. 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.
3) 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.
4) Many regions are not completely filled with brush strokes. Instead, some of them have border areas intentionally left blank. Blank areas increase spatial non-uniformity and also give a pleasant look. They are primarily used for indicating highlight areas, as well as areas near occlusion boundaries on partially occluded objects or regions.

III. System Overview

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, as discussed in the next section.

During the online stage, as shown in Fig. 3b, the user draws sparse pencil strokes to delineate the shape of objects using the under painting technique (i.e. drawing on a semi-transparent canvas over the reference photo). 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 during the preprocessing stage as a suggestion near the stroke being drawn by the user (Fig. 3e). 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 (Fig. 3c). It further computes the color of the gouache paints that should be applied in each region (Fig. 3d), the layout and the rendering order of the brush strokes that will be used for filling it (Fig. 3g). Our system also determines the areas that should be left blank according to occlusion boundaries detected in the preprocessing stage (Fig. 3f).

IV. Preprocessing

A. Region Extraction

Given an input image, our system first scales it to have length of longer side equals to 400 pixels and automatically segments it into coherent regions. The boundaries of these regions are used to generate auto-completion suggestions in the online drawing stage. The precomputed regions are obtained using a hierarchical segmentation algorithm [18]. We chose this algorithm because of its high accuracy on the Berkeley Segmentation Benchmark [19] 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 [18] and extract region boundaries at three levels according to three different threshold (0.15, 0.30, 0.50 respectively) in the generated UCM map, as shown in Fig. 4. The extracted boundaries are further divided into segments at T-junctions and local maxima of curvature [20].

B. Occlusion Boundary Detection

As mentioned in Section II-B, artists tend to leave blank areas near boundaries of occluded areas. To enable the quick handling of occlusion boundaries in an interactive drawing session, we pre-compute them and a depth map from the

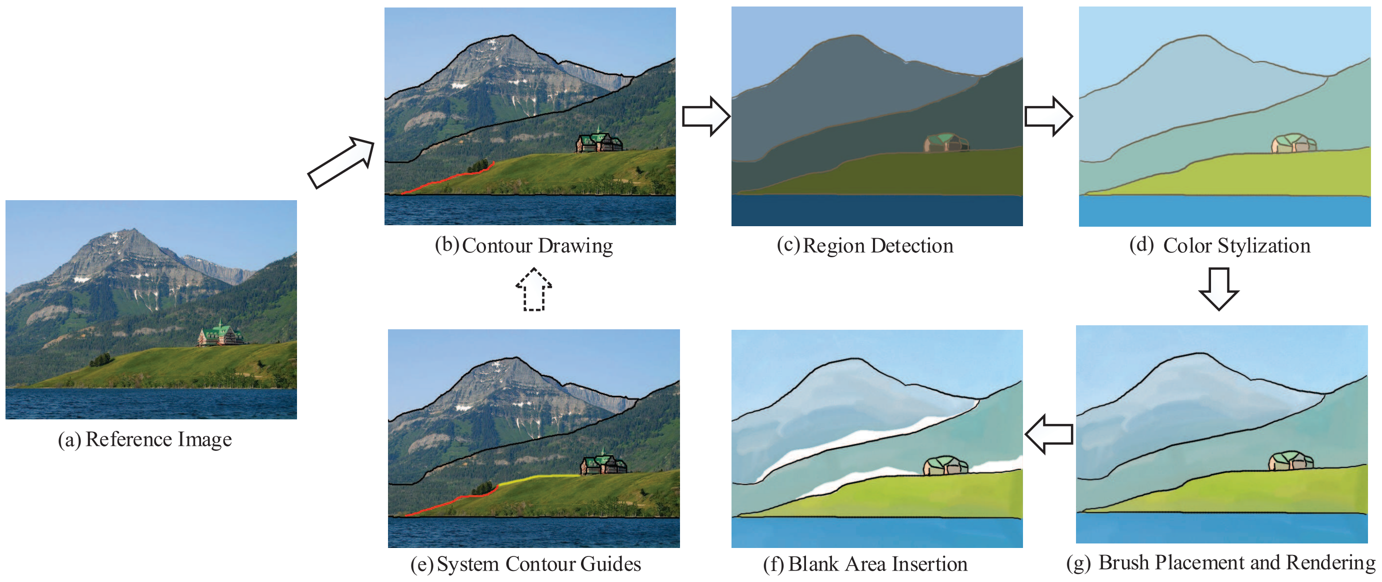


Fig. 3: The work flow of our ColorSketch system

input image using a scene-based occlusion reasoning method proposed in [21] with the source code and parameters setting provided by the author. During an interactive session, when the user adds a new pencil stroke, if it is part of a pre-computed 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 [21].

V. Interactive Color Sketching

A. User Interface

The user interface of our system consists of a tool bar and two windows placed side-by-side. The left window is used for drawing, it shows the reference photo overlaid with a translucent canvas. The right window is the output window, it displays the automatically completed color sketching result and update it in realtime. 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, with the guidance of the underneath reference photo. When a stroke is being drawn by the user, our system continuously analyzes it in the background, searching through a set of automatically extracted strokes during preprocessing for finding the best candidate stroke that the user is most likely meant to draw. This candidate stroke is then shown as a shadow on the canvas, and can be directly accepted into the output window by a keyboard event triggered by the user. Accepted shadow strokes usually are more accurately positioned than the user's unfinished strokes. In this way, our system can greatly improve the user's drawing efficiency. In case there are multiple candidate strokes near the unfinished stroke, one of them is chosen 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 0.5; (2) if more than one candidate strokes exist under rule (1), the amount of pixels overlap between the two strokes is considered. If all of the candidate strokes in this stage have overlapping pixels less or equal 50, choose the one with the largest overlap, otherwise choose the one according to rule (3); (3) the total length of the candidate stroke (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 close shape, we implicitly join nearby strokes by connecting 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 (8 pixel 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 realtime 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 whose boundaries 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 also a brush tool useful for fine-tuning automatically placed brush strokes and blank areas.

B. Region Boundary Smoothing

We have observed that pencil strokes drawn by artists are relatively smooth, while 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 the endpoints of a boundary segment 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 is dependent 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, we only need to smooth its remaining boundary segments that have not been smoothed earlier. Note that all points on image borders are fixed during smoothing. Fig. 5a demonstrates the result before and after boundary smoothing.

C. Color Stylization

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

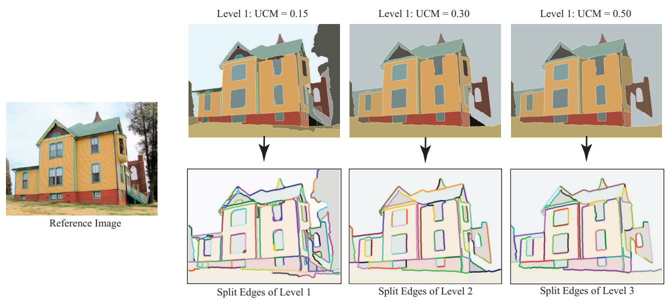
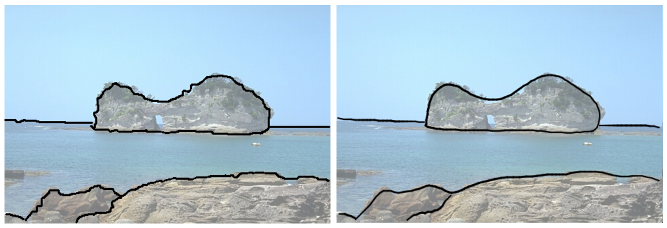
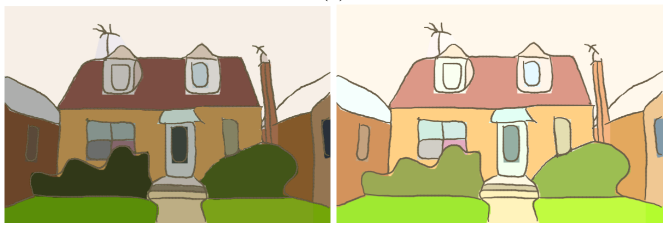


Fig. 4: An example of hierarchical image segmentation and boundary segment (stroke) generation. Segmentations at three different levels of details are extracted from the hierarchical segmentation result, and region boundaries therein are further divided into segments at T-junctions and local maxima of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



(a)



(b)

Fig. 5: **Top:** Sample pencil strokes with (right) and without (left) smoothing. **Bottom:** Region color before (left) and after (right) stylization.

Given a sufficient number of color pairs collected in this way, we can learn a mapping function that converts the original average pixel color to its corresponding painting color.

To learn 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 find that a linear transformation can describe the relationship between the original color and the drawn color well. 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. Noted 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 assure that the absolute distance is less than 180 by applying the equation as follows,

$$\begin{cases} h_a = 360 + h_a & \text{if } h_a \leq h_b \\ h_b = 360 + h_b & \text{if } h_a > h_b \end{cases} \tag{1}$$

.

For our training dataset, the learned matrix is as follows:

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

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

D. Brush Stroke Placement and Rendering

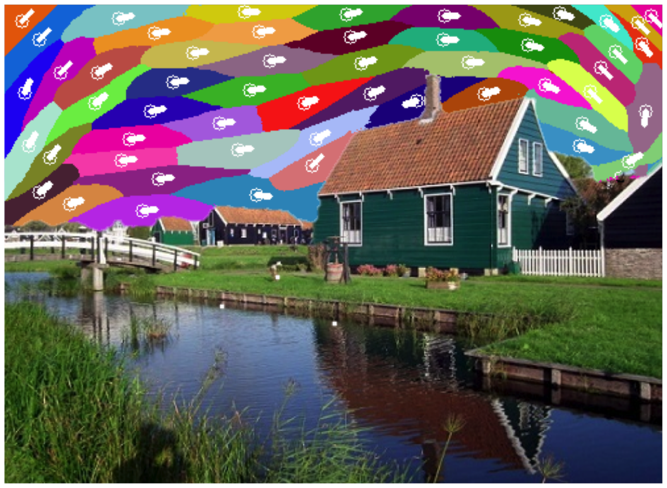
In a color sketch, every region needs to be completely or partially filled with colored brush strokes. To carry out this automatically, we have developed a method that consists of two steps. In the first step, our system determines the center and orientation of every brush stroke. It then determines 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 performs 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 towards the interior of the region using a distance field of the boundary and the fast marching method [22]. That is, each interior pixel receives a unit vector from the nearest boundary pixel. Afterwards, 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 cases, we simply set them as an initial direction (i.e.$(x = 0.57, y = 0.81)$) as suggested by some skilled painters.

Since 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 above 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 $\mathbf{h_c}$ at a pixel C, the anisotropic distance from another pixel $P = (x, y)$ with orientation (u', v') and color $\mathbf{h}$ is defined as follows.

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

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 above 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, as shown in Fig. 6a. Note that we have a database of 450 colored brush strokes painted by artists, and the shape of brush strokes in our system is taken from this database, not from the cells in the anisotropic vector quantization.


(a)

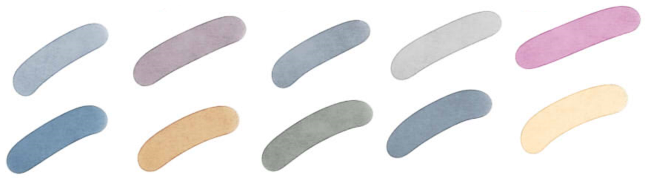
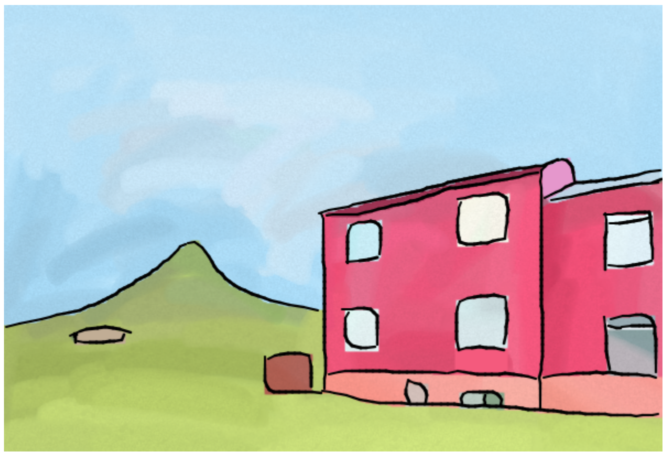

(b)

Fig. 6: **Top:** 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. **Bottom:** Sample brush strokes in our brush database.

We 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 [7]. As shown in Fig. 7, 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 learned in the previous subsection. To make the brush marks less visible, as discussed in Section II-B, we need to control the amount of permissible color variation within a region. Let ΔH, ΔS and ΔV be respectively the maximum variation permitted for the three channels of the HSV color space. We empirically found $\Delta H = 8$, $\Delta S = 20$ and $\Delta V = 12$ (out of the largest possible range of 100) can give rise to 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) 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


(a) Brush placement with anisotropic clustering

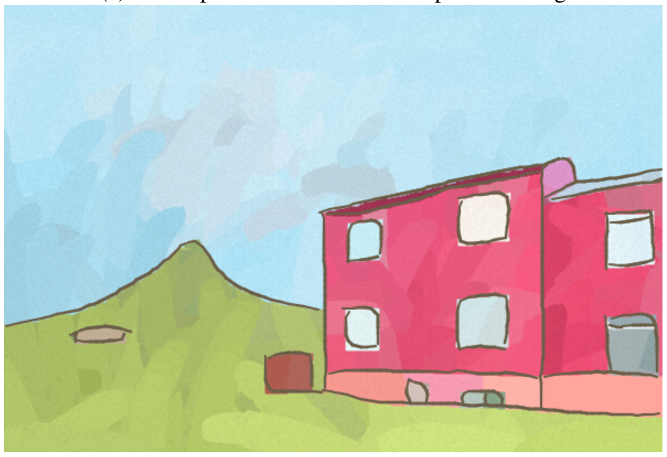

(b) Brush placement using random locations and image gradients

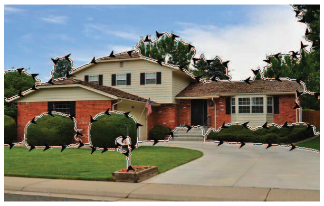
Fig. 7: A comparison of brush placement strategies. **Top:** Brush placement with our anisotropic clustering, **Bottom:** Brush placement using random locations as brush centers and directions orthogonal to image gradients as brush orientations. Our brush placement strategy generates more spatially coherent results.

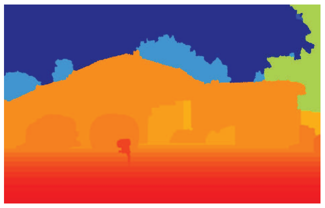
brush colors as follows:

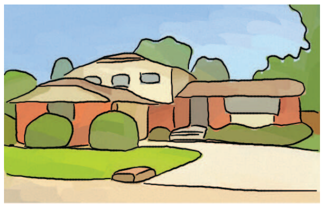
$$H_i^* = \begin{cases} H_{ave} + \frac{H_i - H_{ave}}{H_{ave} - H_{min}} \Delta H, & \text{if } H_i < H_{ave}; \\ H_{ave} + \frac{H_i - H_{ave}}{H_{max} - H_{ave}} \Delta H, & \text{if } H_i > H_{ave}, \end{cases} \quad (3)$$

where $H_{min} = \min_i H_i$ and $H_{max} = \max_i H_i$. Same operations are applied to the S and V channels. Noted that the hue channel is periodic on a circle, the difference between two hue value 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 moduloed by 360. After such color adjustment, each new color is mapped to a painting color using the mapping learned in Section V-C. We then search our brush database for a brush stroke with the most similar color (Fig. 6b).

Since gouache paints are mostly opaque, different rendering orders of the same set of brush strokes could give rise to very 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 (i.e.$(x = 0.57, y = 0.81)$) as suggested by some skilled painters. 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 projections to top right ones), 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) Occlusion Boundary


(b) Depth Map


(c) Without Blank Areas

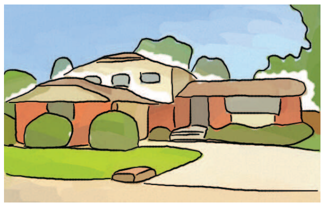

(d) With Blank Areas

Fig. 8: Flow chart for blank area creation. Blank areas are created in partially occluded regions near depth discontinuities.

E. Blank Area Creation

As discussed in Section II-B, some regions in a color sketch are not completely filled with colored brush strokes. Certain areas in these regions are intentionally left blank for various purposes, such as indicating occlusion or highlight. 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 shape of the blank areas is automatically determined by the solution of a Poisson equation.

As we have discussed in Section IV-B, 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 make use of the pre-calculated 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. Fig. 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 $\partial S'$. Our idea is to 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. f is greater than τ in one of the subregions and smaller than τ in the other. The former subregion should be left blank since it is next to $\partial S'$. In practice, we obtain the function f by solving the following "membrane" equation:

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

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 5 and the threshold τ is typically set to 4 if pixel intensities $\in [0, 1]$. The above equation is actually a special case of the more generic Poisson equation [23]. It can be discretized into a sparse linear system that can be solved very efficiently [24].

To create blank areas inside a region to indicate highlights, the process is slightly different. The user can first draw line segments at desired locations 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 (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, where τ is again set to 4 in our experiment.

VI. Evaluation

To evaluate the effectiveness and performance of our color sketching system, we have conducted a two-stage user study. In the first stage, subjects were required to produce two color sketches of a given reference image using Photoshop and our ColorSketch interface. In the second stage, a separate group of subjects were invited to evaluate the drawings collected in the first stage.

a) Drawing Stage: We invited 20 people to participate in the first stage of the user study. 10 of them have previously received formal training in drawing and painting, and the remaining subjects are novice users that have little experience in drawing. 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, lasso tool, quick selection tool, etc., and the other one with our ColorSketch interface. Considering that the order of the two tools used may influence the user experience and results, we let half of the participants used Photoshop first and the rest of them began with ColorSketch. The users were encouraged to do the best as they can without any restriction on choosing the tools in both Photoshop and ColorSketch. We used 10 reference images, each of which was used by a skilled painter and a painting layman. The content of the reference images include 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 to get familiar with both systems. After 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. Sample color sketches drawn with our sketching system can be found in Figs. 1 and 10. As a comparison, sample color sketches drawn with Photoshop are also shown in Fig. 10.

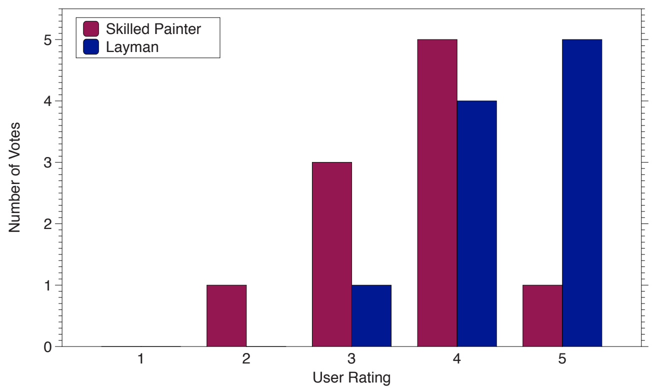
b) Evaluation Stage: The evaluation consists of two parts. In the first part, we asked the participants who had participated in the drawing stage to complete a short questionnaire that has four questions:

1) Rate the usefulness of our system with numbers from 1 to 5, where 1 means useless and 5 means quite useful;
2) Rate whether our system is easy to use with numbers 1 to 5. 1 means very difficult to use, and 5 means very easy to use;
3) Compare the sketch generated using our system with the one generated in Photoshop, and rate it with numbers from 1 to 5, 1 for much worse, 5 for much better;
4) Additional comments.

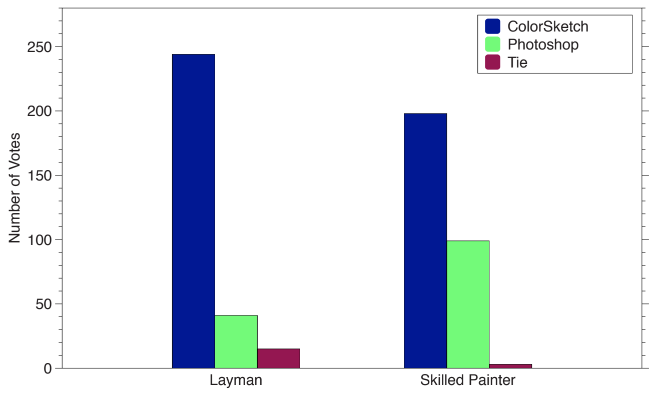
In general, through this survey, we would like to know whether our system is easier to use and whether it can help users create better color sketches compared with traditional fully-manual tools.

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 we put the two sketches side-by-side in random order, and asked the participants to choose one that is visually more appealing. And if the subject could not decide, he/she could choose a "Tie" option. Every participant was asked to evaluate all 20 pairs of sketches.

c) Results: We first present the survey results in the first part of the evaluation. We have evaluated our system in term of usefulness, usability, as well as its helpfulness. Regarding the usefulness and usability of our system, the mean usability rating given by novice users is 4.6/5.0 while the same rating given by skilled painters is 4.2/5.0; and the mean usefulness rating given by novice users is 4.1/5.0 while the same rating given by skilled painters is 3.4/5.0. These ratings indicate all users agree our system is useful in producing color sketches and it is also easy to use. In particular, novice painters found our system more useful than skilled painters.



(a) Users' self-rating on the helpfulness of ColorSketch



(b) #votes received by all sketches generated with ColorSketch and Photoshop

Fig. 9: User study results. (a) Users' self-rating of the sketches produced with our system in comparison with those produced with Photoshop. (b) Total number of votes received by all sketches produced with ColorSketch and Photoshop, respectively.

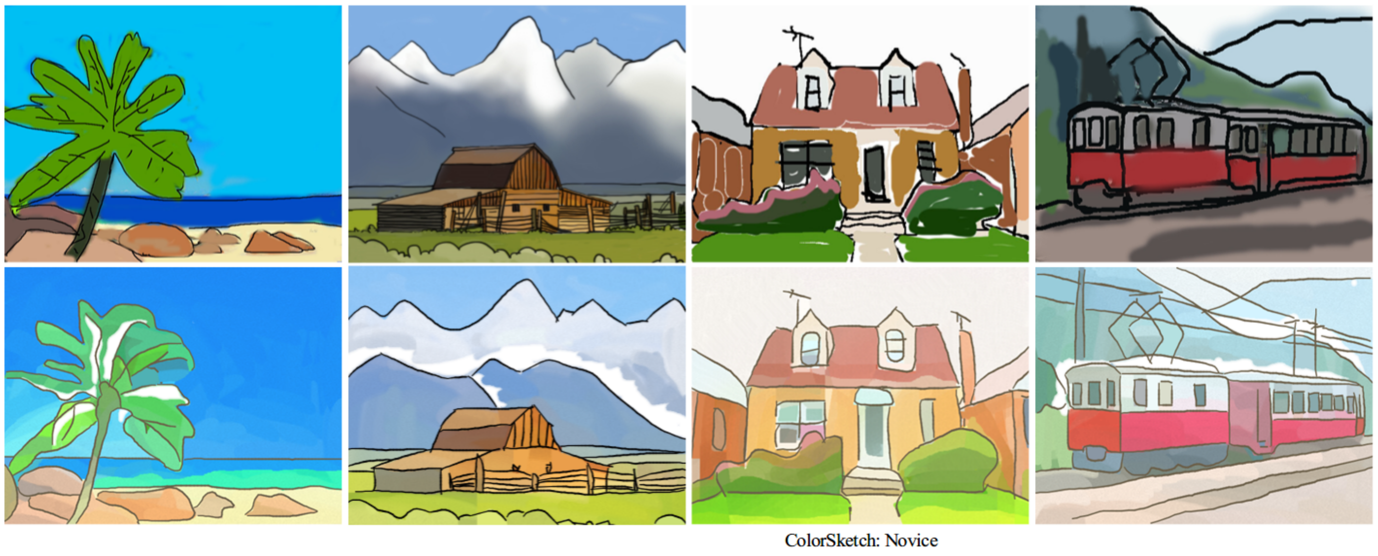


Fig. 10: A visual comparison of color sketches generated with Photoshop and our sketching interface during the user study. In each vertical pair, the upper one was created with Photoshop, and the bottom one was created with ColorSketch. From left to right, the first two columns were drawn by skilled painters while the last two were drawn by novice users.

Fig. 9a presents the self ratings of the sketches produced with our system in comparison with the sketches produced with Photoshop. We can see that most people, regardless of skilled painters or painting laymen, found our system helpful in assisting them generating color sketches. In addition, 6 out of 10 skilled painters found that they could produce much better results with our system. This percentage is even higher among painting laymen. That is, 90 percent of them claimed that better results were achieved with our system.

Next we report the user study results in the second part of the evaluation. In 7 out of 10 pairs of color sketches created by skilled painters, the one generated with ColorSketch is considered better than the one generated with Photoshop. The percentage is even higher for sketches produced by painting laymen, which shows that all users have generated better results with our system. Fig. 9b 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, while 80 percent of the votes went to our system in the group of painting laymen.

To make a fairer comparison, we also asked 8 of the participants (including 4 laymen and 4 skilled painters) to draw a color sketch with CoreDRAW (vector based tool). All painting laymen did not think CoreDRAW easier to use than Photoshop for color sketches drawing since Photoshop has some tools like "Lasso Tool" and "Quick selection Tool" which can help them to quickly select a region for editing (adding color and brushes) and Photoshop also has a pen tool to help them efficiently delineate the contour as CoreDRAW provides. Most of the skilled painters thought both CoreDRAW and Photoshop could be used to create color sketches very conveniently since they were skilled in using the Wacom. Though CoreDRAW can be used to create vectorized images, it is not necessary here for comparison since our ColorSketch could only create bitmap images.

Overall, ColorSketch is shown to be able to assist users to obtain better color sketches, and it is considered to be more helpful for novice users. During the user study, we found that users had difficulty in selecting proper colors and brush sizes in Photoshop, which resulted in poor sketches. This is 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 6 minutes to generate a color sketch with our system, however, it takes an experienced painter more than 15 minutes to draw a color sketch with a similar quality.

VII. Conclusion and Future Work

We presented an intelligent interface that can help users draw color sketches more efficiently. Our interface provides two unique features: (1) a realtime auto-completion suggestion for drawing pencil strokes along object boundaries; and (2) a data-driven approach for rendering colored brush strokes inside image regions, with proper color conversion and blank area generation. User study results suggest that users, especially novice ones, can generate much better color sketches more efficiently with our system than using traditional manual tools.

Our system mainly focused on generating color sketches imitating the style of gouache paints. Utilizing the data-driven methods to generate other style paintings is also worth exploring in the future. What's more, generalizing this system to give users more artistic freedom (i.e. more color selection and brush style, drawing parts of the objects in an image or synthesizing different parts from multiple images) is another direction in our future research.

Acknowledgements

The first author was supported by Hong Kong Postgraduate Fellowship. This work was partially supported by Hong Kong Research Grants Council under General Research Funds (HKU17209714). Ying-Qing Xu was supported in part by the National Basic Research Program of China under Grant 2012CB725300 and in part by 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 Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 27, 2011.

[4] Q. Su, W. H. Li, J. Wang, and H. Fu, "Ez-sketching: Hierarchical optimization for error-tolerant image tracing," *ACM Trans. Graph.*, vol. 21, no. 3, 2014.

[5] J. J. Cewu Lu, Li Xu, "Combining sketch and tone for pencil drawing production," in *International Symposium on Non-Photorealistic Animation and Rendering*, 2012.

[6] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin, "Orientable textures for image-based pen-and-ink illustration," in *Proc. SIGGRAPH*, 1997, pp. 401–406.

[7] K. Zeng, M. Zhao, C. Xiong, and S.-C. Zhu, "From image parsing to painterly rendering," *ACM Trans. Graph.*, vol. 29, no. 1, pp. 2:1–2:11, Dec. 2009. [Online]. Available: http://doi.acm.org/10.1145/1640443.1640445

[8] A. Bousseau, M. Kaplan, J. Thollot, and F. X. Sillion, "Interactive watercolor rendering with temporal coherence and abstraction," in *Proc. NPAR*, 2006, pp. 141–149.

[9] D. Dixon, M. Prasad, and T. Hammond, "icandraw: Using sketch recognition and corrective feedback to assist a user in drawing human faces," in *Proce. SIGCHI*, 2010, pp. 897–906.

[10] S. Grabli, E. Turquin, F. Durand, and F. X. Sillion, "Programmable rendering of line drawing from 3d scenes," *ACM Trans. Graph.*, vol. 29, no. 2, 2010.

[11] E. Iarussi, A. Bousseau, and T. Tsandilas, "The drawing assistant: Automated drawing guidance and feedback from photographs," in *ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 2013.

[12] L. Benedetti, H. Winnemöller, M. Corsini, and R. Scopigno, "Painting with bob: assisted creativity for novices," in *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 2014, pp. 419–428.

[13] Y. Gingold, E. Vouga, E. Grinspun, and H. Hirsh, "Diamonds from the rough: Improving drawing, painting, and singing via crowdsourcing," in *Proceedings of the AAAI Workshop on Human Computation (HCOMP)*, 2012.

[14] A. Limpaecher, N. Feltman, A. Treuille, and M. Cohen, "Real-time drawing assistance through crowdsourcing," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 54:1–54:8, 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2462016

[15] D. DeCarlo and A. Santella, "Stylization and abstraction of photographs," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 769–776, 2002.

[16] J. Wang, Y.-Q. Xu, H.-Y. Shum, and M. Cohen, "Video tooning," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 574–583, 2004.

[17] F. Wen, Q. Luan, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Color sketch generation," in *The 4th international symposium on Non-photorealistic animation and rendering*, 2006, pp. 47–54.

[18] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.

[19] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. ICCV*, 2001, pp. 416–423.

[20] D. G. Lowe, "Organization of smooth image curves at multiple scales," *International Journal of Computer Vision*, vol. 3, no. 2, pp. 119–130, 1989.

[21] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *11th International Conference on Computer Vision*, ser. ICCV, 2007, pp. 1–8.

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

[23] P. Pérez, M. Gangnet, and A. Blake, “Poisson image editing,” *ACM Transactions on Graphics (TOG)*, vol. 22, pp. 313–318, 2003.

[24] S. Toledo, V. Rotkin, and D. Chen, “Taucs:a library of sparse linear solvers. version 2.2,” Tel-Aviv University, Tech. Rep., 2003.