Fast Accurate and Automatic Brushstroke Extraction

YUNFEI FU, iArt.ai, China
HONGCHUAN YU, National Centre for Computer Animation, Bournemouth University, UK
CHIH-KUO YEH, School of Computer Science and Software, Zhaoqing University, China
TONG-YEE LEE, Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan
JIAN J. ZHANG, National Centre for Computer Animation, Bournemouth University, UK

Brushstrokes are viewed as the artist's "handwriting" in a painting. In many applications such as style learning and transfer, mimicking painting, and painting authentication, it is highly desired to quantitatively and accurately identify brushstroke characteristics from old masters' pieces using computer programs. However, due to the nature of hundreds or thousands of intermingling brushstrokes in the painting, it still remains challenging. This article proposes an efficient algorithm for brush Stroke extraction based on a Deep neural network, i.e., DStroke. Compared to the state-of-the-art research, the main merit of the proposed DStroke is to automatically and rapidly extract brushstrokes from a painting without manual annotation, while accurately approximating the real brushstrokes with high reliability. Herein, recovering the faithful soft transitions between brushstrokes is often ignored by the other methods. In fact, the details of brushstrokes in a master piece of painting (e.g., shapes, colors, texture, overlaps) are highly desired by artists since they hold promise to enhance and extend the artists' powers, just like microscopes extend biologists' powers. To demonstrate the high efficiency of the proposed DStroke, we perform it on a set of real scans of paintings and a set of synthetic paintings, respectively. Experiments show that the proposed DStroke is noticeably faster and more accurate at identifying and extracting brushstrokes, outperforming the other methods.

CCS Concepts: • **Computing methodologies → Image manipulation;**

Additional Key Words and Phrases: Brushstroke extraction, painting authentication, hard and soft segmentation, Pix2Pix network

ACM Reference format:
Yunfei Fu, Hongchuan Yu, Chih-Kuo Yeh, Tong-Yee Lee, and Jian J. Zhang. 2021. Fast Accurate and Automatic Brushstroke Extraction. *ACM Trans. Multimedia Comput. Commun. Appl.* 17, 2, Article 44 (May 2021), 24 pages.
https://doi.org/10.1145/3429742

This research was supported by EU-H2020-MSCA project: AniAge (No. 691215) and the grant of the Ministry of Science and Technology, Taiwan (Grant No. 107-2221-E-006-196-MY3 and Grant No. 108-2221-E-006 -038 -MY3).
Authors' addresses: Y. Fu, iArt.ai, Shenzhen, China; email: fuyunfei1991@gmail.com; H. Yu and J. J. Zhang, National Centre for Computer Animation, Bournemouth University, Fern Barrow, Poole, UK; email: {hyu, jzhang}@bournemouth.ac.uk; C.-K. Yeh, School of Computer Science and Software, Zhaoqing University, City 701, China; email: simpson.ycg@gmail.com; T.-Y. Lee, Department of Computer Science and Information Engineering, National Cheng-Kung University, City 701, Tainan, Taiwan, China; email: tonylee@ncku.edu.tw.

1551-6857/2021/05-ART44 $15.00
https://doi.org/10.1145/3429742

1 INTRODUCTION

In recent years, paintings processing has received wide attention in computer vision and graphics fields, such as animating paintings [37], style transfer [18, 38], and mimicking paintings [39, 47]. Particularly, deep learning technology is boosting this research. There have been several emerging AI created paintings available, which falls in the category of stroke-based rendering [9, 10, 12, 36], i.e., creating non-photorealistic imagery through placing discrete elements such as paint strokes or stipples. However, for learning styles and mimicking paintings, it usually requires strokes to be extracted from masters' pieces in advance. These applications indeed demand an old masters' brushstroke database for rendering purposes. A rising challenge is to automatically and accurately extract brushstrokes from a brush painting, which inspires the research work of this article.

In some circumstances, computer programs can extract certain patterns from scans more thoroughly than manual attempts, process a much larger number of paintings, and be less subjective. However, despite encouraging results from research groups, none are perfect. For instance, current methods cannot automatically extract all the brushstrokes from a painting, and therefore manual input is required, which is a tedious and time-consuming task due to the nature of hundreds or thousands of brushstrokes intermingling with each other in the painting.

To tackle the rising challenge, this article aims to develop an efficient algorithm that can automatically and correctly detect close to all the brushstrokes on a brush painting and accurately approximate them. The proposed Deep neural network–based brush Stroke extraction (DStroke) method employs deep neural network and image matting techniques (i.e., foreground and background extraction problems) to brushstroke extraction and accurately approximates the real brushstrokes. The challenging problem we encounter is to deal with the scenario of brushstrokes overlapping in a painting. Compared to the existing representative work [7, 25, 41], the proposed DStroke can clearly identify close to all the brushstrokes, while accurately acknowledging the faithful soft transitions between brushstrokes (see the right of Figure 1). This is significant for image segmentation because brushstrokes similar in color due to either overlapping or being adjacent to each other (see the left of Figure 1) tend to be incorrectly classified as one (see the middle of Figure 1), thus requiring human correction. This also sums up the main limitations of [7, 25, 41]. To the best of our knowledge, the other stroke extraction methods don't take into account overlapped strokes. Additionally, for visualization purposes, we label the extracted brushstrokes in colors (see the middle and right of Figure 1). Particularly, to highlight the soft transitions of brushstrokes, the color values depend on the individual alpha values, which results in the blurred boundaries of strokes at the right of Figure 1. The blurred boundaries indeed represent soft transitions.

Our research is inspired by the state-of-the-art work, i.e., Pix2Pix network [17] and Semantic Soft Segmentation [2], both of which facilitate semantic segmentation. Likewise, as for the other GAN neural network–based semantic segmentation methods [6, 11, 15, 40], Pix2Pix [17] cannot also be directly applied to brushstroke extraction due to the following facts:

- Currently, there is no brushstroke training dataset available for deep neural network training. Manually annotating strokes on a painting is a tedious task making it difficult to build up a large training dataset for deep learning purposes.
- The current semantic segmentation methods always fail to segment overlapping strokes on a painting since all the strokes may share the same class.
- The current segmentation methods are not suited to deal with a dense labeling problem since labels are discrete valued. Unfortunately, extracting hundreds or thousands of brushstrokes from a painting is a dense labeling problem.

We modify the Pix2Pix network to cope with these difficulties in this article. Aksoy et al. [2] presented the other implementation of semantic segmentation on soft transitions between image regions, which embeds texture and color features from the image as well as high-level semantic information generated by a neural network. The soft segments are generated through eigendecomposition of the Laplacian matrix. Nevertheless, this method suffers from at least three limitations, which results in the failure of soft segmentation on the application of brushstroke extraction:

- It only supports a small number of segments, and the segment number is usually fixed. This is not suitable for segmenting hundreds or thousands of strokes from a painting.
- It does not support instance-level segmentation, since there is no instance-aware semantic information available. This is not suitable in dealing with the scenario of overlapping strokes of a similar color.
- The spectral decomposition of the matting Laplacian is very expensive, decreasing the efficiency of the algorithm.

In contrast, the proposed DStroke overcomes these deficiencies. Our research focuses on the efficiency of the algorithm, i.e., computational complexity, accuracy, and automation. The main contributions include the following:

- A large, automatically generated painting training dataset. To the best of our knowledge, we are the first to provide an automatic method to build up a large painting sample dataset for deep learning purposes.
- The proposed DStroke method supports soft segmentation on instance level to identify every brushstroke and recover the faithful soft transitions between them.
- The modified Pix2Pix network can output the labels of segmentation without limitation on segment number, which are exactly discrete values.

Moreover, the proposed DStroke method can fulfill soft segmentation for fuzzy boundaries by involving the guided filter [13], which is a linear time algorithm. In fact, the efficiency and accuracy of a guided filter entirely depend on a given guidance image. A merit of DStroke is to rapidly generate accurate guidance images required by the guided filter.

Additionally, an increasing number of problems in the history of art, particularly authenticating and dating brush paintings by the masters, have received wide attention, and many rigorous computer methods have been developed through significant interdisciplinary efforts across computer vision, graphics, and art history in recent years. Herein, brushstroke extraction plays an important role. This is because brushstrokes can be employed to assess the level of distinction between categories of paintings and identify attributes that differ significantly on average [4, 14, 16, 25, 31, 35, 41]. Moreover, identifying the order of brushstrokes and their individual colors is important as well for authentication purposes. In fact, it is not only a request from painting authentication, but also is the basis of many existing layer decomposition methods [7, 30, 33, 34, 37]. However, it still remains challenging since overlapped strokes result in mixed color, and transparency change leads to blur. This article demonstrates that the DStroke method can automatically detect close to all the brushstrokes while accurately recovering the fragile soft transitions between strokes, has high efficiency, and also works well on most brush paintings such as oil brush paintings, Chinese paintings, watercolor, and acrylic paintings.

2 RELATED WORK

Layer Decomposition: In digital image editing systems, artists deposit colors throughout a painting via a set of strokes, which are classified into different layers in terms of opacity values. Skilled artists commonly blend multiple layers, each of which is composed of simple colors and transparency gradients, to represent an object. However, the scans of paintings and photographs have

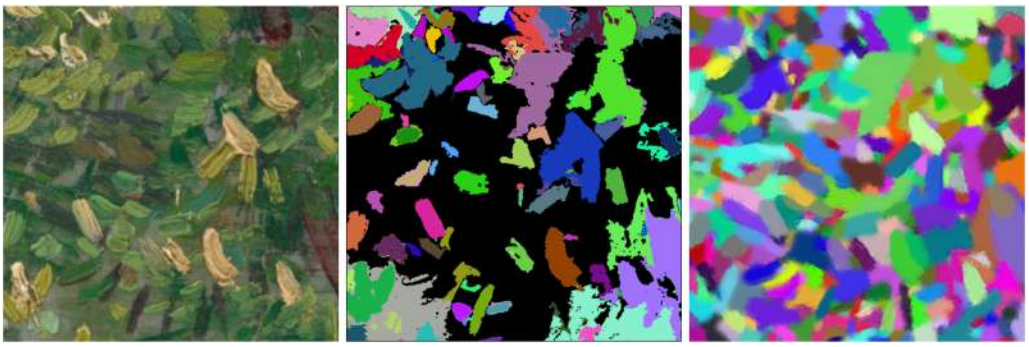
Fig. 1. Paintings with highly mixed brush strokes in similar colors (left); extracted brushstrokes by [7] (middle, many strokes are missed and are classified in dark). Extracted brushstrokes by our DStroke (right, the soft transitions between strokes can be noted).

no such layer information. Even for digital paintings, the available image files usually lack layer information as well. Without layer information, even simple editing may become very challenging. Our previous work [7] attempts to employ layer decomposition and boundary constraints to stroke extraction. Richardt et al. [30] present an interactive approach for decomposing bitmap drawings and photographs into opaque and semi-transparent vector layers. Xu et al. [37] aim to decompose Chinese paintings into a collection of layered brushstrokes with an assumption that an overlapping region contains at most two strokes and has minimal variation in transparency. Moreover, their approach requires the knowledge of the order of strokes and a brushstroke library for recognition, which is built by professional artists. McCann et al. [26, 27] present two generalized layer decomposition methods, in which pixels have individual layers and partially overlap with each other, the layer orderings of which may be manipulated. Aharoni-Mack et al. [3] propose a recoloring painting method for watercolor paintings based on color palette estimation and layer decomposition. Tan et al. [34] present a layer decomposition method based on RGB-space geometry. An assumption is that all possible image colors are convex combinations of the palette colors. Computing the convex hull of image colors and per-pixel layer opacities is converted into a convex optimization problem. Thus, their method can work well without prior knowledge of shape and even with some overlapping strokes. Furthermore, Tan et al. [33] proposed a palette-based layer decomposition algorithm. The distinct advantage is to require no numerical optimization and allow users to interactively edit the palette to adjust the layers. Koyama and Goto [19] argued that the previous methods typically only support linear color-blend modes, and further proposed a layer decomposition method to support any user-specified color-blend modes. However, these methods still cannot accurately extract all the strokes from a painting. The proposed DStroke method will tackle this challenge.

Soft Segmentation: For oil paintings such as van Gogh's artifacts, most of the brushstrokes tend to be opaque, which benefits edge detection. However, the overlapped parts tend to be missed. Li et al. [25] and Lamberti et al. [41] employ the seed growing–based brushstroke extraction scheme to painting authentication and artist identification. Firstly, some pixels are selected as seeds. Then, neighboring pixels are exploited through region growing. Region growing can be controlled through a shape validation method. In their implementation, the metric of shape validation is defined based on 10 examples of brushstroke regions that are manually extracted from van Gogh's paintings. When the boundaries of objects are vague, or if objects are translucent (e.g., water, hair, brushstrokes with transparency), traditional image segmentation methods begin to fail.

Even if manual segmentation is taken, it is no longer reliable. Our proposed DStroke method uses soft segmentation technology to tackle this challenge.

Compared to usual segmentation [46], the key feature of soft segmentation is that pixels may be assigned to the foreground or background in terms of transparency values. Thus, it can capture the fuzzy boundary between objects. The core is image matting. Matting aims to estimate the per-pixel transparency of the foreground region based on the indicators that users provide [1, 5, 23]. The indicators are typically represented using trimap, in which the foreground, background, and unknown transparent regions are distinguished using different colors. Instead of identifying the foreground objects from the image, soft segmentation decomposes an image into multiple layers. As a result, each pixel is likely to be classified into multiple layers. Singaraju et al. [32] presented an approach of segmenting an image into multiple layers by the estimation of alpha mattes, which need to be optimized iteratively. The spectral matting technique in [24] extended spectral segmentation techniques [23] from the extraction of only hard segments, to include the extraction of soft matting components.

However, these soft segmentation methods cannot generate semantically meaningful segmented regions without user indications. Recently, Aksoy et al. [2] leveraged deep network for semantic soft segmentation, using the high-level information (semantic information) from a deep network to define affinities between different regions to generate soft segmentation corresponding to semantic regions. On the other hand, low-level information such as color and matting affinity, are also calculated to construct the matting Laplacian matrix to handle soft boundaries. Like spectral matting, regions with local soft transitions are generated using matting Laplacian and spectral decomposition. However, when applied to brushstroke extraction, their methods suffer at least two limitations. Firstly, a brush painting normally contains hundreds or even thousands of brushstrokes. Unfortunately, the methods in [2, 24] only support a limited number of segmented regions. Secondly, despite using high-level information from a deep network, the semantic soft segmentation in [2] cannot do instance-level segmentation. This will bring about serious errors for brush paintings since hundreds, or thousands of instances (brushstrokes) in one painting may share the same semantic label. In contrast, our proposed DStroke can handle thousands of brushstrokes in a given brush painting and can effectively generate instance-level segmentations corresponding to semantic regions.

Deep Learning–Based Segmentation: Krizhevsky et al. [20] introduce a deep convolution network called AlexNet consisting of eight layers and millions of parameters which was trained on the ImageNet dataset with 1 million images. Since then, even larger convolutional networks have been designed and competed with the state of the art in image segmentation, including [6, 11, 17, 40, 45]. U-Net [29] was proposed for biomedical image segmentation, which provides a pixel-wise accuracy for dense cells segmentation. Deep convolution neural networks were used to learn through minimizing a loss function. It is crucial to design an appropriate loss function. For instance, if simply using the Euclidean distance between the predicted and ground truth pixels as loss, the results may tend to blur the boundaries. Unceasingly involving expert knowledge into loss functions can improve the performance of deep networks. The discriminator network is combined with U-Net in the Pix2Pix network [17], which can automatically learn a loss function appropriate for satisfying this goal. It dramatically improves the accuracy of segmentation that is employed in our method.

The main challenge currently is the lack of a training dataset for brushstrokes extraction. Deep neural networks need a large dataset for effective training. Manually extracting brushstrokes is impractical since a painting usually contains thousands of brushstrokes. To deal with this challenge, we propose an automatic method to build a large training dataset, in which brush paintings consist of a given set of brushstrokes and also contain information of the brushstrokes' edge maps. This addresses the challenges of manually constructing a large dataset for machine learning purposes.

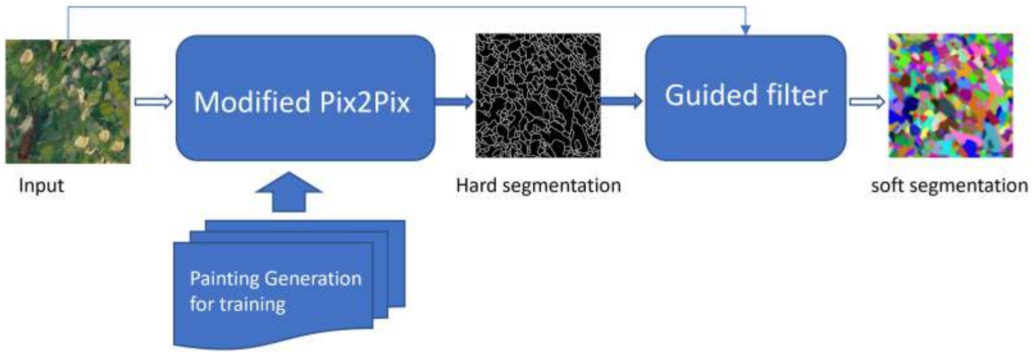

Fig. 2. Overview of the proposed DStroke.

3 DEEP NETWORK–BASED BRUSHSTROKE EXTRACTION (DStroke) METHOD

We focus on the efficiency of the proposed DStroke method, which is illustrated in Figure 2. The first problem is to automatically create a large painting training dataset for deep neural network training purposes. To the best of our knowledge, we are the first to propose an automatic method of building up a large training dataset of brush paintings. The modified version of the Pix2Pix network [17] is applied to brushstroke extraction for edge map detection (also called hard segmentation). We prefer instance-level segmentation on paintings and therefore employ the edge maps of sample paintings as instance-level information to neural network training. After that, the guided filter [13] is employed to soft segmentation, i.e., identifying soft transitions between brushstrokes to refine the boundaries of strokes, which is a linear time algorithm and drastically improves the efficiency of DStroke.

3.1 Training Database Generation

We address the automatic painting generation method at first and then address how to build up a large painting training dataset. Brushstrokes on a painting are usually similar. It tends to classify multiple brushstrokes into one class. To identify every brushstroke from a painting, the training sample dataset is required to contain a set of paintings associated with the individual edge maps of brushstrokes in order to provide instance-level information. The edge map is the collection of the brushstrokes' boundaries on the painting.

Our painting generation method is to convert photos to paintings through a set of given brushstrokes. Although there have been many methods for painting production, they do not usually provide brushstroke information. In our implementation, when painting brushstrokes, the boundaries are cumulated to form the edge map of brushstrokes on the resulting painting.

Moreover, for reality purposes, the brushstroke samples are required to maintain the illumination effect of paintings. Usually, real brush paintings have individual height maps since the pigment associated with each brushstroke on a canvas may be layered to have differing thicknesses. The photo or image of a painting is the realistic appearance of the canvas surface with plausible lighting. Thus, every brushstroke sample is assigned to the individual height map (e.g., let alpha map as height map), and the height field of the created painting is generated by rendering the brushstrokes textured with the height maps. The final painting is rendered by the input of image colors alongside the height maps. Algorithm 1 addresses how to create a painting by mimicking the physical appearance of brushstrokes.

In our implementation, the canvas C is firstly initialized as a blank image plane, and the **region of interest** (**ROI**) is selected in terms of the pixel with the largest color difference between the

ALGORITHM 1: Painting Generation

```
Input: $S$ sample image; $SA$ stroke alpha map; $ST$ stroke thickness/height map; $C$ blank canvas;
Output: $C$ painted canvas; $E$ edge map;
Initialising: $Diff = \infty$ color difference; $H$=0 height field; $R$=0 the ratio of painted area over the canvas area;
while $R < 1$ do                    /* may change $R$ to threshold $\epsilon < \|S - C\|$ for a pleasing visual effect */
  $Diff_1 = S - C$
  $ROI = arg\max_{i,j} |Diff_1(i,j)|$      /* region of interest $(ROI)$ is a neighborhood of the pixel $(i,j)$ */
  Select a stroke from library;
  $color = avg(ROI)$                /* average color of $ROI$ on S is viewed as the selected stroke's color */
  $C = compose(C, Stroke, Color)$         /* paint the selected stroke on canvas with color */
  $Diff_2 = S - C$                  /* update color difference */
  if $\|Diff_1\| > \|Diff_2\|$ then
    Keep the selected stroke on $C$;
    Save the stroke boundary on $E$;
    Remove the overlapped edge from $E$;
    $H = H + ST$                    /* add stroke's height map to height field */
    $R = \frac{\text{painted region}}{\text{canvas}}$            /* $R \in [0,1]$, the painted regions will gradually spread over the whole canvas */
  else
    Remove the selected stroke from $C$;
  endif
end while
$C = Rendering(C, H)$;
```

sample image and the canvas. The gradient within this pixel's neighborhood on the sample image is computed as the ROI's gradient. Secondly, a stroke is randomly picked up from the small stroke library KyleBrush [44] and put over the ROI on the canvas. The orthogonal direction to the gradient usually indicates edges. We compose the stroke along with the orthogonal direction on the canvas. After that, the new ROI is detected over the updated canvas and is overlapped by a new selected stroke from the stroke library until the original canvas is thoroughly covered by strokes. The height map ST of a brushstroke sample is set equal to its alpha map (see Figure 3). The height field H of the canvas is cumulative and updated each time a stroke is painted onto the canvas. The normal of pixels on the canvas is calculated by the directional derivative of the height field. The illumination of each pixel is then calculated under the different illumination models in the rendering step, e.g., the reflection model may be the Phong model or the Cook-Torrance model [43].

Figure 4 shows the process of creating a brush painting through a set of given brushstrokes; particularly Figure 4(c), which shows the effect from the height field of the painting. Herein the goals are to make the painting look like the sample image and limit the number of strokes in some way to make the result look like a painting.

To this end, the terminal criteria include the ratio R of the painted area over the whole canvas area and the differences $Diff$ between the sample image and the rendering. $Diff$ is minimized in a trial-and-error way, i.e., if the change reduces the cost function, the change is incorporated; otherwise, it is discarded. Using R can limit the number of strokes. If pursuing a pleasing visual effect (see Figure 4(d)), we can simply change the ratio R to the threshold ϵ of the $Diff$'s error (see Algorithm 1). However, this will require a longer running time.

We further address how to build up a large training dataset using our automatic painting generation method. In deep learning applications, the training sample set is acquired in a limited set of conditions. But the application may exist in a variety of conditions. It is natural to add synthetically modified data into the training dataset to account for these new situations, which is called data

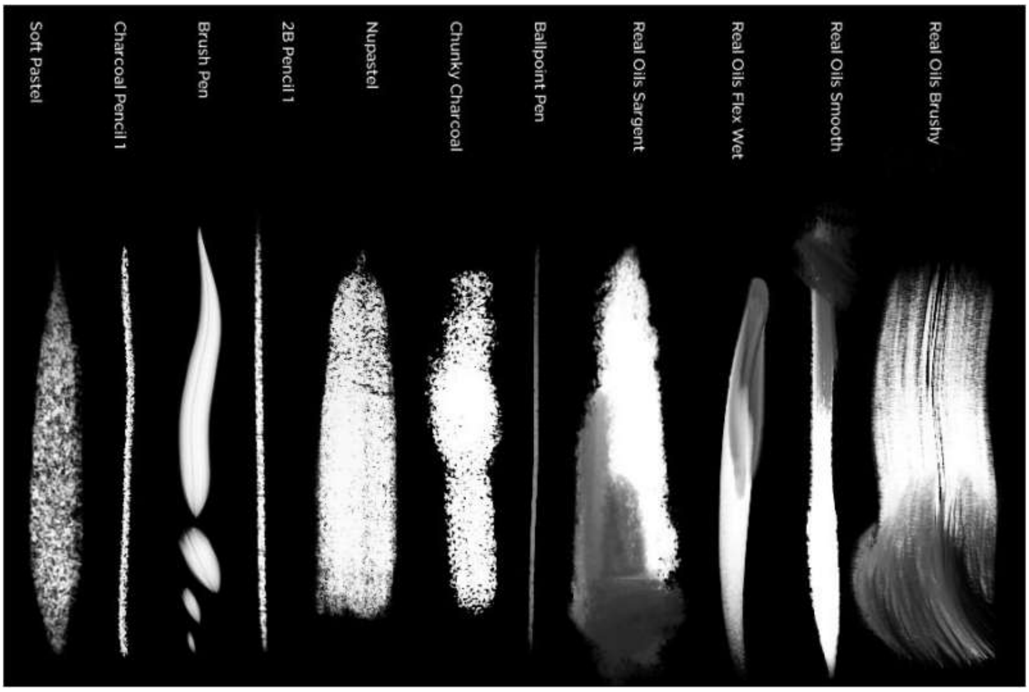

Fig. 3. The alpha maps of different brushstrokes from the KyleBrush library [44].

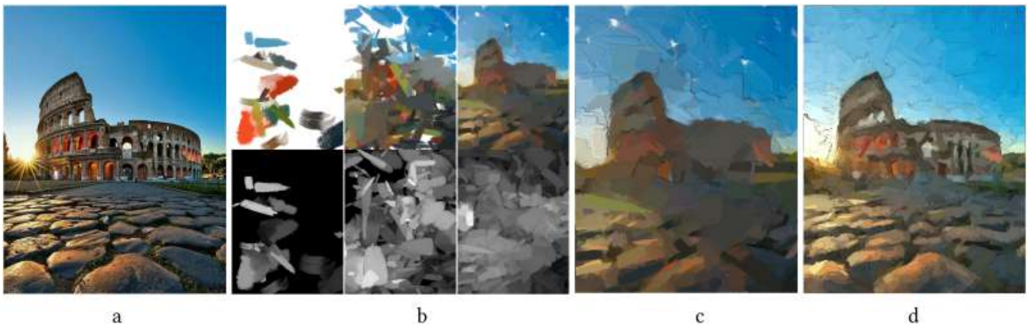

Fig. 4. Process of painting production. (a) Scene image. (b) Iteratively painting strokes on canvas (top row); height map of canvas (bottom row). (c) Illumination of canvas. (d) Effect image using the threshold ϵ, which looks more like (a).

augmentation. To this end, the created paintings in our training dataset are randomly flipped, rotated, and slightly distorted; noise is added (e.g., Gaussian noise, salt and pepper noise) to generate new training samples. We also shift the colors of the created paintings by randomly changing hues for new training samples. Figure 5 shows such synthetic paintings. As the ratio R is employed rather than the threshold ϵ, the running time is acceptable; i.e., Algorithm 1 spends around 2 hours on producing 1,000 paintings for our training dataset. Regarding the core of the training dataset generation, i.e., Algorithm 1, we hope to point out the following distinct advantages when compared with the existing “painterly rendering methods” such as [8–10, 18, 38],

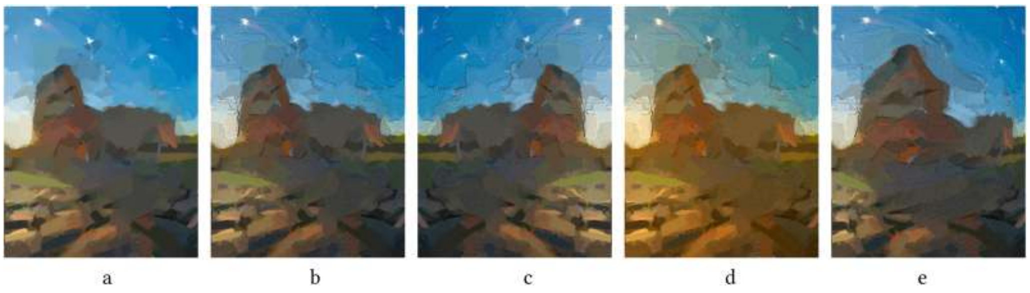

Fig. 5. Illustration of data augmentation. (a) Input painting, (b) adding noise, (c) distortion, (d) changing hue, and (e) flipping.

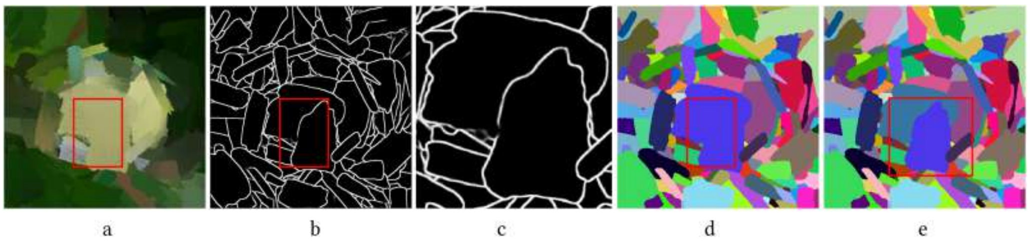

Fig. 6. An example of blurred edges and gaps within strokes. (b) Edge map by Pix2Pix from (a) input image. (c) A small gap from (b). (d) Wrongly merging the adjacent regions due to a gap on their boundaries. (e) Correct segmentation.

- Algorithm 1 can better avoid over-fitting issues in training by utilizing many stroke types for training instead of only a few, e.g., using more than 500 stroke types in our implementation. Moreover, the data augmentation is also applied to increasing the variety of data.
- Utilizing the height maps of brushstrokes, output paintings are rendered under different light settings to further avoid over-fitting issues.

3.2 Deep Network for Hard Segmentation

We apply the deep neural network, i.e., Pix2Pix network [17], to paintings for hard segmentation. The modified Pix2Pix network is end-to-end trainable, i.e., the output is a binary segmentation map instead of an intensity image. The network consists of two parts: the generator is trained to generate the edge maps of brushstrokes from input paintings, and the discriminator is trained to detect the generator output's fakes. For most of the paintings, Pix2Pix can output good edge maps of paintings. However, in some scenarios, edge detection still remains challenging (see Figure 6). It can be noted that there are still some small gaps or blurred edges on the outputs, i.e., the boundaries of brushstrokes are not closed. This is unacceptable since these gaps usually result in wrong segmentation. The following soft segmentation step always relies on accurate hard segmentation.

To tackle this challenge, we add region information into the Pix2Pix network. There are two distinct advantages: (1) improving convergence of deep network training; and (2) obtaining a binary edge map without gaps. The results are very encouraging, i.e., the boundaries of segments are closed. For ease of use, we keep the same mathematical symbols and terminology as in [17]. Figure 7 shows the modified Pix2Pix network structure, i.e., the generator G's inputs are paintings x while the outputs being the brushstrokes' edge maps $G(x)$. Our modification is to add a reference edge map $T(x)$ as one of the G's outputs, which can be obtained by a simple merging operation

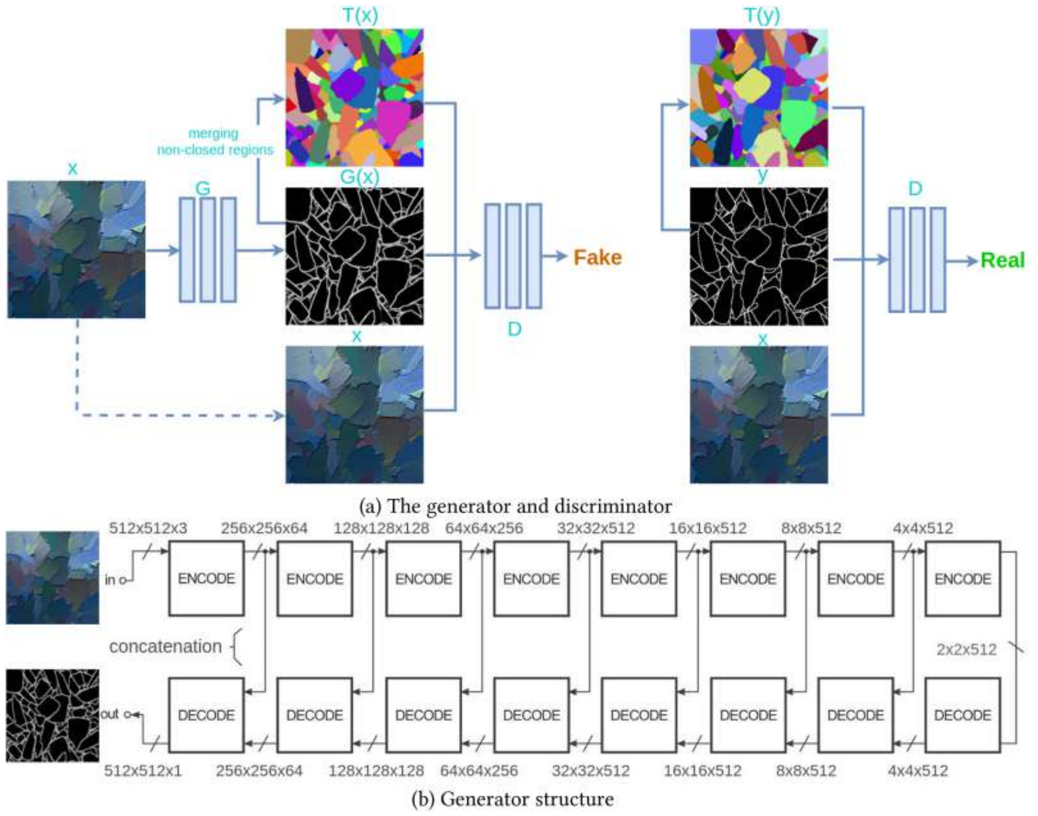

Fig. 7. The modified Pix2Pix network structure. Note that the colors in $T(x)$ and $T(y)$ are used for visualization, but not labels. $T(.)$ denotes edge map that is involved in Pix2Pix network computation. Note that $T(x)$ and $T(y)$ should be of binary image edge maps without colors. Herein we color regions in order to visualize the regions closed.

such as the trapped-ball method [42]. Herein, the $T(x)$ is a binary image, whereas the $G(x)$ is an intensity image. The merging strategy is to apply the region growing to $G(x)$ (see Figure 6(b)) so that the regions' boundaries are closed in the resulting $T(x)$. There is no gap or blurred edge on $T(x)$. It is likely to wrongly merge adjacent regions (see Figure 6(d)). Compared to the output $G(x)$, the difference between the $T(x)$ and the ground truth will likely be enlarged. Thus, it cannot pass the discriminator check unless wrong segmentation is corrected (see Figure 6(e)). The distinct advantage is that the region growing ensures the region boundaries closed and outputs the binary edge map $T(x)$ rather than an ambiguous intensity image. The loss of the modified Pix2Pix network can be rewritten as

$$\begin{aligned} L_{cGAN}(G,T,D) \quad = \quad & E_{x,y\sim p_{data}(x,y)}[\log D(x,y,T(y))] \\ & + E_{x\sim p_{data}(x),z\sim p_z(z)}[\log(1-D(x,G(x,z),T(G(x,z))))], \end{aligned} \tag{1}$$

where $T(.)$ denotes the reference edge map. Note that for the ground truth y, obviously $T(y) = y$ when y is binarized. The regularizer with the l_1-norm metric is written as

$$L_{l_1}(G) = E_{x,y\sim p_{data}(x,y),z\sim p_z(z)}[\|y - G(x,z)\|_1]. \tag{2}$$

We do not add another regularizer for the reference edge map T since both $T(x)$ and $G(x)$ are edge maps and $T(x)$ is derived from $G(x)$. The discriminator is trained by

$$\begin{cases} D[x, G(x,z), T(G(x,z))] = fake \\ \qquad D[x, y, T(y)] \qquad = real. \end{cases} \tag{3}$$

The resulting generator is

$$G^*, T^* = arg \min_{G,T} \max_{D} L_{cGAN}(G, T, D) + \lambda L_{l_1}(G). \tag{4}$$

The network architecture uses the form of convolution-BatchNorm-ReLu. The generator in Figure 7(b) shows the skip connections. Regarding the new added term, i.e., reference edge map T, it may be regarded as a post-process of the generator G. The resulting $T(x)$ is still an edge map as well as the output $G(x)$. In fact, the binarization of the output $G(x)$ likely results in gaps from the blurred edges of the $G(x)$. The $T(x)$ has no gaps due to the merging operation. This is indeed to amplify the difference between the discriminator input and the ground truth if there is wrong segmentation, which is beneficial for deep network training. The input painting x associated with its reference edge map $T(x)$ and edge map $G(x)$ is regarded as a tensor, which is used for training the discriminator patchGAN. Moreover, we test the deep network with/without the reference edge map $T(x)$. It can be noted that the final output, $G^*(x)$, may still contain blurred edges due to the property of intensity image. This will no be longer a big deal here since the final reference edge map, $T^*(x)$, is a binary image and has no such deficiency. Thus, $T^*(x)$ is the desired hard segmentation.

In our implementation, 5,000 training paintings generated by Algorithm 1 are used for 200 epochs, batch size 1, with random jitter and mirroring. To visualize the regions closed, we color every region in the edge maps $T(x)$ and $T(y)$ in Figures 6 and 7. The color values are labels that are randomly assigned and not involved in the computation. It is worth mentioning that the final reference $T^*(x)$ is indeed a label map since the boundaries of all the regions are closed. $T^*(x)$ can be expediently labeled in any form. Currently, the applications of GAN models need to convert the labels from discrete values to continuous-valued variation [17]. Our modified Pix2Pix network can successfully generate "labels" without such troubles.

3.3 Soft Segmentation

Brushstrokes overlapping usually results in fuzzy boundaries. A rising issue is to identify the boundary of a stroke on the overlapping regions. The proposed DStroke employs the guided filter [13] to the scenario of strokes overlapping in a painting and extracts the alpha mattes of brushstrokes as soft segmentation. The challenge is to identify the soft transitions between brushstrokes in terms of transparency. The guided filter has been proved to be a good explicit image matting method, particularly capturing the thin structures in a composite image [13]. This is because with the help of the guidance image, it can make the filtering output more structured and less smoothed than the input. Accurate guidance image (or also called a trimap) plays a vital role in such soft segmentation. We employ the hard segmentation of a painting as its guidance image for brushstroke soft segmentation and show it below by briefly addressing the guided filter. The guided filter outputs the alpha mattes of every brushstroke in a painting, which is used to further refine their boundaries. Additionally, we also apply the resulting alpha mattes to stroke ordering and coloring issue, which is required by stroke analysis for painting authentication. In terms of the pigment colors, stroke orientation, and the order of strokes from master pieces, artists hope to have an insight into the individual old masters' traits [22]. Moreover, it also is the basis of many existing layer decomposition methods [7, 30, 33, 34, 37].

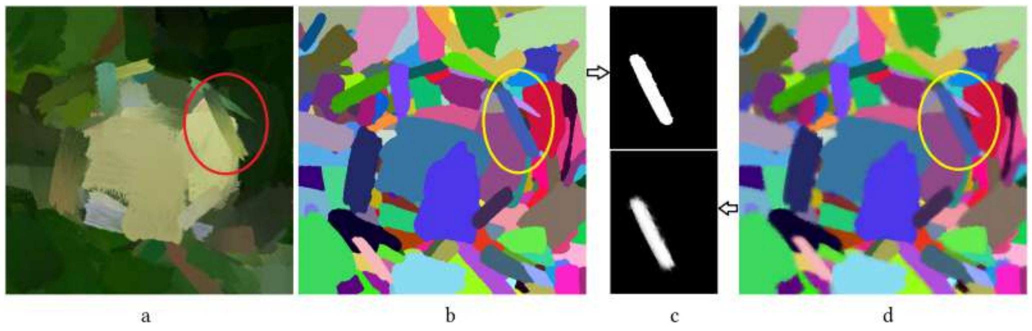

Fig. 8. Illustration of soft segmentation. (a) Input painting. (b) Hard segmentation. (c) Top: A segmented brushstroke mask from (b). Bottom: The selected brushstroke mask from (d). (d) Soft segmentation.

Relating to the matting Laplacian, the guided filter involves an input image I, a guidance image p (or also called a trimap), and an output alpha matte α, and minimizes the cost function,

$$E(\alpha) = (\alpha - p)^T \Sigma(\alpha - p) + \alpha^T L\alpha, \tag{5}$$

where L denotes an $N \times N$ matting Laplacian matrix, and Σ denotes a diagonal matrix encoded with the weights of the constraints. We view hard segmentation as p. The solution to this optimization problem can be approximated [13] by

$$\alpha_i \approx \sum_{j\in\omega_i} W_{ij}(I)p_j, \tag{6}$$

which has an $O(n)$ linear algorithm. The weight of the guided filter kernel, $W_{ij}(I)$, is defined on a small sized window ω_i. Obviously, once trimap p is available, DStroke can take soft segmentation in linear time.

ALGORITHM 2: Brushstroke Ordering and Coloring

```
Input: a pair of overlapped strokes, $A$ and $B$, with the observed colors of each pixel $(AB)_{rgb}(i), i \in A \cup B$;
Output: the order of $A$ and $B$; the colors $A_{rgb}, B_{rgb}$;
/* computing the colors of $A$ and $B$ under two supposed scenarios, respectively */
  Suppose the order $A.\because B$;
  Computing $A_{rgb1}, B_{rgb1}$ by Equation (8)
  Suppose the order $B.\because A$;
  Computing $A_{rgb2}, B_{rgb2}$ by Equation (8)
/* reconstructing the color of each pixel within $A$ and $B$ */
  Using $A_{rgb1}, B_{rgb1}$ with the assumption of the order $A.\because B$;
  Computing each pixel's color $(A.\because B)_{rgb}(i)$ by Equation (8) within $i \in A \cup B$
  Using $A_{rgb2}, B_{rgb2}$ with the assumption of the order $B.\because A$;
  Computing each pixel's color $(B.\because A)_{rgb}(i)$ by Equation (8) within $i \in A \cup B$
/* computing order */
  $order = argmin\left(\sum_{i\in A\cup B}((AB)_{rgb}(i) - (A.\because B)_{rgb}(i))^2, \sum_{i\in A\cup B}((AB)_{rgb}(i) - (B.\because A)_{rgb}(i))^2\right)$
```

We further analyze the errors of the alpha matte α. Let the offset of p be Δp in a vector form. Substituting Δp into Equation (6) in a matrix form yields

$$\Delta\alpha \approx W\Delta p. \tag{7}$$

Note that $\sum_{j \in \omega_i} W_{ij}(I) = 1$. If the elements of error vector Δp share the same value, this value is simply added onto the alpha matte vector α. This implies that the error Δp is transferred linearly to the alpha matte α. As a result, the guided filter performance relies on quickly providing the accurate trimaps. In our implementation, each brushstroke is processed independently, i.e., one stroke is regarded as the foreground object to be segmented from the others. The resulting alpha matte of each stroke is independent of the others. Figure 8 shows the effect of soft segmentation. It can be noted that the mask of the alpha matte is closer to the original brushstroke than the hard segmentation in Figure 8(c).

Additionally, we apply the resulting alpha mattes of brushstrokes to the stroke ordering and coloring issue. The standard Porter-Duff's "*A* over *B*" compositing and blending model [28] is used:

$$(A \therefore B)_{rgb} = \frac{\alpha_A A_{rgb} + (1 - \alpha_A)\alpha_B B_{rgb}}{\alpha_A + (1 - \alpha_A)\alpha_B}, \tag{8}$$

where pixel A with color A_{rgb} and alpha α_A overlays pixel B with color B_{rgb} and alpha α_B and the observed color is $(A \therefore B)_{rgb}$. The ordering is unchangeable since the "*A* over *B*" operation is not commutative. For a pair of overlapped brushstrokes, we can apply Equation (8) within the union region of these two brushstrokes to each pixel, and compute the two stroke colors, A_{rgb}, B_{rgb}. In terms of our observation, it is plausible to assume that every brushstroke contains only one color in a painting. The color change within one brushstroke (e.g., from light to dark) is most likely caused by transparency, i.e., alpha change. For a given painting, the alpha mattes may have a variation within a brushstroke while the brushstroke color remains the same. If the order of stroke "*A* over *B*" is correct, reconstructing the color $(A \therefore B)_{rgb}$ of each pixel by Equation (8) using the resulting colors, A_{rgb}, B_{rgb}, should result in a small error. Otherwise, the order needs to be reversed. Solving the stroke colors A_{rgb} and B_{rgb} will result in an over-determining linear system depending on the number of pixels within these two strokes. This is a least square solution. If the order of stroke "*A* over *B*" is correct, the linear system is compatible and reconstructing the color $(A \therefore B)_{rgb}$ of each pixel will finally result in a small error. Otherwise, the linear system is incompatible, which will accumulate a big error in reconstructing each pixel color. Thus, simply reconstructing pixel colors may discriminate which order and brushstroke colors are acceptable. The brushstroke ordering and coloring is summarized in Algorithm 2.

Remark. Algorithm 2 is useful for the layer decomposition methods [7, 30, 33, 34, 37]. These methods usually solve the opacities for each layer through minimizing a polynomial cost function, which is an optimization problem with the initial guesses of layer number, order, and color values. It is likely to apply Algorithm 2 to a specified region in a painting to estimate layer colors and order of two successive layers, which are utilized as the estimations of layer color and order to solving this optimization problem.

4 RESULTS AND ANALYSIS

The experiments focus on the running time and accuracy of our DStroke. In our implementation, we applied Algorithm 1 to create 5,000 paintings associated with the individual set of brushstroke edge maps as our training dataset. We didn't apply the scans of real paintings to train deep network. This is because (1) for the synthesized paintings, the available edge maps are exact without any error; (2) for the scans of real paintings, the edge maps must be marked by manual. If applying the edge maps marked by manual to deep network training, it will result in a big error. Moreover, to avoid the over-fitting issue, several training strategies are employed such as Data argumentation, i.e., the samples of paintings and edge maps are randomly flipped, rotated, slightly distorted, and added noise. Moreover, the samples of paintings and edge maps are randomly cropped into

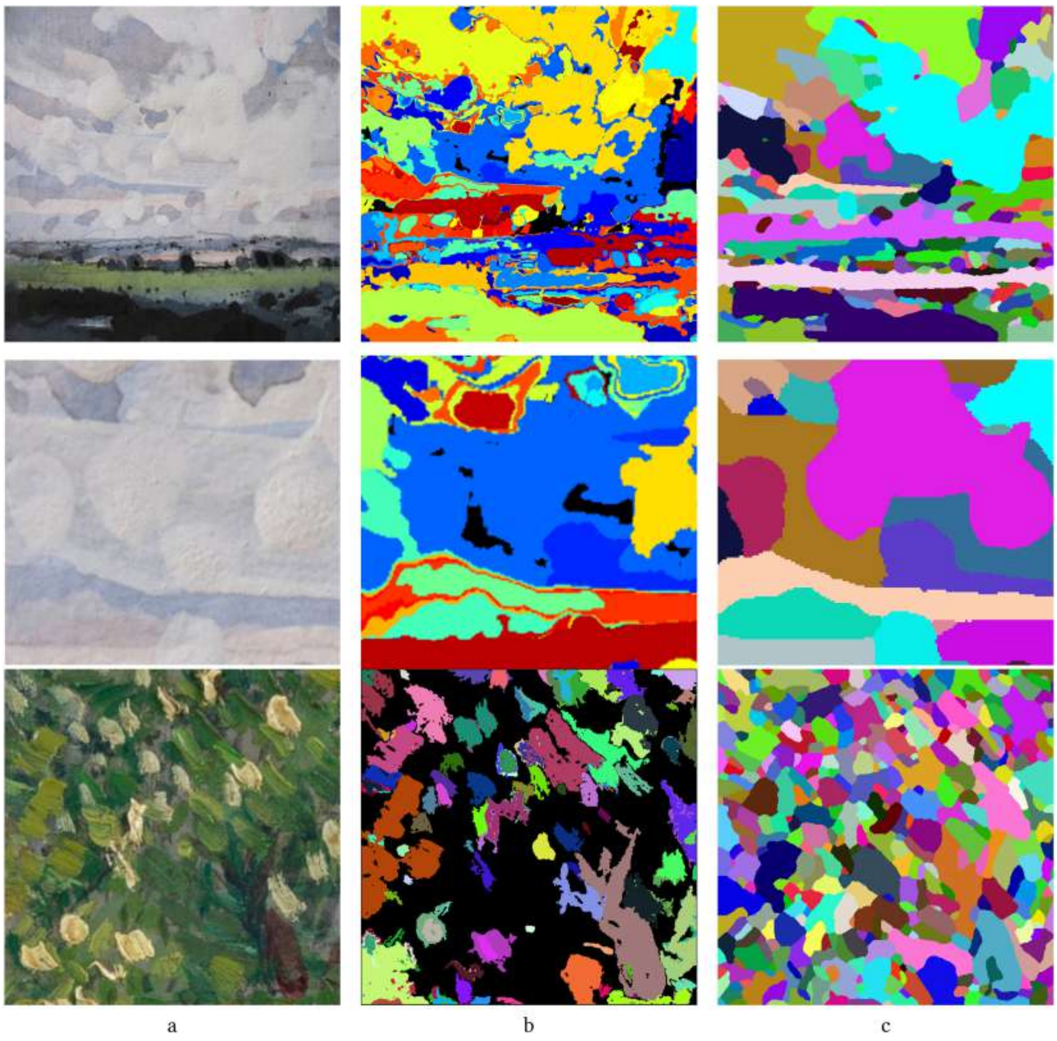

Fig. 9. Highly mixed brushstrokes in similar colors are extracted. (a) Input paintings. (b) Brushstrokes extracted by [7]. (c) Brushstrokes extracted by DStroke (only hard segmentation).

different sizes and resized into 512×512 pixels. Training the modified Pix2Pix network took less than 2 hours on a single GTX 1080 GPU.

In this section, we compare the proposed DStroke method with several existing approaches, including [7], and Semantic Soft Segmentation [2]. The test painting set is composed of two groups: one for the scans of 15 real brush paintings and the other for the synthetic paintings produced by Algorithm 1. For numerical comparison, we have to employ some synthesized paintings here. This is because the ground truth of the brushstrokes in the synthesized paintings is exact without any errors, whereas the ground truth of the scans is delineated by manual. Additionally, for fairness, the synthetic paintings in the test painting set are reproduced rather than taken from the training dataset. Additionally, we also show several examples of applying brushstrokes to image editing, such as recoloring and inserting objects, which herald a great potential in image processing.

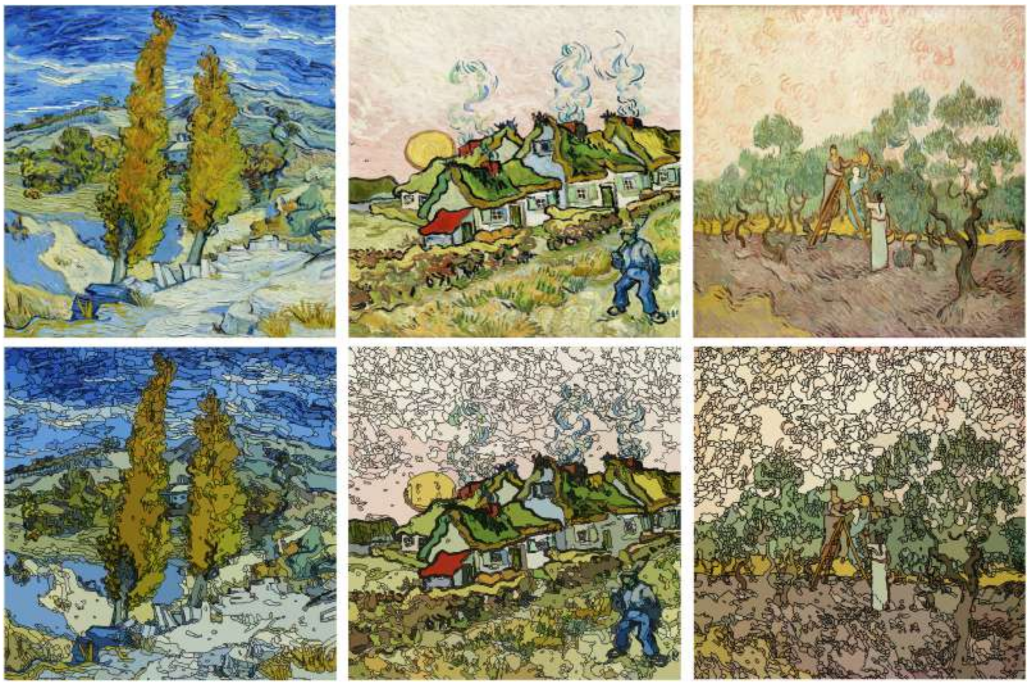

Fig. 10. Illustration of hard segmentation on whole paintings. The second row shows the hard segmentation.

4.1 Comparing DStroke with [7, 41]

We selected the real brush painting group in our test painting set for comparison, including acrylic paintings, watercolor paintings, and oil paintings. To demonstrate the robustness of our DStroke, paintings are carefully picked, in which brushstrokes in similar colors are heavily employed or there are many blending areas, as shown in Figure 9(a). Due to the brushstrokes in similar colors and the complexity of the paintings, some of brushstrokes are undetected by [7] (note that the undetected regions are labeled in black in Figure 9(b)). In contrast, our DStroke successfully extracts close to all the brushstrokes, even in similar color regions as shown in Figure 9(c). Moreover, we selected three well-known van Gogh painting scans and performed our DStroke on the whole painting rather than patches. To better visually match the hard segmentation to their counterpart in the source paintings, we used the dominant color of the brushstrokes and boundaries in the hard segmentation as shown in Figure 10.

To numerically evaluate our results, likewise [7, 25], we created the ground truth through manually marking brushstrokes by the experienced artists, and applied the same accuracy metrics, i.e., valid ratio and detection ratio, to tests (the reader is referred to [25] for the details of these two ratios). Moreover, we introduced a new metric, i.e., **Intersection over Union** (**IoU**), to evaluate the overlap ratio between the target masks (manually masked strokes) and the detected masks (detected strokes) as below,

$$IoU = \frac{Target \cap Detected}{Target \cap Detected}$$

The mean of Intersection over Unions (meanIoU) are evaluated and shown in Table 1. Computing the valid ratio and detection ratio, the valid covering percentage is given as 80% in [25], whereas it is set to 85% in our experiments. Although this change results in the low valid ratios and low

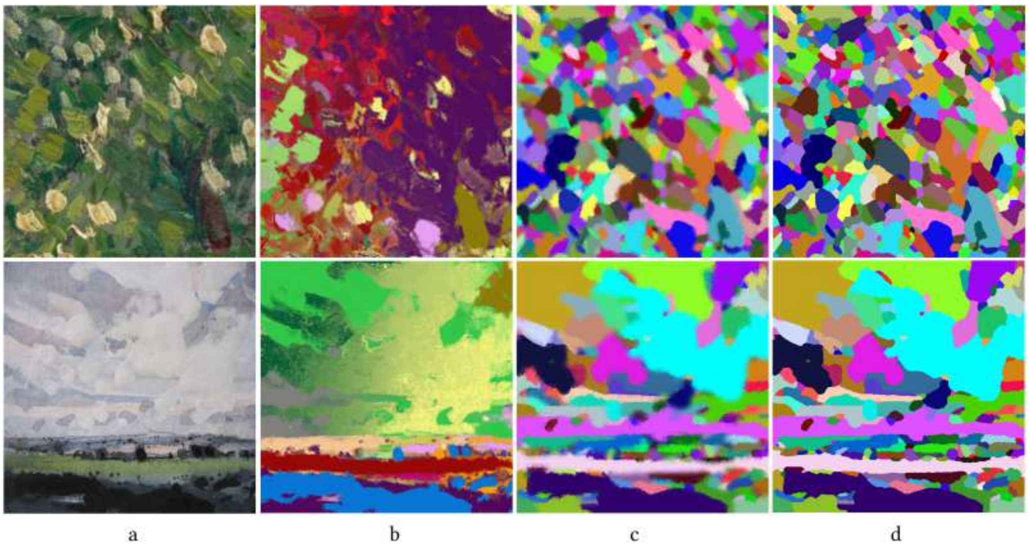

Fig. 11. Comparison of semantic segmentation and instance-level segmentation. (a) Input paintings. (b) The results of Semantic Soft Segmentation. This method accidently merges many strokes into one segment. (c) Soft segmentations and (d) hard segmentation by DStroke. DStroke can do segmentation on instance level to identify all the strokes.

detection ratios in Table 1, the quality of the matched brushstrokes is satisfactory. We also tried to set the percentage to 60%. All the ratios reached 100%, which means that all the brushstrokes can be detected and extracted though the accuracy is low.

Moreover, we further compared the proposed DStroke with [25, 41] using the metrics proposed by [41], including the Mean Square angular Distance of orientation (MSD_{α}), the Mean Square Percentage Distance of length ($MSPD_{\lambda}$), Mean Square Percentage Distance of width ($MSPD_{\phi}$), fill rate, and representativeness rate, and showed the results in Tables 2 and 3, respectively. These quantitative comparisons show that the proposed method can correctly detect on average 84% of the brushstrokes from a painting (see Table 1), and accurately approximate the real brushstrokes with high reliability (see Tables 2 and 3). Thus, we only claim that the proposed DStroke can extract close to all the brushstrokes from a painting. Figure 14 shows 15 real painting scans, the effect images of segmentation and manual marked strokes mentioned in Table 1. Although the modified Pix2Pix network is trained by the synthetic paintings rather than real ones, these experiments show that DStroke can achieve results comparable to manual labeling on the real paintings. To justify it, we hope to further point out the following three facts: (1) Figure 14 shows the comparison of the brushstrokes extracted by DStroke and the manual segmentations on the real paintings rather than synthetic ones; (2) the brushstroke boundaries extracted by DStroke are closed as shown in the edge maps of column b; (3) DStroke does not miss any strokes compared to the manual segmentations as shown in column e. Moreover, our DStroke can work well on a large class of brush paintings, including oil paintings and Chinese paintings.

4.2 Comparing DStroke with Semantic Soft Segmentation

Our DStroke is also compared with the most relevant work: Semantic Soft Segmentation in [2]. The advantages of DStroke are, for hard segmentation, (1) there is no limitation of segment number; (2) it can do segmentation on instance level as shown in Figure 11(d). In contrast, due to lack of

Table 1. Evaluation and Comparison of DStroke and [7] (Higher Value is Better)

Painting ID	[7]			DStroke Method		
	meanIoU (%) (%)	Valid Rate (%)	Detection Rate (%)	meanIoU (%)	Valid Rate (%)	Detection Rate (%)
Mixed strokes1 (Figure 14, row 1, oil painting)	35.01	36.15	11.11	80.42	77.00	82.17
Beach (Figure 14, row 2, acrylic painting)	33.25	40.91	8.37	82.11	80.92	87.07
Mixed strokes2 (Figure 14, row 3, oil painting)	28.30	46.88	17.36	78.16	74.52	80.99
Mixed strokes3 (Figure 14, row 4, oil painting)	32.88	39.06	14.08	81.99	79.27	82.99
Cloud (Figure 14, row 5, acrylic painting)	31.86	38.64	12.86	86.66	88.43	90.95
Dusk1 (Figure 14, row 6, watercolor painting)	35.39	40.54	12.84	76.13	74.81	89.91
Dusk2 (Figure 14, row 7, watercolor painting)	22.56	38.71	12.12	75.68	68.97	79.70
Trees (Figure 14, row 8, acrylic painting)	33.19	48.72	21.09	76.72	74.47	82.03
Mixed strokes4 (Figure 14, row 9, oil painting)	35.23	34.13	11.37	76.04	70.26	77.84
Road (Figure 14, row 10, oil painting)	26.79	46.15	19.59	76.69	74.44	85.57
Landscape (Figure 14, row 11, watercolor painting)	27.53	35.00	11.79	79.22	78.00	88.97
White lotus (Figure 14, row 12, Chinese painting)	17.22	49.57	8.45	65.40	77.19	71.01
Yellow lotus (Figure 14, row 13, Chinese painting)	23.48	45.16	12.14	73.39	88.90	95.71
Flying bird (Figure 14, row 14, Chinese painting)	19.78	42.48	10.11	71.02	88.39	93.09
Peony (Figure 14, row 15, Chinese painting)	33.43	43.33	8.13	64.07	75.49	70.03

Table 2. Comparison with Tables 5 and 6 of [41] (Paintings are from [25] and the Higher Value is Better)

Painting ID	Valid rate (%)				Detection rate (%)				Fill rate (%)			Representative rate (%)		
	[25]	[41]	[7]	DStroke	[25]	[41]	[7]	DStroke	[25]	[41]	DStroke	[25]	[41]	DStroke
F218	42.7	20.2	88.1	86.3	21.6	20.5	48.5	52.3	29.1	55.2	62.2	2.1	22.3	27.3
F386	73.7	41.6	82.4	83.1	68.4	59.7	90.2	90.6	15.0	24.6	35.9	0.7	13.2	22.4
F518	60.7	35.3	71.2	75.2	75.2	60.1	78.9	81.9	24.2	40.3	51.5	8.4	29.8	28.0
F538	49.1	23.5	84.2	87.1	44.9	32.2	81.3	85.1	12.9	31.8	41.8	3.2	23.7	34.6

appropriate hard segmentations, Semantic Soft Segmentation [2] wrongly classifies many strokes into one stroke as shown in Figure 11(b). For soft segmentation, our DStroke works in an $O(n)$ complexity based on the guided filter [13]. For numerical evaluation, we used the synthetic painting group in our test painting set for comparison since all the brushstroke information is available in advance, which can be regarded as the ground truth. We select 500 synthetic paintings with

Table 3. Comparison of Brushstroke's Length, Width, and Orientation (We Ask Experts to Label These Four Paintings and Compare Our Results with Table 7 of [41])

Painting ID	MSD_α			$MSPD_\lambda$			$MSPD_\phi$		
	[25]	[41]	DStroke	[25]	[41]	DStroke	[25]	[41]	DStroke
F218	144.00	137.00	95.00	1.29	1.06	0.54	2.40	0.86	0.45
F386	112.00	94.00	65.00	3.89	3.66	1.25	2.75	0.74	0.42
F518	47.00	68.00	24.00	1.44	0.79	0.26	2.76	0.45	0.15
F538	97.00	83.00	42.00	1.12	0.38	0.12	2.50	0.43	0.28

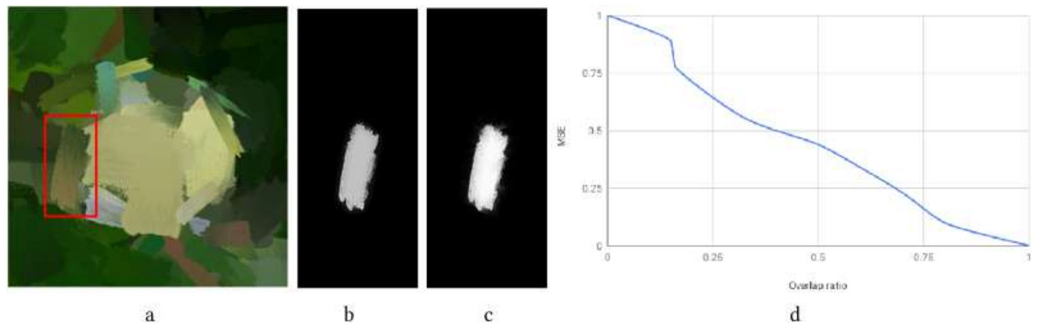

Fig. 12. Comparison of the alpha map of a segmented stroke and the ground truth. (a) Input painting. (b) Alpha map of a stroke segmented by Equation (7). (c) Alpha map of the ground truth. (d) MSE with varying overlap ratio.

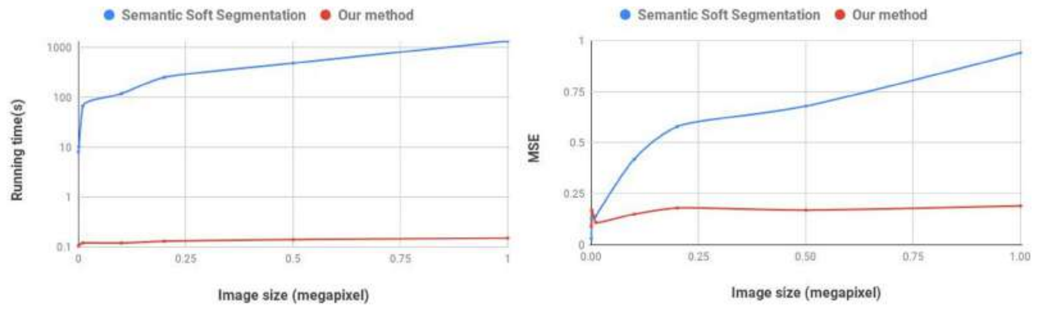

Fig. 13. Comparison of alpha MSE and running time at different size levels.

different sizes (from 0.1K to 1M) for test. The accuracy of brushstroke alpha mattes is computed by **MSE (mean squared error)**,

$$MSE = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N}(\alpha_{ij} - \widehat{\alpha}_{ij})^2\right),$$

where M denotes the stroke number, N denotes the pixel number within one stroke, and the alpha of the i-th pixel on the j-th brushstroke α_{ij} is generated by Equation (7), and $\widehat{\alpha}_{ij}$ is the corresponding ground truth. Figure 12 illustrates the alpha mattes of a stroke segmented by Equation (7) and the ground truth. Moreover, Equation (7) indicates that the errors of hard segmentations are linearly transferred to the alpha mattes. Figure 12(d) shows the example of MSE with varying the ratio of the detected hard segmentation over the ground truth hard segmentation, which is approximately linear. Thus, the soft segmentation entirely depends on the hard segmentation. Figure 13

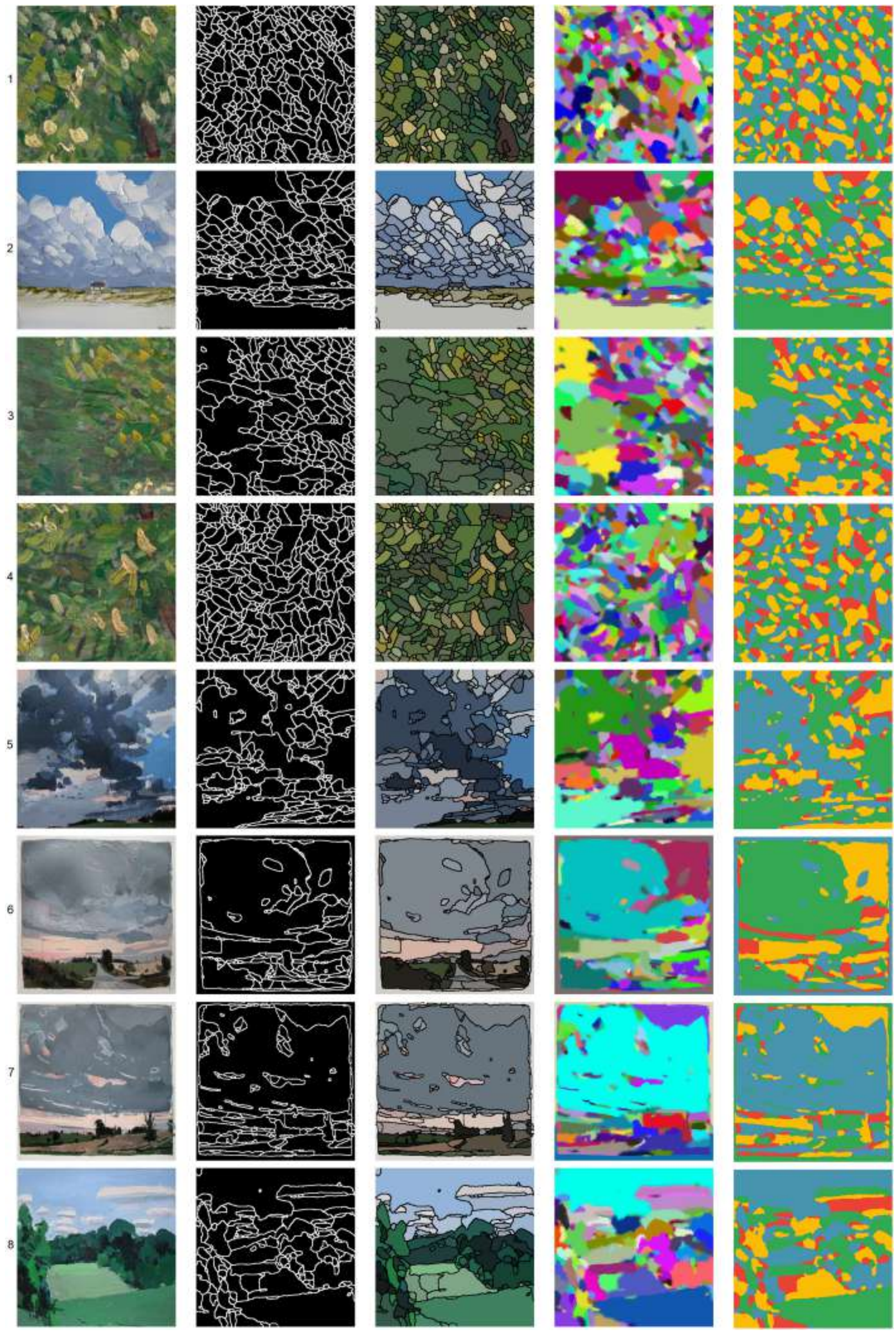

Fig. 14. Comparison of the results by DStroke and manual segmentation. (a) Input paintings. (b) Detected brushstroke edges. (c) Hard segmentation. (d) Soft segmentation. (e) Manual segmentation. We use the dominant color with segmentation boundaries in (c) to easily compare the hard segmentation with the brushstrokes of the paintings in (a). However, to highlight transition areas in soft segmentation, we still use random colorization in (d) to ensure a big color difference between any two adjacent brushstrokes.

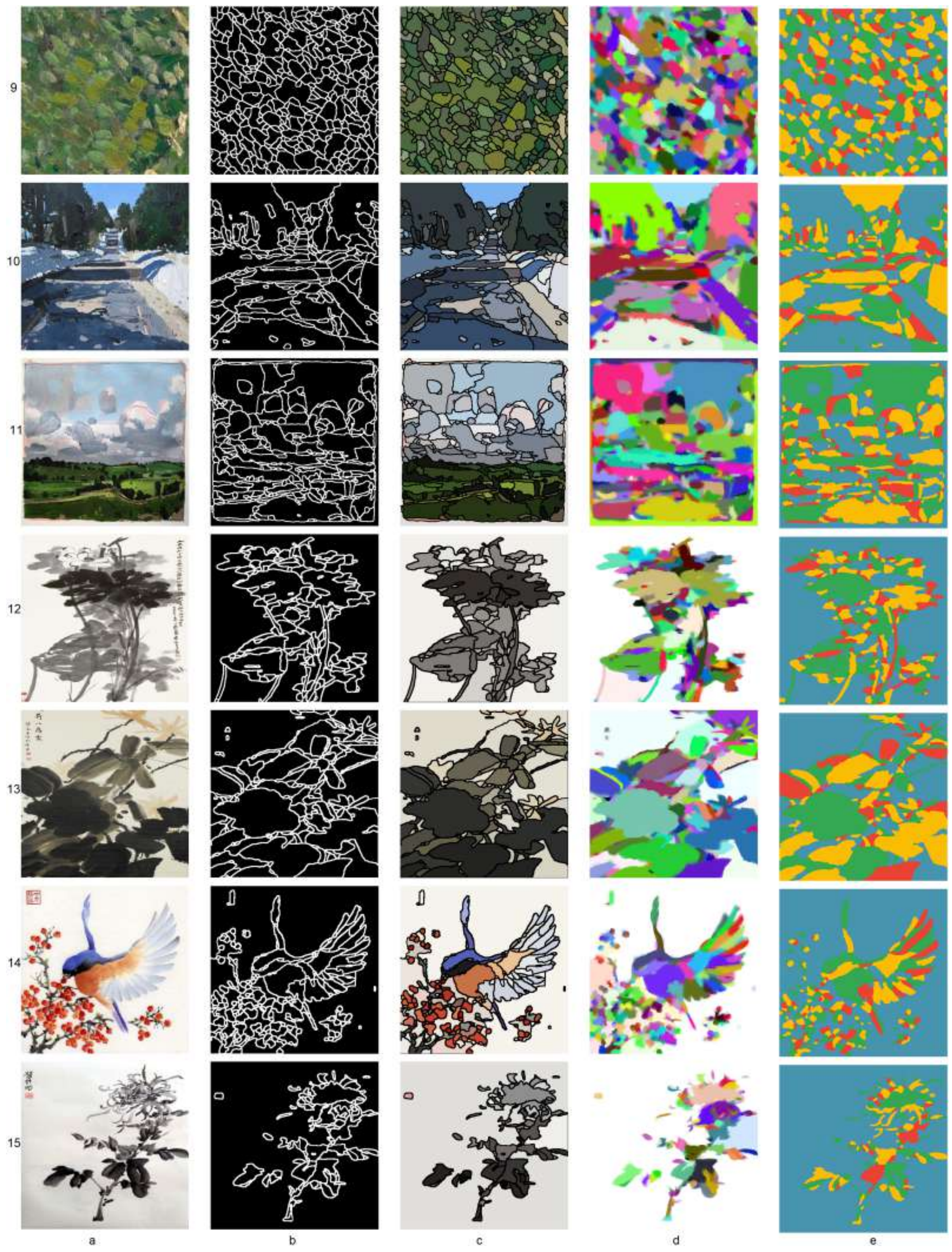

Fig. 14. Continued.

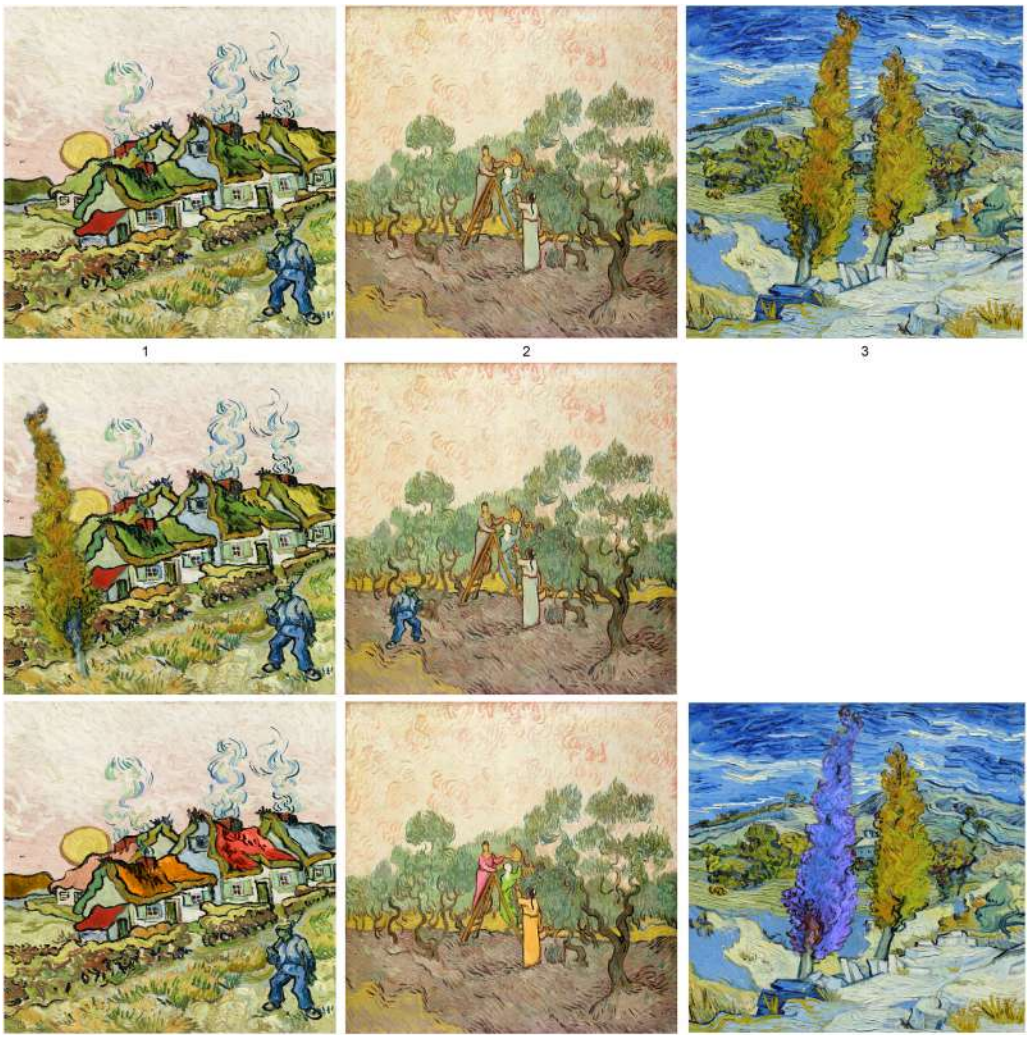

Fig. 15. Image editing. Row 1 shows the whole real paintings. Row 2 shows the results of inserting objects. A tree in the painting 3 is inserted in painting 1. The farmer in painting 1 is inserted into painting 2. Row 3 shows the results of recoloring paintings.

further shows the comparison of the proposed Dstroke method and Semantic Soft Segmentation [2] on both running time and alpha matte accuracy. It can be noted that the MSE of our DStroke is obviously less than [2]. Whether overlapping or changing image size, a big error is introduced by [2] into hard segmentation. In contrast, DStroke can continually provide a more accurate hard segmentation despite an increasing image size, hence reducing MSE.

In summary, compared to Semantic Soft Segmentation, our DStroke method has a better performance in the following aspects:

- DStroke is able to do soft segmentation on instance level.
- DStroke has no limitation on the segment number enabling soft segmentation for thousands of strokes, whereas Semantic Soft Segmentation can only segment a small number of regions.

- The accuracy of DStroke's alpha mattes is noticeably higher than Semantic Soft Segmentation.
- The running time of the DStroke method is much less than Semantic Soft Segmentation. For a 640×480 image, Semantic Soft Segmentation takes around 3 minutes on segmentation, while DStroke takes less than 1 second.

4.3 Applications in Image Editing

When almost all the brushstrokes in a painting are extracted, it is likely to recompose the painting through manipulating the strokes. Figure 15 shows two applications in image editing: recoloring paintings through changing brushstrokes' colors and inserting objects into paintings. To make inserted objects and recolored colors look reasonable and vivid, we performed the DStroke for brushstroke extraction on three whole real paintings and then carried out inserting and recoloring separately.

5 CONCLUSION

In this article, we propose a Deep learning–based brush Stroke extraction method, DStroke, which consists of two parts: (1) edge detection (or hard segmentation) through the modified Pix2Pix network; and (2) soft segmentation by the guided filter. Compared to the state-of-the-art methods, the main merit of the proposed DStroke is its high efficiency, i.e., to automatically and rapidly extract close to all the brushstrokes from a brush painting and accurately recover the faithful soft transitions between brushstrokes. The numerical results show that our DStroke can accurately approximate brushstrokes with high reliability.

Our main contributions include (1) proposing a painting production method that automatically builds up a large brush painting training dataset without the need for manual annotation. To the best of our knowledge, we are the first to provide an automatic method of building up a large painting sample dataset for deep learning purposes. (2) The modified Pix2Pix network can do segmentation on instance level and output "labels" for segmentation. In addition, we also apply the estimated alpha mattes of brushstrokes to identifying stroke ordering and coloring issues for painting authentication purposes. Experimental results further demonstrate that our DStroke method outperforms the current state-of-the-art methods.

Limitations. Due to the diversity of drawing art, our DStroke is unsuitable for traditional (or classic) western paintings, whose style was developed in the Renaissance and emphasized realism such as "Mona Lisa" by da Vinci, and abstract paintings like "Composition 8" from Vasily Kandinsky. Additionally, it cannot deal with the mixed color strokes that appear in watercolor and oil paintings with brush mixing and pickup. These limitations direct our further research.

We plan to build up an old masters' brushstroke database using the proposed DStroke method in the near future. Moreover, we are also interested in the recent work, alphaGAN [21], and aim to integrate hard segmentation and soft segmentation into the GAN model soon.

REFERENCES

[1] Yagiz Aksoy, Tunc Ozan Aydin, and Marc Pollefeys. 2017. Designing effective inter-pixel information flow for natural image matting. In *Proceedings of CVPR'17*. DOI: 10.1109/CVPR.2017.32

[2] Y. Aksoy, T.-H. Oh, S. Paris, M. Pollefeys, and W. Matusik. 2018. Semantic soft segmentation. *ACM Transactions on Graphics* 37, 4 (2018), 72. DOI: doi.org/10.1145/3197517.3201275

[3] Elad Aharoni-Mack, Yakov Shambik, and Dani Lischinski. 2017. Pigment-based recoloring of watercolor paintings. In *Proceedings of the ACM Symposium on Non-Photorealistic Animation and Rendering (NPAR '17)*, Article 1. DOI: doi.org/10.1145/3092919.3092926

[4] I. E. Berezhnoy, E. O. Postma, and H. J. van den Herik. 2009. Automatic extraction of brushstroke orientation from paintings. *Machine Vision and Applications* 20, 1 (2009). DOI: doi.org/10.1007/s00138-007-0098-7

[5] Q. Chen, D. Li, and C.-K. Tang. 2013. KNN matting. *IEEE Transactions on PAMI* 35, 9 (2013), 2175–2188. DOI: 10.1109/TPAMI.2013.18
[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, aAtrous convolution, and fully connected CRFs. *IEEE Transactions on PAMI* 40, 4 (2017), 834–848. DOI: 10.1109/TPAMI.2017.2699184
[7] Y. Fu, H. Yu, C. Yeh, J. J. Zhang, and T. Lee. 2018. High relief from brush painting. *IEEE Transactions on Visualization and Computer Graphics*, DOI:10.1109/TVCG.2018.2860004
[8] B. Gooch, G. Coombe, and P. Shirley. 2002. Artistic vision: Painterly rendering using computer vision techniques. In *Proceedings of the ACM 2nd International Symposium on Non-Photorealistic Animation and Rendering*. 83–ff. DOI: 10.1145/508530.508545
[9] A. Hertzmann. 2003. A survey of stroke-based rendering. *IEEE Computer Graphics and Applications* 23, 4 (2003), 70–81. DOI: 10.1109/MCG.2003.1210867
[10] A. Hertzmann. 2002. Fast paint texture. In *Proceedings of the ACM 2nd International Symposium on Non-Photorealistic Animation and Rendering (NPAR'02)*. 91–ff. DOI: doi.org/10.1145/508530.508546
[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask r-CNN. In *Proceedings of IEEE ICCV'17*. 2980–2988. DOI: 10.1109/ICCV.2017.322
[12] S. Hegde, C. Gatzidis, and F. Tian. 2013. Painterly rendering techniques: A state-of-the-art review of current approaches. *Computer Animation and Virtual Worlds* 24, 1 (2013), 43–64. DOI: doi.org/10.1002/cav.1435
[13] K. He, J. Sun, and X. Tang. 2013. Guided image filtering. *IEEE Transactions on PAMI* 6 (2013), 1397–1409. DOI: 10.1109/TPAMI.2012.213
[14] T. Hurtut. 2010. 2D Artistic Images Analysis, a Content-Based Survey. (2010). Retrieved from http://hal.archivesouvertes.fr/hal-00459401_v2/.
[15] Zhongyi Han, Benzheng Wei, Ashley Mercado, Stephanie Leung, and Shuo Li. 2018. Spine-GAN: Semantic segmentation of multiple spinal structures. *Medical Image Analysis* 50 (2018), 23–35. DOI: doi.org/10.1016/j.media.2018.08.005
[16] Krassimira Ivanova, Peter Stanchev, Koen Vanhoof, and Phillip Ein-Dor. 2010. Semantic and abstraction content of art images. In *Proceedings of the Mediterranean Conference on Information Systems 2010*. http://aisel.aisnet.org/mcis2010/42.
[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of CVPR'17*. DOI: 10.1109/CVPR.2017.632
[18] Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. 2018. Stroke controllable fast style transfer with adaptive receptive fields. In *Proc.eedings of the European Conference on Computer Vision 2018*. 238–254. DOI: 10.1007/978-3-030-01261-8_15
[19] Y. Koyama and M. Goto. 2018. Decomposing images into layers with advanced color blending. *Computer Graphics Forum* 37 (2018), 397–407. DOI: 10.1111/cgf.13577.
[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* 60, 6 (2017), 84–90. DOI: doi.org/10.1145/3065386
[21] S. Lutz, K. Amplianitis, and A. Smolic. 2018. AlphaGAN: Generative adversarial networks for natural image matting. In *Proceedings of the British Machine Vision Conference 2018*. Retrieved from https://arxiv.org/abs/1807.10088.
[22] C. Li and T. Chen. 2009. Aesthetic visual quality assessment of paintings. *IEEE Journal of Selected Topics in Signal Processing* 3, 2 (2009), 236–252. DOI: 10.1109/JSTSP.2009.2015077
[23] A. Levin, D. Lischinski, and Y. Weiss. 2008. A closed-form solution to natural image matting. *IEEE Transactions on PAMI* 30, 2 (2008), 228–242. DOI: 10.1109/TPAMI.2007.1177
[24] A. Levin, A. Rav-Acha, and D. Lischinski. 2008. Spectral matting. *IEEE Transactions on PAMI* 30, 10 (2008), 1699–1712. DOI: 10.1109/TPAMI.2007.1177
[25] J. Li, L. Yao, E. Hendriks, and J. Z. Wang. 2012. Rhythmic brush-strokes distinguish van Gogh from his contemporaries: Findings via automated brushstroke extraction. *IEEE Transactions. on PAMI* 34, 6 (2012), 1159–1176. DOI: 10.1109/TPAMI.2011.203
[26] J. McCann and N. Pollard. 2009. Local layering. *ACM Trans. on Graphics* 28, 3 (2009), 84. DOI: doi.org/10.1145/1576246.1531390
[27] J. McCann and N. S. Pollard. 2012. Soft stacking. *Computer Graphics Forum* 31 (2012), 469–478. DOI: doi.org/10.1111/j.1467-8659.2012.03026.x
[28] T. Porter and T. Duff. 1984. Compositing digital images. In *Proceedings of ACM SIGGRAPH84-Computer Graphics*, Vol. 18, 253–259. DOI: doi.org/10.1145/964965.808606
[29] O. Ronneberger, P. Fischer, and T. Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention (MICCAI'15)*. Springer, LNCS Vol. 9351. DOI: 10.1007/978-3-319-24574-4_28

[30] C. Richardt, J. Lopez-Moreno, A. Bousseau, M. Agrawala, and G. Drettakis. 2014. Vectorising bitmaps into semi-transparent gradient layers. *Computer Graphics Forum* 33 (2014), 11–19. DOI: doi.org/10.1111/cgf.12408

[31] Lior Shamir. 2015. What makes a Pollock Pollock: A machine vision approach. *International Journal of Arts and Technology* 8, 1 (2015), 1–10. DOI: 10.1504/IJART.2015.067389

[32] D. Singaraju and R. Vidal. 2011. Estimation of alpha mattes for multiple image layers. *IEEE Transactions on PAMI* 33, 7 (2011), 1295–1309. DOI: 10.1109/TPAMI.2010.206

[33] Jianchao Tan, Jose Echevarria, and Yotam Gingold. 2018. Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. *ACM Transactions on Graphics* 37, 6 (2018), Article 262, DOI: doi.org/10.1145/3272127.3275054

[34] J. Tan, J.-M. Lien, and Y. Gingold. 2016. Decomposing images into layers via RGB-space geometry. *ACM Transactions on Graphics* 36, 1 (2016), 7. DOI: doi.org/10.1145/2988229

[35] Laurens J. P. van der Maaten and Eric O. Postma. 2010. Texton-based analysis of paintings. In *Proceedings of the SPIE*, Vol. 7798, id.77980H. DOI: doi.org/10.1117/12.863082

[36] Ning Xie, Hirotaka Hachiya, and Masashi Sugiyama. 2013. Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. *IEICE Transactions on Information and Systems* E96.D, 5 (2013), 1134–1144. DOI: 10.1587/transinf.E96.D.1134

[37] S. Xu, Y. Xu, S. B. Kang, D. H. Salesin, Y. Pan, and H.-Y. Shum. 2006. Animating Chinese paintings through stroke-based decomposition. *ACM Transactions on Graphics* 25, 2 (2006), 239–267. DOI: doi.org/10.1145/1138450.1138454

[38] N. Xie, T. Zhao, F. Tian, X. Zhang, and M. Sugiyama. 2015. Stroke-based stylization learning and rendering with inverse reinforcement learning. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15).* 2531–2537. DOI: 10.5555/2832581.2832603

[39] Ningyuan Zheng, Yifan Jiang, and Dingjiang Huang. 2019. StrokeNet: A neural painting environment. In *Proceedings of the International Conference on Learning Representations 2019.* Retrieved from https://dblp.org/rec/conf/iclr/ZhengJH19.html.

[40] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. 2017. Pyramid scene parsing network. In *Proceedings of IEEE CVPR'17.* 6230–6239. DOI: 10.1109/CVPR.2017.660

[41] F. Lamberti, A. Sanna, and G. Paravati. 2014. Computer-assisted analysis of painting brushstrokes: Digital image processing for unsupervised extraction of visible features from van Gogh's works. *EURASIP Journal on Image and Video Processing* 2014 (2014), 53. DOI: doi.org/10.1186/1687-5281-2014-53

[42] S. Zhang, T. Chen, Y. Zhang, S. Hu, and R. R. Martin. 2009. Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics* 15, 4 (2009), 618–629. DOI: 10.1109/TVCG.2009.9

[43] Robert L. Cook and Kenneth E. Torrance. 1981. A reflectance model for computer graphics. *SIGGRAPH Computer Graphics* 15, 3 (August 1981), 307–316. DOI: https://doi.org/10.1145/965161.806819

[44] Retrieved from https://www.kylebrush.com/.

[45] Liqiang Nie, Meng Liu, and Xuemeng Song. 2019. Multimodal Learning toward Micro-Video Understanding, Multimodal Learning toward Micro-Video Understanding, Morgan-Claypool. DOI: 10.2200/S00938ED1V01Y201907IVM020

[46] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. 2009. From contours to regions: An empirical evaluation. In *IEEE CVPR'09.* 2294–2301, DOI: https://doi.org/10.1109/CVPR.2009.5206707

[47] Jingwan Lu, Connelly Barnes, Stephen DiVerdi, and Adam Finkelstein. 2013. RealBrush: Painting with examples of physical media. *ACM Transactions on Graphics* 32, 4 (2013), Article 117, 12 pages. DOI: https://doi.org/10.1145/2461912.2461998

Received April 2020; revised August 2020; accepted October 2020