Pacific Graphics (2014)
J. Keyser, Y. J. Kim, and P. Wonka (Editors)

Image Palette: Brushstroke Synthesis-based Style Transfer

Zheng Miao[1,2] Yan Zhang[†1,2] Zhibin Zheng[1,2] Zhengxing Sun[1,2]

[1]State Key Lab for Novel Software Technology, Nanjing University, China
[2]Department of Computer Science and Technology, Nanjing University, China

Abstract

Painting style transfer is one kind technology where given the sample images with some specific art style, we can render the target images in the same style as the samples after some computation. In this paper we present a new approach of painting style transfer in which such a style transfer work is done by analogy with simulating the process of creation. We take the sample as palette where users can select arbitrary outlines or textures as the current input mode brush strokes. We then analyze brush strokes' style feature information and use such information for the style transfer and synthesis along the stroke curves learned from the specified area in target images to get the same painting style as the samples. The results show that the users can get a style-transferred personalized target image just by the given sample images and least interactions.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation I.3.m [Computer Graphics]: visual arts—image-based modeling

1. Introduction

In recent years, the stylized painting has become a hot topic, for it can simulate the processes and effects of different kind art styles based on the image identifying and analysis with the help of artistic ideas and artistic standards of artistic creation. The painting stylization can be divided into two categories: simulation-based [ZHT07] [ZZXZ09] [ZZ11]and example-based [HJO*01] [LSRY10] [RLC*06] [LFB*13]. Simulation-based paintings focus on simulating the real painting process where the target painting is produced by adding brush strokes of different colors and sizes on the panel. Example-based paintings, however, simulate the samples' properties by texture-synthesis, thus make the targets preserve their own contents meanwhile carry on the style of the samples.

In this paper we propose a novel approach of painting style transfer, which combines two methods of image stylized painting and implement realistic stroke synthesis-based style transfer of images based on analyzing features of the existing stylized image samples. Our algorithm can be divided into two phases – first setting stokes based on areas and then transfer synthesis based on strokes. Specifically, during the setting stroke phase, we refer to the ideas of brush stroke generation in simulation-based rendering techniques. We segment target images effectively and obtain brush strokes covering each one of the regions according to the size and orientation of that region. During the transfer synthesis phase, we use input sample images as a palette in which the user can select any contour or texture as input mode of each brush stroke, and then we analyze texture, color, direction and other features information of selected area. Then, we use the method of strokes transfer synthesis based on irregular quadrangles, which is proposed in this paper, along current stroke curve with help of these feature information to implement transfer synthesis of each stroke. Finally, after all brush strokes are completed, we can finish overall painting style transfer of the target image.

2. Stroke Settings

To complete the style transfer of strokes by areas, firstly, we use sketch tokens [LZD13]to segment the target image. Then we lay strokes along boundaries and within the areas. We define the simulated stroke as $S = (\theta(\text{orientation}), w(\text{width}), l(\text{length}), t(\text{texture}))$.

2.1. Settings of strokes in regional boundaries

As sketch tokens can segment the target image by areas, it is easy to obtain the contour of each region. So we denote the

† zhangyannju@nju.edu.cn

DOI: 10.2312/pgs.20141252

direction of strokes of boundariesas the contour's trend of the region. Stroke's width w will be set according to users' needs. We set the stroke's width to 5-6 boundary pixels in experiments. The reason for choosing this number is that if the width is too small, the stroke synthesis will be difficult, whereas the larger width will make the boundaries too obvious, thus influence the overall effect. Stroke's length l is the length of the contour of the area. Stroke's texture information t will be used to transfer style according to boundary strokes selected by users in the sample image. This is the focus of this paper and will be described in detail in Section 4.

2.2. Settings of strokes in areas

Similarly with strokes of regional boundaries, next we will introduce how to obtain each parameter of the stroke within areas.

Stroke's orientation (θ): We hope that the general orientations of strokes can be visually consistent with the orientation of the texture and the shape of the object, which are often associated with the boundary and the vector fields. Therefore the vector field, which is calculated within the divided region, will be able to effectively guide the orientation of strokes. We set each stroke's orientation as the orientation of the vector field θ.

Stroke's width (w): In principle, users can set width for strokes freely. But the stroke's width set by the user will affect various regions of coverage and computing of subsequent stroke transfer synthesis. If the stroke's width is too small brush strokes that cover each area will be too many, which is not conducive to calculating the sample-based strokes transfer synthesis. If the stroke's width is too large brush strokes that cover each area will be too few, which will reduce randomness of results of the sample-based strokes transfer synthesis. Therefore, we set the stroke's width between 14-20 pixels in our experiment.

Stroke's length (l): The length of stroke is related to the region which needs to be rendered. After determine the orientation of the brush strokes, the deformed scan line should not exceed the boundary of the region. Hence the length of the fitted curve within the region is the length of the current stroke.

Stroke's texture (t): In order to complete the painting style transfer, strokes' texture in the result image should preserve the original color of the target image as well as texture information of the sample image. This is the focus of this paper, and we will introduce this in detail in the next section.

With the above parameters (stroke texture not included) and size of each area in the target image after segmentation, we will be able to get coverage of all the empty strokes of the target image (we will call them empty strokes because there is no texture information of strokes). As shown in Figure 1, Figure 1 (a) is the target image, and Figure 1 (b) is the image after segmentation. In Figure 1 (b), we show the empty strokes along the regional boundaries and within the area in the target image.

Figure 1: *Settings of strokes. (a)Target image. (b)Result of segmentation for (a) and strokes laid along the extension of the texture. (c)Results of style transfer (d)Results of color transfer*

3. Stroke transfer synthesis

After learning each empty stroke, we need to take the sample image as a palette and let users select their preferred areas as inputs then start stroke-based transfer synthesis. In order to maintain the style of the sample image, we choose the boundaries and areas in sample image as the input to do the transfer synthesis in the target images. After getting the corresponding input, we analyze features of the input sample with a certain style to obtain high-level style information. Then we optimize empty strokes along the curves using the synthesis feature in order to get smooth and complete stroke texture.

3.1. Feature analysis on input stylized samples

As mentioned in related work, the existing style transfer method [LSRY10] [RLC*06] [LFB*13]often use pixel-like low level features in the sample image to describe the style and rarely involves some high-level features. However, the formation of many styles is associated with high-level features such as texture and orientation of strokes. Therefore, the good use of these high-level features will help to express the original style of the sample image. In this paper, we mainly analyze the high-level features including orientation of strokes, texture features and color features in the selected sample image. Next we will describe how to obtain these features.

Texture feature: In this paper we choose the LBP texture feature analysis method to complete this work. Ojala proposed LBP [OPM02] based on any neighborhood to improve the LBP operator. Here we use the latter one to encode the texture sample to obtain the texture features.

Color features: We still need to preserve important pixel-like low level features in the existing style transfer methods [LSRY10] [RLC*06] [LFB*13] since during the synthesis process, the continuity of the pixels will be able to better maintain the continuity of the original samples. In this paper, we use color features and texture features to achieve

the strokes optimization transfer synthesis based on irregular quadrangles.

3.2. Stroke transfer synthesis based on irregular quadrangles

After obtaining empty strokes along the boundaries as well as those within areas in Section 3, the main objective is filling these empty strokes based on the style of the sample image selected by users. This is equivalent to synthesizing unknown regions with a known area, which is similar to the idea of texture synthesis. In this paper, we use high-level feature information of the sample combined with the offset statistical methods proposed by He [HS12] to achieve a rapid optimization transfer synthesis for empty strokes' texture.

3.2.1. Segmentation of strokes to be synthesized

Figure 2: *Segmentation of empty strokes. (a)Empty stroke. (b)Segmentation by quadrangles.*

Since empty strokes set in Section 3 each has a certain curvature, it is difficult to split empty strokes, especially the strokes of borders into structured pieces with the same size. But we can divide empty stroke to be synthesized into a number of irregular quadrangles with the help of strokes' width and length. In Figure 2, it is assumed Figure 2 (a) is empty stroke S to be synthesized currently. We know the direction θ, the width w and the length l of the stroke. It is easy to obtain the inner and outer two contours T and B of the S along the stroke's direction, as shown in Figure 2 (b). Sampling along the two contours equidistantly, we can divide empty stroke S to be synthesized into a number of quadrangles. We denote the divided quadrangle as P_i, which has the coordinates$((T_{x_i},T_{y_i}),(B_{x_i},B_{y_i}),(T_{x_{i+1}},T_{y_{i+1}}),(B_{x_{i+1}},B_{y_{i+1}}))$, wherein $(x_i{=}i,y_i{=}i)$. As Figure 2 (b) shows, the quadrangle obtained by the division is relatively small, so the synthesis process can not only ensure the smoothness of the original curve, completeness, but also omit further deformation of the sample to synthesize.

3.2.2. Construction of synthetic optimization function

Our synthetic procedure is to find a matched quadrangle for each quadrangle in the specified sample. Since our strokes to be synthesized are curves (Especially the border of the region, which is a relatively large curvature). In order to maintain a continuous texture under the premise to get the original texture consistent with the direction of the target image drawing brush, we need to rotate the sample strokes along

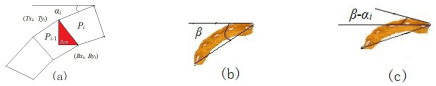


Figure 3: *Rotation of sample strokes. (a)Direction of P_i. (b)Direction of the sample stroke. (c)Rotating the sample stroke.*

the direction of texture in the target image when synthesizing a quadrangle each time to maintain the direction and consistency of brush strokes and the curves. The specific method is shown in Figure 3. For this quadrangle P_i to be synthesized, as shown in Figure 3 (a), we first calculate the angle α_i between the direction of the stroke and the horizontal. Then we also need to obtain angle β between the direction of the sample stroke and the horizontal to make the sample stroke rotate along the texture. As Figure 3 (b) shows, the angle is between the original direction of the sample stroke in the sample image and the horizontal. Finally, to make the sample stroke toward the direction with consistency of the curve, we take the upper right corner as the center of rotation and rotate the sample stroke (β-α_i) degree clockwise. The result after rotation is shown in Figure 3 (c), and the sample here is the sample for further match of P_i in the next. As Figure 4 shows, in pretreatment, we divide the empty stroke into 200 small quadrangles in Figure 4 (b). In order to maintain the texture integrity and the direction of the stroke, we use our method to rotate the input sample strokes in Figure 4 (a) many times and obtain input sample strokes (the first 10) in Figure 4 (c). Each subsequent small quadrangle will be searched for the matched one in the corresponding input sample stroke after rotation when synthesizing. The final result of synthesis is shown in Figure 4 (d). Next, we will describe in detail how to obtain results in Figure 4 (d).

Figure 4: *Generation of strokes. (a)Sample stroke. (b)Empty stroke divided by 200 small quadrangles. (c)Rotating the sample stroke. (d)Result of synthesis.*

In order to maintain the continuity of the original sample, we need a certain amount of overlap between the blocks similar to the idea of texture synthesis. In order to facilitate the subsequent matching, we determine the overlap area by adjacent quadrilateral boundaries. As Figure 3 (a) shows, we take the boundary between the adjacent blocks as a diagonal to generate a triangle *Box* connecting two small quadrangles. Here we assume that the coordinates of the adjacent edge of the current two quadrangles are (T_{x_i}, T_{y_i}), (B_{x_i}, B_{y_i}). It is easy obtain the coordinates of the triangle *Box*, which is $((T_{x_i},$

T_{y_i}),(T_{x_i}, B_{y_i}),(B_{x_i}, B_{y_i})) as shown in Figure 3 (a). Besides, the continuity of the area can be ensured by searching the matched triangle since the triangle connects two quadrangles. The main reason for taking the triangular as the overlap region is to make sure that pixels involved in the calculation have real eigenvalues in the subsequent matching search. Because pixels involved in the triangular has been synthesized in the previous quadrangle, so they have practical eigenvalues.

The process of matching searching can be seen as a problem of coherence measure [BSFG09]. The energy equation can be defined as:

$$d_{cohere} = \sum_{P \in S} \min_{Q \in \Omega} ||P - Q||^2 \quad (1)$$

In equation (1) P is the quadrilateral block divided in the stroke S. Q is the matching block for P in the sample area selected by the user. During our current calculation, to maintain the continuity of the region, we will use the quality of matching for the triangular connecting the two quadrilateral areas, and therefore the above equation can be rewritten as:

$$d_{cohere} = \sum_{Box \in S} \min_{R \in \Omega} ||Box - R||^2 \quad (2)$$

In equation (2) *Box* is the triangle area connecting the adjacent quadrangle. R is the matched area for *Box* in the sample region selected by the user.

3.2.3. Optimization

With the energy equation (2), We use the method of statistics of patch offsets proposed by He et al [HS12] to optimize it. In this paper, we make some improvements according to the characteristics of the problem. The specific implementation process is as follows.

Similar patch matching: He et al use only color information of pixels in each patch to calculate when they obtain match for each patch in [HS12]. However, in Section 4.1 we analyze that not only color information, but also directions, texture, and other high-level information can affect the style of the sample image. In this paper we consider these two factors in patch matching search and make the following improvements. (1) In this paper, the direction in search is vector field direction in the sample selected by the user. (2) We involve the texture feature which is a one-dimensional feature of each pixel obtained in Section 4.1 in the calculation with the color feature when we search the match for each patch. The specific equation is as follows:

$$offset(X) = \arg\min_{offset} ||E(X + offset) - E(X)||^2 \quad (3)$$

$$s.t. |offset| > \tau$$

Among this equation *offset(X)* is the offset between *E(X)* and its optimal match patch *E(X+offset)*. *E(X)*is a patch whose center is X and size is $w*w$. τ is the threshold value set to avoid the match patch falling into the neighborhood. We represent each pixel in patch by a four-dimensional vector which is the RGB color information and the LBP texture feature obtained in Section 4.1. The patch size set in this paper is 3*3. In the search process, due to small user-specified sample space, we use the method of global search to search along the direction obtained in Section 4.1 to build statistics of offset in the sample.

Offset histogram: After obtaining the offsets of all patches in the sample, we used the approach in [HS12] to make two-dimensional histogram. The equation is as follows:

$$H(u,v) = \sum_{X} \delta(offset(X)) = (u,v)) \quad (4)$$

In this equation δ is 1 when the condition is true, otherwise it is 0. *H(u,v)* represents number of blocks when *offset=(u,v)*. In order to reduce the effects of random noise, we use Gaussian smoothing filter on the histogram. We can get dominated K blocks offset after histograms are sorted in descending. We set K = 60, which will be the main basis for subsequent optimization search.

Optimization synthesis based on offsets: After getting the offset histogram, we can modify equation (2) in Section 4.2.2 to the form as follows:

$$d_{cohere} = \sum_{X \in S} \min_{offset \in \Omega} ||Box(X) - Box(X + offset_i)||^2 \quad (5)$$

$$1 \leq i \leq K$$

The equation means that the best match patch for each *Box* will be obtained according to statistics of the offsets in the sample selected by the user. Further, according to the relationship between coordinates of each *Box* and those of each quadrangle, it is easy to determine the final region of quadrangles. After completing synthesis of all quadrangles, we can get a complete stroke, as shown in Figure 4 (d).

Figure 5: *Optimization synthesis based on offsets. (a)Box of strokes in the boundary. (b)Box of strokes in the area.*

The above describes synthesis of a single stroke. In the actual synthesis process, we need to pay attention to the following two points: (1) When synthesizing strokes in the boundary, if the boundary is a closed curve, we use two *Box* to determine the last quadrangle. As Figure 5 (a) shows, we set two *Box* according to left and right two edges of the the last quadrangle and conduct match search. (2)When synthesizing strokes within areas, we synthesize strokes in order of left to right and top to bottom. In order to maintain the continuity of the stroke, when synthesizing intermediate strokes, we also use two *Box* to determine the current quadrangle. As Figure 5 (b) shows, the current quadrangle will be determined by the adjacent two quadrangles which have been synthesized. For example, when we synthesize the current quadrangle in Figure 5 (b), we will use the top and left edges of the quadrangle to determine the corresponding two *Box* and match search according to equation (5). As shown in Figure 5 (b), *Box* in blue and red will be used in the current calculation.

After we implement optimize synthesis for all empty strokes set in Section 3, we can complete the style transfer of the target image.

3.3. Color transfer

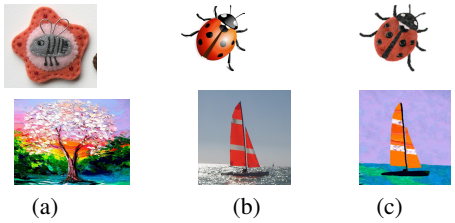
Because we use the color information in sample images in Section 4.2, results of style transfer do not have color information of the original image. We also provide a function of transfer color if users want to keep the color information of the original image. We use a global color transfer method proposed by Reinhard [RAGS01] to transfer color. The result is shown in Figure 1, wherein Figure 1 (a) is the original image, and Figure 1 (c) is the result of transfer synthesis. Figure 1 (d) is the result of color transfer. We can see that the color transfer can not only keep the style of the sample image, but also preserve the color information of the original image.

4. Results and Analysis

We realize the related algorithms based on the C + + language on the PC. The environment is as follows: Intel Core 2 Duo E7300 2.6CHz CPU, 2 GB of memory. We conduct lots of experiments using different styles of sample images, as well as target images of different structures.

Figure 6 shows style transfer results from our method. As for the running time, our method requires relatively long time only in part of image segmentation. Other parts such as empty strokes setting and strokes transfer can be completed in a few seconds. We need to produce a result image in size of 500* 500 on average 3 to 5 minutes(More results are shown in our supplementary material).

Our method proposed in this paper can also transfer style from multiple samples as shown in Figure 7(More results are shown in our supplementary material). Users can take multiple samples as a palette and select the style from different samples.

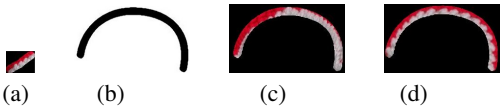


(a) (b) (c)

Figure 6: *Result of style transfer by our method. (a)Sample image. (b)Target image. (c)Result of style transfer from (a) to (b).*

(a) (b) (c)

Figure 7: *Results of style transfer from multiple samples. (a)Sample images. (b)Target image. (c) Result of style transfer from (a) to (b) (Result in (1) is color transferred partially in the leaf. Result in (2) is color transferred integrally.).*



(a) (b) (c) (d)

Figure 8: *Verify the method of rotating sample strokes. (a)Sample stroke. (b)Empty stroke. (c)Result without rotating sample strokes. (d)Result by the method of rotating sample strokes.*

To verify the method of rotating sample strokes mentioned in Section 4.2.2, we focus on brush strokes with a relatively large change rate in curvature and do the experiment shown in Figure 8. For the same input sample of strokes in Figure 8 (a) and the same empty stroke in Figure 8 (b), the synthesis result by the method of rotating sample strokes along the direction of empty strokes is shown in Figure 8 (c). The synthesis result without rotating sample strokes is shown in Figure 8 (d). We can find that the generated stroke is not able to maintain the continuity of the original sample and keep the direction of the stroke without rotating sample strokes.

In addition, similar to the [LFB*13], we also compare our method with the [RLC*06] [Ash01]as Figure 9 shows. We carry out experiment with the target image and the sample image in [LFB*13] where results in [RLC*06] [LFB*13] [Ash01] are shown in Figure 9 (b) - Figure 9 (e). It is easy to see that our result of style transfer shown in Figure 9 (f) is better than those in other papers.

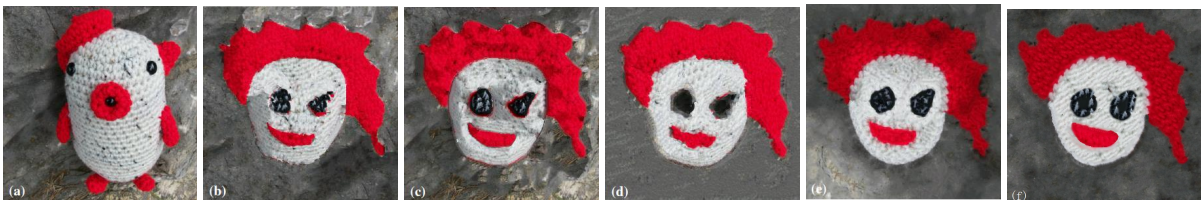


Figure 9: *Comparison of results from different approaches. (a) Sample image. Result of (b)Image Analogies [HJO*01], (c) Painting with Texture [RLC*06], (d) Synthesizing Natural Textures [Ash01], (e)Painting by Feature [LFB*13], (f) our approach.*

The main reasons are as follows: First of all, our method of painting starts from the perspective of simulation based on brushstrokes, which is consistent with the actual painting requirements. Furthermore, we add impact of computing texture features in the process of strokes transfer synthesis, therefore guarantee better smooth, continuity and randomness of strokes synthesis results.

Limitations: There are still some limitations in our method. First, the style transfer results are unsatisfactory when the images have complex structure information. Besides, setting the stroke to a fixed number of pixels makes the method scale-dependent. Lastly, since we allow users to take the sample images as a palette, the synthesis results are closely related to the areas selected by users. Therefore, due to the difference in user's selection each time for the same target image, it is difficult to obtain the same results after the second style transferring.

5. Conclusion and Future Work

This paper presents a new style transfer method based on brush strokes synthesis. In our method users can take any input sample images with a certain style as a palette and complete personalized art style transfer of the target images efficiently and automatically. For future work, we want to extend our method to be able to handle vector images. We also hope that our method will be extended to the video rendering or three-dimensional rendering and can be applied to more areas in the future.

6. Acknowledgement

We would like to thank all anonymous reviewers for their constructive comments. This research has been supported by the National Science Foundation of China (61321491, 61100110, 61272219) and the Science and Technology Program of Jiangsu Province (BY2012190, BY2013072-04).

References

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 217–226. 5, 6

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. 28*, 3 (July 2009), 24:1–24:11. 4

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 327–340. 1, 6

[HS12] HE K., SUN J.: Statistics of patch offsets for image completion. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part II* (Berlin, Heidelberg, 2012), ECCV'12, Springer-Verlag, pp. 16–29. 3, 4

[LFB*13] LUKÁČ M., FIŠER J., BAZIN J.-C., JAMRIŠKA O., SORKINE-HORNUNG A., SÝKORA D.: Painting by feature: Texture boundaries for example-based image creation. *ACM Trans. Graph. 32*, 4 (July 2013), 116:1–116:8. 1, 2, 5, 6

[LSRY10] LEE H., SEO S., RYOO S., YOON K.: Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2010), NPAR '10, ACM, pp. 43–48. 1, 2

[LZD13] LIM J. J., ZITNICK C. L., DOLLÃąR P.: Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR* (2013), IEEE, pp. 3158–3165. 1

[OPM02] OJALA T., PIETIKÄINEN M., MÄENPÄÄ T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 7 (July 2002), 971–987. 2

[RAGS01] REINHARD E., ASHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Comput. Graph. Appl. 21*, 5 (Sept. 2001), 34–41. 5

[RLC*06] RITTER L., LI W., CURLESS B., AGRAWALA M., SALESIN D.: Painting with texture. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2006), EGSR'06, Eurographics Association, pp. 371–376. 1, 2, 5, 6

[ZHT07] ZHANG E., HAYS J., TURK G.: Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (2007), 94–107. 1

[ZZ11] ZHAO M., ZHU S.-C.: Customizing painterly rendering styles using stroke processes. In *NPAR '11: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2011), ACM, pp. 137–146. 1

[ZZXZ09] ZENG K., ZHAO M., XIONG C., ZHU S.-C.: From image parsing to painterly rendering. *ACM Trans. Graph. 29*, 1 (Dec. 2009), 2:1–2:11. 1