Computers & Graphics 35 (2011) 81–91

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Extended papers from NPAR 2010

Directional texture transfer with edge enhancement

Hochang Lee, Sanghyun Seo, Kyunghyun Yoon *

Computer Graphics Laboratory, CS&E, Chung-Ang University, 221 Hukseok-Dong, Dong-Jak Gu, Seoul 156-756, Republic of Korea

ARTICLE INFO

Article history:
Received 28 August 2010
Received in revised form
15 November 2010
Accepted 15 November 2010
Available online 21 November 2010

Keywords:
Non-photorealistic rendering (NPR)
Texture transfer
Example-based rendering
Difference of direction

ABSTRACT

Texture transfer re-renders a target image with high-frequency information (texture) taken from parts of a reference image that is matched locally to the target image using characteristics such as color. In this paper, we add a directional factor based on the flow of the target image, creating a stroke-like effect that follows edges more accurately. In addition, we propose a method to clearly express object boundaries by considering the effect of the medium on edges in the reference image. We also show how to select an appropriate weight for this directional factor from the reference image. We demonstrate the suitability of this improved form of texture transfer for expressing various artistic styles, and compare our results with those from previous texture-transfer algorithms. We find that our algorithm can be adapted to texture synthesis as well.

1. Introduction

Artists construct a painting by the repetitive use of familiar and reliable brush strokes on a two-dimensional canvas. As shown in Fig. 1, paintings can have many different styles, which depend on the medium and the way in which the artist uses the brush to make strokes of different lengths, directions, and sizes. In particular, direction is a key attribute because brush strokes follow object boundaries to express their shape. A major goal of non-photorealistic rendering (NPR) is the mimicking of painting techniques. The different approaches include physical simulation of brushwork through user interaction [3], stroke-based rendering [13,9], and texture transfer based on a reference image [14,2]. In this paper, we focus on the texture-transfer technique. Texture transfer requires two input images: a reference image S, and a target image T. Each pixel in T is updated to the best among several candidate sets taken from S. The resulting image R is composed of the color of T and the texture of S. The strength of the texture-transfer approach is that different styles can be archived by changing S. Existing techniques use distance factors based on average color and spatial frequency to rank candidate pixels and hence find the best texture from the candidate set. Although these factors are effective in transferring the texture of S to T, the directional information in T is lost. Better use of directional information can help viewers to recognize object shapes and produce more artistic impressions. The importance of direction in painting is obvious in works by Van Gogh, whose brushstrokes clearly follow the direction of the edges. In stroke-based rendering, the image gradient is used to adjust the stroke direction. This is more difficult with a texture-transfer algorithm, which is pixel-based.

In this paper, we propose a new texture-transfer algorithm that includes a directional effect based on the image gradient of the target image. Our algorithm is a modification of an existing texture-transfer method that is fast and flexible. To ensure that the texture directions conform to the target image, we add a term to this algorithm to express neighborhood similarity in a way that takes the gradient direction into account. We first calculate the gradient of the target image to find the flow information. In order to find the most coherent texture, we calculate the average intensity of pixels laid down in the direction of flow at the current pixel, and compare it with each pixel in the candidate set, as explained in Section 4. This allows us to build a texture effect based on the image gradient. We can also use our algorithm for texture synthesis. It can create a good texture effect that follows the shape of an object.

A well-expressed boundary has the advantage of recognizing object shape. Materials such as watercolors, sumi-e, and line drawings have various boundary effects. To express a natural texture effect at object boundaries, we propose an edge-enhancement method by considering edge effects as shown in the reference image. We analyze changes in the intensity at the edge of S, and apply the analyzed information to edges of the target image. From this approach, we can express the object shape clearly and the material effects at boundaries.

We also show how to select a weight to apply to the directional factor taken from the reference image. We define the difference-of-direction (DoD) to be the average discrepancy between the direction of the original and interpolated gradients. The DoD value characterizes the flow in a reference image. An image with a coherent direction, like a Van Gogh painting, is likely to have a small

* Corresponding author.
E-mail addresses: hclee0126@cglab.cau.ac.kr (H. Lee), shseo@cglab.cau.ac.kr (S. Seo), khyoon@cau.ac.kr (K. Yoon).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.11.008

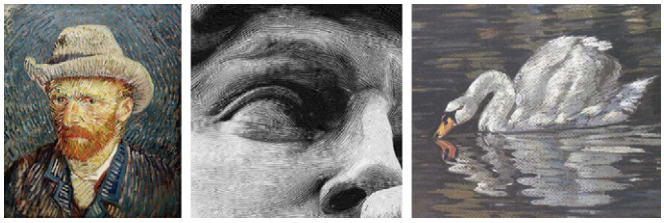


Fig. 1. Various style of painting.

DoD value. A watercolor or a pastel is likely to have a large DoD value. We assign a weight to the directional factor based on the DoD value of S.

The main contribution of this paper is to introduce a novel texture-transfer algorithm that considers the gradient of the target image. Although this is a pixel-based approach, we can express a coherent direction of stroke based on the image gradient. We can also express the edge enhancement of the target image by analyzing edge effects of materials, as shown in the reference image. Secondly, we propose a weight estimation method by analyzing the flow of the image. This approach should be further extended to enhance evaluation methods in NPR research.

2. Related works

Many NPR techniques produce painterly effects. This line of work began in the late 1980s [23,10]. From the late 1990s, [21,20,12,5] proposed techniques based on brushstrokes rather than pixels. Texture transfer is derived from texture-synthesis techniques that generate an arbitrarily large texture from a small sample texture [4,6,19,18,7]. Research in this area has been focused on the creation of self-similarity textures rather than artistic rendering.

In 2001, [14] introduced the image analogies technique. This method produces impressive results by globally optimal and locally coherent texture synthesis; however, it is time-consuming. In addition, the user must provide additional unfiltered information from the target image to achieve an exact correlation with the sample reference texture. In 2003, [2] proposed a fast texture-transfer algorithm, which directly extended a previous technique for locally coherent texture synthesis [1]. This algorithm produces results similar to image analogies in simulating artistic effects. In addition, it is much faster than Hertzmann's technique; it takes a few seconds for an image of 640×480 pixels, and requires only one reference image. Although this algorithm is focused on generating the texture and pattern of the reference image, it does not consider directional information. Ref. [24] proposed a texture-synthesis algorithm for painterly rendering, which takes account of stroke directions. They segmented the target image and created a direction field for each segment. They cut patches from the reference painting and attached them to create the output image. Their approach can express the direction of the target image and the texture of the reference image. However, for selecting good patches from reference image, user must to choose them. This approach is effective when user wants to express various styles in one result image. However, our approach is more effective to express one style in a result image. In addition, although our algorithm has a pixel-based approach, it is very simple and fast. In general, the result is generated in about 5–10 s when the image size is 720×480. Our algorithm also attempts to express the boundary effects of a material as shown in the reference image.

Direction is a very important factor among the many stroke attributes. In Zhang et al. [26], they propose the method to design and control a directional field that guides brush stroke orientations. In order to express stroke direction, many researchers have studied and utilized NPR effects. In 1999, Freeman et al. [8] proposed a method of generating coherent line drawings based on training data. Techniques by Salisbury et al. [22] propose a method for expressing coherent directions of pen effects. Litwinowicz [20], Hays and Essa [11] used interpolation of edge directions to achieve coherent stroke direction. This is relatively easy to do in stroke-based techniques. However, pixel-based approaches encounter difficulty in expressing the stroke direction and texture; therefore, they are not widely studied. In contrast to previous pixel-based approaches [14,2], we propose an algorithm that considers the image gradient of the target image. Our results are generated very quickly and full automatically based on determined parameters. Due to the low time cost, the proposed algorithm can also be applied to artistic animation.

3. Concept for expressing direction in pixel-based texture-transfer techniques

In stroke-based rendering (SBR) research, a stroke is a basic unit. The resulting image is filled with strokes. These techniques can consider the object shape by rotating the stroke based on the image gradient. Patch-based texture-synthesis approaches also express direction by rotating the patch based on the image gradient, as shown in Fig. 2. A general texture-transfer algorithm creates an image by filling in each pixel. As each pixel has only one color value, expressing image flow is difficult. Nevertheless, a pixel-based approach can express image flow by filling pixels with a similar color along the direction of the image gradient, as shown in Fig. 3. This is the basic idea of the directional texture-transfer technique introduced in Section 4.

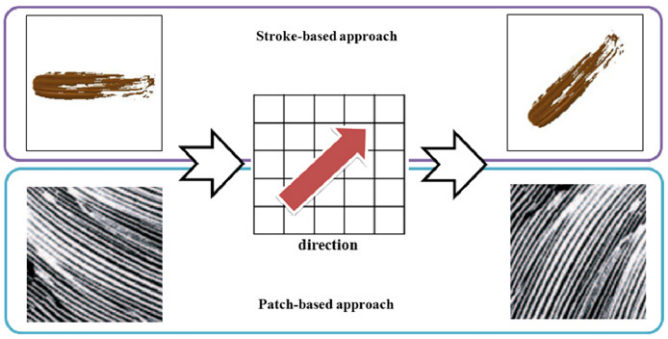


Fig. 2. Expressing direction in stroke-based and patch-based rendering.

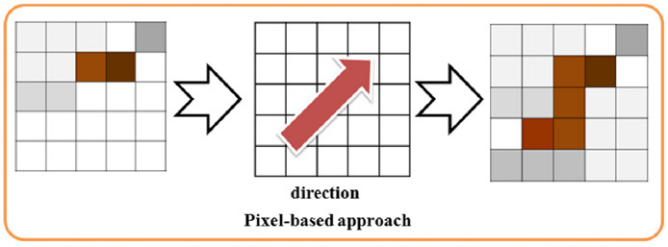


Fig. 3. Directional row in a pixel-based approach.

4. Directional texture transfer technique

Fig. 4 is a flowchart of our system. We extend the fast texture-transfer technique of Ashikhmin [2] with a new term that expresses the similarity of the neighborhoods while taking the direction of the gradient into account. We do this by extracting the flow from the target image and estimating weights from the reference image. We also extract edges from the target and reference images for expressing boundary effects. Let S, T, and R denote the reference image, the to-be-filtered image, and the resulting image. Let $N(x)$ also denote the square neighborhood of a pixel x, and let $L(x)$ denote top-left L-shape neighborhood of a pixel x. Fig. 5 shows these neighborhoods. Suppose that S, T, and R are all single-channel images (luminance image).

The fast texture-transfer algorithm can then be summarized as follows:

1. Initialize R by filling it with pixels randomly taken from S. Let function g maintain the relation between the mapping of pixels: pixel r in R obtains its value from pixel $g(r)$ in S.
2. Visit each pixel r in R, scanning in order from top-left to bottom-right.
   (a) Build candidate set Q.
   (b) Select minimum distance argument from Q, and update r in R.
3. Go back to step 2 if necessary (in the case where the iteration parameter is greater than 1).

For further details on the algorithm, we recommend the reading of Ashikhmin [2].

In Section 4.1, we explain how to build a candidate set for reducing artifacts. In Section 4.2, we explain the distance metric that considers the directional factor proposed in our algorithm for finding the best suitable pixel from a candidate set. In Section 4.3, we propose a method for edge enhancement, as shown in S.

4.1. Build candidate set Q

In Ashikhmin's approach, candidate set Q is constructed as follows:

$$Q \leftarrow \{g(t)+(r-t) | t \in L(x)\}. \quad (1)$$

For extending the search space, we add an extra candidate pixel with a predefined probability p.

This approach is based on random sections. Therefore, the usage of edge pixels in S occasionally creates artifacts such as those shown in Fig. 6. These artifact appeared in areas with an intensity value similar to that of edges in S. To block these artifacts, we use only pixels that are not located at edges. For this, we eliminate edge pixels from S, and then use the remaining pixels to select the candidate set (Fig. 7). This is a very simple approach. However, it can reduce many artifacts in the results.

4.2. Distance metric for finding the best candidate pixel

For selecting the best suitable pixel in a candidate set, we use a distance metric. We calculate the distance of each candidate pixel and select the pixel with the minimum distance:

$$g(r) \leftarrow \underset{q \in Q}{\operatorname{argmin}} \mathbb{D}(r,q). \quad (2)$$

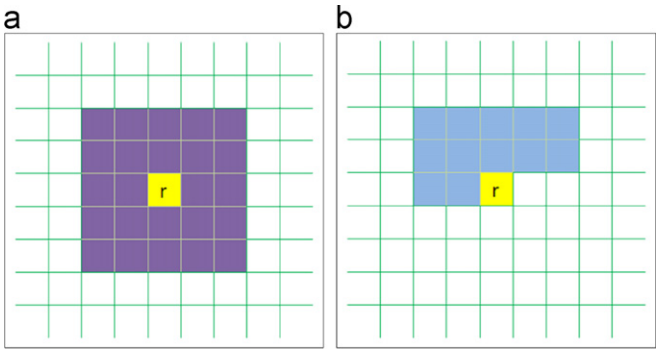


Fig. 5. Types of pixels neighborhood: when *neighbor* = 5: (a) square and (b) L-shape.

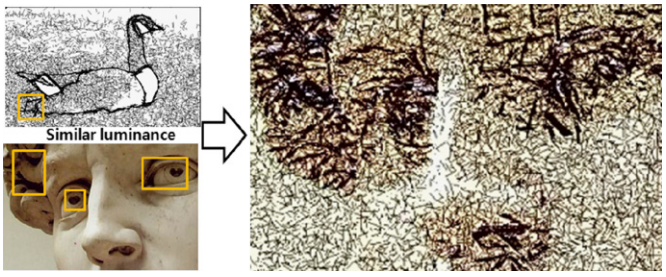


Fig. 6. The usage of edge pixels in S occasionally creates artifacts.

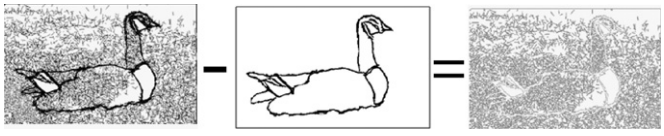


Fig. 7. Valid pixels for candidate sets. (Left: reference image, middle: edge image, right: resulting image obtained by subtracting edge image from reference image). For reducing artifacts, we use only pixels that are not located at the edges.

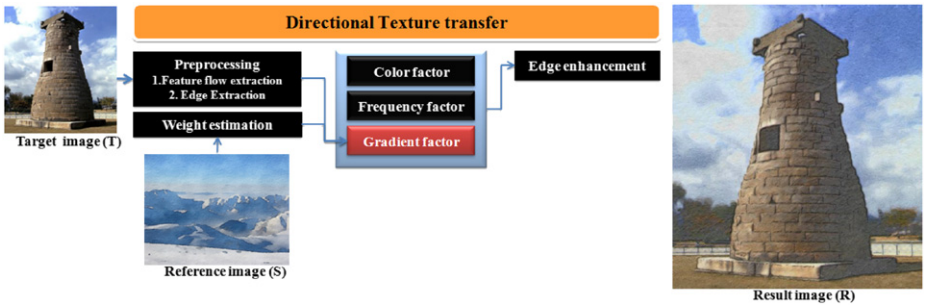


Fig. 4. System flow.

The distance metric used in this paper is constructed as follows:

$$\mathbb{D}(r,q) = D_N(r,q) + w_L \cdot D_L(r,q) + w_I \cdot D_I(r,q). \quad (3)$$

4.2.1. Color and spatial frequency factors

The first two terms in the distance metric are used in previous work. The first term shows the intensity distance between the current position and a candidate pixel q. This factor attempts to select pixels with an intensity value similar to that of the current position r:

$$D_N(r,q) = \|n_r - n_q\|_2, \quad (4)$$

$$n_r = \overline{T(x)}, \quad x \in N(r), \quad (5)$$

$$n_q = \overline{S(x)}, \quad x \in N(q), \quad (6)$$

where $\|\cdot\|_2$ denotes the l^2 norm; n_r and n_q are sample means of pixels in the square neighborhoods of r and q, respectively.

The second term uses the standard deviation. This factor attempts to select pixels with coherent texture effects:

$$D_L(r,q) = (1/|L(r)|)\|H_r - H_q\|_2, \quad (7)$$

$$H_r = [R(x_i) - \overline{R(x)}], \quad x_i \in L(r), \quad (8)$$

$$H_q = [S(x_i) - \overline{S(x)}], \quad x_i \in L(q), \quad (9)$$

where $|\cdot|$ denotes the set size.

In general, w_L is 1, and the neighborhood size is 5.

4.2.2. Gradient directional factor

The fast texture-transfer algorithm, explained in the previous section, produces coherent texture results. However, it cannot express the directional information contained in T. Therefore, we consider the flow of T. To extract coherent gradient information from T, we use Kang's edge tangent flow (ETF) algorithm [15]. This is efficient and preserves edge directions smoothly around important features. The ETF produces a smooth directional field that preserves the flow of the salient image features. This techniques use two parameters. One is *radius* the other is *iteration*. According to parameters, user can control smoothness of directional field. In our algorithm, we generally use *radius*=6 and *iteration*=3. Fig. 8 shows the result of ETF interpolation with Line Integral Convolution(LIC), vector visualization techniques.

We obtain an I-shape neighborhood for each pixel based on the image flow. Let the I-shape neighborhoods be laid down in the normal of interpolated gradient direction belonging to the L-shape (Fig. 5(right)). We use Bresenham's line-drawing algorithm to find the pixels in this I-shape. Fig. 9 shows the method for finding I-shape neighborhood pixels, which are colored red in the figure. In each I-shape neighborhood, we can create a stroke texture effect with the interpolated gradient direction of T. We define the directional factor, $D_I(r,q)$ as follows:

$$D_I(r,q) = \|\overline{R(x_i)} - S(q)\|_2, \quad x_i \in I(r). \quad (10)$$

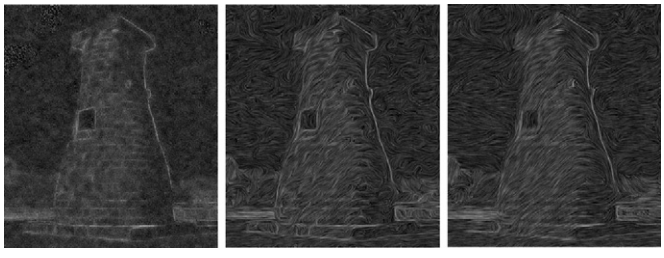


Fig. 8. Original gradients (left), interpolated gradients with radius=6, iteration=3 (middle), radius=16, iteration=3 (right).

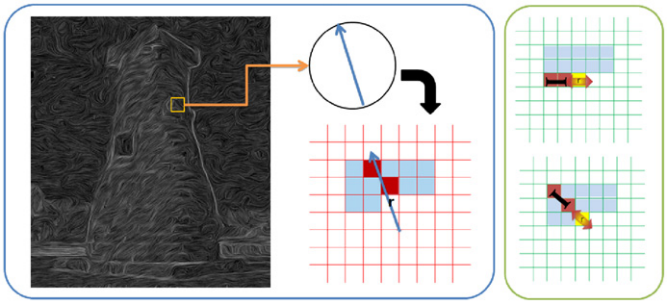


Fig. 9. Method for finding I-shape neighborhoods from image flow (left), and two sample I-shapes (right).

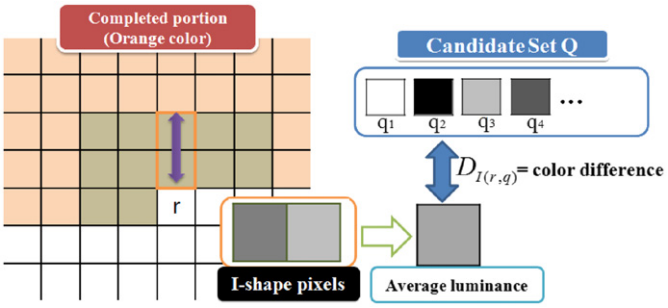


Fig. 10. Method for considering gradient information in I-shape neighborhoods.

Table 1
Distance of each candidate pixel.

Candidate pixel	D_N	D_L	D_I	$\mathbb{D}_{(r,q)}$
$q1$	0.42	0.3	0.6	1.32
$q2$	0.4	0.3	0.53	1.23
$q3$	0.9	0.65	0.1	1.65
$q4$	0.2	0.27	0.4	0.87
...	...	...	...	...

Based on the interpolated gradient value, we then calculate the average intensity of the pixels in the I-shape neighborhoods, and find the difference between the average intensity and the intensity of each candidate pixel q. If the difference is small, q has a color similar to the pixels in the I-shape. The goal is to find the pixel with the minimum difference. The range of $D_I(r,q)$ is between 0 and 1. Fig. 10 shows the process of finding $D_I(r,q)$.

Calculated distance using color, spatial frequency and gradient factors, we normalize each term, select the candidate pixel with the minimum distance and update pixel r with this pixel. For example in Table 1, $q4$ is the most suitable candidate among the calculated pixels (from $q1-q4$).

4.3. Edge enhancement

For expressing more natural and visual effects, we propose an optional method for expressing edge enhancement onto the resulting image based on analyzing the boundary effect in S. First, we extract edge information from T and S at the preprocessing step. We essentially use coherent line drawing [15]. It is occasionally difficult to find clear edge information from the reference image. In that case, we modify the edge image manually by user. From the

extracted edge image, we calculate an edge distance map and utilize pixels located near edges. We analyze intensity changes according to the distance of each pixel from the nearest edge. By averaging the rate of change at each distance, we construct a graph of average intensity changes as shown in Fig. 11. We find average intensity value at each distance, and connect each point. From this, we generate a plot with a linear decrease or increase in each pixel section. Based on this graph from S, we apply this variation to the edges of the resulting image. At each edge point, we adjust the intensity according to the edge distance information. Depending on the material, this graph takes various forms, as shown in Fig. 12. In addition, some references image have no boundary effects.

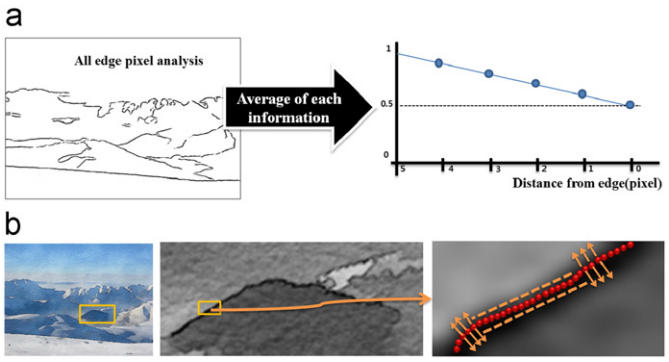


Fig. 11. Analysis of boundary effects in reference image. (a) Constructed graph from edge of S and (b) (left) S, (middle) scale up about bounding box as shown in S, (right) more scale up about edges and an approach for finding intensity changes for each pixel according to the distance from the edges.

Therefore, we use these edge enhancement effect depending on user selection.

5. Estimation of w_I based on reference image

Despite a certain degree of variation across different media, the artistic reference image has a flow of strokes. In line drawings or Van Gogh's paintings, the strokes are laid coherently based on object shape. In other media such as stippling and watercolor, the strokes are difficult to identify. To express the degree of flow in S, we use w_I. By changing the value of w_I, we can control the magnitude of the directional effect in the resulting image. A higher value of w_I gives the directional factor a greater influence on the resulting image. Therefore, the strokes follow the gradient of S more clearly. In this paper, we propose an optional method for automatically estimating w_I from S. This may reduce the amount of trial-and-error necessary to produce an acceptable image. We estimate w_I from S using the difference-of-direction (DoD) value, which is defined as follows:

$$DoD = avg\left(\sum_{x \in S} |(S_{d(x)} - \tilde{S}_{d(x)})|\right), \tag{11}$$

where $S_{d(x)}$ and $\tilde{S}_{d(x)}$ are the original and interpolated gradient information of S, and x is the coordinate of a pixel in S. In this paper, we obtain the original gradient information using a sobel filter. To interpolate direction, we use the ETF interpolation method with $radius = 6$ and $iteration = 3$. The range of the direction is between 0 and 360. As 0 is considered identical to 360, the average direction is calculated in angular coordinates as shown in Lee et al. [16]. Fig. 13 shows the original and interpolated flows for each sample image. In

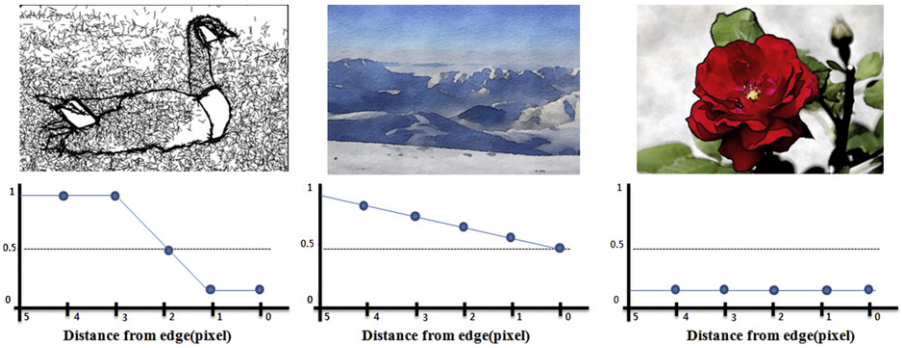


Fig. 12. Plot of average pixel intensity change with respect to the distance of pixels from their nearest edges depending on the reference image line-drawing (left), watercolor (middle) and sumi-e (right).

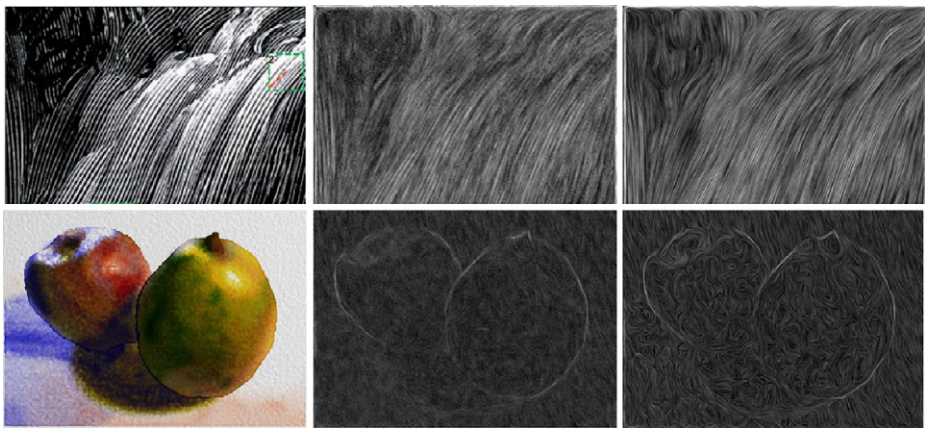
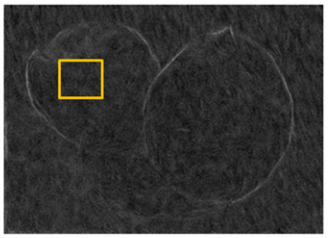
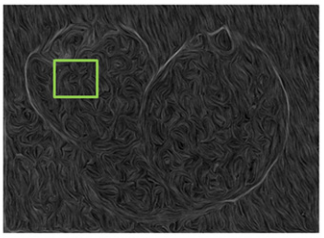


Fig. 13. Two sample images, and original and interpolated directions of each image.





$S_{d(x)}$	$\widetilde{S}_{d(x)}$	$S_{d(x)}-\widetilde{S}_{d(x)}$
30	30	0
38	27	11
50	357	53
...	...	
...	...	

30	38	50	70	170	355	340
80	50	53	28	48	66	57
77	36	39	85	49	116	129
330	289	300	198	28	11	7
328	260	300	15	22	5	359
46	60	36	0	17	27	49

30	27	357	85	199	300	333
65	50	64	28	48	66	57
148	60	70	45	49	116	129
280	249	250	206	56	34	27
308	260	260	15	22	5	10
46	55	36	0	17	27	30

Fig. 14. Example of calculating difference-of-directions (DoDs) from original (left) and interpolated (right) directions.

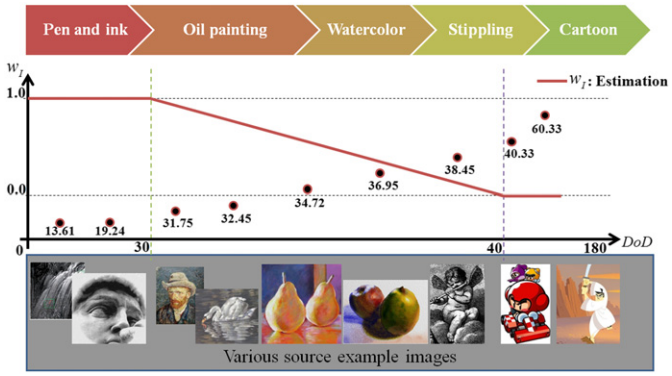


Fig. 15. Estimation of w_I value from the *DoD* of various images.

a line drawing, the flow of the original image gradient is already coherent. Thus, the *DoD* value is small (Fig. 13, middle and right). In watercolor, on the other hand, the *DoD* is greater. Therefore, a low *DoD* value is derived from a line drawing, and a high *DoD* is derived from a watercolor. The range of *DoD* values is between 0 and 180. Fig. 14 shows an example of calculating *DoD* values.

We calculated *DoD* values for 50 sample images and obtained the distribution of values shown in Fig. 15. We found that the *DoD* is less than 30 in a strong line-drawing image. In a cell-animation image, which rarely has a strong flow, the *DoD* tends to be higher than 60. In media such as oils, watercolors, and pastels, the *DoD* is generally between 30 and 40. We can obtain w_I from the *DoD* as follows:

$$w_I = clamp\left(\frac{-1}{10}DoD + 4, 0, 1\right), \tag{12}$$

We set w_I to 1 when the *DoD* is less than 30, and 0 when the *DoD* is higher than 40. When the *DoD* value is between 30 and 40, we calculate w_I by linear interpolation. From this experiment, we can automatically estimate w_I of the target image. If the user uses a higher value than the calculated w_I, s/he can express a directional effect greater than the flow of *S*. In this paper, to express the directional effect specifically, we use a higher w_I value than that calculated from *S*. The definition of artistic style is subjective; success in attaining this style is a matter of personal preference. Therefore, users take a trial-and-error approach and experiment with different parameter values. We performed this estimation to help the user select reasonable parameter values.

Table 2
The usage of parameters and their note.

Parameter	Note
Neighbor size	Size of considering area pixels
p	Probability of adding extra candidate
Iteration	Iteration count
w_L	Weight of D_L
w_I	Weight of D_I

Table 3
The processing time.

Size of *T*	*Neighborhood*	Processing time (ms)
640 × 480	5	43
640 × 480	7	73
1024 × 768	5	76
1024 × 768	7	132

6. Experimental results

We experimented with different brush stroke parameter values and target images. The algorithm parameters are explained in Table 2. The time required for rendering depends on the image size and parameters as shown in Table 3. Including pre-processing time, it takes about 4 s for a VGA image size with a neighborhood size = 5 and Iteration = 1. We used a Pentium 2.8 GHz quad-core PC with 4 GB of memory to produce the results shown in this paper.

Fig. 16 shows the reference images used in this paper. We just apply edge enhancement effect only when we use bottom-row images as reference image. Fig. 17 shows the results according to reference images. In each case, w_I was specified by the user. We tested the reference images of oil, line drawing, pastel, and watercolor. We can see how the stroke texture follows the flow of the cloud and the Eiffel tower. In this figure, we do not apply boundary effects in any image. Fig. 18 shows another result. In the Fig. 18(c), the areas of the sky express white pixels. In the Fig. 18(d), the areas of the sky express some regular patterns. These artifacts

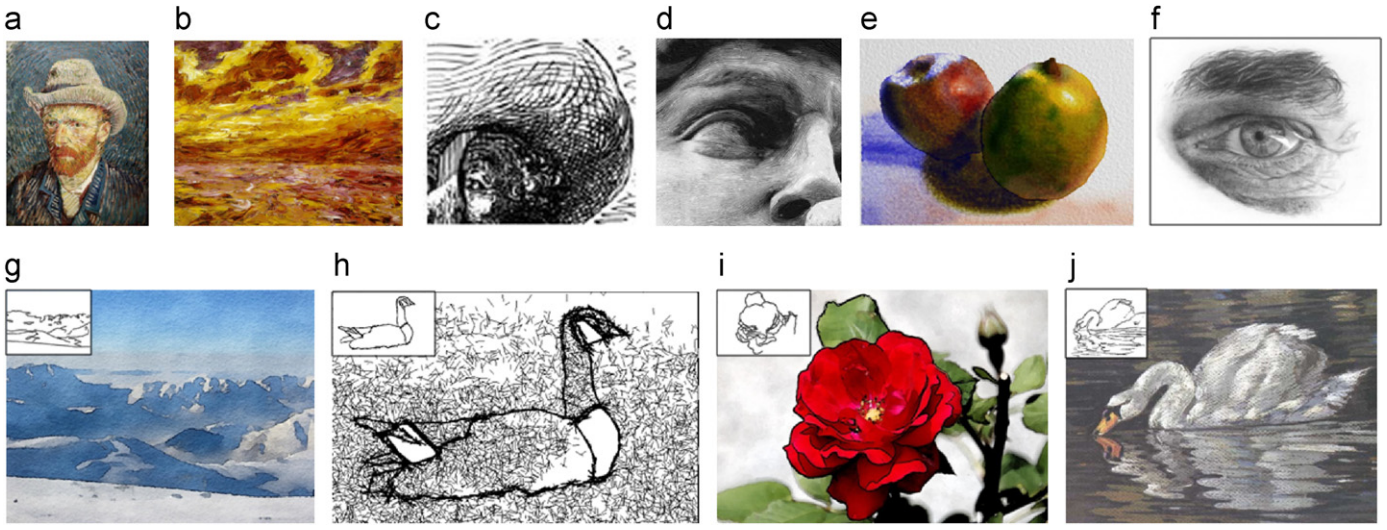


Fig. 16. The reference images used in this paper. We just apply edge enhancement effect only when we use bottom-row images (g) (j) as reference image. (a) Oil painting by Van Gogh's "Self-Portrait with Felt Hat", (b) Emil Nolde's "Autumn Sea", (c) line drawing from Gustave Dora's illustrations for "Don Quixote", (d) line drawing, (e) watercolor painting , (f) charcoal, (g) watercolor painting, (h) line-drawing, (i) sumi-e painting and (j) Pastel.

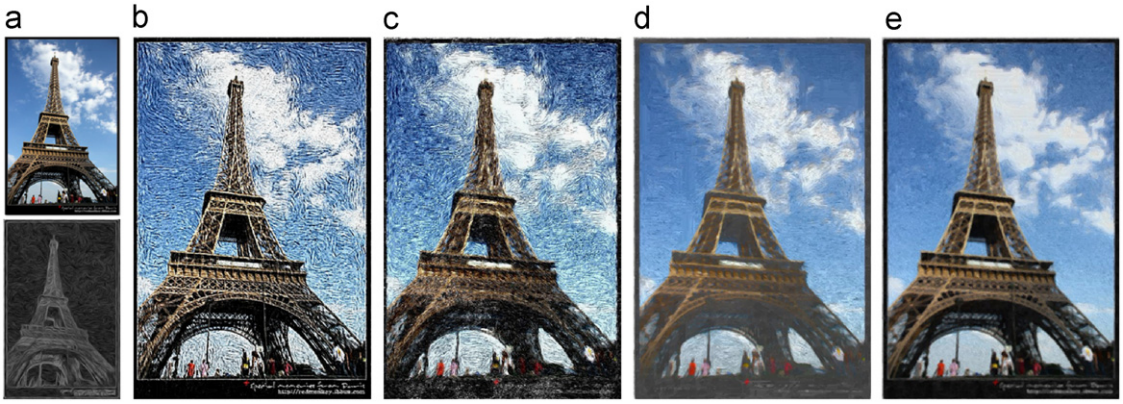


Fig. 17. Various results with different S (*Iteration* $= 1, p = 0.0$). (a) Target image and flow image, (b) S: Fig. 16(a), $w_I = 0.6$, (c) S: Fig. 16(c), $w_I = 0.7$, (d) S: Fig. 16(j), $w_I = 0.8$ and (e) S: Fig. 16(e), $w_I = 0.8$.

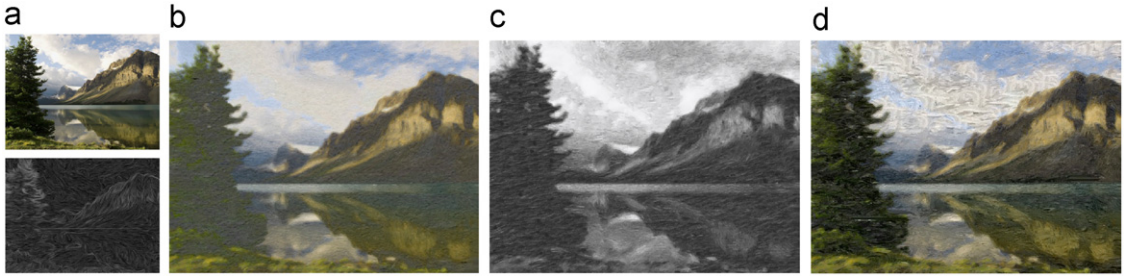


Fig. 18. Results for a landscape (*Iteration* $= 1$, w_I=1, p=0.0). (a) Target image and flow image, (b) S: Fig. 16(g), (c) S: Fig. 16(f) and (d) S: Fig. 16(b).

come from the reference image, which only has a small area of white. Therefore, there are only a small number of candidates to fill the large white area in the target image. Fig. 19 shows new results that are affected by the value of w_I. When $w_I = 0$, direction is not considered. This gives the same result as a classical texture-transfer algorithm. As w_I grows, the directional factor influences the result more than other factors. When w_I is too large, the results are unnatural. Values of w_I between 0 and 1 give good results.

Fig. 20 compares four different results. Fig. 20(a) shows the result that used Ashikhmin's techniques. Because of using random selection for finding the candidate set, it leads to some artifacts. Fig. 20(b) shows the result considering directional factor. By applying the directional factor proposed in our paper, the result is improved. Fig. 20(c) show the result which does not use edge pixels of S and Fig. 20(d) show the result that applied edge enhancement effect. From this techniques, our algorithm produces

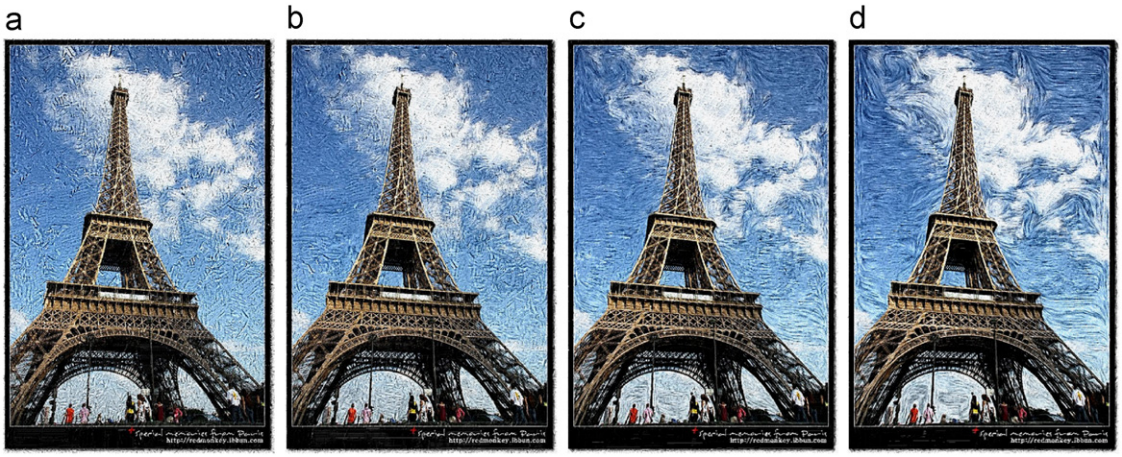


Fig. 19. Results with different w_l values (*Iteration*=1, P=0.0, S: Fig. 16(a)). (a) w_l = 0.0, (b) w_l=0.4, (c) w_l=0.7 and (d) w_l=1.0.

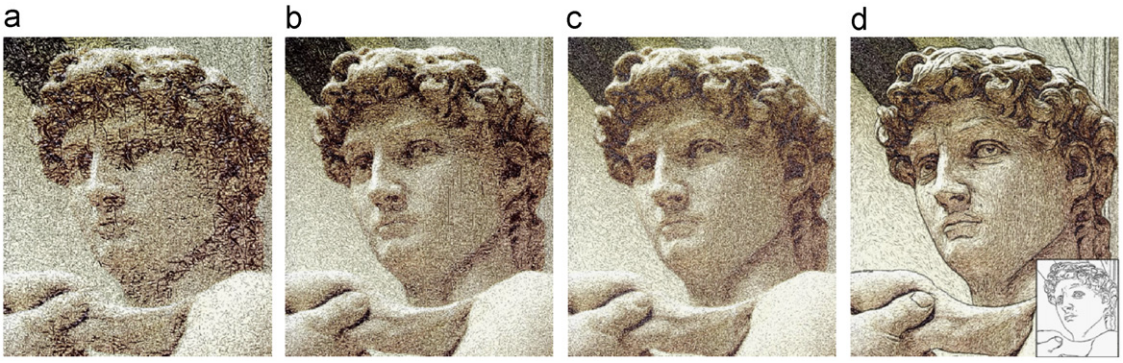


Fig. 20. Results based on randomly selecting candidate sets from the entire or non-edge area and edge enhancement effect. (a) Ashikhmin's technique, (b) original directional texture-transfer (consider only direction), (c) from result (b)+(selecting candidate sets from non-edge area), (d) from result (c)+ (applying edge enhancement effect). (a) w_l=0.0, (b) w_l=0.4, (c) w_l=0.7 and (d) w_l=1.0.

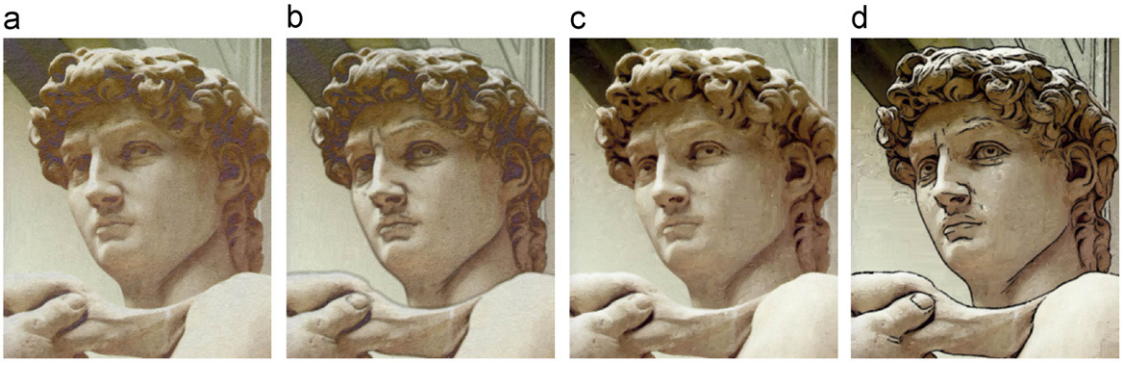


Fig. 21. Results without (first and third) and with (second and fourth) edge effects, figure (a) and (b): S= Fig. 16(g), figure (c) and (d): S= Fig. 16(i).

more compelling results than original directional texture techniques as shown in Fig. 20(b) [17]. Fig. 21 compares the results for different boundary effects. The object silhouette is also a very important cue for visual effects. By applying edge-enhancement effects based on analyzing the edge effects in S, we can produce more enhanced results. However some reference images have no boundary effects. Therefore, we use this approach depending on user selection.

Fig. 22 compares our algorithm with other pixel-based texture-transfer techniques using the same reference image (Fig. 16(c)). Fig. 22(b) was produced using Hertzmann's technique, and Fig. 22(c) is the resulting image from Ashikhmin's method. It is clear that our approach expresses the direction of textures more effectively. Fig. 23 compares our algorithm with a Wang's patch-based algorithm. The image on the left is produced by Wang, whereas that on the right panel shows the results with our technique. Our algorithm can produce results similar to that of patch-based directional expression. However, we think that our algorithm is simpler than the patch-based approach and easy to implement. In Wang's work, they indicate the total processing time of generating as 55 s. However, in our algorithm, it takes 6 s for generating result. Our result also has rougher texture effects as shown in the reference image (Fig. 16(b)).

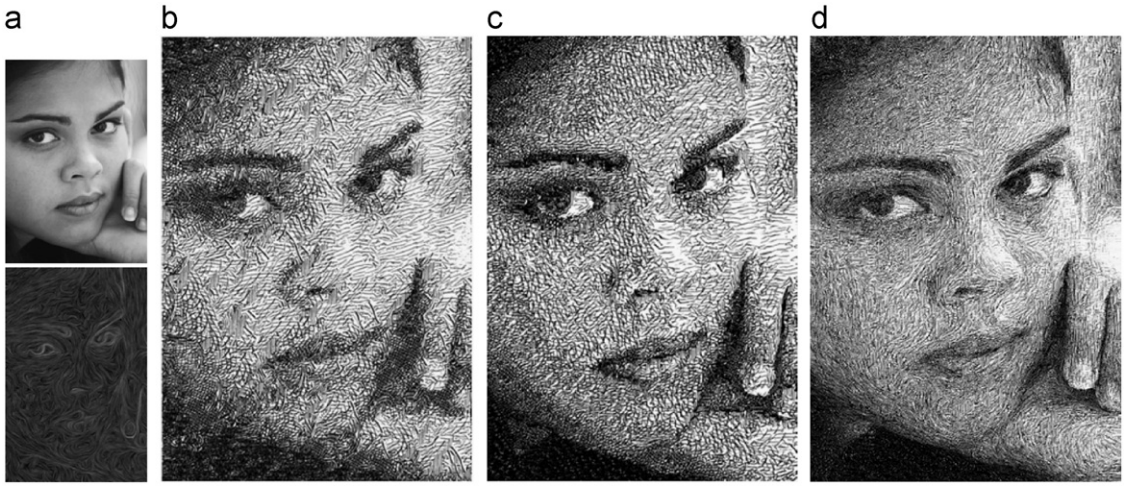


Fig. 22. Comparison results. (a) Target image and flow image, (b) Hertzmann's technique, (c) Ashikhmin's technique(©IEEE2004) and (d) our technique.

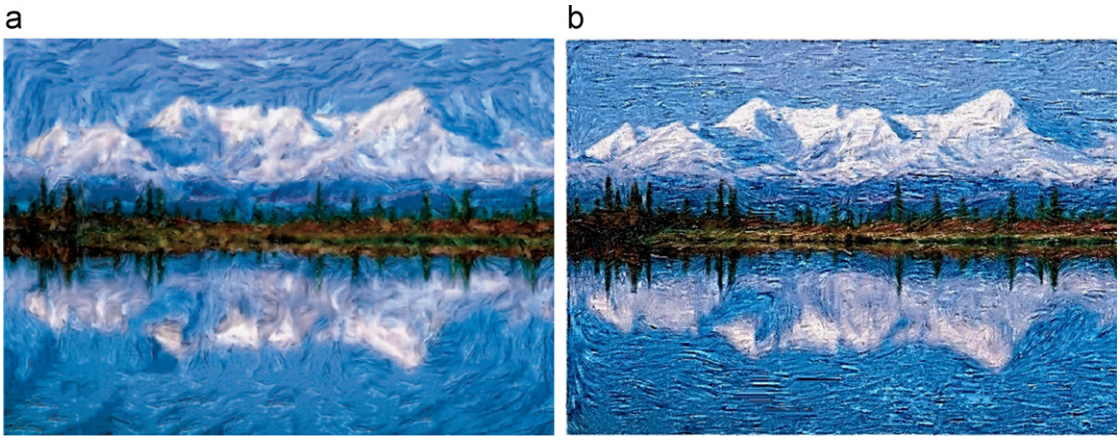


Fig. 23. Comparison of Wang's (a) and our (b) techniques.

Fig. 24. Comparison of results to those of a previous texture synthesis technique (left) and our (right) techniques.

Fig. 24 shows the result of adapting our algorithm to texture synthesis. We test our gradient directional factor onto Ashikhmin [1]. In texture-transfer research, we use only a single channel (luminance). The final result is generated by combining the color of the target image and the updated intensity channel. In texture synthesis, on the other hand, we use all channels (YIQ space) of the reference image, and update the resulting image. When the patch texture has a shape such as that of a flower, we may obtain an undesirable result. This is because our directional factor distorts the shape of the patch. However, if the patch has no shape such as in Fig. 24, our algorithm operates well. The quality of the result is dependent on the viewer. Nevertheless, our results exhibit some flow of image gradient. Fig. 25 shows additional results that demonstrate how our algorithm can transfer texture effectively and pick up gradients from the target image and effect of edge enhancement.

7. Test in smart phone application

Our algorithm has a range of applications. It would be easy to implement in a smart phone. We are currently porting our algorithm onto a smart phone. On a 1 GHz mobile CPU running the android OS, it takes 20–30 s to produce a Qvga image size. We are planning code

Fig. 25. Further results (edge enhancement effect is used in (a), (b), (c) and (d)). (a) $S=$ Fig. 16(h), (b) $S=$ Fig. 16(i), (c) $S=$ Fig. 16(g), (d) $S=$ Fig. 16(h), (e) $S=$ Fig. 16(d), (f) $S=$ Fig. 16(a), (g) $S=$ Fig. 16(a) and (h) $S=$ Fig. 16(d).

optimization to enhance performance on small devices such as digital frame or camera.

8. Conclusion and future works

In this paper, we have introduced a directional texture-transfer algorithm. We extract an interpolated image gradient from a target image. We also define I-shape neighborhoods to express a directional stroke texture. The average color of the pixels in a neighborhood is compared to a number of candidate pixels and the pixel with the minimum distance is selected. Optionally, we adapt edge-enhancement by analyzing graphs of intensity changes from S. We have also proposed an automatic method of estimating the weight of the directional factor. Our algorithm can express a convincing combination of the texture of a reference image and the flow of a target image. We have also shown that our algorithm can perform texture synthesis effectively.

This algorithm contributes to NPR research in many ways. It is a novel algorithm expressing directional effects with a pixel-based approach. We also adapt edge-enhancement based on reference image. In addition, we use the difference-of-direction (DoD) value

to estimate the parameter w_I from the reference image. We anticipate that this *DoD* formulation will be used other NPR applications.

We are planning future studies along the following lines. Firstly, our results tend to exhibit longer strokes than the reference image, so it is impossible to express stippling images in this manner because our algorithm tries to connect pixels into I-shape neighborhoods. It would be helpful to consider the deviation of I-shape pixels. Secondly, we currently use a vector for direction. However, for a more natural flow, we will attempt to use a tensor field, as proposed in Zhang et al. [25]. Finally, we plan to extend this algorithm to the area of non-photorealistic animation.

Acknowledgement

This research was supported by the Chung-Ang University Research Scholarship Grants in 2008.

References

[1] Ashikhmin M. Synthesizing natural textures. In: ACM symposium on interactive 3D graphics; 2001. p. 217–26.
[2] Ashikhmin M. Fast texture transfer. IEEE Computer Graphics and Applications 2003;23(4):38–43.
[3] Baxter W, Wendt J, Lin MC. Impasto—a realistic, interactive model for paint. In: Proceedings of the NPAR'04, New York, NY, 2004. p. 45–56.
[4] Bonet JSD. Multiresolution sampling procedure for analysis and synthesis of texture images. In: Proceedings of SIGGRAPH 1997; 1997. p. 361–8.
[5] Deussen O, Hiller S, van Overveld C, Strothotte T. Floating points: a method for computing stipple drawings. Computer Graphics Forum 2000;19:40–51.
[6] Efros AA, Freeman WT. Image quilting for texture synthesis and transfer. In: Proceedings of SIGGRAPH 2001; 2001. p. 341–6.
[7] Eisenacher C, Lefebvre S, Stamminger M. Texture synthesis from photographs. In: Proceedings of the Eurographics conference; 2008. p. 419–28.
[8] Freeman WT, Tenenbaum JB, Pasztor E. An example-based approach to style translation for line drawings. Technical Report TR-99-11, MERL—A Mitsubishi Electric Research Laboratory, 1999. URL: ⟨http://www.merl.com/papers/docs/TR99-11.pdf⟩.
[9] Gooch B, Coombe G, Shirley P. Artistic vision: painterly rendering using computer vision techniques. In: Symposium on non-photorealistic animation and rendering NPAR 2000; 2000. p. 83–90.
[10] Haeberli P. Paint by numbers: abstract image representations. In: Proceedings of SIGGRAPH 1990; 1990. p. 207–14.
[11] Hays J, Essa I. Image and video based painterly animation. In: Symposium on non-photorealistic animation and rendering (NPAR2004); 2004. p. 120–33.
[12] Hertzmann A. Painterly rendering with curved brush strokes of multiple sizes. In: Proceedings of SIGGRAPH 1998; 1998. p. 453–60.
[13] Hertzmann A. A survey of stroke-based rendering. In: IEEE Computer Graphics and Applications; 2003. p. 70–81.
[14] Hertzmann A, Jacobs CE, Oliver N, Curless B, Salesin DH. Image analogies. In: Proceedings of the 28th annual conference on computer graphics and interactive techniques. ACM; 2001. p. 327–40.
[15] Kang H, Lee S, Chui C. Coherent line drawing. In: Symposium on non-photorealistic animation and rendering (NPAR 2007); 2007. p. 43–50.
[16] Lee H, Lee C, Yoon K. Motion based painterly rendering. Computer Graphics Forum 2009;28(4):1207–15.
[17] Lee H, Seo S, Ryoo S, Yoon K. Directional texture transfer. In: Proceedings of the eighth international symposium on non-photorealistic animation and rendering (NPAR 2010). New York, NY, USA: ACM; 2010. p. 43–8.
[18] Lefebvre S, Hoppe H. Appearance-space texture synthesis. In: SIGGRAPH '06: ACM SIGGRAPH 2006 papers. New York, NY, USA: ACM; 2006. p. 541–8.
[19] Liang L, Liu C, Xu Y-Q, Guo B, Shum H-Y. Real-time texture synthesis by patch-based sampling. ACM Transactions on Graphics (TOG) 2001;20:127–50.
[20] Litwinowicz P. Processing images and video for an impressionist effect. In: Proceedings of SIGGRAPH 1997; 1997. p. 407–14.
[21] Salisbury MP, Anderson SE, Barzel R, Salesin DH. Interactive pen-and-ink illustration. In: SIGGRAPH '94: proceedings of the 21st annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM; 1994. p. 101–8.
[22] Salisbury MP, Wong MT, Hughes JF, Salesin DH. Orientable textures for image-based pen-and-ink illustration. In: ACM SIGGRAPH 1997 proceeding; 1997. p. 401–6.
[23] Strassmann S. Hairy brushes. In: Proceedings of SIGGRAPH 1986; 1986. p. 225–32.
[24] Wang B, Wang W, Yang H, Sun J. Efficient example-based painting and synthesis of 2d directional texture. IEEE Transactions on Visualization and Computer Graphics 2004;10(3):266–77.
[25] Zhang E, Hays J, Turk G. Interactive tensor field design and visualization on surfaces. IEEE Transactions on Visualization and Computer Graphics 2007;13(1):94–107.
[26] Zhang E, Mischaikow K, Turk G. Vector field design on surfaces. ACM Transactions on Graphics 2006;25(4):1294–326.