

# Synthesis of Oil-Style Paintings

Fay Huang<sup>(✉)</sup>, Bo-Hui Wu, and Bo-Ru Huang

Department of Computer Science and Information Engineering,  
National Ilan University, Yi-lan, Taiwan, ROC  
`fay@niu.edu.tw`

**Abstract.** Non-photorealistic rendering is an important research topic in computer graphics, where painterly (or stroke-based) rendering has received intensive attention from researchers in recent years. The goal of this paper is to design a fully automatic algorithm, which is able to turn a photograph into an oil-style painting. Different from the existing approaches that use real brush stroke images as templates, our brush strokes were created in random manner according to the characteristics of the local image region. For determining the direction of a brush stroke, we also proposed a new method based on template-matching to evaluate the major orientation of edge features within a local image window. Moreover, a novel method of deciding stroke locations was proposed, which is simple yet effective. All these features together significantly reduce the undesirable systematic impression, which appears to be a common artifact of painterly rendering.

**Keywords:** Non-photorealistic rendering · Painterly rendering · Stroke-based painting · Texture synthesis

## 1 Introduction

Non-photorealistic rendering (NPR) is an active research topic in computer graphics. One of the interesting applications of NPR technique is to simulate different types of art mediums and different styles of paintings/drawings. The task is to turn a photo into a specific type of painting or drawings by a computer algorithm. Many digital image acquisition devices nowadays include built-in software, which allow users to change the style of photos. Various commercial image editing software provide filters which are able to simulate different styles of artistic impressions. However, none of those is able to deliver quality oil-style painting synthesis automatically. Either manual assistance or the knowledge of specific combination of different filtering functions is required.

There are many artistic-style simulation algorithms for some specific type of art medium, such as pencil, crayon, ink, watercolor, and etc., as well as a well-known artist's style. Among these researches, two major approaches are,

---

F. Huang—This research project is funded by Ministry of Science and Technology, Taiwan (MOST 104-2221-E-197 -020 -MY2).

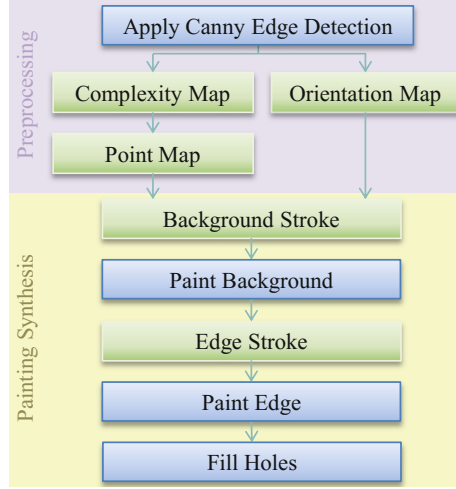
first, considering the simulation task as texture synthesis, and second, imitating the way an artist paints a picture. In the first approaches, an example painting, which is sometimes referred to as a sample image, is required. The essential pre-processing step of approaches in this category is performing texture analysis to the patches of the sample image. The algorithms then transform a photo into an artistic style picture following the painting style of the given example by various texture synthesis techniques [1, 6, 10].

In the second approaches, the major concerns are the design of the brush strokes and the method of applying them (e.g., orientation and order). The painting synthesis results are generated by stacking various sizes and orientations of strokes. Since brush stroke templates are the principal elements for methods belonging to this category, the approaches are often referred to as painterly rendering [2, 3, 8, 9] or stroke-based rendering [4, 5, 7].

Meier [8] at Walt Disney first introduced the concept of painterly-style rendering for animation and Hertzmann [3] proposed the modelling of curved brush strokes. Both papers received great attention in the early development of NPR technology. Brush strokes in [3] are represented by anti-aliased thick cubic B-splines. The termination of each stroke and the painting order are two important subjects discussed in Hertzmann’s paper. Lee et al. [5] refined Hertzmann’s approach by proposing a new painting order which reduced the undesirable color scattering artifacts at the objects boundaries. A 3D curved brush stroke model was introduced by modifying the actual brushstroke samples provided by an artist. Gooch et al. [2] proposed a new method of constructing brush strokes based on the medial axes of the segmented regions. Shiraishi and Yamaguchi [9] defined a set of brush stroke attributes such as color, location, orientation, and size. Stroke distribution was determined according to the stroke area image. Stroke size was specified by user. Kang et al. [4] presented a unified scheme for automatically generating various types of artistic illustrations from photographs. A set of eleven parameters were used to classify different styles of illustrations. However, Kang’s approach required manual assistance to calculate the importance map, which served as a reference for direction map generation. The goal of this paper is to design a fully automatic painterly rendering algorithm, which is able to turn a photograph into an oil-style painting.

## 2 Program Framework

Paintbrushes come in various shapes and sizes, each with a different purpose. One most intuitive way of selecting different sizes of paintbrushes is according to the area of the specific color to be painted on the canvas. Usually, one would use large paintbrushes to paint the large color-homogeneous region, and use smaller paintbrushes for the detailed areas. Therefore, the proposed synthesis algorithm first analyzes the color complexity of the different portions of the input photograph. A complexity map will be established and is used as a reference to select the size of a paintbrush for later painting process. The framework of the proposed synthesis algorithm is depicted in Fig. 1. In this figure, the outlined



**Fig. 1.** The flowchart of the proposed approach. The outlined rectangles indicate the tasks to be performed, and the un-outlined rectangles denote the items to be created.

rectangles indicate the tasks to be performed, and the un-outlined rectangles denote the items to be created.

Once the brush size is determined, the next things to decide are where to apply a stroke and in which direction the brush should be moved. The directions of the brush strokes are essential, especially for creating an oil painting effect. In the preprocessing stage of the framework, a point map and an orientation map are constructed based on the complexity map and the edge features of the input image, respectively. The point map indicates the locations where the brushes are to be applied. The orientation map stores the suggested orientations of the strokes. A novel edge orientation determination algorithm is proposed, which is robust to noise and thus the common preprocessing such as noise removal can be waived. The definition and generations of complexity, point, and orientation maps are elaborated in Sects. 3 and 4.

Artists usually use relatively large brushes to paint the background or large color-homogeneous regions first, and use smaller brushes to add complex details to the foreground objects last. To imitate this behavior, the developed painting algorithm also performs background painting first, and then paints the edge regions on top of the resultant image. In the painting synthesis stage of the framework, the paintbrush generation is divided into two categories: one is background stroke and the other is edge stroke. The size of the background strokes depends on their associated values in the obtained complexity map. Edge strokes are designed to be thin and longish to emphasize their orientations. The detailed stroke generation methods will be explained in Sect. 5.

Finally, after the painting process, if there are still small unpainted regions, then the hole filling will take place to complete the oil-style painting synthesis. The unpainted regions can be filled by colors obtained from the input photograph.

### 3 Complexity Map and Point Map

In this paper, the term complexity for a local image region is defined by the color or intensity variation within that region. The complexity map serves as a reference for designing the brush strokes (i.e., size and shape) and determining the amount of strokes to be applied for a particular image region.

In order to calculate the complexity map, we applied edge detection to the input photograph. The selection of edge detection methods is not critical in our application and we have chosen Canny edge detector and modified it to increase the thickness of the edge to be three pixels. The resulting edge image is stored in binarized form, denoted as  $B$ . Function  $B(x, y)$  returns the binarized intensity value, either 0 or 1, of image pixel  $(x, y)$ , where value 1 indicates the edge location. The image is then partitioned into  $i \times j$  rectangular regions. Each region is of size  $m \times n$  pixels. The complexity map is a  $i \times j$  matrix. The complexity value  $C(i, j)$  of a reference pixel  $(x, y)$  is evaluated by the following equation.

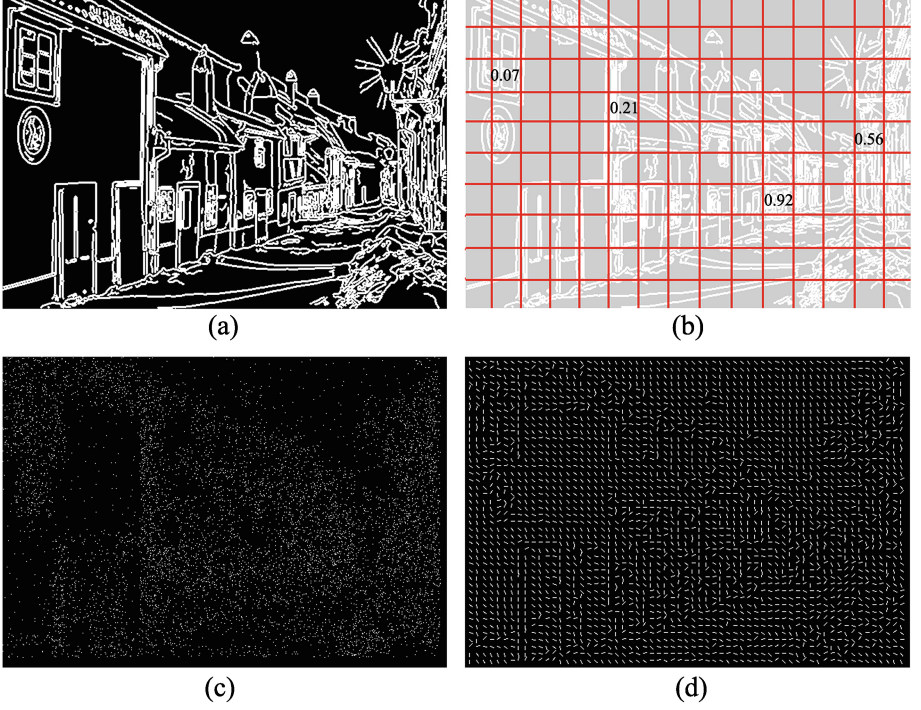
$$C(i, j) = \frac{1}{mn} \sum_{h=0}^{m-1} \sum_{l=0}^{n-1} B(x+h, y+l)$$

where  $x = jn + 1$  and  $y = im + 1$ . Based on the experience from the experiments, according to the size of input image, we would recommend partitioning the image into 120 to 200 rectangular regions. For instance, a  $400 \times 600$  pixels image would be partitioned into  $10 \times 15$  regions, and in this case, values  $m$  and  $n$  are both equal to 40,  $i = 15$ , and  $j = 10$ . Figure 2 illustrates an example of the aforementioned preprocesses, where (a) shows the result after increasing the thickness of Canny edges of the image in Fig. 7. The image partitions are depicted by grids and the associated complexity values of four selected regions are shown in Fig. 2(b).

Next, a point map is generated according to the complexity map. The point map indicates the locations where the brush strokes are to be applied. The task is to assign an amount of random dots within each of the specified rectangular image regions of Fig. 2(b). The total amount of stroke to be applied (i.e., the total number of dots) for the whole image, is defined to be  $\frac{1}{k}$  of the total number of edge pixels within the image. The default value of  $k$  is 10. This parameter  $k$  can be altered by user to control the abstractness of the painting style. (Note: to increase the abstraction level, the value of  $k$  should be increased within the range between 10 to 20, and the source image should be blurred accordingly to be used in the hole filling process.) Based on the complexity value for each region and the total number of strokes, the program assigns dots in random manner to every image region. The number of dots within each region is proportional to the associated complexity value. Figure 2(c) illustrates an example of the point map.

### 4 Orientation Map

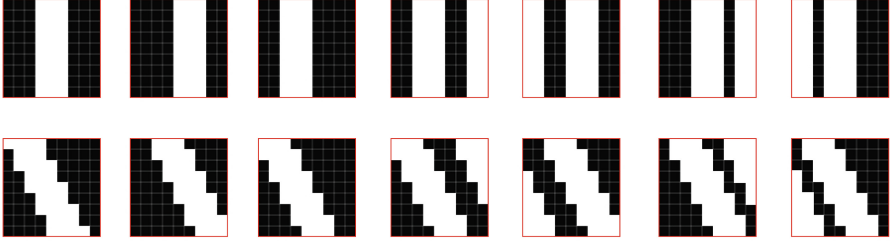
An orientation map is created to guide the painting direction of each stroke. We proposed a new method to obtain the major orientation of edges within a local



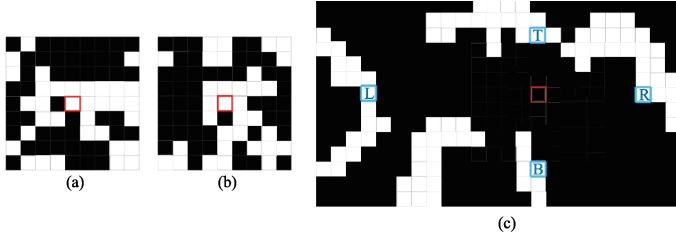
**Fig. 2.** (a) the result after applying a modified Canny edge detection to photo in Fig. 7. Examples of a  $10 \times 15$  complexity map (b), a point map (c), and a orientation map (d). Numbers in the complexity map illustrate examples of the complexity values of that particular regions.

window. This method is very useful especially when multiple edges are clustered within a small region. The artist often paints a stroke along the major orientation of structures within the neighborhood, especially in the abstract-style painting. In the case when there is no edge within a local window, the orientation value of a homogenous region is obtained by considering the orientation values of the closest edge pixels in top, down, left, and right directions.

A set of 56 directional templets were designed and used to determine the major orientation of edges within a  $9 \times 9$  local window region. We defined seven varieties of templets for each of the eight directions, namely  $0, 22.5, 45, \dots, 157.5$  degrees. In particular, zero degree corresponds to vertical edges and 90 degrees to horizontal edges, respectively. Figure 3 shows the zero and 22.5 degrees examples of the directional templets. Let  $T_{dn}$  denote the  $n^{th}$  templet of degree  $d$ , and function  $T_{dn}(u, v)$  returns the intensity value of the  $(u, v)$  pixel, which is either 0 (black) or 1 (white). For each reference pixel  $(x, y)$  in the binarized image  $B$ , if  $B(x, y) = 1$ , we calculate the following:



**Fig. 3.** Examples of  $9 \times 9$  directional templates. First row shows seven pre-defined directional templates of degree zero (i.e., vertical lines). Second row illustrates the other set of seven pre-defined directional templates of 22.5 degrees.



**Fig. 4.** (a) and (b) are two examples of complicated image edge distributions within the  $9 \times 9$  local window. The orientation value was assigned to be 78.75 degrees by our approach for (a), and 168.75 degrees for (b). (c) shows an example of a homogeneous region, and the orientation value for this reference pixel (outlined square) is obtained by linear combinations of orientation values of pixels L, T, R, and B.

$$Similarity(T_{dn}) = \sum_{u=-4}^4 \sum_{v=-4}^4 NOT(XOR(T_{dn}(u+5, v+5), B(x+u, y+v)))$$

for all  $d$  and  $n$ . Then all the similarity values are sorted in descending order. The greater the similarity value indicates the more closely the template matches the local edge distribution. The final orientation value for the edge pixel  $(x, y)$  is set to the average degrees associated with the highest two similarity values. By this way, our approach is able to identify 16 different orientations. Consequently, more variety of strokes can be created. Figure 4(a) and (b) show two examples of complicated distributions of image edges within the  $9 \times 9$  local window. It is not trivial to determine the gradient values for the respective reference pixels (outlined square) using the standard gradient formula. After applying our orientation estimation, Fig. 4(a) obtained an orientation value of 78.75 degrees, and 168.75 degrees for Fig. 4(b).

Figure 4(c) gives an example of the possible homogeneous region. In order to obtain an orientation value for its reference pixel (outlined square), our approach searches for the existing orientation values associated to the nearest neighbors in

four directions, namely top, down, left and right. The distances between the reference pixel and those neighborhood pixels were used to decide a set of weighting coefficients. The final orientation value is set to equal to the sum of weighted orientation values of those neighborhood pixels. For the photograph shown in Fig. 7, the corresponding orientation map is illustrated by a needle map displaying only every tenth pixel in Fig. 2(d).

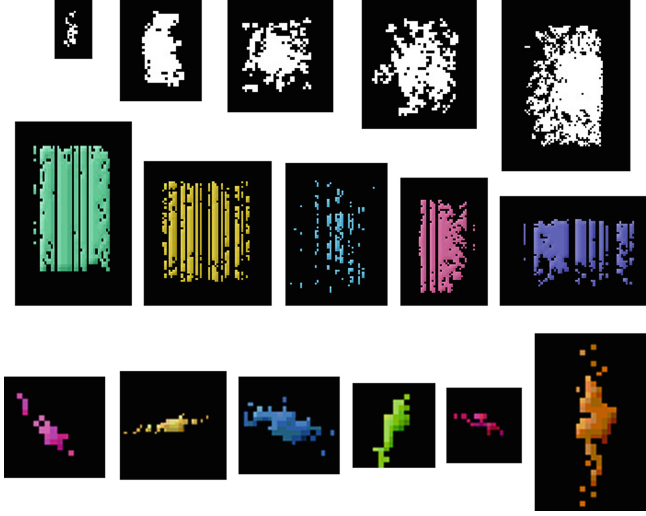
## 5 Stroke Design and Painting

The design of painting strokes is the most essential task in a painting simulating process. Some researchers [5] and commercial products used real brush strokes as templates to paint the entire picture. Only the size and the orientation of the stroke template was varied accordingly during the painting process. In their approaches a pre-established database of real brush strokes is required. Moreover, if painting a picture using only a few types of strokes, then it would create an undesirable systematic impression to the final artwork. Therefore, we proposed a method to design our own brush strokes instead of using a real brush stroke database. In our approach, parameters are the keys to change the appearance of a stroke. The parameter values are determined according to the characteristic of the local edge features, and each stroke is created by random manner. The parameters include the color, the width and height, the density, the orientation, and the thickness of the stroke.

### 5.1 Background Region

Background in this paper is defined to be the non-edge region of the binarized image  $B$ . It corresponds to the locally color-homogeneous region in the input image. First of all, the size (i.e., width and height) of a background stroke is decided based on the complexity value associated with this stroke location on the image. The smaller the complexity value indicates the higher possibility that this location lies within a larger color-homogeneous region. Therefore, the size of the stroke to be applied at this location should likely to be greater in comparison to the other strokes. A maximum stroke length  $L$  is evaluated by  $L = k * \max(W, H) * (1 - C(i, j)) / 100$ , where  $W$  and  $H$  are the width and height of the input image,  $C(i, j)$  is the corresponding complexity value, and  $k$  defines the abstractness level as described earlier. The width  $w$  and height  $h$  of the stroke is obtained independently by  $w$  or  $h = \text{ceil}(\text{rand} * L) + 5$ , where  $\text{ceil}$  and  $\text{rand}$  are the ceiling and the random functions provided in Matlab. The number 5 in the formula is to ensure the size of a background stroke to be at the least five.

Next, the pattern of the stroke is mainly affected by the density value. Here, the density of a stroke can be analogous to the density of the paint (or pigment) on the brush. In general, the overall usage of paints (or pigments) on brushes to create the entire piece of artwork is somehow consistent. Therefore, a single average density value is assigned for each artwork, which influences the style



**Fig. 5.** Top row shows some examples of stroke masks of various intensity values. Middle row illustrates few examples of simulated background strokes. Bottom row shows some examples of generated edge strokes of different orientations.

of the painting. The default value is 0.5. The way to create a stroke mask is by adding random dots within a predefined rectangular region of size  $w \times h$ . The number of dots is equal to  $wh * (randn * 0.1 + 0.5)$ , where *randn* is a normally distributed random function provided in Matlab. After this, more dots are randomly added towards the center region to simulate the appearance of an actual brush stroke. Some examples are illustrated in the top row of Fig. 5.

In order to create more realistic brush textures, the stroke mask is then subtracted from another mask containing randomly placed vertical lines. This gives a good simulation of oil paint texture. Before applying the stroke mask, it would be rotated according to the direction specified in the orientation map. Finally, certain thickness is added to the stroke by applying a proper shading. The middle row of Fig. 5 demonstrates few examples of simulated background strokes without performing the rotation step.

The program generates a new background stroke for each point, say  $(x, y)$ , in the point map which lies in the background region, and applies (or paints) this stroke to the corresponding location  $(x, y)$  on the resulting image. The color of a stroke is assigned to be the same as the pixel  $(x, y)$  in the original input image. By this way, an image pixel is possible to be painted multiple times. When it occurs, the colors of overlapping strokes are averaged. The effect of averaging colors proposed in our approach appears to be somehow similar to the real world situation when mixed colors are used. But, averaging too many difference colors might eventually turn to gray. Therefore, our program accepts at most four colors to be applies on each pixel. The effect of overlapping strokes is shown in Fig. 6.





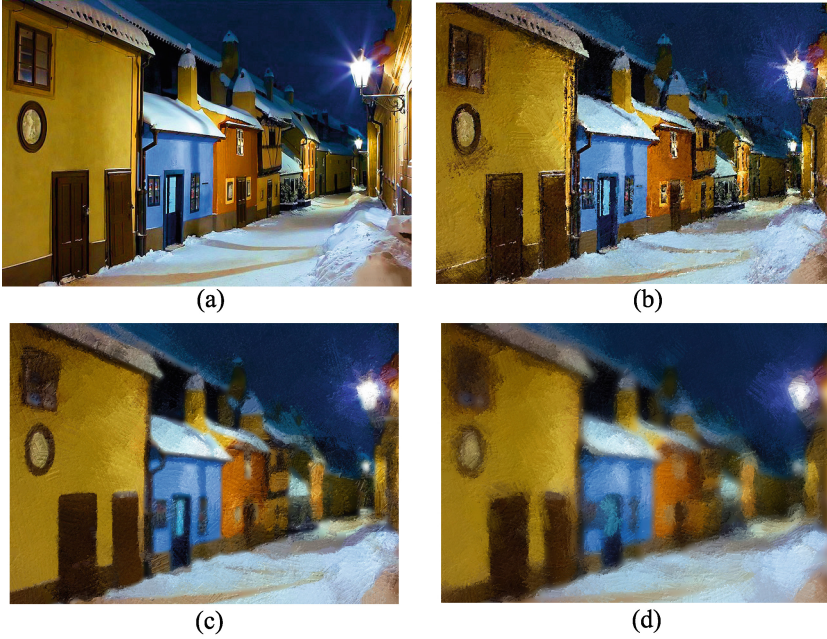
**Fig. 6.** Enlarged portions of the resulting simulation to demonstrate the strokes textures.

## 5.2 Edge Region

The role of the edge strokes is to enhance the outlines of the objects or to add details to the complex image regions. Orientation characteristic is essential for the edge strokes. Artists usually apply brush strokes according to the main direction of surrounding structures. In order to serve those purposes, our program generates thin strokes for edges and rotates it to meet the majority of the local edges directions. The generation procedure for an edge stroke is the same as that for background strokes. The only differences are the values for size and density parameters. The program specifies the aspect ratio of edge stroke to be about 5 : 1, and the length of the stroke is determined by  $k * \min(W, H) * (1 - C(i, j)) / 200$ . The average density for edge stroke is set to 0.7. Bottom row of Fig. 5 shows examples of generated edge strokes. For each point in the point map which lies in the edge region of  $B$ , an edge stroke is created and applied to the resulting image on top of the already painted background region. The edge stroke color will directly overlay the existing background color. Only colors of overlapping edge strokes will be averaged.

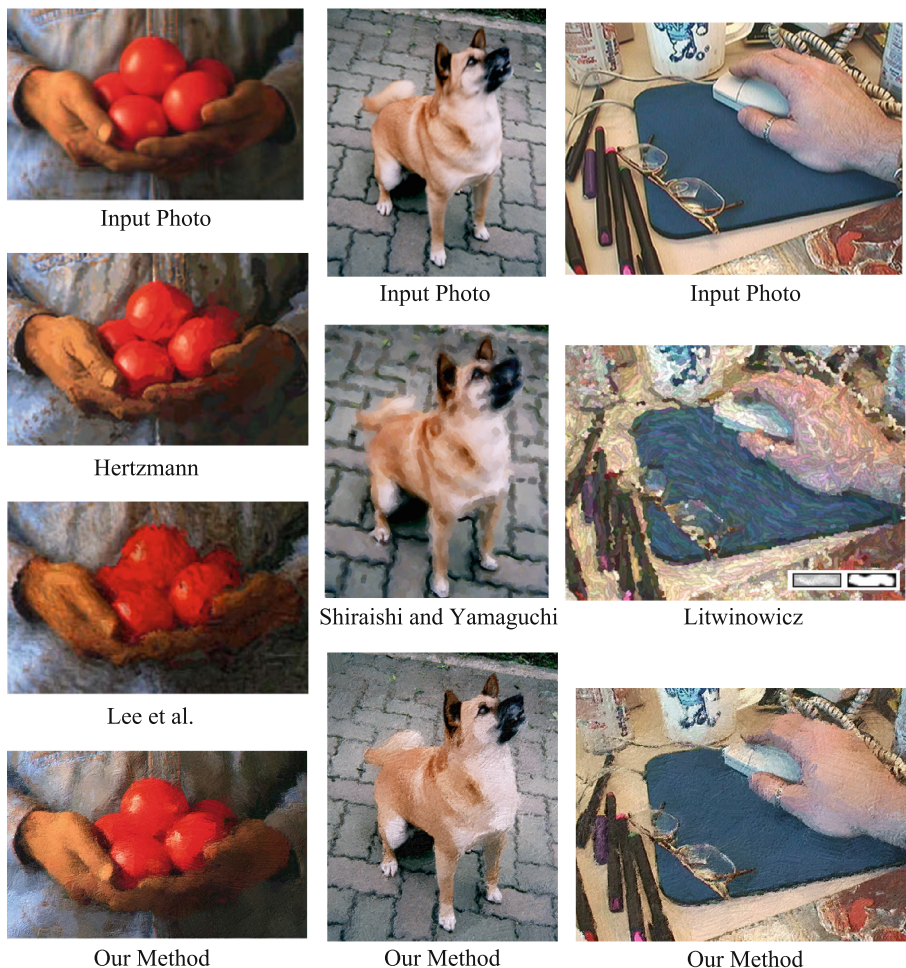
## 6 Experiment Results

The implemented oil-style painting synthesis program has been applied to a diverse set of photos, including sceneries, indoor views, animal photos, and portraits etc. The input photograph that was used to illustrate the concepts of our approach is shown in Fig. 7(a) and the final synthesis result is shown in (b). Figure 7(c) and (d) show the results of increasing levels of abstraction, where the value of  $k$  was set to 15 and 20 respectively.



**Fig. 7.** (a) the original input photo. (b) the final synthesis result. (c) and (d) are results of increasing levels of abstraction to  $k = 15$  and  $k = 20$  respectively.

The proposed method was also compared with existing approaches based on the availability of the original input images. In the first column of Fig. 8, our synthesis result was compared to Hertzmann [3] and Lee et al.’s [5] approaches. Our method demonstrates a good simulation of oil-painting strokes at enhancing the shading of the fruits. In the middle column of Fig. 8, Shiraishi and Yamaguchi’s [9] approach creates a slightly blurring effect to the resulting image. Our method remains the sharpness of the contents and at the same time provides good abstraction of the details, for instance, see the head of the dog. In the right column of Fig. 8, our result was compared to Litwinowicz’s [7] approach. It shows that even in the large color-homogeneous regions, our method is able to simulate the paintbrush effects without the noticeable repetition of brushes. In general, the painting simulation results of our approach contain less systematic impression normally caused by similar shapes of strokes. Our program creates individual unique stroke based on the characteristics of the local image region. For a  $600 \times 400$  pixel image as shown in Fig. 7, the processing time of the developed MATLAB code is 109s running on an Intel Core i7 computer with 16 GB of RAM. The code has not yet been optimized to increase its efficiency.



**Fig. 8.** The comparison of our synthesis results to the exiting approaches.

## 7 Conclusion

A stroke-based oil-painting simulation algorithm has been proposed. The main contributions of the proposed approach different from the existing works include: First, the newly introduced concepts of a complexity map and a point map, both serve as important components in the entire simulation process. Second, the novel method for calculating the main edge direction within a local window, by which an orientation map can be achieved. The proposed method is able to distinguish 16 different orientations. Third, a newly developed stroke generation procedure, by which each stroke can be uniquely created based on the edge distribution within the local image region. Finally, the proposed synthesis approach is also able to simulate different level of abstractions by varying the value of a parameter.

## References

1. Chang, W.-H., Cheng, M.-C., Kuo, C.-M., Huang, G.-D.: Feature-oriented artistic styles transfer based on effective texture synthesis. *J. Inf. Hiding Multimedia Signal Process.* **6**(1), 29–46 (2015)
2. Gooch, B., Coombe, G., Shirley, P.: Artistic vision: painterly rendering using computer vision techniques. In: 2nd International Symposium on Non-photorealistic Animation and Rendering, pp. 83–90 (2002)
3. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: 25th International Conference on Computer Graphics and Interactive Techniques, pp. 453–460 (1998)
4. Kang, H.W., Chui, C.K., Chakraborty, U.K.: A unified scheme for adaptive stroke-based rendering. *Vis. Comput.* **22**(9), 814–824 (2006)
5. Lee, K.J., Kim, D.H., Yun, I.D., Lee, S.U.: Three-dimensional oil painting reconstruction with stroke-based rendering. *Vis. Comput.* **23**(9), 873–880 (2007)
6. Lee, H., Seo, S., Ryoo, S., Yoon, K.: Directional texture transfer. In: Symposium on Non-Photorealistic Animation and Rendering, pp. 43–48 (2010)
7. Litwinowicz, P.: Processing images and video for an impressionist effect. In: 24th International Conference on Computer Graphics and Interactive Techniques, pp. 407–414 (1997)
8. Meier, B.J.: Painterly rendering for animation. In: 23rd International Conference on Computer Graphics and Interactive Techniques, pp. 477–484 (1996)
9. Shiraishi, M., Yamaguchi, Y.: An algorithm for automatic painterly rendering based on local source image approximation. In: First International Symposium on Non-Photorealistic Animation and Rendering, pp. 53–58 (2000)
10. Wang, B., Wang, W., Yang, H., Sun, J.: Efficient example-based painting and synthesis of 2D directional texture. *IEEE Trans. Vis. Comput. Graph.* **10**(3), 266–277 (2004)

Image and Video Technology

7th Pacific-Rim Symposium, PSIVT 2015, Auckland, New Zealand, November 25-27, 2015, Revised Selected Papers

Bräunl, Th.; McCane, B.; Rivera, M.; Yu, X. (Eds.)

2016, XVI, 793 p. 399 illus., Softcover

ISBN: 978-3-319-29450-6