Computational Aesthetics in Graphics, Visualization, and Imaging (2009)
O. Deussen and P. Hall (Editors)

A Painterly Rendering Based on Stroke Profile and Database

SangHyun Seo[1], JinWan Park[1] and KyungHyun Yoon[2]

[1]The Graduate School of Advanced Image Science, Multimedia & Film, Chung-Ang University, Seoul, Korea
[2]The School of Computer Science & Engineering, Chung-Ang University, Seoul, Korea

Abstract

We present a method for producing painting-like images composed of predefined brush strokes. Our proposed method is based on image retrieval method and uses a stroke database. The stroke database consists of transformed copies of the several brush stroke profiles which are obtained from an actual brush stroke. An input image can be reordered in a painterly manner by combination of brush strokes retrieved from the database.

Our method is able to produce a painting with diverse media by changing the type of database, such as oil and pastel. In this process, we present a search algorithm to select an appropriate brush stroke from database and an assessment algorithm to judge whether to draw the retrieved brush stroke on the canvas or not. We also introduce a efficient brush stroke model and way of achieving the appearance of thick paint without physical simulation.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.8]: Application—Graphics Utilities [I.3.4]: Paint systems—Computer Applications [J.5]: Arts and Humanities - Fine Arts—

1. Introduction

NPR (Non-Photorealistic Rendering) is an area of computer graphics concerned with the generation of diagrammatic and artistic images. Various traditional media can be mimicked, such as oils [Her98] [HP00], pen-and-ink illustration [SWHS97] [JNLM05], watercolor [CAS*97], cartoon [KKY01], pencil [LKL06], mosaic [Hau01] and charcoal [MG02].

In particular, image-based painterly rendering algorithms analyze input image using image process and vision technique, and create diverse brush strokes with different attributes, such as orientation, position, color, and length. They have focused on how correctly it can draw or generate attributes of brush stroke automatically in order to depict the features in the source image. Meanwhile, this paper introduces a method different from the paradigm pursued in the existing image-based painterly rendering method.

Our painting model is based on the idea that a painter makes each brush stroke to represent part of the subject as clearly as possible. The painting is built up by repeating this process with increasingly fine sketch with smaller and smaller brushes. In the painting process, a painter has a "slip" as a necessity. A slip is an error in carrying out the intention [Nor83]. The painter's unavoidable slip is one of the important factors that artistic works give inspiration to the humans and occur necessarily during the drawing. Figure 1 illustrates the general painting process.

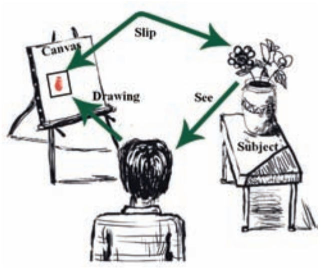


Figure 1: *The painting process*

Although the process may inevitably bring about some slip being against a painter's original intention, it is rather a distinguishing factor between painting and photograph, and is the essence of the beauty of painting. The mechanism under which a human is unable to make a painting just like a photograph as he pleases, this limitation is the staring point to make a creative work. The inevitable slips can be implied by using the several random or restricted factors on the algorithm. By using the random factor in the determination

DOI: 10.2312/COMPAESTH/COMPAESTH09/009-016

process of the brush stroke position and the stroke DB that confine the number and shape of the stroke expressible, we can express the slip of a human to the rendering results.

Although the random factor is one of the approaches many researchers are to evade, some of the theoretically oriented members of the computer art movement suggested that random number generators would take the role of intuition in a human artist's creativity. Random numbers have remained important in all genres of digital art up to present days [DG07] [Nak07].

The painterly image is generated from combination of numerous sets of brush strokes. However, it is very exhausted to generate numerous sets. We construct a stroke database which contains sets of user brush strokes that represent various painting styles, such as oil, pastel and pencil. By retrieving and combining these pre-defined stroke sets, we can create images that appear to have been painted in diverse styles. This approach has some similarities to Photomosaic [Sil97] [BP05] which are images tiled with small piece of different pictures or photographs. But our 'tiles' are brush strokes and not photographs.

We construct our stroke database from user profiles which consist of real stroke samples and retrieve candidate brush strokes from the database, and then decide which are to be used. We also introduce a new method of rendering brush strokes which have the visual effect of the final painting without requiring any physics-based simulation. This involves embossing data which contains predefined height information for each brush stroke. The results that we present are mainly in an oil painting style. But we provide additional examples that show how we can mimic different media by the changing rendering parameters and brush stroke sets.

2. Related work

In this paper, we present a new method of painterly rendering that allows more diverse media than existing techniques. Researches on painterly rendering can be classified by representation methods, which include: physical simulation of brushing through user interaction [BWL04], object-based method using additional information from the 3D data [SGS05], 2D image-based method generating the strokes from two dimensional images [KM06], and painterly animation from video [HE04] [PY08].

We restrict ourselves to two dimensional painterly rendering which was first attempted by Haeberli [Hae90], who introduced a brush stroke as the basic unit of rendering, starting position, color, size, orientation and shape. By changing these attributes he was able to produce image reduced in various styles.

Litwinowicz [Lit97] used straight brush strokes which were aligned with gradients located from the input image, whereas previous researchers relied on user input to orientate the brush strokes.

Hertzmann [Her98] used realistic curved brush strokes of various sizes, combined with a system of layered grids. An input image was divided into layers corresponding to different brush sizes, so that the reordered image is produced by an artist-like progression from coarse to fine brush strokes. Subsequently, Hertzmann [HP00] extended his algorithm to animation, based on optical flow, and improved the appearance of brush strokes using height field data. Hays et al. [HE04] are responsible for a related technique which maintains the temporal coherence of brush strokes in painterly animation.

In other work, Nehab [NV02] introduced a method in which a local region with the same color information are approximated by a single brush stroke using image moment, and Shiraishi et al. [SY00] similarly approximated local regions of the source image with square brush stroke. Gooch et al. [GCS02] used segmentation and morphological operations to create a painting by planning and rendering a resolution-independent set of strokes. This method is differentiated from previous research because it does not rely on the gradient of the image to determine the orientation of brush strokes.

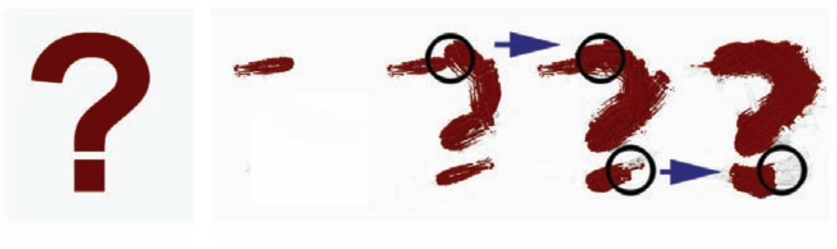
Salisbury et al. [SWHS97] introduced an interactive method which draws oriented strokes based on a user-specified set of example strokes. This algorithm creates pen-and-ink-style line drawings but similar to our fundamental approach which use the pre-defined stroke set.

3. Our approach

Artists construct a painting by a repetitive use of familiar and reliable brush strokes. We are trying to simulate the way in which a painter gradually resolves the inconsistency between the subject and what in on the canvas. The shape and accuracy of the brush strokes depend on the artist's skill and technique. It is often the case, especially with contemporary style, that a small number of strokes are deemed to result in a painterly picture. A usage of finite stroke according to artist's talents can make more natural and artistic works than infinite numbers of stroke. This way is very similar to the process in which human being makes a painting. Our technique also attempts to reduce the difference between the input and output images by repetition of a limited numbers of candidate strokes.

Figure 2 illustrates this evolutionary process of our algorithm in which the painting becomes closer to the source image through successive repainting with several strokes of different colors and shapes. Figure 2-(b) shows a short 'clip' from this process: a red brush stroke is made in the top circle and a white brush stroke in the bottom circle, both of which move the picture closer to the source image. It will be getting closer and closer to the source image if there are more attempts or repainting with more adaptive strokes.

However, in the process of transferring the subject to a canvas, it is always accompanied the distortion of the source



(a) Subject (b) Painting sequence

Figure 2: *Painting by combining strokes*

image caused by the finite brush strokes. Since this distortion by slip is the source of artistic beauty of painting. It is necessary to be controlled properly. Although we use a finite number of brush strokes, we never match the input image exactly. The difference between the images encapsulates the merit of our painting process, and our algorithm controls the nature of that difference by painting the strokes that the computer can choose in the stroke database.

In order to simulate the painting process shown in Figure 2, we present a painting method that limits the strokes the computer can choose, and achieves the optimum or the second optimum result through the combinations of limited strokes. We define the set of the limited strokes as the stroke database.

This method is fundamental difference from previous painterly rendering methods. Earlier methods focus on the ***generation*** of brush strokes that reduce the difference between the subject and the canvas, whereas our algorithm ***selects*** and ***access*** the best brush strokes from a real stroke database.

4. Constructing the stroke database

4.1. Brush stroke profiles

Our database consists of several user brush stroke profile, each of which consists of several brush stroke samples obtained from an actual painter or extracted from existing graphic software such as Photoshop.

The oil brush stroke profiles which are mainly used in this paper was obtained by painting with actual brushes on a back-lighted glass plate and capturing the resulting shape, as shown in Figure 3. The thickness of a stroke is obtained from brightness information and characteristic shapes of strokes are obtained through multiple tries. Some post processing was required to eliminate the noise that occurred during the capture process. Areas of the stroke that are brighter than a user-defined value are regarded as transparent(see Section 5). The other types of stroke profiles discussed in section 7 were derived from commercial software or real painted medium.

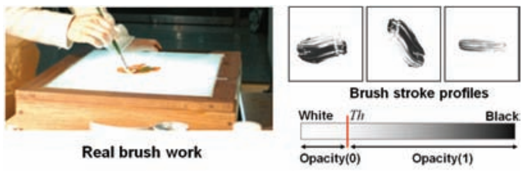
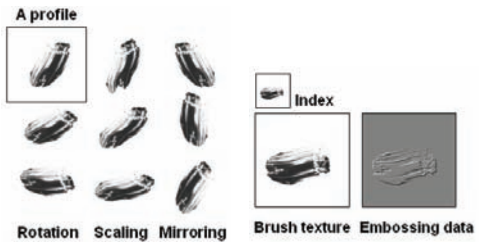


Figure 3: *Example of oil brush stroke profiles*

4.2. Stroke database

The database of oil-painting brush strokes is based on captured strokes, which are transformed by rotation, scaling and mirroring, as shown in Figure 4(a), so as to have enough data without making the input process too arduous. Rotating makes it possible to depict diverse orientations and scaling allows to make various sized brush strokes. Mirroring can increase the diversity of the brush.

For each stroke in the database there is an additional index image and an embossing data. The index image is a reduced copy of a stroke, and is used to increase the speed of the search. The index image is used in the process of searching the appropriate brush stroke from stroke database. In the rendering process, we use high resolution stroke texture and embossing data. Figure 4(b) shows a structure of a database element. The embossing data represents the degree of impasto, which is the thickness of the paint.



(a) Transformed copies of profiles (b) A database element

Figure 4: *Constructing the database from strokes profiles*

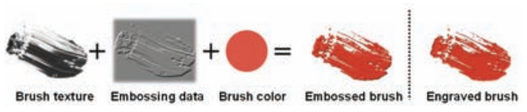
5. Expressing the depth of stroke

The thickness of paint in each brush stroke is modeled by texture and embossing data which are stored in the stroke database. The stroke texture contains opacity information, so that pixels brighter than a threshold value(Th) are not drawn. The embossing data is used to give illusion of depth to the stroke. Embossing is performed by a filter of the same type as that used in Adobe Photoshop.

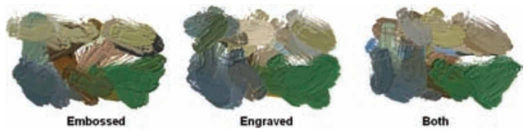
Embossing data is given as difference value of pixel height relying on the orientation of a nominal light source and is stored as relative height value to 128 in the range of 0-255. We make the natural assumption that the light comes from the upper left.

We use the HSV color model to render brush strokes because it represents perceptual color relationships more accurately than RGB. Each opaque pixel of stroke texture is replaced by the brush color. Then we change the RGB color of the brush stroke into HSV. Saturation and hue are inherited without correction and the value is adjusted using the embossing data. Negative embossing data can be used to represent engraved strokes which reverse the orientation of the shadowing. Engraved strokes frequently occur when a paint is applied thickly, and subsequent strokes are made before the first layer has dried.

Figure 5(a) explains this method of obtaining a depth effect. This algorithm is an effective way of producing a depth without physical simulation. Figure 5(b) shows an example of brush strokes with both engraved and embossed brush effect which has very natural depth effect.



(a) Stroke generation



(b) Rendering image combined with embossed and engraved brush strokes

Figure 5: *Stroke generation process and rendering examples*

Herzmann [Her02] obtained a depth effect by shading the accumulated height of all the brush stroke textures. This is simple and effective method but it requires the final height information and normal vector. The Impasto program [BWL04] produced an excellent visual effect by using a physical simulation but it requires special hardware to run in reasonable time. Our technique is much easier to compute because it only uses a simple pixel operation.

The Herzmann's method can control degree of depth by calculating the height field variously. Our algorithm also can control the degree of stroke depth through the control of differences in the height of embossing data. Additionally, existing color of previous strokes drawn on the canvas can be revealed by changing the opacity threshold (Th).

6. Rendering algorithm

Our iterative algorithm, which is shown in figure 2, consists of three stages;

1. Decide the location and the size of the next brush stroked.
2. Search for the brush stroke texture that is next relevant to the location from the stroke database.
3. Decide whether this stroke complements the source image: if so, it is applied; otherwise not.

This process is repeated with decreasing size of brush, from coarse to fine brushes.

6.1. Position, size and color of brush stroke

We determine the positions and sizes of strokes in a similar way to Herzmann [Her98].

We also use the list of brush sizes $B_i = \{B_1, B_2, ..., B_n\}$ and a list of different-sized grids $G_i = \{G_1, G_2, ..., G_n\}$. The grid size for each brush size is given by $G_i = F_g \times B_i$. This approach follows Herzmann's grid system.

We do not use for gradient information from the input image, because we simply search the database for the best stroke for a given grid cell at each grid level, and the strokes in the database have already been rotated. The stroke position is randomly selected in the grid block and the color of the pixel at the position is used as the color of the brush stroke. The stroke positions chosen at each brush size are sorted by intensity before any strokes are drawn, so that the painting process follows a progression from dark to light.

6.2. Stroke Search

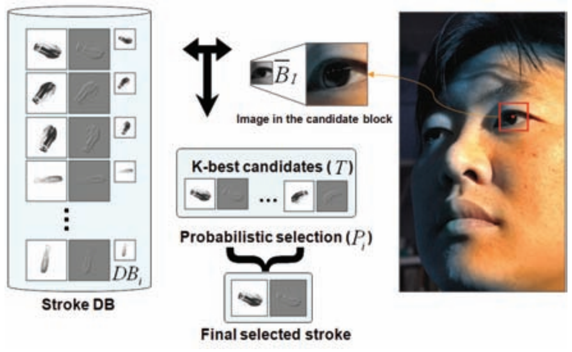

$\overline{B}_1$
Image in the candidate block
K-best candidates (T)
Probabilistic selection (P_i)
DB_i
Stroke DB
Final selected stroke

Figure 6: *Selecting strokes.*

Given a candidate center position(stroke position), we need to search the database for the most suited block to depict the input image. We do this by computing the average difference in color between the candidate block and the brush stroke in the database. This is the criterion that is used to find tiles for a Photomosaic [Sil97].

The average color difference, $D(I_1, I_2)$ between two images I_1 and I_2 is the average of the Euclidean distance between the pair of pixels $I_1(x,y)$ and $I_2(x,y)$ at each coordination (x,y) and is as follow:

$$D(I_1,I_2) = \frac{\sum_x^w \sum_y^h |I_1(x,y) - I_2(x,y)|}{w \times h}, \tag{1}$$

where w is the width of the image and h is its height.

Figure 6 shows the search process. A candidate block image B_I is first extracted from the input image I and then transformed to a grayscale image and reduced to the size of an index image. $\bar{B}_I$ is transformed image. K stroke textures, $T = \{T_1, \ldots T_k\}$ are extracted in ascending order of color difference between $\bar{B}_I$ and the database index image DB_i, and the closest matches are selected.

$$T = Min_K\{D(DB_i, \bar{B}_I) | For\ DB_i \in Stroke\ DB\} \tag{2}$$

K is related with correctness of drawing. Namely, K is slips that are preventable errors in human-computer interaction. A small value of K tends to lead to repetitious strokes in on area of the image, because the same stroke texture is repeatedly selected to be best match to the local color. This problem can be seen at the border and the sides of the nose of the image in Figure 7(b). but if K is too large, as seen in Figure 7(c), the strokes become too irregular and the output image becomes unintelligible.

When K is large, we therefore expand the search for a stroke, using a sampling method to improve the probability of selecting a stroke texture that is similar to the input image.We take the probability method in order to achieve a balance between excessive irregularity and repetitive brush strokes.

For probable approach, we reuse the value of average color difference pre-calculated in the previous step. We must select one stroke texture from T. The probability of choosing a stroke texture T_i from T is denoted P_i . P_i is calculated by follow steps. Firstly, we use normalized color difference (ND_i) between T_i and $\bar{B}_I$. ND_i is calculated by the equation 3:

$$ND_i = \frac{D(\bar{B}_I, T_i)}{\sum_{j=1}^{k} D(\bar{B}_I, T_j)}. \tag{3}$$

As ND_i is small, P_i must be lager. Therefore, ND_i value is in inverse proportion to P_i. P_i is calculated by the ratio of reciprocal of ND_i. P_i is represented by the equation 4:

$$P_i = \frac{1}{(\sum_{j=1}^{k} 1/ND_j) \times ND_i}, \tag{4}$$

where $\sum_{i=1}^{k} P_i = 1$. The appropriate stroke texture is chosen by the probability value.

The number of stroke samples in the stroke database is possible to be expanded massively. An expansion of database allows depicting image variously than the one with small number of stroke samples.

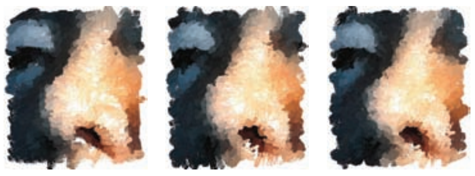

(a) $K = 1$ (b) $K = 10$ (c) $K = 20$

Figure 7: *Output images corresponding to different value of K, with a database of 100 strokes*

6.3. Stroke assessment

Before making a stroke, we need to determine whether its brush texture actually complies with our intention to make the canvas closer to the input image. A real artist may make a similar judgment before applying their brush to the canvas. The criterion that we use for this assessment is whether the average color difference between the areas of the canvas in which the new stroke will be placed and the corresponding block in the input image will be reduced.

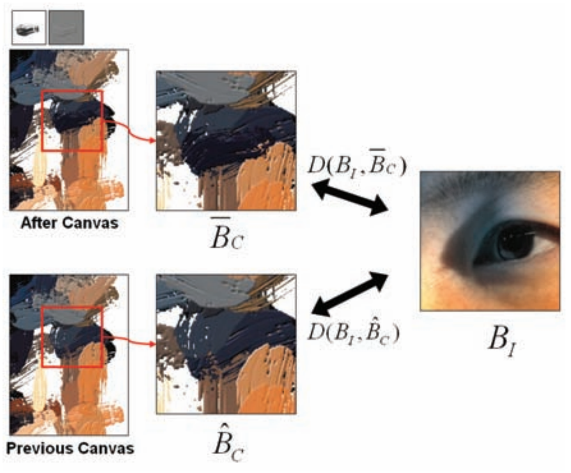


Figure 8: *Stroke assessment*

This requires the input image area B_I, the current canvas image area $\hat{B}_C$, before the new stroke is applied, and next canvas image area $\bar{B}_C$, after the new stroke has been applied. The result image on the canvas B_C corresponding to the candidate block is determined by the following formula 5:

$$B_C = \begin{cases} \hat{B}_C, & \text{if } (D(B_I,\bar{B}_C)\text{-}D(B_I,\hat{B}_C)) < F \\ \bar{B}_C, & \text{otherwise} \end{cases} \quad (5)$$

Figure 8 illustrates the stroke assessment. If $D(B_I,\hat{B}_C)$ is smaller than $D(B_I,\bar{B}_C)$, applying the new stroke will damage the image, so it is not applied to the canvas.

A constant F controls the acceptance ratio, which is the proportion of strokes that are actually drawn. F has the same range as the average color difference. If F is -1, no strokes are ever drawn, and if F is 1, all strokes are drawn. In practice we use value of F between -0.1 and 0.1 because our method is sensitive between them.

Figure 9 shows how affects the acceptance ratio and figure 10 its effect on the changes of strokes.

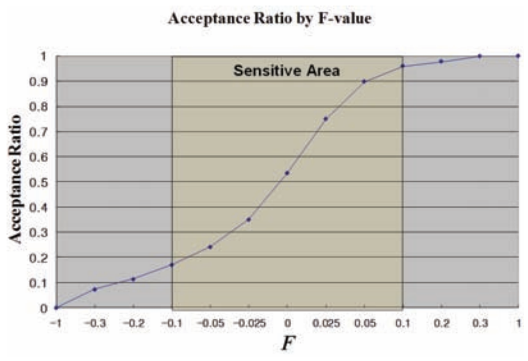
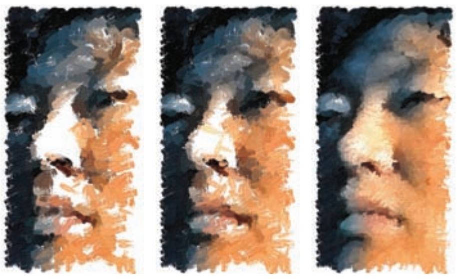


Figure 9: *Relationship between F and acceptance ratio*



(a) $F = -0.2$ (b) $F = -0.1$ (c) $F = 0.0$

Figure 10: *Output images obtained at different value of F ($K = 5$)*

7. Experimental and results

We mainly experimented with an oil painting style, although other painting styles can be produced by changing the stroke database (see Section 3.1). We varied the following parameters in our experiments:

- Stroke DB type (DB_{type}): oil, pastel, pencil, colored pencil, and user defined types.
- Number of strokes in the DB (DB_{num}): affects the diversity of stroke shapes and orientation in the DB.
- Brush size (B_i): size list ($B_1 \ldots B_n$) with n sizes of candidate block (see Section 6.1).
- Grid size (F_g): controls the grid size which interacts with the brush size list (see Section 6.1).
- Brush search count (K): the number of strokes retrieved from the stroke DB (see Section 6.2).
- Acceptance control factor (F): Controls whether to draw the retrieved stroke or not on the canvas (Section 6.3).

Figure 11 shows output images rendered with databases that mimic stroke profiles of oil, pastel, pencil, and colored pencil styles. The images on the left top are the original stroke profiles used to construct the database. Each database contains a hundred transformed copies of each profile. The number of strokes in the database is able to be expanded by user. The resolution of two sources in figure 11 is 1024×768 and each painting took under 30 seconds to generate on a 2.4GHz Pentium IV processor without graphics hardware.

Figure 12 shows unique results archived with a bespake user-defined profile.

Figure 13 shows the results of high resolution and quality that we have got by permitting the redundancy of brush sizes. The thickness of stroke is revealed well and the features of the image are expressed delicately. Our method permits the change of profile to match each artist's personality, and therefore, can include the creativity of the artist to the algorithm. And, it can be utilized frequently in interactive computer art field as we can secure appropriative performance through the control of brush size and quantity of database.

To mention another application of the suggested algorithm, it is a type of process that can immediately apply to animation. By using the process of stroke assessment explained in section 6.3, we can reproduce the painting on glass method that can modify the changed places only without any difficulty. Figure 14 shows a few frame of animation result.

8. Conclusion and future works

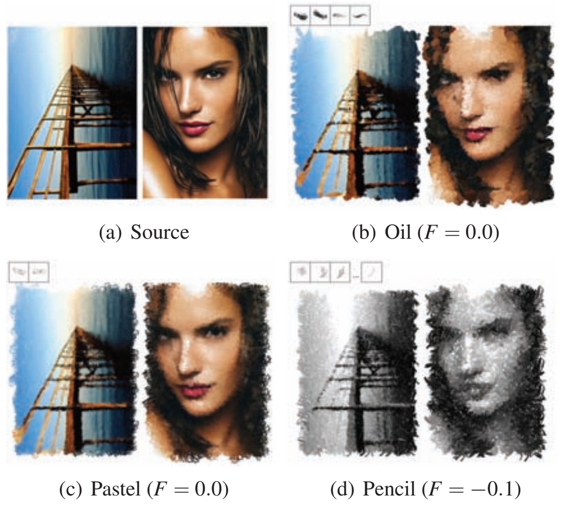
We have introduced a new method of painterly rendering based on image searching technology. Instead of generating each stroke repeatedly, a database of strokes is generated our algorithm selects the most suitable stroke for each situation. This is a simple, efficient and flexible process.

We allow the user to define the types and shapes of strokes in the database, and also allow variation of the result by providing rendering parameters. We also introduced a simple method of giving a depth effect to brush strokes, which does not use a physical simulation by using only two types of simple data, stroke texture and embossing data. Our method is able to:

- make stroke database using real stroke samples drawn by painters.
- make a painting represented with diverse media by changing the stroke database.
- select the level of abstraction by controlling how the strokes are combined.
- give the impression of solid media without physical simulation.

The number of strokes determines the quality of image produced by our algorithm and computing performance. If the database contains many strokes with different shapes, orientations and sizes, we can get good results. However, we need to reduce the size of the database in order to use our method in real-time because its performance is limited by the time required to search for the best stroke in the stroke database.

We hope to overcome this limitation by improving the speed of searching the database, a better brush shape distribution, and may require a better organization of the brushes in the database. Another interesting line of work is processing of video for painterly animation. New techniques will be required to maintain temporal coherence for our approach.


(a) Source (b) Oil ($F = 0.0$)

(c) Pastel ($F = 0.0$) (d) Pencil ($F = -0.1$)

Figure 11: *Images produced with different stroke profiles and databases($DB_{num} = 100, Bi = \{200,150,60,40\}, K = 3, Fg = 0.25$)*

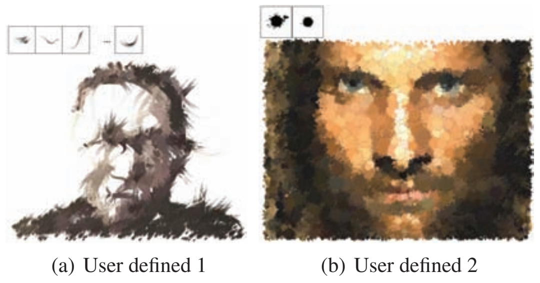

(a) User defined 1 (b) User defined 2

Figure 12: *Images created using a user-defined database DB($Fg = 0.25$)*

Acknowledgement

This work was supported by the Korea Science and Engineering Foundation(KOSEF) through the National Research Lab. Program funded by the Ministry of Science and Technology (No. R0A-2008-000-20060-0)

References

[BP05] BLASI G. D., PETRALIA M.: Fast photomosaic. In *Poster of WSCG05* (2005), pp. 15–16.

[BWL04] BAXTER W., WENDT J., LIN M. C.: Impasto - a realistic, interactive model for paint. In *Proc. NPAR'04* (2004), pp. 45–56.

[CAS*97] CURTIS C. J., ANDERSON S. E., SEIMS J. E., FLEISCHER K. W., SALESIN D. H.: Computer-generated watercolor. In *Proc. SIGGRAPH'97* (1997), pp. 421–430.

[DG07] DIPAOLA S. R., GABORA L.: Incorporating characteristics of human creativity into an evolutionary art algorithm. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation* (2007), ACM, pp. 2450–2456.

[GCS02] GOOCH B., COOMBE G., SHIRLEY P.: Artistic vision: Painterly rendering using computer vision techniques. In *Proc. NPAR'02* (2002), pp. 83–90.

[Hae90] HAEBERLI P.: Paint by numbers: Abstract image representations. *ACM SIGGRAPH Computer Graphics 24*, 4 (1990), 207–214.

[Hau01] HAUSNER A.: Simulating decorative mosaics. In *Proc. SIGGRAPH'01* (2001), pp. 573–580.

[HE04] HAYS J., ESSA I.: Image and video based painterly animation. In *Proc. NPAR'04* (2004), pp. 113–120.

[Her98] HERTZMANN A.: Painterly rendering with curved brush strokes of multiple sizes. In *Proc. SIGGRAPH'98* (1998), pp. 453–460.

[Her02] HERTZMANN A.: Fast paint texture. In *Proc. of NPAR'02* (2002), pp. 91–96.

[HP00] HERTZMANN A., PERLIN K.: Painterly rendering for video and interaction. In *Proc. NPAR'00* (2000), pp. 7–12.

[JNLM05] JEONG K., NI A., LEE S., MARKOSIAN L.: Detail control in line drawings of 3d meshes. *The Visual Computer 21*, 8 (2005), 698–706.

[KKY01] KANG D., KIM D., YOON K. H.: A study on the real-time toon rendering for 3d geometry model. In *Proc. IV'01* (2001), pp. 391–396.

[KM06] KASAO A., MIYATA K.: Algorithmic painter: a npr method to generate various styles of painting. *The Visual Computer 22*, 1 (2006), 14–27.

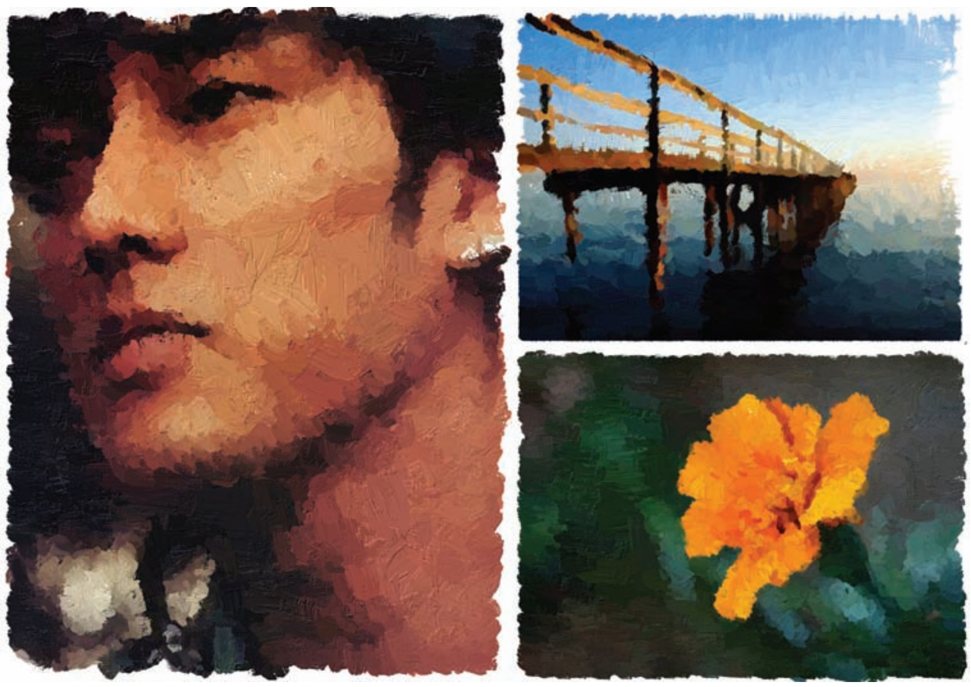


Figure 13: *High quality images created by duplicated brush sizes*($DB_{num} = 300, Fg = 0.25, Bi = \{200,150,100,100,80,80\}, K = 10, F = 0.0$)

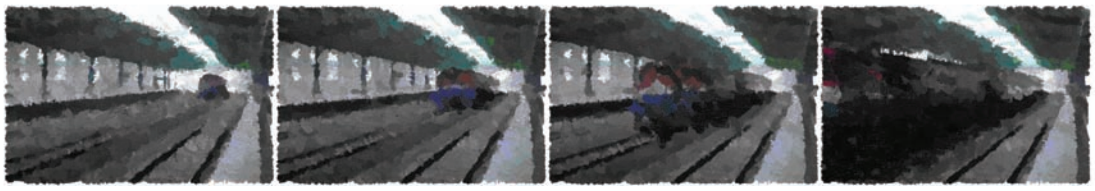


Figure 14: *The examples of animation. F is* 0.0 *in the first frame and F is* −0.15 *in the remainders*

[Lit97] LITWINOWICZ P. C.: Processing images and video for an impressionist effect. In *Proc. SIGGRAPH'97* (1997), pp. 407–414.

[LKL06] LEE H., KWON S., LEE S.: Real-time pencil rendering. In *Proc. NPAR'06* (2006), pp. 37–45.

[MG02] MAJUMDER A., GOPI M.: Hardware accelarated real time charcoal rendering. In *Proc. NPAR'02* (2002), pp. 59–66.

[Nak07] NAKE F.: Computer art: creativity and computability. In *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition* (2007), ACM, pp. 305–306.

[Nor83] NORMAN D. A.: Design rules based on analyses of human error. *Commun. ACM 26*, 4 (1983), 254–258.

[NV02] NEHAB D., VELHO L.: Multiscale moment-based painterly rendering. In *Proc. of SIBGRAPI'02* (2002), pp. 244–251.

[PY08] PARK Y., YOON K.: Painterly animation using motion maps. *Graph. Models 70*, 1-2 (2008), 1–15.

[SGS05] SCHLECHTWEG S., GERMER T., STROTHOTTE T.: Renderbots-multi-agent systems for direct image generation. *Computer Graphics Forum 24*, 2 (2005), 137–148.

[Sil97] SILVER R.: *Photomosaics*. Heny Holt, 1997.

[SWHS97] SALISBURY M. P., WONG M. T., HUGHES J. F., SALESIN D. H.: Orientable textures for image-based pen-and-ink illustration. In *Proceedings of SIGGRAPH 97* (Aug. 1997), pp. 401–406.

[SY00] SHIRAISHI M., YAMAGUCHI Y.: An algorithm for automatic painterly rendering based on local source image approximation. In *Proc. NPAR'00* (2000), pp. 53–58.