# Painterly Rendering Framework from Composition

Chi Chu
Department of Computer Science
National Chiao Tung University
1001 Ta Hsueh Rd., Hsinchu,
Taiwan 300, R.O.C
maktub.cs94g@nctu.edu.tw

Zen-Chung Shih
Department of Computer Science
National Chiao Tung University
1001 Ta Hsueh Rd., Hsinchu,
Taiwan 300, R.O.C
zcshih@cs.nctu.edu.tw

## ABSTRACT

Painterly rendering has recently drawn considerable attention from graphics researchers. However, the state of the art is neither systematic nor evaluative. This work presents a novel painterly rendering framework. The painting process is decomposed into three stages to satisfy the needs of developers and users of painterly rendering algorithms and programs. The framework comprises three systems, namely primitive mapping, rendering and mark systems, and is inspired by John Willats' perceptual decomposition of the painting process presented by [Wil97]. Moreover, the rendering system is further decomposed into four independent modules, namely initial point, path, cross-section and color. The independence of each module makes new styles easy to generate by combining existing styles, or constructing complex styles from simple styles. The proposed framework shows the power of painterly rendering algorithm, which can not only imitate existing styles, but also generate new styles. Furthermore, parameters in rendering systems are specified hierarchically. Users only need to specify the user parameters, which are then automatically converted into system parameters during rendering. This approach is crucial to facilitating the use of the program by end-users.

### Keywords

painterly rendering, non-photorealistic rendering, hierarchical composition

## 1. INTRODUCTION

Painterly rendering is of priority concern in non-photorealistic rendering. The process takes an ordinary image (probably captured by a digital camera) as the input, and generates another image, representing a particular painting style, as its output. Although this problem has been addressed for several years, the state-of-art is far from the original aim. First, the algorithms are hard-wired to their objective painting styles, and therefore are able to generate only a few particular styles, but are neither systematic nor evaluative. Thus, these algorithms cannot be easily integrated to generate desired new styles. Although painting involves creation, but the current algorithms can not achieve this function, these algorithms generate various painting styles by changing the parameters. However, the parameters are related to their implementation, rather than to the painting style. Therefore, present systems are not intuitive for end users. This work develops a general framework for painterly rendering to alleviate these limitations.



The proposed framework is inspired by the book "Art and Representation" by John Willats [Wil97]. Willats divided the painting process into two systems, the drawing and the denotation systems. Fr´edo Durand [Dur02] recently extended Willats' framework into four sub-systems, namely the spatial, primitive, attribute and mark systems.

The proposed framework resembles that of Durand's work, except that the spatial system replaced by perspective projection (projection in photograph). The mark system is similar to that of Durand's. His primitive system and attribute systems are combined into the proposed primitive system, along with the rendering system, because assigning visual properties to a picture primitive is different from depicting it. The rendering system manages depicting a picture primitive. Moreover, primitive mapping does not simply choose from different primitives, but also concerns the mapping of attributes in primitives. For instance, an impressionist prefers high-tone pure colors (for example: yellow, green, orange), so a

primitive mapping system can map an ordinary color in the primitive to a high-tone pure color.

Thus, the whole framework comprises three components, namely primitive mapping, rendering and mark systems. The primitive mapping system relates to the mapping among various scene primitives. The rendering system synthesizes different painterly styles based on various scene primitives. The mark system depicts the actual physical implementation of strokes generated by the rendering system.

The rendering system is further decomposed it into four independent modules, determining stroke initial point, path, color and cross-section, to introduce creation. The independence of each module makes a new style easy to generate by combining existing styles or constructing complex style from simple styles. This novel system shows the power of the painterly rendering algorithm, which can not only imitate existing styles, but also generating new styles.

The parameters in rendering system can be classified hierarchically as user parameters, system-dependent parameters and system-independent parameters. The user only needs to specify the user parameters, which are automatically converted into system parameters in rendering. We believe that this parameter classification is crucial to ensure that end-users can easily run the program. Figure 1 illustrates the flow of the proposed framework.

## 2. Related works

Painterly rendering algorithms have been studied for several years [Chi06, Col02, Goo02, Hae90, Hay04, Her98, Her03, Lit97, Mei96, Ols05, Sch05]. Haeberli [Hae90] provides a paradigm for painterly rendering, in which a painting is synthesized by an ordered collection of brush strokes, each having its own color, shape, size and orientation. Various painting effects can be created by adapting these strokes. Many painterly rendering algorithms follow this paradigm and are designed to be automatic. However, these algorithms are hard-wired to their codes, so do not provide much variation on possible painting styles, and cannot be used by artists to guide the synthesizing process. On the contrary, the proposed framework involves artist's creation by module composition.

Artificial intelligence is applied to painterly rendering algorithm in [Col02, Sch05]. However, because of the weak power of current computer artificial intelligence, these algorithms do not differ from automatic algorithms much. Besides, it is not intuitive for user to design salience map [Sch05] or agent behavior [Col02] because they are relative to the underlying algorithm rather than artist's view.

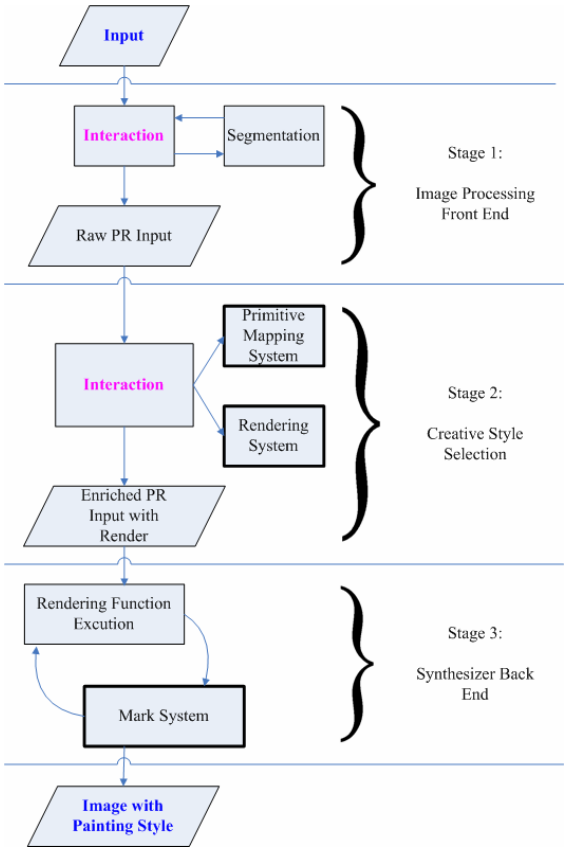



Figure 1. The flow of proposed framework.

Semi-interactive and interactive algorithm [Chi06, Goo02, Hae90, Hal02, Ols05, Kal02] are those in which an artist can become involved in the process of synthesis. Interaction can be achieved by simply modifying parameters or mimic artist's painting process [Gra04, Hal02, Kal02].

Halper [Hal02] present user a way to design nonphotorealistic images based on series of elementary operations. These elementary operations include scene modifiers, stroke modifiers and image modifiers. By linking these operations, image of novel styles can be synthesized. Halper's elementary operations are similar to the independent modules in rendering system of the proposed framework. However, the independent module is more fundamental because it is only a partial style rather than a complete style represented by Halper's elementary operation.

Stéphane [Gra04] presented a programmable interface for non-photorealistic line drawing, in which the topology of a view map of lines is extracted from a three-dimensional polygon mesh. Three user definable modules, namely selecting, chaining and splitting, are then applied. Each module refines the lines, which are then drawn on the final image. This framework is very flexible to synthesize different line drawing styles. The primitive mapping

system of the proposed framework is similar to Stéphane's framework.

## 3. Overview

The proposed framework comprises three main stages (Fig. 1): Image Processing Front End (IPFE), Creative Style Selection (CSS) and Synthesizer Back End (SBE). Each stage corresponds to painters' actual painting process: determine what to paint (IPFE), choose the painting style (CSS) and paint (SBE). The first two stages (IPFE and CSS) involve user interaction. The third stage (SBE) is automatic.

### 3.1 Image processing front end stage

As mentioned before, a painterly rendering algorithm has many inputs, which have to be unified to construct a framework, since the image itself (as a set of pixels) is not intuitive for end-users or artistic who want to give guidelines to a particular painterly rendering algorithm. A unified input is also needed to combine different algorithms.

The input of the proposed framework, Raw PR Input in Fig. 1, comprises a hierarchy of objects and primitives. Objects are a high-level concept, namely what the end user wants to paint. The object hierarchy denotes the way that painters view the scene. For instance, a scene comprises a chair and an apple, and the chair comprises four legs. Besides, object relationship in the hierarchy provides rendering system with necessary information, for example, the "ImpMonet" rendering function in Section 5.3. A primitive is a low-level concept that is involved in the painting process, and denotes the painter's perception of a single object. For instance, a painter may perceive a region with red color from the apple.

The Image Processing Front End stage converts an input image into a hierarchy of objects and primitives. The interaction stage of IPFE stage requires a user to use image segmentation tools to segment areas in an input image of interest. Any commercial tool can be used to finish this task. Figure 2 illustrate an example of an object hierarchy.

### 3.2 Creative style selection stage

The *Creation Style Selection* (CSS) stage helps a user to determine the painting style, and consists of two parts: modification and selection of primitives, and choosing the style for each selected primitive. These two parts correspond respectively to the primitive mapping and rendering systems [Dur02], and are discussed in detail in Sections 4 and 5 respectively.

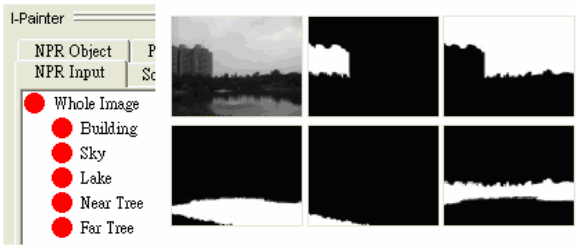



Figure 2. A hierarchy of objects and the corresponding primitives: one base object and five first layer sub-objects. Primitives are two-dimensional images represented by masking images.

### 3.3 Synthesizer back end stage

The system paints automatically once the CSS stage is finished. The mark system in the Synthesizer Back End stage performs the painting process. The proposed framework currently supports oil painting. Section 6 discusses this part in detail.

## 4. Primitive system

A primitive system helps users to select the object to be painted and the modification that should be applied before painting. The modification is required for two reasons. First, humans represent scene objects in their own way, similar to the salience map in previous approaches. Second, different artists interpret scene objects differently. For instance, Van Gogh would draw twisted contour lines, while Renoir preferred draw smooth contour lines. This modification must be separated from rendering functions, since it simplifies the design of rendering functions, and makes the whole painting process easy to understand.

A primitive has two fundamental properties, namely shape and color. Shape is represented as a two-dimensional mask, and color is represented as a two-dimensional color buffer in which pixels adopt the RGB color model. Additionally, the concept of extendedness [Wil97] is employed to represent human perceptions of object shape.

### 4.1 Extendedness

Willat [Wil97] employed the concept of extendedness to describe human perceptions of shape. The extendedness specifies the relative extensions of primitive in different directions of space. This work extends Willat's extendedness concept to synthesize the quick drawing effect of Impressionism (Section 5.4). The extendedness in the proposed framework is defined as a list of spans, each with a starting direction, ending direction and intensity. These spans are obtained by first threshold the length from pixel position to object center and then merge pixels with similar length. Extendedness is used in the

framework to generate a “shape direction” for each point in primitive. The shape direction is then used to derive stroke direction in a rendering system.

## 4.2 Elementary operators

Four selection operators are available, namely selection, merging, subtraction and sorting. The selection operator takes a primitive as input, and decides whether to select it based on the information included in the primitive. For instance, selection may depend on the importance value or type of an object where the primitive belongs. Users can implement their own selection operators based on complex functions. Several built-in selection operators are available, including selection by object importance, object id and object type.

The merge operator takes two primitives as input and the merged result as output. Users can customize the layering behavior by merging insignificant primitives. For instance, far primitives can be merged together, and near or important primitives should be treated separately. The built-in merge operator is implemented using set union, in which the shape mask and color buffer of primitive are considered as sets.

The subtraction operator can be employed in rendering functions. The built-in subtraction operator is implemented by set subtraction, in which the shape mask and color buffer of primitive are considered as sets.

The order of primitive drawing strongly influences the result. The sorting operator is provided to determine the order of primitives, and results in a partial order of primitives in which primitives of the same anti-chain are in the same painting layer. Trivial build-in sorting operator is not available, since this operation involves creativity, and depends entirely on the user.

# 5. Rendering system

The rendering system comprises a set of rendering functions, each taking a mapped primitive as input and generating stroke definitions. A stroke definition includes the path, cross-section at each control points, initial bristle attributes and physical-effect parameters. All these parameters are required in the mark system. The rendering function can also access global information, i.e. the input and Canvas object used in mark system. The generation of stroke definition comprises several steps. Stroke definitions generated from each step are fed into the mark system, whose results affect the next stroke definition generation step.

## 5.1 Module composition

Most current painterly rendering algorithms are black boxes that generate all stroke definitions. These algorithms are hard to modify or combine, so cannot be employed to create a new style. This problem is solved herein using module composition. The basic idea is that although artist’s creation can not be realized by computer algorithm, common fundamental painting techniques do exist among these artists and these techniques can be achieved by computer. Thus, partial styles are developed instead of complete styles. Users can apply their creation to the painting synthesizing process.

The rendering function is divided into four modules. Each module is responsible for generating one kind of stroke definitions, namely stroke initial point, stroke path, stroke color and stroke cross-section. The stroke initial point module determines the distribution of strokes, and consists of a set of points representing the initial point of the path of each stroke. The stroke path module creates a path from a given initial point, and consists of a set of control points for each stroke definition. The stroke color module determines the color of the stroke. The stroke cross-section module determines the stroke cross-section of the stroke path, and consists of a set of cross-section definitions corresponding to each control point. Figure 3 illustrates these four modules.

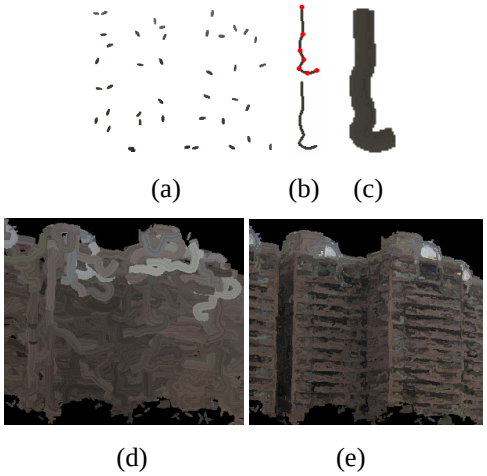



Figure 3: Rendering modules and a basic style. (a) stroke initial points; (b) stroke path formed by control points (red dots); (c) stroke with cross-sections and color defined; (d) first layer of painting by applying four modules sequentially; (e) a basic style resulting from three layers.

To combine different rendering functions, these four modules are made independent to each other. A new style is created by simply choosing these four modules from existing rendering functions and combining them. This novel approach shows the power of the proposed painterly rendering algorithm,

which can not only imitate existing styles, but also generate new styles. Existing rendering functions are also easy to modify. Modification can be applied on individual modules without affecting the other modules.

## 5.2 Parameter hierarchy

Most painterly rendering algorithms adopt parameters to control the variation of styles. However, they derived parameters are typically derived from the algorithm designing stage, making them unintuitive for end-users. The proposed algorithm solves this problem by providing a hierarchical representation of parameters. Conceptually, users who are unfamiliar with the algorithm can simply specify high-level parameters, which are automatically converted to low-level parameters.

The parameters are classified into four levels, namely the style, user, system-dependent and system-independent parameters. The lowest level is system-dependent parameters. Only these parameters are adopted in the rendering process. Parameters in all other levels are converted to the lowest level. System-dependent parameters are parameters that depend on the rendering target, for instance, the "surrounding color" in "ImpMonet" rendering function in Section 5.3. System-dependent parameters can be determined by rendering function or specified by user.

User and style parameters are high-level parameters. User parameters have to be determined by users due to algorithm's limitations. Style parameters are summaries system parameters. Different style parameters in the same rendering function represent minor variations of the same style. Figure 4 illustrates the concept of parameter hierarchy.

Several painting styles were implemented to show the effeteness of our framework. The following sub-sections discuss these styles in detail.

## 5.3 Style one: impressionism, Monet

The Monet rendering function was used to synthesize a series of paintings by a series of paintings by Monet during 1899~1901. The subjects in these paintings are buildings, rivers and skies immersed in the morning mist. To depict the mist, all objects are painted casually and burred. However, the painting style for each subject (building, river and sky) is slightly different.

To synthesize these effects, three rendering functions were implemented to synthesize the building, river and sky. First, the "ImpBuilding" rendering function would automatically find the two most distinct base colors representing the surrounding color (obtained from the "surrounding" object, i.e. the sky) and the object's instinct color (i.e. the diffuse color). These two colors are blended to form the stroke color. The direction of the stroke path is modified to follow the shape of the object. The initial point is seeded randomly to mimic casual painting effects. Figure 4 illustrates some results of this function.

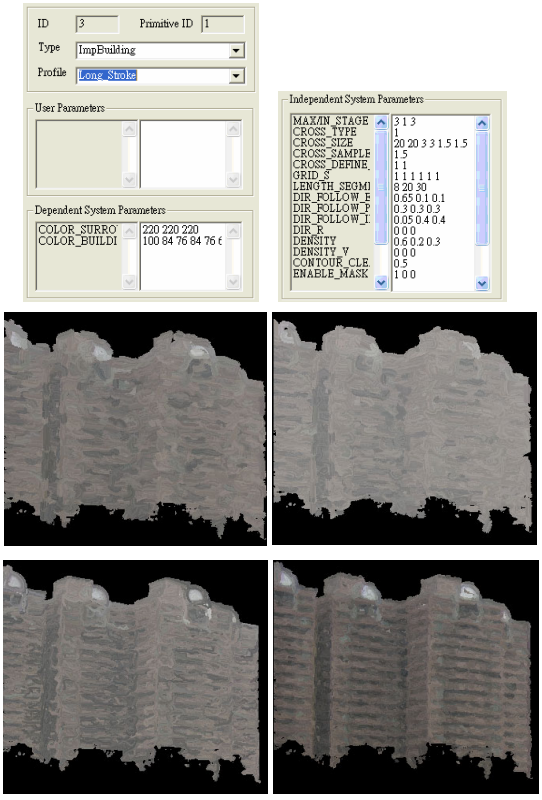



Figure 4: Parameter hierarchy of rendering function "Impressionist Building (ImpBuilding)": Four style parameters: "default", "more impression", "long stroke" and "clear"; User parameters are empty; Two system dependent parameter: surrounding color and building color; Nineteen system independent parameters.

After the "ImpBuilding" rendering function is designed, the other two rendering functions, "ImpWater" and "ImpSky", were implemented as extensions of "ImpBuilding". Because of the separation of modules and the concept of inheritance, the similarities of these three styles could be preserved, thus focusing only on the difference. For instance, the "ImpSky" rendering function uses three base colors: the surrounding color, the object's instinct color and the sun color. Thus, the "ImpSky" was obtained by modifying the color module of "ImpBuilding".

### 5.4 Style two: impressionism, quick draw

The quick draw rendering function was used to synthesize the quick drawing effect of Impressionist paintings. The following objective must be achieved to synthesize this effect. First, the number of strokes representing the foreground object should be minimized. These strokes should reveal the shape of the object. The color distribution of the foreground object should be much less noticeable than its shape during synthesis. Second, neither the shape nor the color of the background objects should be noticeable during synthesis.

To attain the first aim, the stroke path did not follow the normal direction of image gradient. The blending "shape direction" and normal direction of image gradient were used instead. The shape direction was obtained from primitive extendedness, as explained in Section 4.1. Additionally, the stencil buffer of canvas was employed to avoid overlapped strokes. To attain the second aim, the color buffer of the primitive in the background object was blurred by blending each pixel's color value with the mean color of the entire color buffer. Figure 6 illustrates some results of this function.

### 5.6 Other styles

Pen-and-ink styles including half-toning, stippling and mosaics [Str02] also have been developed. By combining modules in pen-and-ink style with modules in painterly rendering style, novel styles can be synthesized. Corresponding results are displayed in Figure 7.

## 6. Mark system

The list of stroke definitions generated by the rendering system is fed into the mark system that draws these strokes on canvas. The physical implementation of stroke definition depends strongly on the target painting media. For instance, the water color mark system obviously should differ from oil painting mark system a lot.

The proposed mark system has three components, a brush model, a bristle canvas interaction system and a random system. The brush model includes objects such as pigment, canvas, stroke paths and stroke cross sections (round and flat). In the bristle canvas interaction system, the bristle location is determined by the stroke path and stroke cross section. Each bristle contains pigment, which is placed on canvas. The interaction between bristle and canvas occurs on every contact along the stroke path. This mark system is extended from Way's [Way01] stroke model.

## 7. Results

Results, including the individual styles and the compositions of styles, are now presented in Figure.5~7. The framework is implemented in C++ language on a laptop with a Pentium 1.5G CPU and 512 MB RAM. Running time ranges from thirty seconds to twenty minutes. Full size images and additional results are contained in the supplementary files.

## 8. Conclusion and future work

This work has presented a flexible painterly rendering framework. A developer can extend this framework by customizing its components, enabling different painterly algorithms to be placed within it. The division of this framework is also based on the actual process of painters, which means that the generated results can be evaluated esthetically. Several painting styles were synthesized to indicate the effectiveness of the proposed framework. Several results are presented in each style, requiring extension or modification of each component of this framework. The extension and modification processes are clear and intuitive.

Future work will attempt to improve the proposed framework in the following ways. A spatial system will be added to the framework. Such a system is especially important in oriental painting styles. We will attempt to discover how to establish a complicated projection system in painting processes. The mark system will also be improved. Only simple effects of oil painting have been synthesized so far. Further synthesis experiments will be performed in the near future.

## 9. REFERENCES


[Bom91] Bomford, D., J. Leighton, J. Kirby, and A. Roy, Art in the Making Impressionism, Yale University Press, New Haven, CT, USA, 1991.

[Chi06] Chi, M.T. and T.Y. Lee, "Stylized and Abstract Painterly Rendering System Using a Multiscale Segmented Sphere Hierarchy", In IEEE Transactions on Visualization and Computer Graphics, Volume 12, Issue 1, IEEE Computer Society, Los Alamitos, CA, USA, 2006, pp. 61-72.

[Col02] Collomosse, J.P. and P.M. Hall, "Painterly Rendering using Image Salience", In Proceedings of the 20th Eurographics UK Conference (June 2002, Leicester, UK), IEEE Computer Society, Los Alamitos, CA, USA, 2002, pp. 122-128.

[Dur02] Durand, F., "An invitation to discuss computer depiction", In Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering (June 3-5, 2002,

Annecy, France), ACM Press, New York, USA, 2002, pp. 111-124.

[Goo02] Gooch, B., G. Coombe, and P. Shirley, "Artistic Vision: Painterly Rendering Using Computer Vision Techniques", In Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering (June 3-5, 2002, Annecy, France), ACM Press, New York, USA, 2002, pp. 83-90.

[Gra04] Grabli, S., E. Turquin, F. Durand, and F.X. Sillion, "Programmable Style for NPR Line Drawing", In Rendering Techniques 2004: Eurographics Symposium on Rendering (June 21-23, 2004, Norrköping, Sweden), Eurographics Association, Switzerland, 2004, pp. 33-44.

[Hae90] Haeberli, P., "Paint by numbers: abstract image representations", In Proc. 17th Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH), Volume 4, ACM Press, New York, USA, 1990, pp. 207-214.

[Hal02] Halper, N., S. Schlechtweg, and T. Strothotte, "Creating Non-Photorealistic Images the Designer's Way", In Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering (June 3-5, 2002, Annecy, France), ACM Press, New York, USA, 2002, pp. 97-104.

[Hay04] Hays, J. and I. Essa, "Image and Video Based Painterly Animation", In Proc. 3rd ACM Symposium on Non-Photorealistic Animation and Rendering (June 7-9, 2004, Annecy, France), ACM Press, New York, USA, 2004, pp. 113-120.

[Her98] Hertzmann, A., "Painterly Rendering with Curved Brush Strokes of Multiple Sizes", In Proc. 25th Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH), ACM Press, New York, USA, 1998, pp. 453-460.

[Her03] Hertzmann, A., "A Survey of Stroke-Based Rendering", In IEEE Computer Graphics & Applications, Special Issue on Non-Photorealistic Rendering, Vol. 23, No. 4, IEEE Computer Society, Los Alamitos, CA, USA, 2003, pp. 70-81.

[Kal02] Kalnins, R.D., L. Markosian, B.J. Meier, M.A. Kowalski, J.C. Lee, P.L. Davidson, M. Webb, J.F. Hughes, and A. Finkelstein, "WYSIWYG NPR: Drawing Strokes Directly on 3D Models", In Proc. 29th Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH), ACM Press, New York, USA, 2002, pp. 755-762.

[Lit97] Litwinowicz, P., "Processing Images and Video for An Impressionist Effect", In Proc.24th Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH), ACM Press, New York, USA, 1997, pp. 407-414.

[Mei96] Meier, B.J., "Painterly Rendering for Animation", In Proc. 23th Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH), ACM Press, New York, USA, 1996, pp. 447-484.

[Ols05] Olsen, S.V., B.A. Maxwell, and B. Gooch, "Interactive Vector Fields for Painterly Rendering", In Proceedings of Graphics Interface 2005 (May 9-11, Victoria, Canada), volume 112 of ACM International Conference Proceeding Series, Canadian Human-Computer Communications Society, A K Peters, LTD., 2005, pp. 241-247.

[Sch05] Schlechtweg, S., T. Germer, and T. Strothotte, "RenderBots--Multi-Agent Systems for Direct Image Generation", In Computer Graphics Forum, Volumn 24, number 2, Eurographics Association, Switzerland, 2005, pp. 137-148.

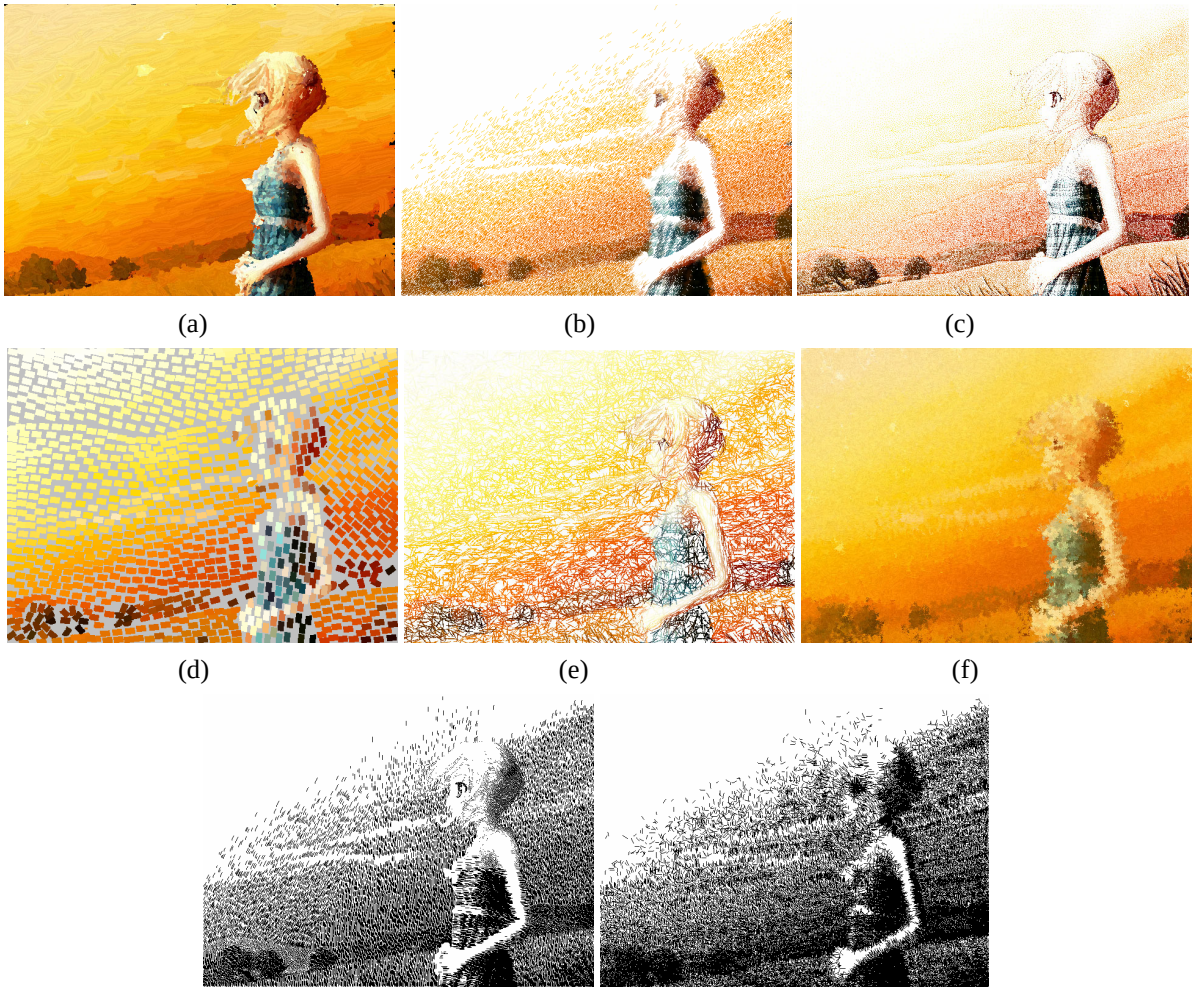
[Str02] Strothotte, T. and S. Schlechtweg, Non-photorealistic Computer Graphics: Modeling, Rendering, and Animation, Morgan Kaufmann, San Fransisco, CA, USA, 2002

[Wil97] Willats, J., Art and Representation, Princeton University Press, Princeton, NJ, USA, 1997.

[Way01] Way, D.L. and Shih Z.C., "The Synthesis of Rock Textures in Chinese Landscape Painting", In Computer Graphics Forum, Volumn 20, number 3, Eurographics Association, Switzerland, 2001, pp. 123-131.

Figure 5: Synthesized painting of one NCTU scene using Impressionism Monet style set

Figure 6: Synthesized painting using impressionism quick draw style.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Figure 7: Synthesized paintings by combining modules from painterly rendering style and pen-and-ink style. (a) Painterly style. (b) Half-toning with line using color module of painterly style. (c) Stippling using color module of painterly style. (d) Mosaics using color module of painterly style. (e) Painterly style using cross-section module of half-toning with line. (f) Painterly style using cross-section module of stippling. (g) and (h) Half-toning with line using path module of painterly style.