Real-time Rendering of 3D Ink-wash Painting Based on Geometry Buffers

YU GUA* Non-member,  NAOKI ONO†‡ Member
KIICHI URAHAMA* Non-member

(Received December 26, 2016, revised March 2, 2017)

Abstract: We propose a technique using geometry buffers for real-time rendering of 3D ink-wash paintings. According to the characteristics of hand-drawn ink-wash paintings, contour lines and coloring area in the ink paintings are stylized by rendering 3D models to 2D texture images. Contour lines are drawn according to curvature of object surfaces. Coloring areas are rendered by alpha blending with stroke textures. Stroke textures are blended at sparsely sampled points for saving computation. Finally, integration of contour lines and colored area yields an ink-wash painting image. Entire rendering computation is implemented with a GPU for enabling its real-time processing on commercially available computers.

Keywords: Non-photorealistic rendering, Ink-wash paintings, Geometry buffer, Real-time animation.

1. Introduction

Ink-wash painting is a non-photorealistic rendering (NPR) technique currently focused attention. This technique is aimed at generating computationally images resembling hand-drawn ink-wash paintings such as in Fig.1. This NPR method is expected by media artists to be used in virtual reality systems. For its use in the virtual reality, generation of 3D CG is required instead of the conventional method transforming 2D images to an ink-painting style. Additionally, user interaction in the virtual reality demands high efficient real-time rendering of scenes.

Lee et al.[1], Yuan et al.[2] and Park et al.[3] presented GPU-based real-time 3D NPR of ink-paintings. Their drawing of surfaces, however, seems only limited to primitive 3D models. Li et al.[4] simulated the diffusion process of Indian inks, however, their computation is not real-time. Way et al.[5] stylized procedural stroke textures, however, their method is specialized to trees. The footprint particles by Xu et al.[6] are applicable to broad scenes, while rendering of ink surfaces is insufficient and diffusion effect is weak.

We propose, in this paper, a technique for real-time rendering of ink-paintings of 3D scenes on the basis of geometry buffers. In our method, all of geometry information are firstly rendered into 2D texture images. Next, contour-line strokes drawn with inks and naturally diffused object surfaces are generated from the geometry buffers. Finally, they are fused into the frames in animation movies. This approach has the advantage over the conventional NPR methods as it can deal with various type of objects and scenes not limited to specific objects.

2. Characteristics of Ink-paintings

Ink paintings are pictures representing shades of objects by black ink drawing on a white paper. Various traditional drawing techniques have been developed in Asian regions.

2.1 Qualitative Characteristics Fine reality is not pursued in ink paintings, instead objects are drawn with abstracted lines and surfaces. Some rules are followed as contour lines are attached on the edge of hard objects such as stones while soft materials are painted with blurred surfaces. The thickness of lines is also adjusted according to fine or rough materials.

2.2 Drawing Techniques in Ink Painting Techniques used in the ink painting are generally categorized into three groups: contouring, coloring and texturing. Each group contains two techniques. The contouring includes drawing outlines and sketching contours, the coloring is composed of stippling and dyeing painting, the texturing contains techniques of drawing the shades and textures of rocks and mountains. Among these techniques, the contour-
ing and the coloring are popularly used, that is, ink paintings are mainly composed of contour lines and colored surface areas.

3. Rendering Technique of Ink-paintings

We propose a rendering technique of ink-paintings focusing on the contouring and the coloring. The flow of the rendering procedure is illustrated in Fig. 2.

As is shown in Fig. 2, our method renders an inputted 3D model into a geometry buffer composed of four texture data: depth map, normal map, diffuse map and grayscale map. From these maps, the edge map is constructed and the contouring lines are rendered by using it. In parallel, the coloring area is calculated and the contour lines and the coloring area are integrated to generate ink-painting frames.

Inputted 3D models are rendered into the above four textures and whole geometry information is written in a 2D image, and all of computation are executed on a pixel level.

In other words, from the geometry buffers, the color, size and orientation of stroke textures are computed and pasted at every pixel in the screen space to generate an output frame of our method.

3.1 Basic Data Structure

We represent the geometry information and surface texture by 2D images for their calculation on the pixel level. We call that set of 2D images the geometry buffer which is composed of four images: depth map, normal map, diffuse map and grayscale map. The depth map contains the geometry information, the normal map includes the information of surface normal vectors. Because only black ink is used in ink-painting, the surface textures, which store material information for ink-painting rendering, are rendered into diffuse map which is used in the same way as general geometry buffer. The property of the materials is preserved in the diffuse map in which the thickness of contour lines is saved in its R-channel, G and B channel denote the darkness of coloring area and the index of materials is saved in the alpha-channel. This grayscale map contains the following data:

\[ G = (N \cdot V_{pix})^3, \]

where \( N \) is the normal direction at the pixel and \( V_{pix} \) a unit vector from the viewpoint to the pixel. We set \( \gamma = 0.35 \). The grayscale map is used for rendering coloring areas.

The information stored in grayscale map is computed by depth and normal. In addition, because of the multiple using of these information, we compute grayscale map in advance for better performance.

3.2 Drawing of Contour Lines

Contour lines are silhouettes of objects. In the NPR, contour lines are drawn not only at object boundaries but also at the interior in objects at high curvature surface. We, therefore, extract contour lines using the curvature of surfaces. Firstly, the curvature is calculated from the data of normal vectors in the normal map [6]. Variations in the angle of normal vectors along the four nearest neighbors \( k_{\text{left}}, k_{\text{right}}, k_{\text{up}}, k_{\text{down}} \) are computed. Next, we apply the Laplacian filter to the depth map to get \( L_{ij} \). We decide the pixel where \( L_{ij} \) is lower than -0.4 as the edge pixel. We compute four weights \( w_{up}, w_{right}, w_{down}, w_{left} \) as

\[
\begin{align*}
w_{up} & = \begin{cases} 0 & L_{i,j-1} > \delta \\ 1 & L_{i,j-1} \leq \delta \end{cases} & w_{right} & = \begin{cases} 0 & L_{i+1,j} > \delta \\ 1 & L_{i+1,j} \leq \delta \end{cases} \\
w_{down} & = \begin{cases} 0 & L_{i,j+1} > \delta \\ 1 & L_{i,j+1} \leq \delta \end{cases} & w_{right} & = \begin{cases} 0 & L_{i-1,j} > \delta \\ 1 & L_{i-1,j} \leq \delta \end{cases}
\end{align*}
\]

for edge pixels. For pixels where \( L_{ij} \) is greater than 0.4, we set these four weights to 0. For remaining pixels, we set these four weights to 1. We set \( \delta = 0.6 \). From these weights multiplied to the above mentioned \( k_{\text{left}}, k_{\text{right}}, k_{\text{up}}, k_{\text{down}} \), the curvature is computed by

\[
k = \frac{k_{\text{max}} + k_{\text{min}}}{2},
\]

where

\[
k_{\text{max}} = \max[w_{up}k_{\text{left}}, w_{right}k_{\text{right}}, w_{down}k_{\text{down}}, w_{left}k_{\text{left}}] \quad k_{\text{min}} = \min[w_{up}k_{\text{left}}, w_{right}k_{\text{right}}, w_{down}k_{\text{down}}, w_{left}k_{\text{left}}].
\]

Finally, this curvature is normalized to \([0, 1]\) by

\[
E = \left(1 - \frac{\varepsilon k}{\pi}\right)^3,
\]

where we set \( \varepsilon = 0.8 \). For edge pixels, we multiply 0.5 to this \( E \) for emphasizing those pixels and thus calculated value of \( E \) is rendered into the edge map.

The size \( S_{\text{contour}} \) of the ink strokes on contours, which is the stroke texture’s width when rendered on screen in pixels, is computed by

\[
S_{\text{contour}} = \max[\eta p_{\text{contour}} D_{\text{red}} S_{\text{contour-min}}, D_{\text{red}} S_{\text{contour-min}}],
\]

where \( \eta \) is a parameter controlling the size of stroke textures, \( S_{\text{contour-min}} \) is a parameter to avoid the size of strokes becoming too small. We use \( R \) channel \( D_{\text{red}} \) in diffuse map to allow artists to control strokes’ size manually.
The parameters $\rho_{\text{contour}}$ and $S_{\text{color}}$ are given by

$$
\rho_{\text{contour}} = 1 - E,
$$

$$(7)

$$
S_{\text{color}} = \begin{cases} Z_{\text{near}}/\text{Depth} & \text{Depth} > \alpha \\ Z_{\text{near}}/\alpha^2 \text{Depth}^2 + \beta & \text{Depth} \leq \alpha \end{cases},
$$

$$(8)

where $\rho_{\text{contour}}$ is the thickness of ink strokes, $S_{\text{color}}$ controls the size of ink strokes, $\text{Depth}$ is the value in the depth map, $Z_{\text{near}}$ is the distance from the view frustum to the projection plane. The parameters $\alpha$ and $\beta$ are coefficients to ensure the size of strokes changing smoothly. We set $\eta = 3250, \alpha = 17, \beta = 0.009, S_{\text{color-min}} = 9$. The orientation of ink strokes is computed from the direction of normal vectors which are projected to the 2D screen plane from 3D normal vectors in the normal map. The ink strokes are rendered using the three mask images in Fig.3. These three masks are quadratically interpolated between masks’ pixel colors according to the thickness $\rho_{\text{contour}}$ of inks. The darkness of ink strokes $C_{\text{contour}}$ is set to 0.3 and the transparency is computed by $A_{\text{contour}} = \rho_{\text{contour}} \phi$ where $\phi$ is a parameter controlling colors. We set $\phi$ to 0.5. Resulted stroke textures are rendered by grayscale value $C_{\text{contour}}$ with the alpha blending with $A_{\text{contour}}$.

3.3 Drawing of Coloring Areas

Coloring areas are regions where the surface of 3D models is drawn with inks. The size $S_{\text{color}}$ of strokes in the coloring area, which is width of texture when rendered on the screen in pixels, is calculated by

$$
S_{\text{color}} = \min \left\{ \max \left( \frac{\mu Z_{\text{near}}}{\text{Depth}}, AS_{\text{color-min}} \right), S_{\text{color-max}} \right\} ,
$$

$$(9)

where

$$
A = \min \left\{ \max \left( \frac{\text{Depth}}{d}, 1 \right), 2 \right\} ,
$$

$$(10)

and $S_{\text{color-min}}$ are $S_{\text{color-max}}$ parameters for restricting the minimum and the maximum of the stroke size. We set $\mu = 2150, d = 150, S_{\text{color-min}} = 10, S_{\text{color-max}} = 33$. The orientation of stroke textures is set random and its ink thickness $\rho_{\text{color}}$ is calculated from the $G$ channel $D_{\text{green}}$ in the diffuse map using the mask shown in Fig.4. The darkness of ink strokes $C_{\text{color}}$ is set to 0, and the transparency $A_{\text{color}}$ is calculated as

$$
A_{\text{color}} = (1 - \phi)\rho_{\text{color}} + \phi,
$$

$$(11)

where $\rho_{\text{color}} = 1 - D_{\text{green}}$ and $\phi$ is a parameter controlling colors. We set $\phi = 0.1$. Resulted stroke textures are rendered by grayscale value $C_{\text{color}}$ with the alpha blending with $A_{\text{color}}$.

Rendering of all of stroke textures is computational intensive even if we use GPU. Hence, stroke textures are rendered by skipping some pixels as is shown in Fig.5(a). However, if stroke textures are rendered with fixed spacing as in Fig.5(a), aliasing occurs around contours as is seen in Fig.5(b). In order to prevent such jaggies, strokes are pasted with variable spacing as in Fig.5(c). This interval is adjusted by the value in the grayscale map. If the grayscale map is small, then such pixels are near contours and their spacing should be small. This adaptive spacing resolves the aliasing as is shown in Fig.5(d).

3.4 Integration

Integration of contour lines and colored area yields an ink-wash painting image.

The first step in the integration is blending of Gaussian-blurred black regions with coloring area. This is secondarily alpha-blended with a paper texture. This is finally alpha-blended with contour lines [8].

In order to preserve temporal coherency in the animation of rendered frames, each frame is smoothed using a temporal Gaussian filter. Thus temporally smoothed movie is an output of our method.

4. Results

The above method was implemented using C++ and HLSL running on the platform of Intel Core i7 and GeForce GTX 970M. An example of results is shown in Fig.6 where (a) is a 3D model of deer, (b) is its surface texture and (c) is a rendered result. We only control the shade in a frame with our method. However, in our experiments, we adjusted the white balance in order to make a classic presence.
Other examples are illustrated in Fig.7. The horse, rabbit, chicken, bird and goat in Fig.7(a)–(e) use grayscale maps. These are rendering of only one object, while multiple objects are rendered in Fig.7(f)–(h). In Fig.8, a rendering result of 3D scene composed of several objects is shown. Finally, six frames in an animation are shown in Fig.9.

Comparisons of our method and other methods are given in Fig.10. Our method expresses a diversity of material such as the difference between beak and other bird’s feathers shown in Fig.10(a) and (b). Fig.10(c) and (d) show that our method has better balance between dry ink and diffusion effect.

Our method renders 3D scenes on a screen space, hence its performance varies with the resolution of the screen as is shown in table 1. The value of FPS larger than 25 is sufficient for real-time processing. All cases in table 1 satisfy this condition.

Table 1: Rendering speed at various resolutions.

<table>
<thead>
<tr>
<th>resolution</th>
<th>vertices</th>
<th>Average FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>640×480</td>
<td>34835</td>
<td>213</td>
</tr>
<tr>
<td>640×480</td>
<td>69145</td>
<td>155</td>
</tr>
<tr>
<td>1024×768</td>
<td>34835</td>
<td>100</td>
</tr>
<tr>
<td>1024×768</td>
<td>69145</td>
<td>79</td>
</tr>
<tr>
<td>1920×1080</td>
<td>34835</td>
<td>49</td>
</tr>
<tr>
<td>1920×1080</td>
<td>69145</td>
<td>44</td>
</tr>
</tbody>
</table>

Figure 6: Example (deer).

(a) model (b) surface texture (c) rendered result

Figure 7: Other examples

(a) horse (b) rabbit (c) chicken (d) bird (e) goat (f) rock & grass (g) houses (h) mountains

Figure 8: Rendering of 3D scene.
objects can also have better performance. Our resultant animations well preserve temporal coherency. Experimental results show the proposed method has the advantage over the conventional NPR methods. Improvement in natural expression of ink-painting strokes and computational efficiency is under study.

This work was supported by JSPS KAKENHI Grant Number JP16K00241.

References


Yu Gua (Non-member) was born in 1991 at China. He is currently a student in the Master’s Course in Kyushu University. His interest includes real-time non-photorealistic rendering and animation of images in computer graphics.

Naoki Ono (Member) was born at Japan, in 1961. He received a Ph.D. degree in engineers from Kyushu University in 1997, and is presently an associate professor at Kyushu University. He has worked on image processing and pattern recognition. He is member of IIAE and IEICE.
Kiichi Urahama (Non-member) was born in 1953 at Japan. He is currently a professor in the department of design, Kyushu University. His interest includes image processing, computer vision and computer graphics.