Hindawi
Scientific Programming
Volume 2021, Article ID 1190912, 8 pages
https://doi.org/10.1155/2021/1190912

Research Article

Research on Visual Image Texture Rendering for Artistic Aided Design

Yahui Xiao

Changsha Normal University, Changsha 410000, China

Correspondence should be addressed to Yahui Xiao; sunny@csnu.edu.cn

Received 17 June 2021; Revised 21 July 2021; Accepted 2 August 2021; Published 9 August 2021

Academic Editor: Muhammad Usman

The rendering effect of known visual image texture is poor and the output image is not always clear. To solve this problem, this paper proposes a visual image rendering based on scene visual understanding algorithm. In this approach, the color segmentation of known visual scene is carried out according to a predefined threshold, and the segmented image is processed by morphology. For this purpose, the extraction rules are formulated to screen the candidate regions. The color image is fused and filtered in the neighborhood, the pixels of the image are extracted, and the 2D texture recognition is realized by multilevel fusion and visual feature reconstruction. Using compact sampling to extract more target features, feature points are matched, the coordinate system of known image information are integrated into a unified coordinate system, and design images are generated to complete art-aided design. Simulation results show that the proposed method is more accurate than the original method for extracting the information of known images, which helps to solve the problem of clearly visible output images and improves the overall design effect.

1. Introduction

Software design and hand-drawn design are usually used in art design. With the rapid development of multimedia technology, art design is more and more inclined to be combined with computer technology, resulting in various auxiliary design tools and software design functions [1]. Various software design tools can improve the quality of work by reducing the cost of manual design. At the same time, designing samples using 2D and 3D software design models enrich the creativity of visual effect [2] and can better represent the designer's design concept and innovative ideas, change the original form of artistic design, improve the working mode of artistic design, and bring earth-shaking changes to traditional artistic design.

In computer science, the scene visual understanding is one of the most widely used technologies in the field of art design. Visual comprehension of scene allows the use of computer to replace human eyes and brain to perceive, recognize, and understand 3D scenes and objects in the real world [3, 4]. It is used to analyze the complex distribution of objects in the scene's image by combining with natural language processing for accurately describing the information obtained in a reasonable manner. The main objective of visual comprehension is to allow the designers to extract the scene information. Applying the visual comprehension algorithm to the visual scene of artistic aided design can help the designer to solve the problem when the output image is not clear because of the imprecise information.

With the development of image processing technology, 2D texture recognition of color image is carried out by using image processing technique from computer vision. Moreover, 2D texture feature extraction and analysis method of color images is combined to analyze the texture feature of color images, improve the image quality and detection ability of color images, study the 2D texture recognition method of color images, and improve the accurate analysis and 3D feature resolution ability of color multitexture image. In [5], the authors used a combination of macro- and local aspects to obtain multiscale data information for building an image information model. The authors in [6] put forward a method of image segmentation based on the induction and application of multifeature information in remote sensing images. This method combines the features of spectrum, texture, and shape, respectively. In [7], the authors put forward a method of remote sensing image segmentation by combining spectral and texture features, which can improve the segmentation efficiency and accuracy of different objects.

Edge sharpening feature decomposition, scale decomposition, and multimode feature reconstruction methods are used to realize 2D texture recognition of color images [8]. However, the traditional methods for 2D texture recognition face numerous challenges such as low precision and bad self-adaptive ability. Hence, in this study, 2D texture rendering based on computer vision is proposed to detect the saliency areas of the 2D texture image.

The rest of this paper is organized as follows. In Section 2, graphics rendering is discussed which is the building block of 2D texture rendering. In Section 3, color multitexture image acquisition and regional fusion filtering is discussed. The experimental results and analysis are provided in Section 4. Finally, this paper is concluded and future research directions are provided in Section 5.

2. Graphics Rendering

Rendering pipeline is a conceptual model in computer graphics that describes the steps a graphics system needs to perform for rendering a 3D scene onto a 2D screen [9]. For this purpose, we first discuss graphical rendering process in Section 2.1 followed by vertex processing and 3D observation in Section 2.2.

2.1. Graphical Rendering Process. Commonly referred to as a rendering pipeline, it is a series of data processing for application's data into the final rendering of an image [10]. The rendering process is shown in Figure 1. First, the vertex and attribute required for the geometry is set on the client side of the application, and then, the data are entered into a series of shader stages for processing. The output of one element is used as an input for the next stage/element, resulting in an image that can be rendered to a 2D screen. Next, the rendering pipeline can be divided into several main stages, namely, vertex processing, rasterization, slice processing, and output integration operation.

During the vertex processing phase, vertices and primitives, such as conversion operations, stored in the buffer are processed. In the rasterization phase, the updated pixels are passed to the rasterized unit upon clipping [11, 12] by converting each pixel into a set of slices. Here, the slice is defined as a set of data, i.e., pixels that can not only be placed in the frame cache but also can be culled out and the pixels in the color buffer defined as a memory space that stores the pixels displayed on the screen. During chip processing, the chip testing is mainly carried out, and then, the color value of the chip is determined by various operations of the chip shader. During the output merge phase, the pixels in the slice and color buffers are compared or merged, and the color values of the pixels are updated [13].

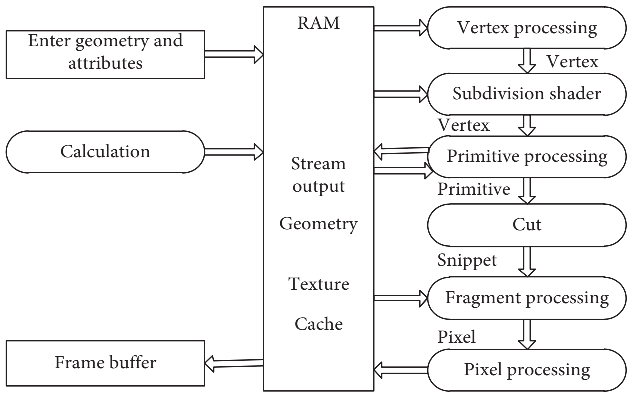

Figure 1: Rendering a graph.

2.2. Verx Processing and 3D Observation. Vertex processing and 3D observation perform various 3D geometric transformation operations on each input vertex stored in the buffer. The vertex processing stage is programmable. Based on the vertex processing transformation operation, 3D objects can be transformed from object space to clipping space. The transformation pipeline flow is shown in Figure 2.

Each object has a local coordinate system, or it can be assumed that each object is defined in its own object space. Also, multiple objects can be integrated into a single world space provided that the coordinate transformation takes the form

$$\begin{aligned} x' &= a_{xx}x + a_{xy}y + a_{xz}z + b_x, \\ y' &= a_{yx}x + a_{yy}y + a_{yz}z + b_y, \\ z' &= a_{zx}x + a_{zy}y + a_{zz}z + b_z. \end{aligned} \tag{1}$$

The coordinates x', y', and z' are derived from the linear transformations of the original coordinates x, y, and z, which are called affine transformation. Translation, rotation, scaling, reflection, and tangent are special cases of affine transformations. Any affine transformation can always be expressed as a combination of these five transformations [14].

In the 3D homogeneous coordinate representation, the 3D translation of the coordinate position can be expressed in a matrix form using

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 0 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{2}$$

In the 3D scene environment, the model object can be transformed by translating its vertex coordinates.

The three-dimensional rotation operation requires defining the corresponding rotation axis. First, the three-dimensional z-axis rotation needs to be obtained using equation (3). Next, the secondary coordinate is obtained using equation (4):

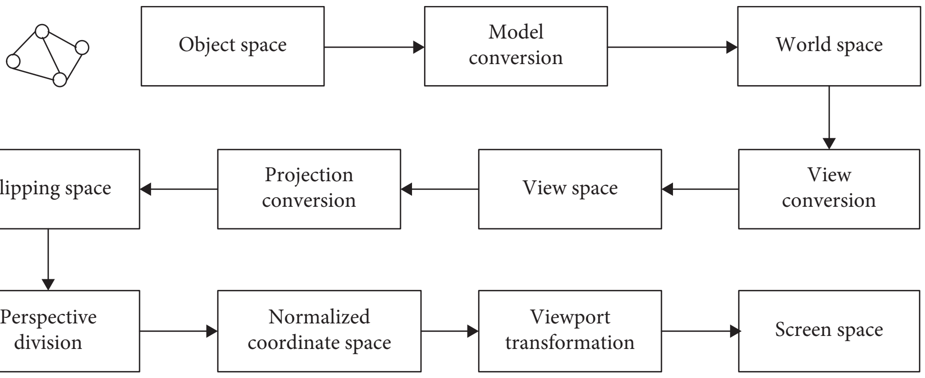

FIGURE 2: Transformation pipeline flow.

$$\begin{aligned} x' &= x \cos\theta - y \sin\theta, \\ y' &= x \sin\theta + y \cos\theta, \\ z' &= z, \end{aligned} \tag{3}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{4}$$

where θ is the angle of rotation.

The equation for rotating around the other two axes can be replaced by the coordinate parameters x, y, and z in equation (3):

$$x \longrightarrow y \longrightarrow z \longrightarrow x. \tag{5}$$

Using equation (5), the transformation equation for rotation around the x and y axes can be obtained as follows:

$$\begin{aligned} x' &= x, \\ y' &= y\cos\theta - z\sin\theta, \\ z' &= z\sin\theta + z\cos\theta, \\ x' &= z\sin\theta + x\cos\theta, \\ y' &= y, \\ z' &= z\sin\theta + x\cos\theta. \end{aligned} \tag{6}$$

Three-dimensional scaling can be represented by the following matrix:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} t_x & 0 & 0 & 0 \\ 0 & t_y & 0 & 0 \\ 0 & 0 & t_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{7}$$

Among them, the scaling parameters t_x, t_y, and t_z are arbitrary positive values that are prespecified. The display of the scaling transformation relative to the origin is expressed as

$$\begin{aligned} x' &= x \cdot t_x, \\ y' &= y \cdot t_x, \\ z' &= z \cdot t_z. \end{aligned} \tag{8}$$

When the object is modeled, it has its own local coordinate system and belongs to its own object space. In the rendering pipeline, the first task is to integrate the model objects of the independent object space into the world space, i.e., the world coordinate system. The world space can be regarded as the coordinate system of the entire virtual scene. The integration process is to apply model conversion, i.e., a model conversion matrix (Mmod) is obtained by multiplying the matrices of the above series of affine transformations, and the position coordinates of the model object in the object space are multiplied by Mmod to obtain the model object in the position coordinates of world space.

3. Color Multitexture Image Acquisition and Regional Fusion Filtering

In this section, first, we discuss the color multitexture image acquisition for rendering followed by plane projection of the area to be mapped. Finally, we discuss the procedure to calculate the coordinate of texture.

3.1. Color Multitexture Image Acquisition. To realize the two-dimensional texture recognition of color images based on computer vision, first, we build a color multitexture image acquisition model [15], use the local window feature detection method to extract the contour feature points Q and P of the color multitexture image, and combine the correlation. According to the fusion rule [16], the maximum value pixel_A of the two-dimensional edge pixel feature components of the color multitexture image is

$$\text{pixel\_}A = \max\left(\sum_{i=1}^{8} (Q - P)\right). \tag{9}$$

Using the local information entropy fusion model for color multitexture image collection, extract the contour points of the color multitexture image, perform local information entropy fusion processing on the color multitexture image, extract the active contour model of the color multitexture image, and combine the color multitexture image The regional features of the active contour are matched with edge pixel features, and the local information entropy rect(t) is extracted, and the output of the pixel feature quantity collected by the color multitexture image is reflected:

$$u(t) = \frac{1}{\sqrt{T}}\operatorname{rect}\left(\frac{t}{T}\right)\exp\left\{-j\left[2\pi K\,\ln\left(1-\frac{t}{t_0}\right)\right]\right\}. \tag{10}$$

In equation (10), the pixel feature quantity $|t| \le 1$ and K are the number of pixels and j represents the singular point of the boundary of the color multitexture image. Assume that the position information associated distribution length of the color multitexture image is $L = x_{\max}-x_{\min}$ and the width is $W = y_{\max}-y_{\min}$ and $H = z_{\max}-z_{\min}$. Set the number of super-pixels to obtain the one-dimensional histogram distribution of the color multitexture image, determine the number of super-pixels K, and combine the scattering model to obtain the 2D texture feature of the color multitexture image. Splines' biorthogonal wavelet transform method is used to obtain the texture high frequency component, according to the USV decomposition result, which realizes the feature decomposition and 2D texture recognition of the color multitexture image.

3.2. Plane Projection of the Area to Be Mapped. Assume that the three noncollinear points in the area to be mapped are point P, point M, and point N, where P controls the source point of texture mapping that corresponds to the coordinate origin of the two-dimensional texture image, and the vectors $\overline{PM}$ and $\overline{PN}$ control the direction of texture image μ and axis v, respectively.

From the plane equation$A(x - x_o) + B(y - y_o) + C(z - z_o) = 0$, it is known that, to determine a plane, one needs to know the coordinate ruler $P(x_o, y_o, z_o)$ of any point on the plane and the plane normal vector $M\{A, B, C\}$. Given that the three vertices of the plane are P, M, and N, set the vectors $M_1 = M - P$ and $M_2 = N - P$, and then, calculate the cross product of the vectors to obtain the plane normal vector $M = M_1 \times M_2$. In this way, a plane composed of points P, M, and N is obtained, which serves as the reference plane T of the projection.

Knowing that the coordinate of any point in the area to be mapped is $Q_i(x_i, y_i, z_i)$, the coordinate $Q_i'(x_i', y_i', z_i')$ of its projection point on the reference plane T needs to be obtained. According to the simultaneous equations,

$$\begin{cases} A(x - x_o) + B(y - y_o) + C(z - z_o) = 0, \\ x_i' = x_i + kA, \\ y_i' = y_i + kB, \\ z_i' = z_i + kC. \end{cases} \tag{11}$$

The value of k can be obtained using

$$k = \frac{A(x_o - x_i) + B(y_o - y_i) + C(z_o - z_i)}{A^2 + B^2 + C^2}. \tag{12}$$

Thus, the projected coordinate Q can be obtained in this fashion, i.e., $Q_i'(x_i', y_i', z_i')$.

3.3. Texture Coordinate Calculation. After projecting the vertices in the area to be mapped onto the reference plane T, a coplanar three-dimensional point set is obtained. According to this coplanar point set, a two-dimensional coordinate system S-T can be established, as shown in Figure 3.

The origin of the S-T coordinate system is the first point P of the three vertices in the reference plane. Take the side $PM = M-P$ as the horizontal axis, the length of the horizontal axis is $|\overline{PM}|$, and the length of the longitudinal axis is $|\overline{PN'}|$:

$$\left|\overline{PN'}\right| = |\overline{PN}| \cdot \cos\alpha, \tag{13}$$

where α is the angle between the vector PN and the ordinate. Given that the projection coordinate of the point $Q_i(x_i, y_i, z_i)$ on the plane is $Q_i'(x_i', y_i', z_i')$, the vector $|\overline{PQ'}|$ can be calculated. Assuming that the angle between the vector $|\overline{PQ'}|$ and the transverse coordinate positive vector S of the texture coordinate X_u is θ, the coordinates of $Q_i'(x_i', y_i', z_i')$ in the two-dimensional coordinate system can be obtained as

$$\begin{cases} S_{Q_i} = L\cos\theta, \\ T_{Q_i} = L\sin\theta, \end{cases} \tag{14}$$

where

$$L = \sqrt{(x_i' - x_o)^2 + (y_i' - y_o)^2 + (z_i' - z_o)^2},$$

$$\begin{cases} \cos\theta = \dfrac{\overline{PQ'}\cdot S}{\left|\overline{PQ'}\right|\cdot S}, \\ \sin\theta = \left|\sqrt{1-\cos\theta^2}\right|. \end{cases} \tag{15}$$

Since, the angle is in the range $[0, \pi]$, the sinusoidal value will be negative, so the sinusoidal value must be absolute to get the correct coordinates.

Using the above calculation, every projection point in the plane can get the coordinate points in the ST coordinate system. The texture coordinate system u-v is located in the range of [0, 1]; hence, it is necessary to normalize the coordinate points. If $S_{\max}$ and $T_{\max}$ are the maximum values of S_{Q_i} and T_{Q_i}, respectively, the final texture coordinates of the fixed points of the model obtained by proportional transformation are shown using

$$\begin{cases} U_{Q_i} = \dfrac{S_{Q_i}}{S_{\max}}, \\ V_{Q_i} = \dfrac{T_{Q_i}}{T_{\max}}. \end{cases} \tag{16}$$

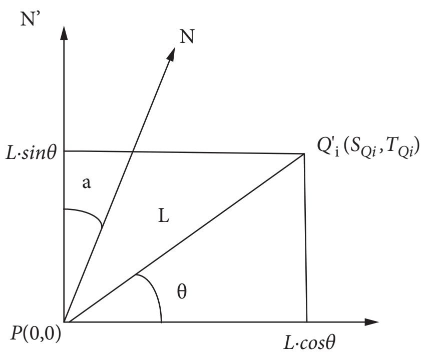

FIGURE 3: Schematic diagram to construct the S-T coordinate system.

The texture coordinates' system schematic is shown in Figure 4.

As shown in Figure 4, the user needs to change the mapping position, i.e., change the point P, which controls the origin of the texture mapping. It corresponds to the coordinate origin in the texture space, and the user needs to change the mapping direction, i.e., change the point M and point N, which control the direction of the texture mapping, respectively. In this figure, vectors PM and PN correspond to the u- and v-axis in the texture space, and the user needs to change the mapping size, namely, change the point M and point N, which control the size of the texture mapping.

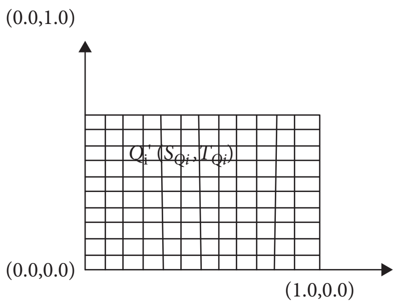

FIGURE 4: Texture coordinates' system.

3.4. Comparison of Changes to the Parameters. Three parameters, P, M, and N, are used to control the effect of local texture mapping in determining the coordinates of vertices. Here, P is the origin of texture mapping and corresponds to the origin of texture image in the 2D coordinate system. It controls the position of local texture mapping. The vector $\overrightarrow{PM}$ (S-axis) of point M and point P controls the direction of the u-axis of the texture image, which corresponds to the x-axis in a two-dimensional coordinate system. The change in the direction of $\overrightarrow{PM}$ also affects the axial direction of S-axis, according to its determination that results in the rotation of the texture map. By changing the size of vector $\overrightarrow{PM}$, the stretching and shrinking of texture can be achieved. Similarly, the vector $\overrightarrow{PN}$ (T-axis) of point N and point P controls the direction of the texture image axis, and the stretching and shrinking effects in the longitudinal direction can be achieved by changing its size. If the texture image is no longer required to be rotated, one can fix the S-axis in the positive direction so that the mapping direction does not change.

4. Experimental Analysis

This paper chooses Windows 10 as the experimental device, the model is CubiB171 N 8GL009BCN BN5000, the CPU memory is 6 GB, and the experimental platform is MATLAB9.0, and this paper takes Figure 5(a) as the experimental object, takes the visual communication effect as the foundation, and uses the designed method to render and analyze the system performance.

In order to verify the performance of this system, the images rendered in this system are compared with the images rendered in Linux graphics rendering system [6] and fluid cloud simulation rendering system [7]. The comparison results are shown in Figure 5.

As can be seen from Figure 5 and 5(b) is a picture rendered through the methodology proposed by this paper. Figure 5(c) is an image rendered by the Linux graphics rendering system [6] which has color differences, many color stripes, and serious distortion. Figure 5(d) is an image rendered through the fluid cloud simulation rendering system, on which many noise spots appear, resulting in blurred image rendering, which cannot be accurately displayed, resulting in a part of the chromatic aberration and a lower effect than that of the Linux graphics rendering system. Figure 5(b) is the poster image after the system rendering. It can be seen from the image that, after the system rendering, the poster image is clear, there is no noise interference, the color difference is rectified by the filter, and the color is more real. Compared with the other two kinds of rendering systems, this system has better rendering effect on the poster image and has excellent rendering effect.

When the system is rendering, it will be affected by factors, such as operation wait time. Compared with the other two systems, the result is shown in Table 1.

As can be seen from Table 1, the waiting time of all rendering operations in this system is not more than 0.5 s, followed by fluid cloud simulation rendering system, the maximum waiting time is 0.90 s, the system with the longest waiting time is the Linux graphics rendering system, and the maximum waiting time is 1.12 s. The average waiting time of

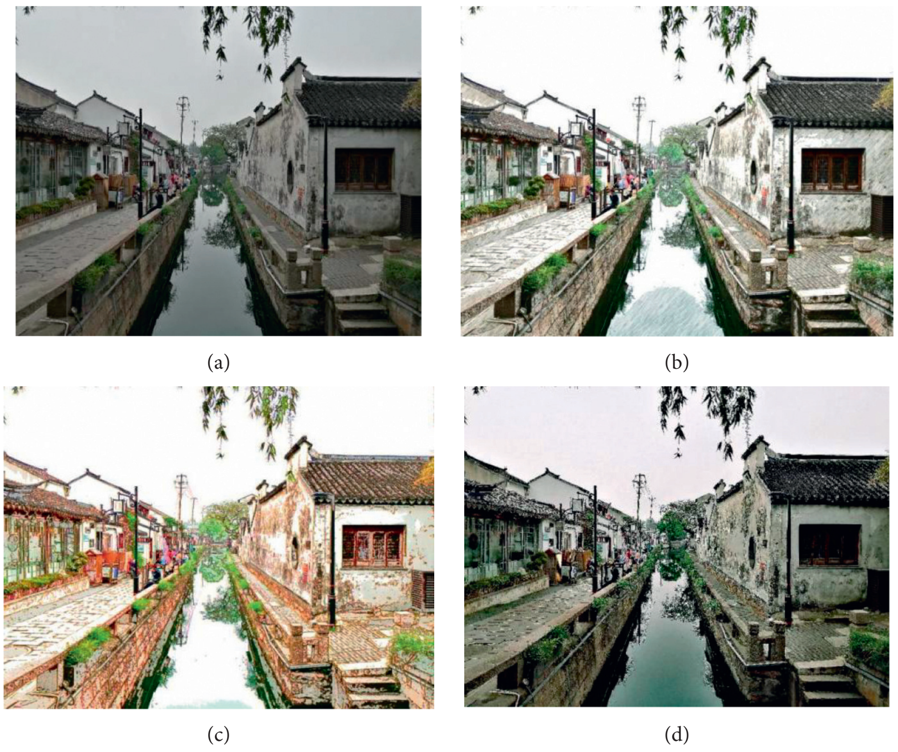

Figure 5: Render contrast effect. (a) Original image. (b) Images rendered by this paper. (c) Images rendered by [6]. (d) Images rendered by [7].

Table 1: Render operation waiting time.

Rendering operation	System waiting time in this paper	Linux graphics rendering system waiting time	Liquid cloud simulation rendering system waiting time
Image rendering exchange waiting	0.13	0.53	0.42
Hardware submission waiting	0.24	0.80	0.63
Image rendering processing waiting	0.15	0.77	0.54
Image data waiting	0.31	1.12	0.90
Image buffer waiting	0.15	0.43	0.27
Wait for completion	0.28	0.76	0.67
Average value	0.18	0.67	0.53

the system is 0.18 s, which is better than that of the Linux graphics rendering system and the fluid cloud simulation rendering system. It proves that the system has the best performance, and the poster image is better.

After image processing, the pixel change is affected by image size change. Comparing the pixel change of the system and other two systems in different image sizes, the result is shown in Figure 6.

Figure 6 shows that no matter how the size of the poster image changes, the poster image rendered by this system keeps a stable pixel, while the image pixel change of the Linux graphics rendering system and the fluid cloud simulation rendering system fluctuates greatly, and there is no obvious trend change, which shows that the image pixel change of these two systems does not vary according to the image size, and at the same time, it indicates that the rendering effect of these two systems is extremely unstable. Compared with other systems, the rendering effect of this system is more stable, and the rendered poster image is more effective.

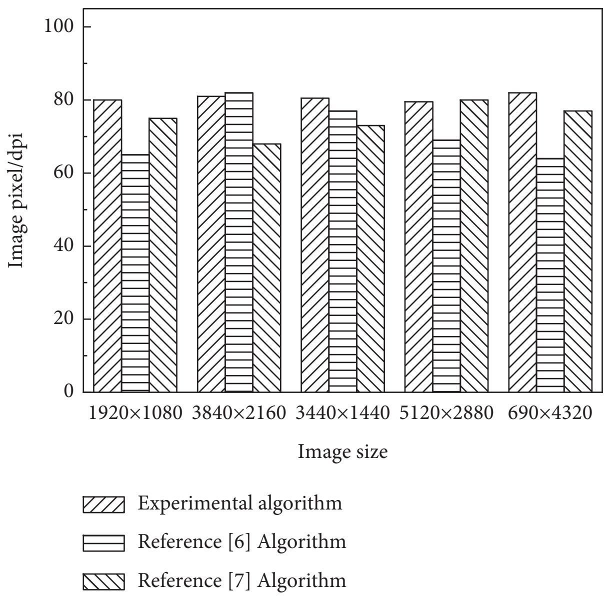

Figure 6: Image rendering effects of different sizes.

5. Conclusions

In this paper, an artistic aided design method based on scene vision comprehension algorithm is proposed. The proposed approach can effectively extract the known scene image information, detect the salient region texture features of the collected color multitexture images by super-resolution fusion method, and identify the 2D texture features according to the texture and color feature components of the color multitexture image. The proposed approach is helpful to solve the problem of unclear output image, improve the output quality of the images, and improve the visual effect of artistic aided design. To verify the efficiency of this approach, a comparison is made with the images rendered in Linux graphics and fluid cloud rendering. Simulation results show that the proposed method is more accurate than the existing methods for extracting the information of known images, which helps to solve the problem of clearly visible output images and improves the overall design effect.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares that he has no conflicts of interest.

References

[1] L. Liu, W. Xu, M. Habermann et al., "Neural human video rendering by learning dynamic textures and rendering-to-video translation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, 2020.

[2] A. Panotopoulou, X. Zhang, T. Qiu, X.-D. Yang, and E. Whiting, "Tactile line drawings for improved shape understanding in blind and visually impaired users," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 1–89, 2020.

[3] S. Chen and Z. Jiu, "A method of stereoscopic display for dynamic 3D graphics on android platform," *Journal of Web Engineering*, vol. 19, pp. 818–829, 2020.

[4] A. Kerim, C. Aslan, U. Celikcan, E. Erdem, and A. Erdem, "NOVA: Rendering virtual worlds with humans for computer vision tasks," *Computer Graphics Forum*, vol. 6, 2021.

[5] D. Beattie, W. Frier, O. Georgiou, B. Long, and D. Ablart, "Incorporating the perception of visual roughness into the design of mid-air haptic textures," in *Proceedings of the ACM Symposium on Applied Perception 2020*, pp. 1–10, Barcelona, Spain, September 2020.

[6] M. Colombo, A. Dolhasz, and C. Harvey, "A texture superpixel approach to semantic material classification for acoustic geometry tagging," in *Proceedings of the Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–7, Yokohama Japan, May 2021.

[7] K. Rematas and V. Ferrari, "Neural voxel renderer: Learning an accurate and controllable rendering tool," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5417–5427, Seattle, WA, USA, December 2020.

[8] J. Wu, "Two-dimensional texture recognition method of color image based on computer vision," *Journal of Jixi University*, vol. 20, no. 1, pp. 31–36, 2020.

[9] M. Salgado, H. Hettiarachchi, T. U. Munasinghe, K. Fernando, and N. C. Cooray, "Assist: rendering, pipeline management, and pipeline tracking software," in *Proceedings of the 2020 2nd international conference on advancements in computing (ICAC)*, Malabe, Sri Lanka, December 2020.

[10] M. Kim and N. Baek, "A 3d graphics rendering pipeline implementation based on the opencl massively parallel processing," *The Journal of Supercomputing*, vol. 3, pp. 1–17, 2021.

[11] J. Chen, Z. Tian, X. Wu, and X. Lou, "Hardware modeling of GPU geometric pipeline rasterization based on UML & SystemC," *Application of Electronic Technique*, vol. 45, no. 1, pp. 23–26, 2020.

[12] W. Guo, Z. Wu, R. Xu, Q. Zhang, and M. Fujigaki, "A fast reconstruction method for three-dimensional shape measurement using dual-frequency grating projection and phase-to-height lookup table," *Optics & Laser Technology*, vol. 112, pp. 269–277, 2019.

[13] J. W. Lee, J. H. Kim, Y. H. Lee, Y. J. Jeon, B. S. Jeong, and J. H. Choi, *U.S.PatentNo.10,466,530*, U.S.patent and trademark office, Washington, DC, USA, 2019.

[14] Z. H. A. N. G. Pingmei, J. I. N. Lizuo, and L. I. Jiu-xian, "Image stitching based on affine transformation and image block," *Information Technology & Informatization*, vol. 5, no. 1, pp. 61–65, 2020.

[15] P. Tiwari, S. N. Sharan, K. Singh, and S. Kamya, "Content based image retrieval using multi-level 3D color texture and low level color features with neural network based classification system," *International Journal of Circuits, Systems and Signal Processing*, vol. 15, pp. 265–270, 2021.

[16] K. M. Hosny, T. Magdy, and N. A. Lashin, "Improved color texture recognition using multi-channel orthogonal moments and local binary pattern," *Multimedia Tools and Applications*, vol. 80, pp. 1–16, 2021.