Computer Science and Application 计算机科学与应用, 2023, 13(12), 2387-2398
Published Online December 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.1312239

Hans汉斯

实时像素化非真实感渲染方法

侯建峰*，刘端烨，谢含纳，陈立哲，石文杰

北方工业大学信息学院，北京

收稿日期：2023年11月26日；录用日期：2023年12月21日；发布日期：2023年12月29日

摘　要

传统像素化的非真实感渲染方法，大多存在操作复杂，需要手动调节众多参数，时间复杂度极高等相关问题。针对以上问题，本文提出了一种高性能实时像素化非真实感渲染方法，该方法首先设计光照模型，随后通过边缘检测提取出目标对象的边缘，进行插值计算对图像做像素化处理，最后将光照模型效果与抖动效果处理相结合，进行抖动风格化效果实时渲染，实验表明本文提出的高性能实时像素化非真实感渲染方法与传统的像素化方法相比，降低了计算的时间复杂度，简化了调节步骤，提升了渲染效果。

关键词

非真实感渲染，像素，实时渲染

Real-Time Pixelated Non-Photorealistic Rendering Method

Jianfeng Hou*, Duanye Liu, Hanna Xie, Lizhe Chen, Wenjie Shi

College of Information, North China University of Technology, Beijing

Received: Nov. 26th, 2023; accepted: Dec. 21st, 2023; published: Dec. 29th, 2023

Abstract

Traditional pixelated non-photorealistic rendering methods often suffer from problems such as complicated operations, manual adjustment of numerous parameters, and extremely high time complexity. To address these issues, this paper proposes a high-performance real-time pixelated non-photorealistic rendering method. Firstly, a lighting model is designed, followed by edge detection to extract the edges of the target object. Interpolation is then applied to pixelate the image, with the lighting model effect combined with dither effect processing for real-time rendering of

*通讯作者。

文章引用: 侯建峰, 刘端烨, 谢含纳, 陈立哲, 石文杰. 实时像素化非真实感渲染方法[J]. 计算机科学与应用, 2023, 13(12): 2387-2398. DOI: 10.12677/csa.2023.1312239

dither stylization effects. The experimental results show that the proposed high-performance real-time pixelated non-photorealistic rendering method reduces the time complexity and simplifies the adjustment steps compared with traditional pixelation methods, while improving the rendering effect.

Keywords

Real-Time Rendering, Pixels, Non-Photorealistic Rendering

1. 引言

近年来计算机图像的非真实感渲染已经成为计算机图形学领域的研究热点和难点[1]。随着动漫化渲染类游戏的兴起，人们开始不仅仅满足于基于物理的渲染，非真实感渲染逐渐进入市场成为主流。

其中，像素风格化是非真实感渲染中具有鲜明特色的一种渲染方式，同时像素风格化类游戏给予玩家一种不同于现实的明确抽离感，将不那么清晰的图案，赋予丰富的象征感，给予玩家更大的想象空间。传统的像素类游戏制作需要美工进行大量 UI 绘制，需要人工将像素色块赋予给物体，花费大量人力资源。并且在制作 3D 像素类游戏过程中需要考虑光线模型问题，无法实现实时的光线追踪像素化。本文主要通过优化边缘检测算子进行采样，实现高性能的实时像素化非真实感的渲染方法，同时基于抖动算法对像素化的物体进行风格化处理。

2. 相关工作

在当前 CG 行业中，像素艺术风格已经成为了一种热门的艺术表现形式。“像素艺术”具有很强的表现能力。虽然相较于精致画风，像素画风略显简陋，但其传递和承载信息的能力也能表现较丰富的内容和剧情。而像素画风的最大优点，是制作成本远低于精致画风，表达效果好不逊色于精致画风。

20 世纪八十年代初期，计算机对图形数据的处理能力相对较弱，无法实现真实感的物理渲染效果。为了解决这一问题，在游戏和电影领域采用了像素化技术。通过将图像数据分割成离散的块状元素，并且使用粗略的着色和渲染技术来降低计算复杂度，以实现更快的渲染速度和更高的图形帧率。

在非真实感渲染领域 Q. Wang 提出了基于非真实感绘制的草图仿真技术仿真研究，对素描绘制方法和手绘风格的实现效果进行了研究。近年来，随着计算机硬件性能的不断提升和图形编程技术的不断改进，实时像素化渲染的效果和速度都得到了极大的提高。现代渲染管线和基于 GPU 的硬件加速技术可以在短时间内完成实时渲染，并且提供了更多的细节和艺术效果，使像素化渲染技术变得更加流行和普及。关于像素化的非真实渲染方面的研究也已经越来越多。2013 年 Takashi Kanai [2]介绍了一种基于像素化的边缘感知和轮廓线渲染方法，使用超像素分割的技术，基于 SLIC (Simple Linear Iterative Clustering)算法，将颜色分割成多个局部区域，通过比较相邻区域的平均颜色信息得到最终的渲染结果。虽然 SLIC 算法在分割效果和处理速度上都有较好的表现，但仍存在超像素的形状和大小不规则，表现力不足。随后，学者们对于像素化的着色模型边缘获取方法进行了更深入的探索。2016 年 Chi-Han Peng [3]提出将获取到的深度图分割成若干个大小相等的矩形区域，通过计算深度图中每个像素的坐标，并按照相应的坐标范围将像素分配到与其重叠的像素块中。这种块状划分的方法可以有效地减少计算量并提高渲染速度，

因为它将像素数据压缩到了离散的块中，而非逐个像素进行处理。此外，在生成近似轮廓线时，像素块之间的边界也更容易确定和控制，从而进一步提高了渲染效果和精度。同年 J. Lee [4]提出一种高效的移动矢量图形抗混叠算法，使优化的 S-MSAA (选择性多样本消除混叠)比原来的 16 × MSAA 快 29.4 倍。

2022 年 Wanru Dai [5]提出了基于三样条插值的子像素轮廓提取与测量，通过引入三次样条函数，将边缘**定位**在更准确的位置，从而提取边缘轮廓。

常规的像素化非真实感渲染常常使用低分辨率的像素进行绘制，从而导致图像失去真实感和细节，并且需要同时对像素尺寸，颜色选取，像素间隔等要对多个参数进行调节，需要一定美术功底，还需进行额外的后处理，增加渲染流程和成本。基于以上文献和像素化非真实感渲染现状启发，本文进行了高性能像素化非真实感实时渲染效果的实现方法。

3. 系统化实时光线追踪渲染方法描述

传统的像素化非真实感渲染普遍存在时间复杂度较大，渲染调参过程复杂，需要有一定美术功底和经验的人才能达到较为理想的效果，且无法精确的进行实时光线追踪渲染等问题。本文首先对模型进行光照模型的分析设计；然后选用合适的边缘检测方法实现对物体进行边缘检测实现像素化；接着，对待处理的物体进行抖动风格化处理；最后，微调参数设计艺术风格获得最终的渲染结果。

Input
(1)物体光照信息 V
(2)相机深度 depth
(3)屏幕尺寸 ScreenSize
(4)抖动处理常数 X
(5)摄像机颜色纹理 Color

Output
(1)光照模型输出 L
(2)像素化调节参数 PixelSize
(3)风格化抖动效果调节参数 DitherSize

1.a=Diffuse(V)
//设计漫反射光照模型，得到 a
2.b=Specular(V)
//设计高光光照模型，得到 b
3.L=a+b
//将漫反射光照模型与高光光照模型线性相加
4.PixelSize=Pixelate(depth,ScreenSize)
//获取相机深度以及相机屏幕尺寸，进行边缘检测
和像素化处理
5.利用随机数生成矩阵 Mr8*8
6.posInput=GetPositionInput(depth)
//获取相机顶点坐标信息
7.f=Dither(Mr8*8,posInput)

//进行抖动计算

8.max(f,Color*X)

//将得出的抖动参数 f 与摄像机颜色纹理相比较，

取两者最大值

9.将得到的 L，PixelSize 与 DitherSize 利用线

性叠加的方式，获得最终的渲染效果

3.1. 设计光照模型效果

像素化非真实感渲染中，像素化的渲染光照处理基本符合传统的真实感渲染光照效果，但仍有部分效果需要进行特殊的光照处理。为了更加真实的表现出物体表面与光线的相互作用，实现实时的非真实感渲染，本文参考 Blinn-Phong 光照模型的思路，将光照分为漫反射，高光两部分，并且在传统的光照模型基础上进行改进优化。

3.1.1. 分析光照模型效果

传统的真实感渲染技术通常采用基于物理的光照模型，例如 Phong 光照模型、Blinn-Phong 光照模型和 Cook-Torrance 光照模型等。这些光照模型能够较好地模拟真实世界中不同材质的反射特性。但对于像素化非真实感渲染，我们更加关注的是艺术效果和风格化表现。因此，在选择光照模型时，考虑与所需风格匹配的简单光照模型，例如漫反射表面的光照效果可以采用 Lambert 光照模型或卡通渲染中常用的半兰伯特光照模型；高光效果则可以采用较为简单的 Blinn-Phong 光照模型进行计算。

3.1.2. 设计漫反射光照效果

物体表面的漫反射光照模型对于计算场景中的真实感和细节至关重要，使用漫反射光照模型可以更好地模拟物体表面与周围环境之间的光照交互，增强物体的真实感，并且能够呈现出更多的细节和纹理，是场景更具有吸引力和视觉效果。本文基于 Lambert 模型的思路进行漫反射光照模型效果设计。入射光线经过物体表面后，会以相同的强度在所有方向上反射。因此，物体表面每个点的亮度只取决于入射光线与表面法线的夹角。为了更好的实现漫反射效果，实现入射光线的光强大小的可控性，本文引入变量 N 用来控制光线强度，设计了新得漫反射计算方法。考虑到光强应恒大于 0，且随着光强的增加，对漫反射光照效果逐渐趋于平缓，因此将变量 N 进行开根处理。公式(1)如下

$$Ld = \sqrt{N} * k_d * \left(\frac{I}{r^2}\right)\cos\theta \tag{1}$$

其中，Ld 为漫反射光光强，k_d 为光照吸收值即漫反射系数，I 为入射光线光强，$\cos\theta$ 表示光线与法线的夹角余弦值。N 为入射光线的散射程度，使入射光线在物体表面强度可以进行调节，与 k_d 光照吸收值相互结合。其取值控制在 0.5~1 之间可以获得较为完美的变化。

3.1.3. 设计高光效果

在光照模型中，高光通常被描述为一种亮度较高的光照分量，能够为物体表面添加光泽感和反光特性，让场景看起来更具有真实感，表现出物体的几何特性和层次感。本文基于 Blinn-Phong 高光计算公式对其进行优化重新设计。如公式(2)所示

$$Ls = M \cdot K_s \cdot I_p \cdot (\cos\alpha)^{\beta_s} \tag{2}$$

其中 Ls 为高光，M 为遮蔽系数，通过调节该系数，使得物体表面受到周围物体阻挡而产生变暗，可以产生更加真实的遮蔽效果，K_s 为镜面反射系数，I_p 为光源的强度，α 是视线方向和半程向量 H 之间的夹角，

β_s 是高光锐度参数。其中 H 半程向量的计算可以通过公式(3)表示得到：

$$H=\frac{l+v}{\|l+v\|} \tag{3}$$

其中，l 是从表面点指向光源的向量，v 是从表面点指向观察者的向量。

3.1.4. 合并光照模型

由于上述两种光照模型皆为线性结构，因此将上述得到的两种光照模型部分通过线性运算合并。可以得到完善的光照模型。

3.2. 设计像素化处理

通过研究传统的像素化方法，了解到边缘检测算法是图像处理中常用的技术。通过边缘检测可以实现简化图像边缘，在保留重要特征的同时，去除过多的细节信息，同时可以实现边缘的实时跟踪和识别，对图像进行增强和优化，在像素化的同时提升图像的可读性[6]。本文通过采用边缘检测算子进行像素化辅助处理[7]，弥补传统方法的不足之处，进行优化。

3.2.1. 对图像进行像素化

在进行边缘检测算法时，我们首先考虑了使用计算简单，速度较快的 Sobel 算法进行边缘检测。在进行边缘检测之前，我们首先对原始图像进行高斯处理，减少噪声对边缘检测的影响。接着，对需要像素化的场景进行灰度化处理，即将 R、G、B 三个通道的值取平均值作为灰度值。然后，计算图像的梯度幅值和方向。Sobel 算子对应的两个卷积核如公式(4)所示：

$$G_x=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix},\ G_y=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \tag{4}$$

随后，对两幅梯度图像进行合并，得到梯度幅值图像 G，如公式(5)所示，其中 G_x 与 G_y 分别为水平方向和竖直方向的梯度图像。

$$G=\sqrt{G_x^2+G_y^2} \tag{5}$$

3.2.2. 优化像素化算法

通过 Sobel 边缘检测初步实现了像素化，但获得的边缘会出现明显的杂边效果。针对杂边效果的出现，考虑到由于 Sobel 边缘检测在灰度变化不明显的地方会产生较强的边缘，因此将 Sobel 边缘检测算子更换为 Canny 边缘检测算法[8]。

因此，在进行 Sobel 算子运算前，对灰度图像进行高斯滤波，根据高斯函数计算卷积核[9]，对输入图像进行平滑处理，高斯核如公式(6)所示，卷积核如公式(7)所示。

$$\begin{bmatrix}0.0751136 & 0.1238414 & 0.0751136\\ 0.1238414 & 0.2041799 & 0.1238414\\ 0.0751136 & 0.1238414 & 0.0751136\end{bmatrix} \tag{6}$$

$$G(x,y)=\frac{1}{2\pi\sigma^2}\mathrm{e}^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}} \tag{7}$$

然后，再次进行与 Sobel 算子计算，将得到的梯度幅值进行非极大值抑制和双阈值处理后，完成 Canny 边缘检测[10]。随后，将 Canny 边缘检测得出的图像结果进行进一步处理，进行形状重建处理将二值图

像转换为连续的像素坐标表示，公式如(8)所示。

$$P_n(x)=\sum_{k=0}^{n}G_k l_k(x) \tag{8}$$

其中 $P_n(x)$ 表示 n 次 Lagrange 插值多项式函数，G_k 表示第 k 个数据点的函数值 $l_k(x)$ 则表示 n 个数据点的 Lagrange 基函数，利用插值计算以此来获得更好的边缘效果。使用 Canny 算法会在边缘检测前会进行高斯滤波和非极大值抑制，去除了边缘附近的噪声干扰，减少了误差，使得产生的边缘更加清晰明显，在图像中同时获得较高精度的边缘，并且利用插值计算将二值图像转化为连续的像素坐标，上述实现像素化非真实感渲染效果如图 1 所示。

Figure 1. Pixel based non realistic rendering edge detection process
图 1. 像素化非真实感渲染边缘检测流程

3.3. 设计风格化抖动效果

8 位像素画(8-bit Pixel Art)以及 16 位像素画(16-bit Pixel Art)这类像素风格的色彩能以极少的色彩配合抖动效果呈现出不同灰度色彩，通过简单的基本几何形状和线条来表示物体和场景，同时运用强烈的颜色对比度增强画面张力，并且在每一帧的绘画中，通过抖动效果实现物体微小的变化，产生动画效果。本文基于此类像素风格色彩将像素进行风格化处理。

3.3.1. 分析抖动算法

目前大多数半色调算法可以归纳为规则抖动算法和误差分散算法[11]，抖动算法需要在每个像素周围的邻近像素中引入噪声，通过改变图像灰度，实现改善图像的视觉效果和表现。其中 Dither 抖动处理既可以是误差分散算法，也可以是规则抖动算法，同时降低色带现象，提高图像的质量和细节表现，通过抖动技术使图像的颜色变得更加柔和，接近自然，增强视觉感受性[12] [13] [14]。因此可以较好的实现 8 位像素画以及 16 位像素化的风格化效果。

3.3.2. Dither 抖动处理实现

首先，将需要渲染的图像利用 RGB 加权平均值的方式来进行颜色处理转换为灰度图像；然后获得像素对应的亮度，并对其坐标求余，如公式(9)所示。

$$M=\sum_{i=1}^{n}\sum_{j=1}^{m}p_{ij}*G_{ij}\bmod x \tag{9}$$

其中 G_{ij} 表示灰度图像中 i 行 j 列的像素值，P_{ij} 是一个权重矩阵，一般情况下选取人眼对颜色的敏感度确定的权重系数，mod 表示取余运算符，x 是一个常数，用于进行取余操作。

通过传入一个随机生成的抖动矩阵来判断当前像素点是映射到的灰度信息。其中抖动矩阵的尺寸用来控制抖动的强度以及均匀性，矩阵尺寸越大，抖动效果越明显，但过大的矩阵会导致过度抖动，增加图像的细节，影响图像整体观感并且增加了计算时间。因此本文实现选择采样的 8*8 的矩阵如公式(10)所示：

$$M_{dither}=\begin{bmatrix}1&49&13&61&4&52&16&64\\33&17&45&29&36&20&48&32\\9&57&5&53&12&60&8&56\\41&25&37&21&44&28&40&24\\3&51&15&63&2&50&14&62\\35&19&47&31&34&18&46&30\\11&59&7&55&10&58&6&54\\43&27&39&23&42&26&38&22\end{bmatrix} \tag{10}$$

最后将像素对应的亮度值与随机生成的抖动矩阵进行通过如公式(11)计算后得到的对应矩阵的值进行比较后

$$R=Y \bmod 8*8+X \bmod 8 \tag{11}$$

其中 R 为 M_{dither} 所对应的数组下标，X 为矩阵行数，Y 为矩阵列数。

得到处理后的像素值将生成实现抖动风格化处理，效果如图 2 所示。

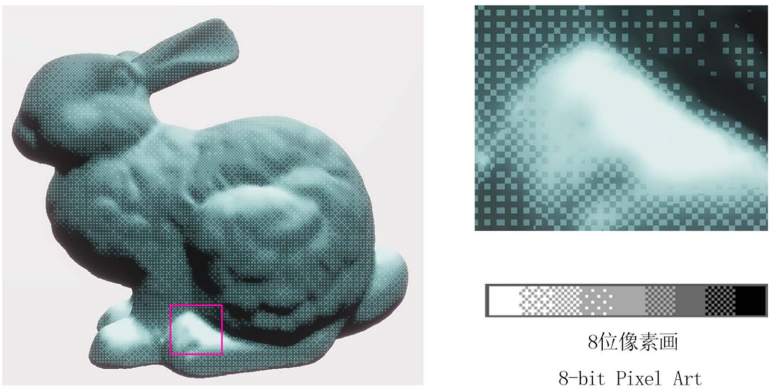

Figure 2. Dither dithering stylized effect
图 2. Dither 抖动风格化效果

3.4. 合并结果

将设计好的光照模型，像素化与抖动处理风格化三者效果相互结合，调整参数后得到效果如图 3 所示。

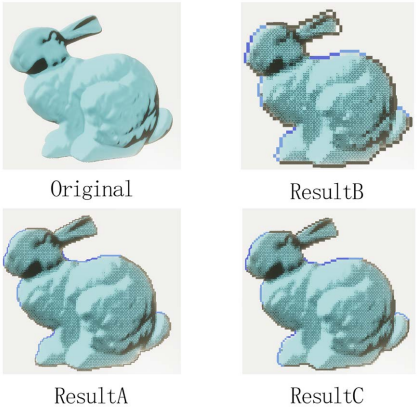

Figure 3. Merge renderings
图 3. 合并效果图

3.5. 时间复杂度理论分析

在进行光照模型设计时，漫反射模型的时间复杂度主要取决于光源数量 n，因此时间复杂度约为 $O(n)$。在实际应用中，高光模型的应用区域比较小，时间复杂度约为 $O(1)$。

实时像素化处理过程中，该算法的时间复杂度取决于图像大小以及卷积核大小，本文所提到的像素化过程基于边缘检测，因此实时像素化过程的时间复杂度与边缘检测算法所需时间复杂度相同，需要进行高斯滤波操作其时间复杂度为 $O(whks)$，其余操作时间复杂度为 $O(wh)$ 故总时间复杂度为 $O(whks)$，其中 $m*n$ 为图像尺寸，k 是高斯滤波器大小，s 为高斯滤波器 σ 值。

风格化抖动处理中，算法的时间复杂度主要取决于抖动矩阵的大小 8，以及图像尺寸 $w*h$，因此时间复杂度约为 $O\left(w*h*8^2\right)$。

将上述时间复杂度综合计算，本文所提出的方法总时间复杂度为 $O\left(\max\left((w*h*k*s),\left(w*h*8^2\right)\right)\right)$。

该方法与传统的像素化方法如 MSAA(多重采样抗锯齿 Multisample anti-aliasing)相比，节省了多次采样的次数，减少了所产生的时间复杂度，提高了实时像素化的性能。

4. 实验与结果评估

本文为了探究实时像素化的非真实感渲染效果，并通过对多个场景进行渲染测试，进行对比试验。本实验使用 OMEN Laptop 15-en0xxx 设备，处理器为 AMD Ryzen7 4800 with Radeon Graphics 2.90 GHz，16GB 内存，显卡为 NVIDIA GeForce RTX2060 开发运行环境为 Windows 10 的操作系统，开发语言为 C++。

4.1. 实验准备

实验准备的初期阶段，我们选取 100 个场景进行测试。测试场景包括由正方体，球体组成的简单几何模型，以及动漫人物模型，房屋场景，植物模型等。以此用来测试本文提出的渲染方式的鲁棒性。

4.2. 评价指标

本文通过渲染时间，渲染效果准确性，内存占用大小。采用这三项测试指标来对本文算法进行实验判断。渲染准确性指对不同面片数量下的模型进行像素化是否能够精准像素化。稳定性指在不同数量的光照环境下，像素化风格渲染是否能够正确分布。渲染时间通过 CPU 和 GPU 的渲染时间进行反馈得到。内存占用大小通过测试算法运行时所需的内存空间得到。

4.3. 可行性实验

通过可行性实验，我们可以对新的图形算法或技术进行初步的评估，了解它们的优点和缺点，为后续的深入研究和应用奠定基础。本文设计了两个可行性实验，即时间复杂度分析与空间复杂度分析。

4.3.1. 时间复杂度分析

通过逐个场景进行渲染，记录在相同光照环境下得到在不同场景面片数量情况下，所需渲染的平均时间。如表 1 所示，可以分析观察到本文所提出的像素化非真实感渲染方式与场景面片数量对其的影响并不大。

随后，本文在不同光照环境下对同一场景进行像素化非真实感渲染，得出结果如表 2 所示。

通过分析可以观察到在不同光照环境下对同一物体渲染所需要的时间复杂度与光照环境数量成线性关系。

Table 1. Relationship between the number of scene patches and overhead time
表 1. 场景面片数量与开销时间关系表

场景面片数量	场景个数	CPU 平均开销时间(ms)	GPU 平均开销时间(ms)
0~10	15	15.32	35.3
10~100	15	35.64	42.4
100~1000	10	21.3	23.31
1000~10,000	10	45.29	67.45
10,000~50,000	15	33.68	95.36
50,000~100,000	15	45.21	59.84
100,000~150,000	20	30.5	88.65

Table 2. Relationship between the number of lighting environments and overhead time
表 2. 光照环境数量与开销时间关系表

光照环境数量	CPU 平均开销时间(ms)	GPU 平均开销时间(ms)
1	16.47	14.88
5	34.24	22.15
10	56.45	30.21
25	78.24	36.97
50	99.52	43.23
75	123.78	65.12
100	154.123	80.21

4.3.2. 空间复杂度分析

通过对逐个场景进行渲染，与时间复杂度分析相同，记录在相同光照环境下，不同面片数量的场景情况下渲染过程对性能利用率的影响。以及在不同数量光照环境下对同一场景渲染过程中对性能利用率的影响。如表 3，表 4 所示。分析可知，场景的不同面片数量与不同数量的光照环境都与内存成线性关系。

Table 3. Relationship between the number of scene patches and utilization
表 3. 场景面片数量与利用率关系表

场景面片数量	场景个数	CPU 平均利用率	GPU 平均利用率
0~10	15	14.3%	15%
10~100	15	16.70%	27.20%
100~1000	10	22.50%	34.10%
1000~10,000	10	26.40%	39.86%
10,000~50,000	15	32.10%	45.38%
50,000~100,000	15	34.20%	51.54%
100,000~150,000	20	37.48%	59.87%

Table 4. Relationship between the number of lighting environments and their utilization
表 4. 光照环境数量与利用率关系表

光照环境数量	CPU 平均利用率	GPU 平均利用率
1	15.54%	16.5%
5	20.21%	21%
10	25.98%	30.15%
25	28.49%	44.10%
50	30.84%	49.3%
75	33.57%	54.12%
100	36.15%	58.90%

4.4. 与传统的像素化方法对比实验

传统的像素化方法是利用 MSAA 描边的本质就是找出颜色值变化较大的像素边缘，通过比较相邻像素的颜色值差异，找出需要描边的像素。使用该方法会产生较高的时间复杂度以及在固定模糊采样过程中导致频域中的低频信息丢失引起画面缺失或导致锯齿仍旧存在，从而降低像素化的精确程度。

为验证本文所提供的方法的可靠性，将本文所提供的方法与传统的像素化方式进行对比实验。即对同一场景进行像素化并对比其内存消耗，以及像素化稳定性进行分析。如图 4，图 5 所示。可以观察到本文所用方法的内存消耗小于传统的像素化方法。

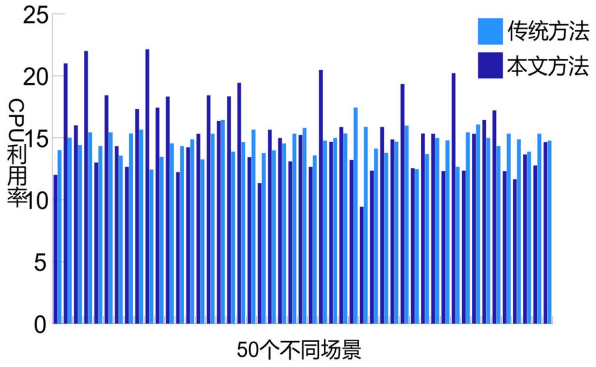

Figure 4. CPU utilization comparison
图 4. CPU 利用率对比图像

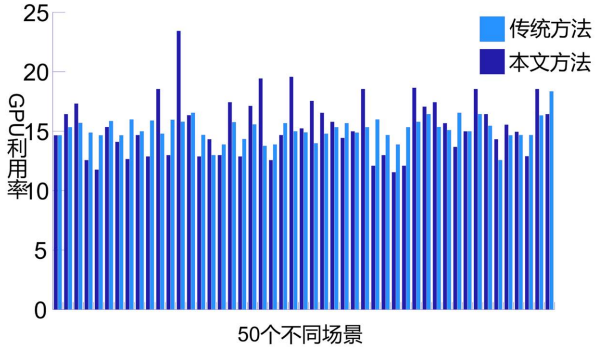

Figure 5. GPU utilization comparison
图 5. GPU 利用率对比图像

通过增加场景中的光照环境个数，再对本文提出的像素化非真实感渲染方式与传统的像素化方式进行对比，如图 6，图 7 所示，通过分析可以得出本文所提供的方法在多个光照环境中消耗内存以及渲染时间都要优于传统所使用的方法。但通过对 CPU 以及 GPU 平均利用率进行方差得出的稳定性略劣于传统方法，如图 8 所示。

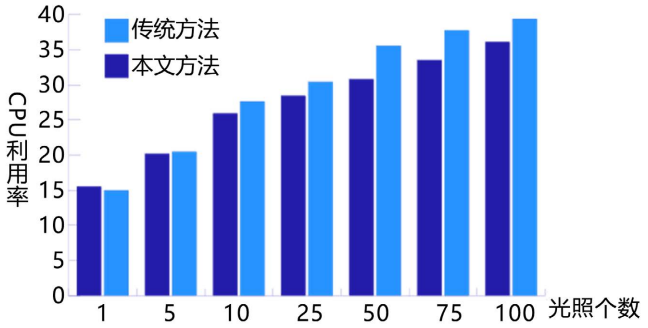

Figure 6. Comparison of CPU utilization under different lighting amounts
图 6. 不同光照数量下 CPU 利用率对比图像

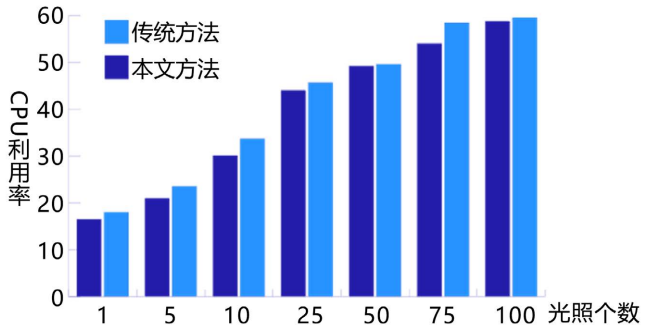

Figure 7. Comparison of GPU utilization under different lighting amounts
图 7. 不同光照数量下 GPU 利用率对比图像

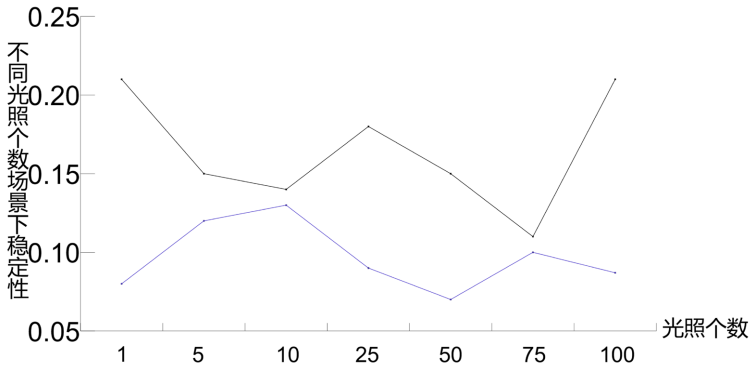

Figure 8. Stability comparison
图 8. 稳定性对比图像

通过以上两组实验的对比，可以得出本文提出的方法是在性能上优于传统方法，但稳定性稍劣于传统的方法。可以认为本文所提供的方法是在损失一定的渲染精度的情况下更加高效的替代方法。

5. 结论

针对于像素化的非真实感渲染效果，本文提出了一种可以将三维场景经过高性能实时的像素化的非真实感渲染实现像素化。本文通过对 Blinn-Phong 的光照模型进行改进，设计特化的光照模型，改进传统的像素化方式，采用 Canny 边缘检测进行算法优化[15]，使得物体边缘更加清晰，避免当场景内出现多个物体有遮蔽关系时，边缘不清的效果。同时使用了 8 位像素化风格的抖动效果进行风格化处理，从而增强场景的层次感[16]。并且通过实验测试发现，本文所提出的方法在性能以及效果上都要优于传统的像素化渲染方法。然而风格化抖动效果在特定光照角度下，偶尔会出现与所需像素化渲染场景的割裂感，渲染失真等问题。基于此，在后续的工作中会着重优化风格化和像素化渲染方式的相互结合。避免渲染失真等问题。

参考文献

[1] 汤力, 常晋义, 卞负. 基于分形维数向量的图像类推技术[J]. 计算机工程与设计, 2009, 30(13): 3142-3143, 3156. https://qikan.cqvip.com/Qikan/Article/ReadIndex?id=31022397&info=6jRdmp7GMFLoNkma9DLAArm3ed13lTxBZFYOq0bcDx8%3d

[2] Kawata, H. and Kanai, T. (2013) A Projection Operator for Representing Sharp Features Using Visibility. *Computer-Aided Design and Applications*, **10**, 33-44. https://doi.org/10.3722/cadaps.2013.33-44

[3] Peng, C.-H. , Yang, Y.-H. , Chuang, Y.-Y. and Tseng, Y.-H. (2016) A Pixelation-Based Shading Model for Stylized Rendering. *ACM Transactions on Graphics*, **35**, Article No. 131. https://doi.org/10.1145/2897824.2925935

[4] Lee, J., Yoo, J.-J., Ryu, S. and Kim, J. (2016) Selective Multi-Sample Anti-Aliasing for Mobile Vector Graphics. 2016 *IEEE International Conference on Consumer Electronics* (ICCE), Las Vegas, 7-11 January 2016, 174-175. https://doi.org/10.1109/ICCE.2016.7430569

[5] Dai, W. (2023) Extraction and Measurement of Sub-Pixel Profile Based on Three Spline Interpolation. *Proceedings of the* 2022 7*th International Conference on Biomedical Imaging, Signal Processing* (*ICBSP* '22), New York, 28-30 October 2022, 28-33. https://doi.org/10.1145/3578892.3578897

[6] Zhang, H.X., Wang, C., Liu, X., *et al*. (2018) Image Edge Detection Algorithm and Its New Development. *Computer Engineering and Applications*, **54**, 11-18.

[7] Yang, D.P., Peng, B., Al-Huda, Z., *et al*. (2022) An Overview of Edge and Object Contour Detection. *Neurocomputing*, **488**, 470-493. https://doi.org/10.1016/j.neucom.2022.02.079

[8] 张加朋, 于凤芹. 基于 canny 算子改进型的影像测量边缘检测[J]. 激光与光电子学进展, 2020, 57(24): 258-265. https://doi.org/10.3788/LOP57.241024

[9] Tong, Z.S., Liu, Z.T., Hu, C.Y., *et al*. (2021) Preconditioned Deconvolution Method for High-Resolution Ghost Imaging. *Photonics Research*, **9**, 1069-1077. https://doi.org/10.1364/PRJ.420326

[10] 于新善, 孟祥印, 金腾飞, 等. 基于改进 Canny 算法的物体边缘检测方法[J/OL]. 激光与光电子学进展, 2023: 1-19. http://kns.cnki.net/kcms/detail/31.1690.TN.20230309.1733.054.html

[11] Liang, Z.-Y., *et al*. (2019) Fast Fourier Single-Pixel Imaging Based on Sierra-Lite Dithering Algorithm. *Chinese Physics B*, **28**, 189-194. https://doi.org/10.1088/1674-1056/28/6/064202

[12] Cai, N., *et al*. (2019) Multi-Objective Strategy to Optimize Dithering Technique for High-Quality Three-Dimensional Shape Measurement. *Chinese Physics B*, **28**, 381-386. https://doi.org/10.1088/1674-1056/ab427b

[13] 郝建新, 谢剑斌, 蔡宣平, 孙茂印. 基于查找表的像素处理器新算法[J]. 国防科技大学学报, 1998, 20(5): 81-85.

[14] 刘政林, 郭旭, 邹雪城, 肖建平. 基于改进 Bayer 抖动算法的图像色彩增强技术[J]. 华中科技大学学报: 自然科学版, 2006, 34(5): 68-70.

[15] Wei, D., Ling, Y. and Zhang, W. (2023) Canny Edge Detection Algorithm Based on Sparse Representation Denoising. *Proceedings of the* 2022 6*th International Conference on Electronic Information Technology and Computer Engineering* (*EITCE* '22), Xiamen, 21-23 October 2022, 1707-1712. https://doi.org/10.1145/3573428.3573730

[16] Wang, Q. (2020) Simulation Research of Sketch Simulation Technology Based on Non Photorealistic Rendering. 2020 5*th International Conference on Smart Grid and Electrical Automation* (*ICSGEA*), Zhangjiajie, 13-14 June 2020, 619-622. https://doi.org/10.1109/ICSGEA51094.2020.00140