一种后处理式的改进抗锯齿算法

邵 鹏[1] 周 伟[1] 李光泉[1] 吴志健[2]
(江西农业大学计算机与信息工程学院 南昌 330045)[1] (武汉大学计算机学院 武汉 430072)[2]

摘 要 FXAA 算法是一种后处理式的抗锯齿算法,由于它是基于图像像素的边缘检测算法,在很多情况下边缘检测的不准确会造成许多不必要的抗锯齿计算,导致图像过度模糊。为了提高抗锯齿的性能,提出了一种基于 FXAA 算法的改进抗锯齿算法(IAAFXAA)。该算法将相对视点的深度和法线保存到纹理中,当使用深度和法线信息时从 G-buffer 中提取。该算法能利用深度、法线信息进行较为精确的边缘检测。大量的实验结果与分析表明,该算法在保证抗锯齿效果的同时,能更加精确地确定抗锯齿区域,生成高质量的边界,避免图像的过度模糊,提升图像质量。

关键词 延迟渲染,G-Buffer,抗锯齿,边缘检测
中图法分类号 TP301 **文献标识码** A

Improved Anti-aliasing Algorithm Based on Deferred Shading

SHAO Peng[1] ZHOU Wei[1] LI Guang-quan[1] WU Zhi-jian[2]
(School of Computer and Information Engineering, Jiangxi Agricultural University, Nanchang 330045, China)[1]
(School of Computer, Wuhan University, Wuhan 430072, China)[2]

Abstract FXAA is a post-processing anti-aliasing algorithm. Because it is an edge detection algorithm based on image pixel, it causes a lot of unnecessary anti-aliasing computation. In order to improve the performance of anti-aliasing, an improved anti-aliasing algorithm based on FXAA (IAAFXAA) was proposed. The depth and normal of the relative view are saved into the texture. The algorithm extracts depth and normal information from G-buffer, and uses depth and normal information to perform more accurate edge detection. A large number of experimental results and analysis show that while ensuring the good anti-aliasing effect, the proposed algorithm can determine the anti-aliasing region more accurately to generate high-quality boundaries, and avoid excessive blurring of images to improve image quality.

Keywords Deferred rendering, G-Buffer, Anti-aliasing, Edge detection

1 引言

锯齿(走样)是由于对连续信息进行离散采样、存储或表示等导致的一种信号失真现象,其是计算机图形学研究的基本问题之一[1]。图形抗锯齿可以有效地重建出几何、纹理、运动等各种细节,从而提高绘制图形和生成动画的质量,帮助用户获得更好的视觉体验。

抗锯齿技术(Anti-Aliasing, AA),也称为边缘柔化、消除混叠、抗图像折叠有损等,是一种消除显示器输出画面中图物边缘凹凸锯齿的技术,其随着 3D 技术的不断发展有着更加广泛的应用[2],如虚拟现实、3D 电影(视频处理[3])、3D 游戏(3D 显卡)、飞行模拟[4]以及其他场景渲染等。随着计算机图形图像技术的快速发展,当前的抗锯齿技术主要分为两种:基于硬件技术的多重采样技术[5]和基于图像像素颜色值或距离的边界混合权重采样技术[6]。这两种技术有其自身的不足,采用的都是"前处理"的思想,非常依赖于渲染管线,缺乏独立性[7-8]。延迟渲染技术[8-9]是目前渲染引擎设计的主流渲染方式,能满足虚拟场景实时渲染的要求,相比前向渲染(Forward Rendering),延迟渲染具有光照所耗费的资源独立于场景复杂度、计算开销小等优点,具有较为广泛的应用。延迟渲染可以避免同一像素的多次计算,提升渲染效率。然而,延迟渲染也有其不足,比如由于延迟渲染技术无法兼容硬件多重采样反走样技术(MSAA)[9],会导致几何信息的缺失,进而变成暴力超级采样反走样(SSAA)。正是因为这点,传统的 MSAA 显得力不从心。快速近似抗锯齿(Fast Approximate Anti-Aliasing, FXAA)作为一种新型后处理式抗锯齿算法,由于其速度快且效果好而被广泛地使用[10,15]。然而,由于 FXAA 抗锯齿技术是基于图像像素进行边缘检测的,往往会进行许多不必要的计算,导致图像过度模糊。因此,提出一种改进的抗锯齿算法(IAAFXAA),该算法从 G-buffer 中提取深度和法线信息并借助该信息进行更加精确的检测,以提高抗锯齿性能,避免图像过度模糊。

2 相关工作

传统超采样抗锯齿算法(Super-Sampling Anti-Aliasing, SSAA),通过直接增加图像的采样频率来减弱锯齿现象,其需要在每个像素中计算多个采样点的渲染信息,但在许多实际应用中渲染计算模型由于其高度复杂性,并不能承受

本文受国家自然科学基金(70971043,61763019),江西省教育厅科技项目(GJJ160409,GJJ161076)资助。
邵 鹏(1983—),男,博士,讲师,主要研究方向为智能计算、图像处理,E-mail:sp198310@163.com(通信作者);周 伟(1998—),男,主要研究方向为图像处理。

SSAA 带来的数倍的渲染计算开销[11-12]。多重采样抗锯齿算法(Multi-Sampling Anti-Aliasing,MSAA)是对 SSAA 的改进算法,其主要思想是把渲染计算和深度模板测试分开处理,即对每个像素进行深度模板测试,但在像素内只在每个几何片元中进行一次渲染计算,通过这种方式,在不容易发生锯齿的非边界区域,该算法省去了不必要的渲染计算[13]。

除上述抗锯齿算法外,还有其他一些算法。如前文所述,这些算法都不能很好地兼容延迟渲染技术。相比前向渲染,延迟渲染是一种在图像空间中处理光照的技术,它将场景信息离散化并保存到一系列屏幕分辨率大小的纹理中,然后通过对纹理中的数据进行后期处理来实现渲染计算。它将一些需要使用的法线信息、漫反射信息等一系列与渲染计算有关的量都存储到被称为几何缓存(G-buffer)的纹理中[14]。使用 G-buffer 技术,整个计算过程都在二维空间中执行且可以节省大量的计算开销。

为了在抗锯齿技术中使用流行的延迟渲染技术,一些使用延迟渲染技术的改进抗锯齿算法被提出。这些算法都是基于后期处理(Deferred Shading)的抗锯齿算法,与延迟渲染技术有着很好的兼容性。后期处理的抗锯齿算法的主要思想是把场景信息以二维纹理的形式存储,并借助这些信息进行场景的抗锯齿重建。FXAA 是该类算法的经典代表之一,其通过快速的锯齿特征进行查找并以颜色衰减的方式进行抗锯齿颜色的重建,但它是一种以牺牲抗锯齿质量和精度来换取性能的算法[15]。为了进一步提高抗锯齿性能,针对 FXAA 的不足,在 FXAA 的基础上结合 G-Buffer 提出一种改进的抗锯齿算法(IAAFXAA)。

3　基于 FXAA 的抗锯齿算法设计

3.1　算法原理

由于屏幕是由像素点阵构成的,但是现实中的事物往往是连续的,因此采用离散型数据来表示连续量时通常会造成图像失真。为了解决这一问题,SSAA,MSAA 等硬件抗锯齿算法应运而生。但是,这些算法在带来更高视觉效果的同时也带来了巨大的性能开销。FXAA 是 MSAA 效果的高性能近似,同时占用更少的资源。但是,FXAA 基于图像像素的边缘检测,易受光照、纹理、阴影等因素的影响,所得出的结果并不完全准确,产生了许多多余的计算。针对此问题,本文提出了一种改进的抗锯齿算法(IAAFXAA)。

该方法首先从存储场景的子像素几何信息的 G-buffer 中提取深度和法线信息,再通过子像素间的几何相似性来估计子像素的颜色,并以此为依据判断其是否为边缘。若不是,则不进行任何处理直接跳过,若是,则应用计算模型对其进行处理,以达到平滑的目的。

3.2　算法实现

3.2.1　构建深度法线纹理

深度法线纹理的本质是一张渲染纹理,其存储了视点到每个像素的距离和法线值,并且这个距离并非是笛卡尔坐标系中的欧几里德距离,而是视点到每个像素的相对距离。由于其储存在渲染纹理中,因此需要先把深度值映射到[0,1]中。由于深度值与观察空间下 z 值的倒数是线性的(参考透视投影矩阵的推导过程),因此深度 $depth$ 可以表示为:

$$depth=a\times\frac{1}{z}+b \tag{1}$$

其中,a 和 b 为常数。接下来需要将深度值映射到[0,1],因此可以得到如下方程组:

$$\begin{cases}0=a\times\dfrac{1}{near}+b\\[2ex]1=a\times\dfrac{1}{far}+b\end{cases} \tag{2}$$

解得:

$$\begin{cases}a=\dfrac{far\times near}{near-far}\\[2ex]b=\dfrac{far}{far-near}\end{cases} \tag{3}$$

整理可以得到:

$$depth=\frac{far\times near}{near-far}\times\frac{1}{z}+\frac{far}{far-near} \tag{4}$$

式(4)即为深度 $depth$ 的计算公式。其中,z 是观察空间下的 z 值,far 是远剪裁面到视点的距离,$near$ 是近剪裁面到视点的距离。

通过式(4)的计算对深度值进行映射,使其保存到纹理中,然后再编码到 a 和 b 两个通道中,再将深度值映射到[0,255]中,如式(5)所示:

$$enc=depth\times(1.0,255.0) \tag{5}$$

然后用原始数据减去映射后的小数部分,即:

$$enc=frac(enc) \tag{6}$$

$$enc.x=enc.x-enc.y\times\frac{1}{255} \tag{7}$$

其中,$frac$ 可以获取浮点数中的小数部分。最后再将 enc 的 x 和 y 分量分别保存到 b 和 a 通道中。

接下来是计算法线值,首先把三维坐标转化为二维,并且映射到[0,1]中。假定 k 为缩放系数且 k 的取值 1.777,然后进行降维,即:

$$enc=\frac{n.xy}{n.z+1} \tag{8}$$

$$enc=enc/k \tag{9}$$

其中,x,y,z 分别为法线 n 的 3 个分量。

此时得到的法线值在[-1,1]中,因此需要对其进行再次映射,即:

$$enc.xy=enc.xy\times0.5+0.5 \tag{10}$$

这样得到的法线值就被映射到[0,1]中。

由于只需要检测法线值的差异,因此在后续的检测中可以不需要对其进行解码。

3.2.2　边缘检测

边缘检测对于大多数抗锯齿算法来说尤为重要,一个好的边缘检测算法可以减少计算量,提高渲染效率,提升画面质量。常用的算法一般是基于颜色、亮度、深度、法线中的一种或多种的组合。FXAA 算法采用的是以像素(颜色、亮度)为对象的边缘检测,但是这种方法往往存在误差。因此,在 IAAFXAA 算法中使用深度+法线进行边缘检测,快速、准确地重建原始边缘信息。

对此,在 IAAFXAA 算法中取对角线上的 4 个采样点并使用 *roberts* 算子[16]来进行检测。表 1 列出了 *roberts* 算子对应的卷积模板,并由此得到该点的梯度值。如果该点的梯度值大于某个阈值(其大小取决于表 2 中的两个参数 SD 和

SN),则认为在这个点附近存在一条边界。

表 1 *roberts* 边缘检测算子

gx		*gy*	
1	0	0	1
0	−1	−1	0

下面分别从 3 个方面进行边界判定的分析。

(1)深度信息

利用深度信息进行后续计算首先要获得深度值,但是直接对纹理采样得到的不是真正的深度值而是经过压缩编码后的值,因此需要先将其还原才能得到深度值。

由于深度编码是将映射后的深度值的小数部分与原始数据分开存放,因此将小数部分还原后再与原始数据相加,即:

$$kdecodedot=(1,\frac{1}{255}) \tag{11}$$

$$depth=kdecodedot\cdot encd \tag{12}$$

其中,*encd* 为直接从纹理中采样得到的深度值,利用该深度值即可进行边缘检测,*kdecodedot* 为常数。

设有一点 p,若 d_1 和 d_2 分别为对角线上两个点的深度值,则有:

$$isedgedepth=\frac{|d_1-d_2|}{d_1}>depththreshold \tag{13}$$

然后,由 *isedgedepth* 就可以判断当前两个像素之间是否存在边界。*isedgedepth* 是由两点间的深度差异得到的,表示这两点间是否存在边界。

图 1 给出了仅使用深度值进行边缘检测的结果,可以看出该结果存在着相当大的误差,许多地方(如地面)被误当成边界。

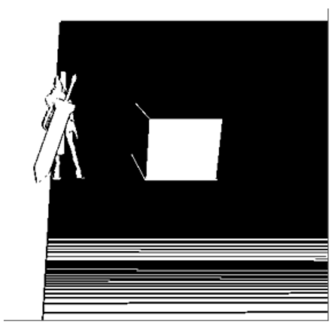


图 1 仅用深度值进行边缘检测的结果

(2)法线信息

如果场景中两点存在边界,则这两点间的法线会相差较大;若为光滑的平面则差异较小。因此,可以以法线差异作为边缘检测的依据。

设有一点 p,假设 n_1 和 n_2 为对角线上两个点之间的法线值,则法线差异 $diffnor$ 为:

$$diffnor=n_1-n_2 \tag{14}$$

再将法线的差异与阈值 *northreshold* 进行比较,以弥补深度检测的不足,进而检测出深度变化较小的边缘。

$$isedgenormal=(diffnor.x+diffnor.y)>northreshold \tag{15}$$

其中,*isedgenormal* 表示通过法线差异判断是否存在一条边界。*northreshold* 与下文实验参数中的 *SensitiveNormal* 有关。

图 2 给出了仅使用法线进行边缘检测的结果,仔细观察可以看到,角色表面几乎都被处理成边界,因此仅仅使用法线信息进行边缘检测得到的结果是不够准确的。

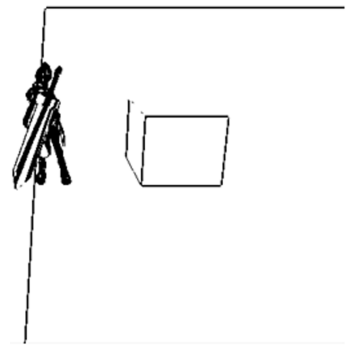


图 2 仅使用法线值进行边缘检测的结果

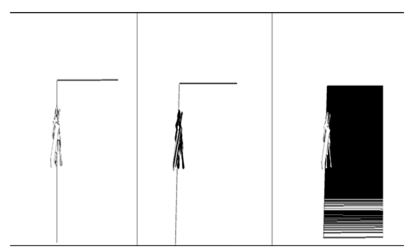
(3)综合深度信息和法线信息

由上文可知,无论是使用深度还是法线,使用单一的元素得出的效果均较差。因此,若要获得较好的边缘检测质量则需要结合两种元素同时进行判别。由上文的描述中可以得知,使用深度值时,较平坦(法线变化较小)的地方容易被误判,使用法线值时,距离较近(深度变化小)的地方容易被误判。因此,当深度和法线的变化都足够大时,得出的结果是比较理想的。

$$isedge=isedgedepth * isedgenormal \tag{16}$$

其中,*isedge* 表示综合法线、深度信息后两点间是否存在边界。

由图 3 可以看出,综合两种元素进行的边缘检测弥补了单一元素进行检测时容易被误判的不足,将场景边界完整地提取了出来。



(a)深度+法线 (b)仅使用法线 (c)仅使用深度

图 3 使用各元素进行边缘检测的结果对比

3.2.3 确定偏移方向

针对提取出的边缘,若要消除其中的锯齿则需要将边缘与其周边的像素进行混合,使其过渡更加平滑。在此之前,需要找出像素偏移的方向,为了取得较好的平滑效果,需要找出像素值变化最快的方向,而图像边缘的特性就是灰度值(彩色图像中称之为亮度值)变化最大的方向,也就是像素的偏移方向。

首先需要计算出颜色的亮度,如式(17)所示:

$$Luminance=col.r\times 0.2125+col.g\times 0.7154+col.b\times 0.0721 \tag{17}$$

其中,r,g,b 分别为颜色值的 3 个分量。

然后分别计算左上、左下、右上、右下 4 个点的亮度值。接下来计算混合方向,首先计算水平方向,即:

$$dir.x=lumaSW-lumaNE-lumaSE+lumaNW \tag{18}$$

$$dir.y=lumaSW-lumaNE+lumaSE-lumaNW \tag{19}$$

其中,*lumaSW* 是中心点左下偏移后采样得到的亮度;

$lumaNE$ 是中心点右上偏移后采样得到的亮度；$lumaSE$ 是中心点右下偏移后采样得到的亮度；$lumaNW$ 是由中心点左上偏移后得到的亮度，其偏移量为单个像素的大小与采样距离（可参考下文的参数设置）之积，亮度可由式(17)计算得出，因此得到了像素的偏移方向 dir。

3.2.4　混合平滑

混合平滑是抗锯齿的最后步骤，将四周的像素进行混合替代边缘处的像素，减轻图像边缘与周围像素的差异感从而减轻图像的锯齿感。

假定 $offset$ 为像素的初始偏移值，即：

$$offset=\begin{cases}0.25\times\_MainTex\_Texel.xy\\-0.25\times\_MainTex\_Texel.xy\\0.75\times\_MainTex\_Texel.xy\\-0.75\times\_MainTex\_Texel.xy\end{cases}\tag{20}$$

其中，$\_MainTex\_Texel.xy$ 为单个像素的长度和宽度，存在 x 和 y 两个分量，分别表示像素的宽度和高度。

每个像素的具体偏移值为：

$$pixel\_offset=offset\times dir\tag{21}$$

接下来对每个像素都进行采样，然后将其相加取平均值即可得到最终的颜色值，即：

$$color=color\_s/4\tag{22}$$

其中，$color$ 为最终颜色，$color\_s$ 为混合相加后的颜色。

4　实验结果与分析

4.1　实验平台

处理器：Intel(R) Core(TM) i5-6300HQ CPU @ 2.30 GHz 四核；内存：8GB；显卡：NVIDIA GeForce GTX 960m；显示器分辨率：1920×1080；操作系统：windows10(64 位)。

4.2　参数设置

通过实验可以得出以下参数的最佳值：采样距离(Sample Distance，SD)，深度敏感(Sensitivity Depth，SDe)，法线敏感(Sensitivity Normal，SN)，边缘最小阈值(Edge Threshold Min，ETM)，边缘阈值(Edge Threshold，ET)，边缘锐利度(Edge Sharpness，ES)，在此条件下可以取得较好的渲染效果，如表 2 所列。

表 2　各算法的最佳参数

	SD	*SDe*	*SN*	*ETM*	*ET*	*ES*
IAAFXAA	0.55	15.0	3.0	—	—	—
FXAA	—	—	—	0.125	0.25	4.0

4.3　实验结果与分析

4.3.1　综合深度法线信息的边缘检测结果比较

检测结果对比如图 4 所示。

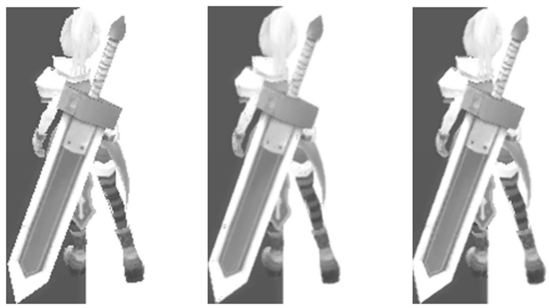
(a)基于像素　　(b)基于深度与法线

图 4　检测结果对比

由图 4 可以看出，基于像素的方法对平面进行检测并不理想，许多地方存在漏检，有时也会认为纹理上的颜色变化是边界，而且对于阴影这些不必检测的细节也添加到了其中。而基于深度与法线进行的边缘检测不仅没有添加阴影、纹理这些不必要的细节，而且对于边缘的检测更加准确、完整。

4.3.2　各算法的性能对比

原图、FXAA 与 IAAFXAA 算法的对比如图 5 所示。



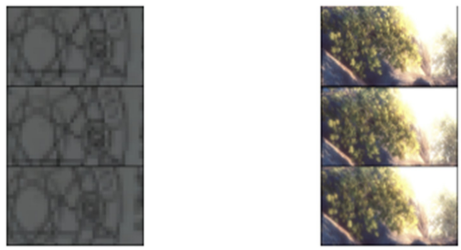
(a)原图　　(b)FXAA　　(c)IAAFXAA

图 5　各算法的性能比较

由图 5 可以看出，原图经过 IAAFXAA 和 FXAA 算法处理后锯齿感明显消除了。同时，对比图中细节部分可以看出，使用 FXAA 算法得到的画面有些许模糊，并且例如高光等细节也被模糊掉了，纹理质量也有一定程度的损失。而使用 IAAFXAA 算法时虽然在画面上也有轻微的模糊感，但是保留了大部分的细节内容。因此，IAAFXAA 算法对于边界的识别能力强于 FXAA 算法，减轻了 FXAA 算法在视觉上的模糊感，提供了更好的视觉效果。

4.3.3　其他场景下的测试

图 6 给出了在不同场景下，本文算法与 FXAA 算法、原图的性能比较。图 6 中，从上到下依次为原图、本文算法实验放大图和 FXAA 实验图。



(a)场景 1　　(b)场景 2

图 6　各算法在不同场景下的性能比较

通过细节图可以看出，经过处理的图片都有不同程度的模糊。仔细观察可以看出，FXAA 在处理纹理时往往会模糊纹理，颜色直接被淡化了。在处理树叶这样密集的图形时也时常因为边缘检测的不准确使得图像看起来模糊。而本文算法则避免了这些问题，使图像可以清晰地显示出来。

4.3.4　平均帧率

表 3 列出了在相同时间内的平均帧率。

表 3　各算法的平均帧率

序号	原图	FXAA	IAAFXAA
示例场景	1114.875	1049.625	1050.380
场景 1	172.500	169.139	167.657
场景 2	127.940	127.592	127.318

(下转第 225 页)

表 2　两种算法输出位姿的对比

算法 \ 位移 x/cm	10	15	20
MR 算法	9.246	14.125	18.341
本文算法	10.523	15.431	20.875

从实验结果可以看出,本文所提出的特征提取算法继承了 MR 算法的准确度,同时还提高了 MR 算法的效率,这不仅能够充分提取图像信息,还能大幅度加快图像匹配的速度。

结束语　本文提出了一种基于四叉树的 ORB 特征提取均匀改进算法。针对图像特征提取过程中特征点过于集中而导致的图像局部特征信息丢失的问题,对图像特征点进行了四叉树划分,同时增加了区域边缘检测以及特征点去冗余,最后在每个节点中取 Harris 响应值最大的角点,保证了特征提取的均匀化。仿真实验验证了本文所提算法在图像特征均匀化处理方面的快速性以及准确性。

本文提出的算法能够作为计算机视觉领域提取并处理特征点的一种有效方法,对于后期机器人建图和自定位的研究具有重要的意义。

参 考 文 献

[1] LOWE D G. Distinctive image features from scale-invariant keypoints[J]. International Journal of Computer Vision,2004,60(2):91-110.

[2] BAY H,TUYTELAARS T,VAN GOOL L. Surf:Speeded up robust features[J]. European Conference Computer Vision,2006,110(3):404-417.

[3] RUBLEE E,RABAUD V,KONOLIGE K,et al. OR-B:An efficient alternative to SIFT or SUR-F[C]// International Conference on Computer Vision. 2012:2564-2571.

[4] MUR-ARTAL R,MONTIEL J M M,TARDOS J D. ORB-SLAM:a versatile and accurate monoc-ular SLAM system[J]. IEEE Transactions on Robotics,2015,31(5):1147-1163.

[5] BENTLEY J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM,1975,18(9):509-517.

[6] MUR-ARTAL R,TARDÓS J D. Orbslam2:An open-source slam system for monocular, stereo, and rgb-d cameras[J]. IEEE Transactions on Robotics,2017,33(5):1255-1262.

[7] ROSTEN E,DRUMMOND T. Machine learning for high-speed corner detection[C]// European Conference on Computer Vision. Springer-Verlag,2006:430-443.

[8] HARRIS C,STEPHENS M. A combined corner and edge detector[C]// Alvey Vision Conference. Manchester,UK,1988:5244.

[9] CALONDER M,LEPETIT V,STRECHA C,et al. Brief:Binary robust independent elementary features[C]// European Conference on Computer Vision. 2010:778-792.

[10] MUJA M,LOWE D G. Fast Matching of Binary Features[C]// Computer and Robot Vision. 2012:404-410.

[11] LEE M K. Pointing device of augmented reality:U. S. Patent Application 13/581,548[P]. 2011-2-28.

[12] TRIGGS B,MCLAUCHLAN P F,HARTLEY R I,et al. Bundle adjustment—a modern synthesis[C]// International Workshop on Vision Algorithms. Berlin Heidelberg Springer:1999:298-372.

(上接第 221 页)

由表 3 可以看出,由于使用更加精确的边缘检测算法,在渲染场景时 IAAFXAA 算法在性能上有所提升。由此可知,在渲染较复杂的场景时,该算法可以保证渲染的实时性,或者是在保证足够的帧率时,能够进一步提升画面质量。

结束语　为了提高抗锯齿的效果,提出了一种后处理式的抗锯齿算法,使用了更加精确的边缘检测算法。在保证 FXAA 算法处理效果的同时,进一步提升了渲染质量。由于边缘检测算法的改进使得抗锯齿区域的识别更加准确,因此能够保留诸如高光等光照效果并且不会模糊纹理细节,此外还可以生成高质量的边界,用于场景描边等特殊画面效果。在性能方面,IAAFXAA 算法处理后的场景帧率有所下降,但都在 5%左右,相对于提升的画面质量而言这些性能损失是完全值得的。因此,相较于 FXAA 算法有相对好的性能提升。今后将继续改进 IAAFXAA 算法的边缘检测部分,增强斜边检测的准确性,同时提高参数的通用性,使其能够不加修改地应用于更多的场景中。

参 考 文 献

[1] 杜文俊. 基于几何的实时绘制反走样[D]. 杭州:浙江大学,2015.

[2] 吴玉培,王斌. 基于 GPU 的实时抗锯齿算法[J]. 现代计算机,2016(3):54-57.

[3] 王栋,刘文波. 基于变混合系数的自适应抗锯齿视频叠加算法[J]. 电子测量技术,2013,36(12):20-24.

[4] 黄炳,陈俊丽,万旺根,等. 飞行视景仿真系统研究与实现[J]. 计算机仿真,2009,26(11):235-238.

[5] BAVOIL L,SAINZ M,DIMITROV R. Image-space horizon-based ambient occlusion[C]// Proceeding SIGGRAPH'08 ACM SIGGRAPH. ACM,2008.

[6] COOK R L,CARPENTER L,CATMULL E. The Reyes image rendering architecture[C]// Conference on Computer Graphics and Interactive Techniques. ACM,1987:95-102.

[7] 王斌,吴玉培,吴志红. 基于 SRAA 延迟渲染抗锯齿[J]. 四川大学学报(自然科学版),2015,52(6):1230-1236.

[8] 罗德宁,张建伟. 基于延迟着色技术的大场景反走样渲染架构设计[J]. 四川大学学报(工程科学版),2017,49(4):158-166.

[9] PHARR M,FERNANDO R. GPU Gems 2:Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems) [M]. Addison-Wesley Professional,2005:143-166.

[10] 佚名. FXAA 如此这般[J]. 电脑迷,2012(2):51-51.

[11] FUCHS H,GOLDFEATHER J,HULTQUIST J P,et al. Fast spheres,shadows,textures,transparencies,and imgage enhancements in pixel-planes[J]. Acm Siggraph Computer Graphics,1985,19(3):111-120.

[12] MAMMEN A. Transparency and Antialiasing Algorithms Implemented with the Virtual Pixel Maps Technique [J]. IEEE Computer Graphics and Applications,1989,9(4):43-55.

[13] AKELEY K. Reality Engine graphics[C]// Conference on Computer Graphics and Interactive Techniques,SIGGRAPH. DBLP,1993:109-116.

[14] GÁBOR L,DACHSBACHER C. Decoupled deferred shading for hardware rasterization[C]// Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM,2012:143-150.

[15] LOTTCS T. FXAA[R]. Nvidia Corporation,2011.

[16] 章慧,陈宏明. 融合 SUSAN 算法和 Robert 算法的图像边缘检测滤波处理技术[J]. 计算机科学,2013,40(3):302-304.