软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2012,23(Suppl.(2)):149−157

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 边界斜率预测的形态学反走样方法*

钟宇琛, 王 锐+

(CAD & CG 国家重点实验室(浙江大学),浙江 杭州 310058)

## Slope-Based Straight Line Predicting Morphological Antialiasing Technique

ZHONG Yu-Chen, WANG Rui+

(State Key Laboratory of CAD & CG (Zhejiang University), Hangzhou 310058, China)

+ Corresponding author: E-mail: rwang@cad.zju.edu.cn, http://www.cad.zju.edu.cn/home/rwang/m

**Zhong YC, Wang R. Slope-Based straight line predicting morphological antialiasing technique. *Journal of Software*, 2012,23(Suppl.(2)):149−157 (in Chinese).** http://www.jos.org.cn/1000-9825/12034.htm

**Abstract**: Aliasing is a long-standing problem in computer graphics. Many techniques have been developed to anti-alias fast and efficiently. Based on the most advanced antialiasing algorithm, this article presents a new approach to deal with the drawback in edge detection and reconstructs the original morphological antialiasing technique. The new approach uses local edge information to calculate the range of the line slope, and then predicts more probable end-points for straight lines. While the reconstructing of straight lines is much better, this technique use minor additive cost. Compared with previous morphological antialiasing techniques, this work not only reconstructs the local edge shapes, but also captures the global edge shapes, which in return helps enormously in antialiasing lines. The use of such global slope-based shape information improves the gradients of straight lines, makes inclined lines more continuous, and therefore, obtains better rendering result.

**Key words**: edge detection; global morphological antialiasing

**摘 要**: 采样不足造成的图像像素走样是一个计算机图形学领域长期存在的问题,寻求速度快、效果好的反走样算法是研究人员一直以来的目标.基于当前最新的形态学反走样算法,针对该算法在倾斜直线边界局部检测与重建上存在的不足,提出了边界斜率预测的形态学反走样方法.该方法利用局部直线边界斜率的信息对直线边界在全局范围内的端点位置进行预测与检验,从而重建出与实际边界更加相符的边界信息,在只增加较少计算量的情况下获得了更准确的直线边界形态.与前人的形态学反走样算法相比,所提出的方法基于全局形态学边界,能够更加准确地重建直线边界,将其应用于反走样计算中,可以进一步改善直线边界的颜色过渡,使倾斜直线边界具有更高的连续性,获得更好的反走样效果.

**关键词**: 边界预测;全局形态学反走样

由采样不足造成的图像像素走样是计算机图形学领域长期存在的一个问题,它会造成静止图像失真,动态图像闪烁.直接增加采样数目可以解决走样问题,但是随采样数目线性增长的计算代价限制了这种方法在实时绘制中的使用.寻求速度快、效果好的反走样算法是当前反走样技术研究的主要目标.

* 基金项目: 国家自然科学基金(60903037); 国家高技术研究发展计划(863)(2012BAH35B03)
收稿时间: 2012-05-20; 定稿时间: 2012-09-29

传统上,超采样反走样(SSAA)和多重采样反走样(MSAA)是最近几十年中运用最广泛的反走样算法.然而,单纯增加采样数的代价是高昂的.Jorge 等人在他们的论文[1]中指出,MSAA 算法无论是内存消耗还是 GPU 占用率都很大,同时,多重采样也很难与时下流行的推延渲染相结合(根据 Jorge 等人的测试,多重采样算法的 GPU 占用率在推延渲染的框架下甚至会高达 30%).

近年来,Reshetov 提出了形态学反走样的方法[2].与超采样反走样(SSAA)和多重采样反走样(MSAA)不同,该方法是一种图像后处理反走样方法.该方法通过对帧缓存中图像的边界进行检测,根据边界具有的不同线型推测重建边界的原始形态,利用重建的边界形态实现对边界像素的反走样计算.该方法证明了在没有经过多重采样的帧缓存上,通过对几何边界的重建也可以得到很好的反走样效果.然而,形态学反走样算法是有其内在缺陷的:由于像素采样的不足,图像上已损失了物体的原始几何边界信息,利用形态学检测获得的边界信息是基于局部线型的,与原始边界不可避免地存在一定的偏差,从而影响了反走样的质量.为了进一步提高反走样的效果,在 Reshetov 的形态学反走样算法[2]提出之后,研究人员提出了多种基于图像形态学检测的后处理反走样技术,其中包括快速近似反走样(FXAA)[3]、距离到边反走样(DEAA)[4]、有方向的局部化反走样(DLAA)[5]、亚像素还原反走样(SRAA)[6]、亚像素改良形态学反走样算法(SMAA)[1]等等.这些算法在面积计算、亚像素特征等多个方面对原始的形态学反走样算法进行了改进,然而,这些算法在解决形态学重建的边界与原始边界偏差方面仍存在缺陷,倾斜直线边界的高质量反走样依然是一个亟待研究的问题.

本文发现,形态学反走样算法中造成重建的倾斜直线边界与原始边界偏差的主要原因是由于这些算法都是通过检测直线像素在水平或是垂直方向上的跳变来实现对直线边界的重建.由于不同斜率直线边界所生成的像素锯齿阶跃宽度不一,仅仅基于局部的线型重建边界与真实的图像之间是存在一定偏差的.如图 1 所示,形态学反走样算法每检测到一个阶梯就对线型进行局部重建,该算法将全局中一条完整的直线(图 1 左,虚线)分成了一组线段(图 1 左,折线).在实际使用中,该误差会造成反走样后直线边界的起伏效果(图 1 右).如果要获得更加完整的直线的全局信息,采用这些算法的像素搜索策略会带来很高的计算代价(具体分析请见第 2 节).

折线:形态学反走样算法对边界的重建
直线:本文算法对边界重建的修正

没有经过反走样处理的边界

经过形态学反走样处理的边界
直线边界有起伏效果

图 1

针对传统形态学算法的这一问题,本文通过对图像边界斜率的分析,发现给定一个锯齿型的边界,如果其是某条直线的一部分,那么这条边界直线的斜率的范围是有限的,从而可以对该边界直线形成的像素阶跃形态进行预测,快速地确定下一个阶跃边界可能出现的位置,进一步将边界直线的两个端点向前延伸,从而重建出与全局中实际边界更加相符的边界信息(图 1 左,直线).基于这样的思想,本文提出了边界斜率预测的形态学反走样算法,通过实验结果的验证,本文的方法更加准确地重建了图像边界,进一步改善了直线边界的颜色过渡,使倾斜直线边界具有更高的连续性,获得了更好的反走样效果.

## 1 相关工作

传统的反走样算法——超采样反走样和多重采样反走样算法,在十几年的时间里一直是反走样算法的标准.超采样反走样在一个像素内部进行多次采样,然后计算多次采样的颜色的平均值.多重采样反走样的思想和超采样的思想较为相似,通常被认为是超采样反走样的一种特殊情况.根据 OpenGL 的定义[7],多重采样反走样是指对超采样反走样的一种优化,一般是指最终图的某些组成部分没有经过超采样,比如只对深度缓存和模板

缓存进行超采样,作为这个像素的最终颜色.然而,由于渲染效率和资源占用上的诸多不足,再加上与目前流行的推延渲染的冲突,已经不能适应时代的要求.尽管最近一些相关的技术,比如 CSAA[8]和 EQAA[9]通过把颜色、深度和样本分离减少了带宽和存储的消耗,但这些方法仍然不能避免超采样反走样的缺点.

Yang 等人的基于过滤器的反走样算法[10]是在反走样领域上的一个很重要的进展,他们的有向可调整边界反走样过滤器是基于 GPU 的实现方法,达到了超采样反走样的灰度变化水平.他们的做法是用等值线穿过一个像素的长度作为亚像素的权重,这种做法可以达到采样数为其 2~3 倍的传统做法才能达到的效果.该方法认为,对图像中每个像素都进行多重采样反走样是没有必要的,反走样最严重的地方一般只出现在几何边界附近,因此只需要在这些边界处进行反走样处理.在每个需要反走样的区域,该算法依旧沿用了传统的多重采样反走样算法.虽然其最终的渲染效果并没有优越于原始的反走样算法,但其过滤器的思想为反走样算法指明了另一个值得改进的方向.

Reshetov 的形态学反走样(MLAA)[2]的提出也证明了在没有多重采样的帧缓存上,通过边界重建和面积计算也能够得到很不错的反走样结果.形态学反走样是一种后处理反走样算法,需要重建因为采样不足而在渲染过程中丢失的信息.简单来说,该算法将图像的局部边界分为 U 型、L 型和 Z 型,然后根据不同的线型推测边界的原始形态.理论上,形态学反走样可以分成以下 3 步:(1) 边界检测;(2) 计算像素混合权重;(3) 与临近像素混合.在 Reshetov 的形态学反走样算法之后,出现了不少形态学反走样算法,但基本上都遵循了这个框架,也包括本文的算法.

Reshetov 的形态学反走样算法是基于 CPU 的,缺乏实用性,这也促使不少研究人员提出了基于 GPU 的实现方法,包括 Jimenez[10],Biri[11],AMD 公司[12]等等.此外,NVIDIA 公司的 Timothy 提出的快速近似反走样(FXAA)[3]算法充分利用了硬件资源,比如,快速近似反走样会利用一种非均质图来在线段查找过程中估计线段的终点.但是以上几种算法都没有提出处理倾斜直线边界的反走样问题的解决方法.Jimenez 的改良亚像素形态学反走样(SMAA)[1]也是基于 GPU 实现的形态学反走样算法,文献[1]首次提出了对倾斜图样的处理方法,对倾斜的边界进行了一定程度上的反走样处理.但是 Jimenez 等人的算法只考虑了接近 45°角的倾斜直线的边界反走样,而忽略了其他倾斜角度的处理.

当图像没有被多重采样时,一些亚像素特征的丢失是不可避免的.为了能够更好地重建图样的原始形态,一些科研人员也采用了在渲染中记录下必要信息的做法.Chajdas 等人提出了亚像素还原反走样法(SRAA)[6],在形态学反走样算法的基础上增加了一个可以保存亚像素特征的缓存区.Malan 提出了距离到边界反走样法(DEAA)[4],在渲染过程中,它会计算像素中心到三角形边界的距离,并把这些信息存在缓存区中,最后根据这些距离来得到混合的权重.与距离到边反走样算法的做法类似的还有 Persson 的几何后处理反走样(GPAA)[13]算法等.通过在渲染过程中得到的额外的信息,这些算法都可以增加反走样的准确性,在一定程度上更好地应对倾斜直线边界的走样问题.但是渲染过程中记录额外的数据需要额外的缓存.另外,这样的设计方式也让程序丢失了原始的形态学反走样后处理的特点,这无疑会给算法的实际应用带来不必要的困难.

本文是 GPU 实现的基于走样图形的后处理形态学反走样算法,沿用了形态学反走样算法边界检测和颜色混合的主要思想,对其中权重计算过程中的直线边界重建进行了改进,使重建的直线更加逼近全局中的真实直线.本文算法在增加很少额外计算的前提下,更加准确地重建了直线的边界信息.

## 2 边界斜率预测的形态学反走样方法

### 2.1 算法概述

形态学反走样(MLAA)[2]尝试估计原几何物体在像素中的覆盖面积.要准确地将一个三角形栅格化,必须要获得每个三角形在像素内的覆盖面积.在初次渲染时得到的是一张没有经过任何反走样处理的缓存图像,为了估计这个覆盖面积,该算法需要对缓存中的图画的边界进行再向量化处理.最后,根据三角形覆盖面积的比例将像素的颜色和附近的像素进行混合.

本文的算法遵循了形态学反走样算法的基本框架,分为 3 个步骤:(1) 边界检测;(2) 面积(混合权重)计算;

(3) 颜色混合.下面首先简要介绍边界检测与颜色混合的内容,具体算法细节详见文献[2];其后再详细说明本文在形态学反走样算法的核心步骤——权重计算上的改进内容.

边界检测是形态学反走样算法的第 1 步,没有检测到边界将不会进行反走样处理.图像中存在很多的信息可以用来做边界检测,其中包括颜色值、深度值、法向量值等等.本文主要是通过比较两个相邻像素的 Luma 值来确定边界是否存在,当相邻两个像素的 Luma 值相差大于某个阈值时就认为边界存在.对于一个像素,只需要检测其上边界和左边界,右边界和下边界的信息可以从对应的右侧和下方的像素得来.具体有关边界检测的算法参见文献[1].

颜色混合是形态学反走样的最后一步.形态学反走样算法在第 2 步时会根据覆盖面积计算出混合权重,在颜色混合阶段,再根据混合权重将像素颜色与附近像素进行混合作为最终的输出颜色.

### 2.2 权重计算与边界延伸

传统形态学反走样算法将边界的反走样计算分为两个方向——水平和垂直,分别进行.每一条边界线段都由两个端点定义,要修正不正确的线段本质上是要找到更加准确的线段的端点.传统的形态学反走样算法在遇到一个 Z 型(图 2,虚线)边界时会假设线段的端点分别是 $A$ 和 $B$,然而,$A$ 和 $B$ 很可能并不是直线的真实端点.这种检测方法无法准确反映倾斜直线边界的形态,将其用于反走样计算,可能会导致最终的渲染结果呈现倾斜方向的起伏效果(图 1 右).

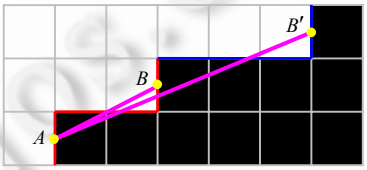


图 2 线段 $AB$ 是原始的传统形态学反走样算法的线段重建结果,本文的目标是将 $B$ 点修正到 $B'$点

改良亚像素形态学反走样算法(SMAA)[1]提出了一种解决倾斜方向的反走样方法.该算法受到传统形态学反走样算法中对正交图样的处理方法的启发,制定了对角线图样的处理机制.然而他们的算法只能处理 45°左右的倾斜直线.本文提出了一种更加高效、更加全面的解决方法.我们认为,在边界检测中出现的一段边界,很可能是一条更长的直线边界的一部分.本文的算法针对形态学反走样中对边界的不正确的估计做出了修正,使其更加逼近真实的边界.

下面以如何修正 $B$ 点位置为例,具体阐述边界斜率预测的形态学反走样方法.如图 2 所示,我们需要将端点 $B$ 修正到 $B'$位置.如果依次遍历虚色线上的 3 个像素,当阶梯的宽度增加之后,算法的代价将非常高.本文的做法是直接预测 $B'$点的位置,并检测 $B'$点位置的边界图样是否与原来 $B$ 点位置的边界图样相同,如果相同,就认为这条边界的右端点更可能是 $B'$而不是 $B$.

根据 Bresenham 直线绘制算法[14],一条斜率小于 45°的直线在屏幕上光栅化之后两个水平相邻的像素的 $y$ 坐标保持不变或者相差 1.要预测 $B'$的位置,本文引入如下定理:

**定理** 1. 一条直线边界,每个边界像素阶跃的宽度之差不会超过 1.

证明:用反证法来进行证明.首先,设阶梯的宽度为 $L$,如图 3 所示,线段的左端点在 $P_1$ 和 $P_2$ 之间,线段的右端点在 $P_3$ 和 $P_4$ 之间,那么阶梯的斜率 $k$ 满足下面的不等式:

$$\frac{1}{L+1}<k<\frac{1}{L-1}.$$

假设一条直线上的 3 个阶梯的宽度分别是 $L_1$,$L_2$ 和 $L_3$,它们分别都满足上面的不等式.

$$\frac{1}{L_1+1}<k<\frac{1}{L_1-1},$$

$$\frac{1}{L_2+1}<k<\frac{1}{L_2-1},$$

$$\frac{1}{L_3+1}<k<\frac{1}{L_3-1}.$$

不妨假设 $L_1<L_2$,即 $L_1+1\leqslant L_2\leqslant L_1-1$,取其倒数可得:$\frac{1}{L_1-1}\leqslant\frac{1}{L_2}\leqslant\frac{1}{L_1+1}$,将该不等式右边代入第 1 条不等式,左边代入第 3 条不等式,可得:

$$\frac{1}{L_2}<k,\ k<\frac{1}{L_2}.$$

这样的 $k$ 不存在. □

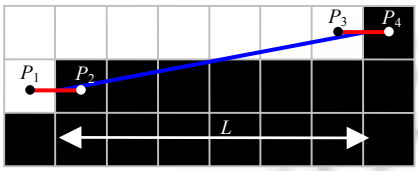


图 3 阶梯的宽度为 $L$,那么线段的左右端点肯定局限在了 $P_1P_2$,$P_3P_4$ 线段的区域内

再根据定理 1,在边界检测中得到了一个阶梯的阶跃宽度是 $L$ 之后,如果这个阶梯是某条直线的一部分,接下来的阶跃的宽度只可能是 $L$–1 或者 $L$+1.换言之,如果出现 3 个锯齿的阶梯宽度分别是 $L$,$L$–1 和 $L$+1,那么这 3 个锯齿肯定不会是在一条直线上.

**算法实现**

假设某个像素的上边界或下边界存在一条边(这里提到的边是像素层面上的边,而不是真实的物体的边界),首先需要沿着这条边找到其到左端点和右端点的距离,假设分别为 $D_{\text{left}}$ 和 $D_{\text{right}}$(如图 4 所示).同样地,如果这个像素的左边界或者右边界存在一条边,也需要沿着这条边找到其到上端点和下端点的距离,假设分别为 $D_{\text{up}}$ 和 $D_{\text{down}}$.图中 $A$ 点是当前像素所在位置;沿水平方向查找到左端和右端的距离 $D_{\text{left}}$ 和 $D_{\text{right}}$,这个阶梯的阶跃宽度为 $L=D_{\text{left}}+D_{\text{right}}+1$;预测下一个阶跃宽度为 $L$,$L$–1 或者 $L$+1;比较新的可能的端点位置处的线型(右上方框)和原端点处的线型(中间方框).

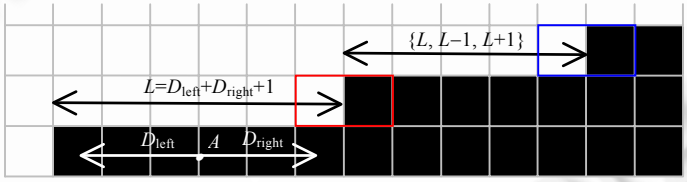


图 4

接下来需要先比较 $D_{\text{up}}+D_{\text{down}}$ 和 $D_{\text{left}}+D_{\text{right}}$ 的大小,如果 $D_{\text{up}}+D_{\text{down}}$ 较大,可以判断,这个像素周围的边界的分布是趋于竖直的(如图 5 中的位置 1);如果 $D_{\text{left}}+D_{\text{right}}$ 较大,则可以认为,这个像素周围的边界是趋于水平的(如图 5 中的位置 2).

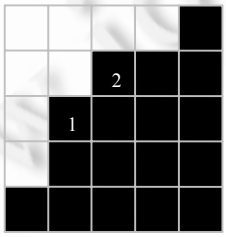


图 5 位置 1 附近的像素边界是趋向于垂直的,位置 2 附近的像素边界是趋于水平的

假设线型以水平方向为主,需要先计算出其宽度 $L=D_{\text{left}}+D_{\text{right}}+1$,然后向左向右分别对直线进行延伸,试图找到更加准确的端点.以向右查找为例,基于前面的理论可知,下一个阶跃宽度可能是 $L$,$L$–1,$L$+1.先后检验这 3

个位置的边界信息是否与之前的一个右端点处的线型一样(如图4中的方框所示).当然,由于受到计算能力的局限,需要预先设定一个循环次数上限$N$.如果$L$,$L$–1,$L$+1这3个位置都没有和原端点位置的边界线性一致,则可认为该阶梯不是长直线的一部分,结束查找.如果检测到新的阶跃宽度为$L'$($L$或者$L$–1),可知下一个阶跃宽度只可能是$L$或$L''$.

程序伪代码如下:

```
x=p
for i=0 to 循环次数上限 N
    if 像素位置 x+L 处为线段端点
        x=x+L
        继续循环
    else if 像素位置 x+L–1 处为线段端点
        x=x+L–1, L'=L–1,跳出循环
    else if 像素位置 x+L+1 处为线段端点
        x=x+1, L'=L+1,跳出循环
    else
    return x
end for
for j=0 to N–i
if 像素位置 x+L 处为线段端点
    x=x+L
    继续循环
else if  像素位置 x+L'处为线段端点
    x=x+L'
    继续循环
else
    return x
end for
```

线型趋于竖直方向和水平方向计算类似,这里不再赘述.

代码第1步将当前像素位置$x$初始化为需计算反走样的像素$p$.在第1个循环里依次检测{$L$,$L$–1,$L$+1}这3个阶跃宽度的像素是否为边界端点,当发现端点位置宽度为$L$时,继续循环;当发现端点位置宽度为$L$–1时,那么该直线阶跃宽度确定为$L$或$L$–1,因此我们将$L$–1赋值给$L'$,跳出循环;当发现端点位置宽度为$L$+1时,那么该直线阶跃宽度确定为$L$或$L$+1,因此我们将$L$+1赋值给$L'$,跳出循环;当发现3个相邻像素都不是边界端点时,结束端点检测,返回当前端点位置$x$.

确定直线阶跃为$L$与$L'$两种宽度后,程序进入第2个循环,分别检测在$L$或者$L'$两个阶跃宽度的像素是否为线段端点.如果是端点,那么继续循环,如果不是,那么结束端点检测,返回当前端点位置$x$.

**算法复杂度分析**

由上述伪代码可以看出,第1个循环中最多进行1次3个端点的检测,在第2个循环中最多进行2次端点的检测.端点检测时,我们采用文献[1]中形态学边界检测的方法,一次纹理采样可以获得两个相邻像素的边界端点信息.因此,我们的方法最多需要的采样次数为$N$+2,算法复杂度为$O(N)$.如果不采用本文提出的基于斜率预测的端点检测的方法,采用边界逐像素端点检测方法.同样,按照沿着阶梯边界每两个像素进行1次纹理采用计算,线段阶跃平均宽度为$L$的情况下,共需进行$L/2{\cdot}N$次采样,算法复杂度为$O(L{\cdot}N)$.从算法复杂度的比较上来看,本文提出的算法具有更优的算法效率.

## 3 结 果

本文在实验平台为 Intel Core2 的 CPU(2G 内存)、NVIDIA GeForce GTX560ti 的 GPU(1G 显存)上验证了我们的算法.在所有结果中,本文的结果图渲染采用的都是 4 次循环次数上限,亚像素还原形态学反走样算法选用的是 SMAA1x 的像素反走样配置[1].每增加 1 次循环,我们的方法沿着直线的两个方向各向前搜索 1 次,在当前的实验平台上,增加 1 次循环额外增加的代价为 0.07ms.

我们在多个场景上验证了本文提出的算法.由于形态学反走样算法为图像后处理方法,我们在图 6 和图 7 中示意了以绘制三维几何体获得的帧缓存为原始图像输入的反走样效果.图 8 和图 9 示意了从文件直接导入复杂场景的绘制图片作为原始图像,并对该图像进行反走样处理的结果.由于本文算法的操作都是基于像素的,建议读者放大图片进行比较.

图 6 为绘制三维场景的反走样结果,场景为倾斜角度不一的 5 根圆柱,我们比较了多采样反走样算法(MSAA4x)、形态学反走样算法(MLAA)、亚像素还原形态学反走样算法(SMAA1x)以及本文算法的结果.由图 6 中圆柱的边界可见,形态学反走样算法(MLAA)和亚像素还原形态学反走样算法(SMAA1x)的渲染结果都在直线边界处出现比较明显的起伏,本文的算法通过对直线边界的斜率修正,在很大程度上改善了渲染效果.

为进一步比较上述几种算法,我们展示了图 6 场景类似的三维场景的局部放大图.如图 7 所示,其中,上框、中框和下框分别是 1 倍、4 倍和 8 倍的显示结果,从最左边放大 8 倍的未经过反走样的图中可以看出,直线边界阶梯的阶跃的宽度在 2 个像素和 3 个像素之间变化,其中位置 2 阶跃宽度是 3 个像素.观察图中位置 1 和位置 3,可以发现其他算法在位置 1 的像素颜色略浅,在位置 3 像素颜色略深,这是造成渲染结果在位置 2 出现起伏的原因.通过本文的修正,位置 1 颜色加深,位置 3 颜色变浅,从而使直线边界显得更加平滑.

图 8,图 9 给出我们比较的本文算法对复杂场景倾斜直线边界的反走样效果,方框中的显示结果为图中对应位置放大后的显示结果.从图 8 中可以看到,在岩石的下边界和绳索的边缘的反走样上,我们方法获得的结果显得更加平滑,而另外两种方法出现起伏比较明显.图 9 为另外两个复杂场景的反走样结果比较,对比不同方法的广告牌下边界和屋子左侧屋顶边界的反走样效果,本文算法的处理结果也更加平滑与自然.

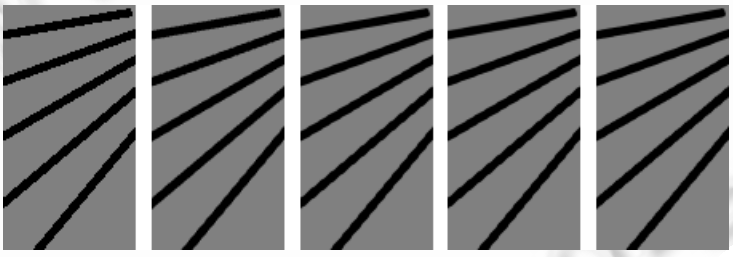


图 6 不同斜率的边界反走样结果比较

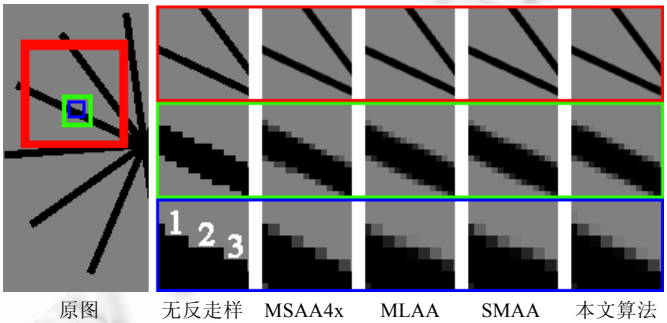


图 7 左大图是未经过处理的原始图,右边上下分别是对应原图中的小方框、中方框和大方框区域内的反走样结果,右图上方框区域没有经过放大,中间和下面方框区域分别是放大 4 倍和 8 倍的结果

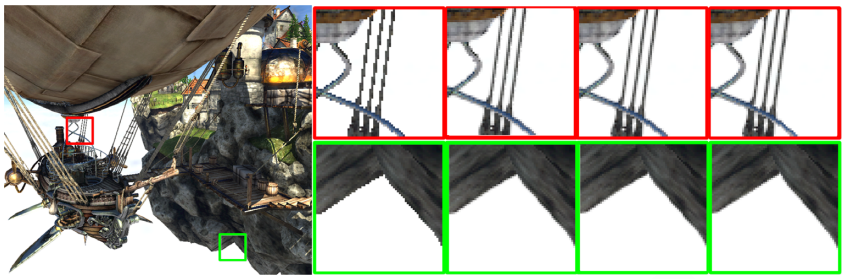


图 8 复杂场景反走样结果(1),原始图片来自文献[1]

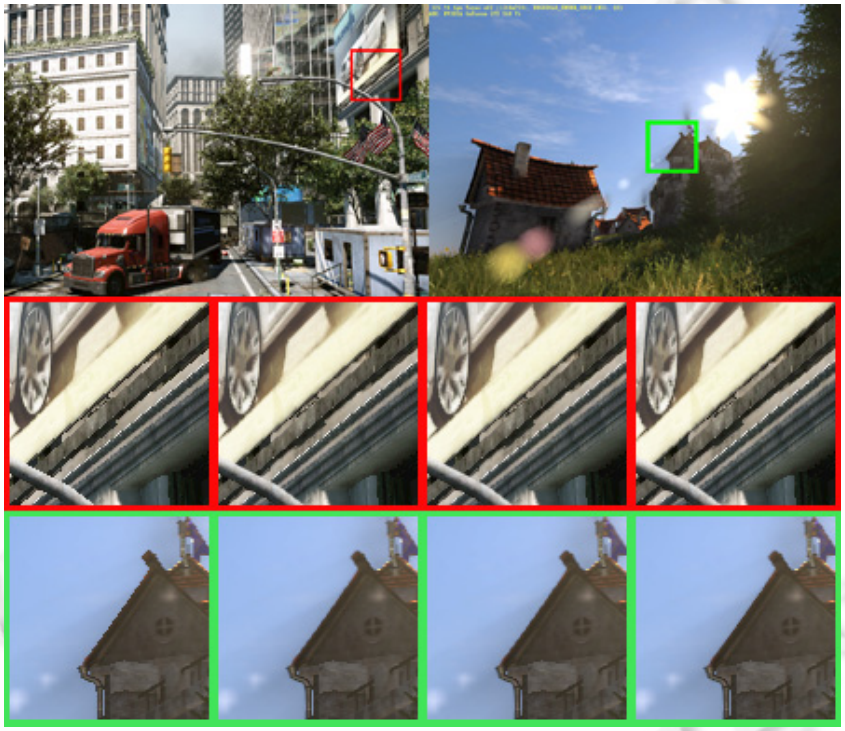


图 9 复杂场景反走样结果(2),原始图片来自文献[1]

## 4 结 论

本文针对形态学反走样的不足,提出了利用直线的斜率对直线进行准确的端点修正的方法,本文算法在增加了少量计算代价的同时,改进了形态学反走样在处理倾斜直线边界上的不足,提出了基于倾斜直线斜率的形态学反走样方法,该方法使形态学反走样算法的反走样效果更好,这必定能使形态学反走样算法的应用得到进一步的推广.

**References**:

[1] Jimenez J, Echevarria JI, Sousa T, Gutierrez D. SMAA: Enhanced subpixel morphological antialiasing. In: EUROGRAPHICS. 2012.

[2] Reshetov A. Morphological antialiasing. In: Proc. of the Conf. on High Performance Graphics 2009. 2009. 109−116.

[3] Lottes T. FXAA: Fast approximate anti-aliasing. Technical Report, NVIDIA, 2011.

[4] Malan H. Distance-to-Edge anti-aliasing. In: Proc. of the ACM SIGGRAPH Course. 2011.

[5] Andreev D. Anti-Aliasing from a different perspective. In: Proc. of the Game Developers Conf. 2011.

[6] Chajdas MG, Mcguire M, Luebke D. Subpixelrecontruction antialiasing for deferred shading. In: Proc. of the Symp. on Interactive 3D Graphics and Games. 2011. 15−22.

[7] OpenGL 1.5 specification, section F.3. http://www.opengl.org/registry/doc/glspec15.pdf

[8] Young P. Coverage sampled antialiasing. Technical Report, NVIDIA, 2006.

[9] AMD: EQAA modes for AMD 6900 series graphics cards. Technical Report, 2011.

[10] Jimenez J, Gutierrez D, Yang J, Reshetove A, Demoreuille P, Bergpoff T, Perthuis C, Yu H, Mcguire M, Lottes T, Malan H, Persson E, Andereev D, Sousa T. Filtering approaches for real-time antialiasing. In: Proc. of the ACM SIGGRAPH Courses. 2011.

[11] Biri V, Herubel A, Deverly S. Practical morphological antialiasing on the GPU. In: Proc. of the ACM SIGGRAPH 2010 Talks (2010), SIGGRAPH 2010. 2010. 45:1.

[12] AMD: Morphological anti-aliasing. 2010. http://sites.amd.com/us/game/technology/Pages/morphological-aa.aspx

[13] Person E. Geometric post-process anti-aliasing. 2011. http://www.humus.name/3D/GPAA.zip

[14] Bresenham JE. Algorithm for computer control of a digital plotter. IBM Systems Journal, 1965,4(1):25−30.

**钟宇琛**(1989－),男,广东湛江人,本科生,主要研究领域为实时渲染.

**王锐**(1978－),男,博士,副教授,主要研究领域为实时绘制,真实感绘制.