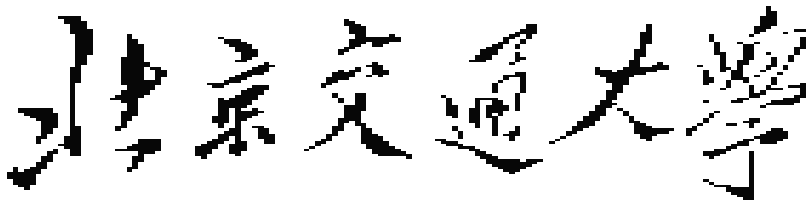


学号	姓名	论文规范性 (10)	问题分析与调研 (30)	方案创新性 (20)	实验结果分析与讨论 (40)	结课论文总成绩 (100)
21301128	贾东原	8	25	16	34	83



游戏中常见的抗锯齿图像增强技术比较

Comparison of Common Anti-Aliasing Image Enhancement Technologies in Games

学 院： 软件学院

专 业： 软件工程

学生姓名： 贾东原

学 号： 21301128

指导教师： 吴雨婷

北京交通大学

2024 年 6 月

## 中文摘要

随着计算机图形学和高质量显卡的不断发展，有越来越多的图像增强技术应用于游戏领域甚至于影视领域，从基础的 MSAA 到如今 DLSS3.5（最近更新是 dlss3.7.0，但和 dlss3.5 没啥差别），这将使得支持这些技术的游戏在视觉效果和性能上得到显著提升，不管是游戏画面和游戏帧率上，都越来越有质的飞跃，为玩家提供更加沉浸式的游戏体验. 本文将简单实现 MSAA, MFAA, SMAA, DLSS 等抗锯齿技术对图片进行简单处理，并比较这些抗锯齿技术之间的优劣性。需要注意的是，不同的抗锯齿技术适用于不同的场景和需求，不存在真正的孰好孰略。。

**关键词：**抗锯齿技术、计算机图像学、显卡、游戏、MSAA、MFAA、SMAA、DLSS

## 目 录

中文摘要.....	i
目 录.....	ii
1 引言.....	3
1.1 项目背景.....	3
1.2 国内外研究现状.....	3
1.3 项目工作内容和组织结构.....	4
2.实验准备工作.....	4
2.1 实验环境和准备工作.....	4
2.2 代码库作用.....	4
3 代码实现.....	5
3.1 MSAA 和 MFAA 实现.....	5
3.1.1 MSAA.....	5
3.1.2 MFAA.....	5
3.3 SMAA 实现.....	5
3.4 DLSS 实现.....	6
4 实验结果和结论.....	7
5 总结与展望.....	9
5.1 全文总结.....	9
参考文献.....	10

## 1 引言

本章着重介绍该技术的应用背景、发展现状和实用价值，同时介绍本项目的主要工作及论文组织结构。

### 1.1 项目背景

随着大家对于计算机视觉的不断探索和对于高画质游戏和场景的需求提高，抗锯齿技术应运而生，然而由于设备性能的瓶颈，我们很少能直接实现原生画质下的高分辨率和高帧率，而抗锯齿技术在计算机图形学中的重要性主要体现在改善图像质量和视觉体验两个方面，因此抗锯齿技术在计算机图形学中的重要性毋庸置疑。

### 1.2 国内外研究现状

目前的抗锯齿技术已经经历了多个阶段的发展，主要包括传统的硬件抗锯齿技术和最近的基于深度学习的方法。传统抗锯齿技术有 MSAA (Multisample Anti-Aliasing)、CSAA (Coverage Sampled Anti-Aliasing, NVIDIA 引入的一种变种，通过改进采样覆盖率来提高效果)、EQAA (Enhanced Quality Anti-Aliasing, AMD 的一种扩展，类似于 CSAA，能提供更高质量的图像输出)。基于后处理的抗锯齿技术：FXAA, SMAA (Subpixel Morphological Anti-Aliasing): 其中 SMAA 是一种结合了 MLAA (Morphological Anti-Aliasing) 和 SRAA (Subpixel Reconstruction Anti-Aliasing) 的技术，提供了更好的图像平滑度和细节保留。

目前最火的是基于深度学习的超采样技术，即 DLSS (Deep Learning Super Sampling)，这是一个由 NVIDIA 引入的通过深度学习模型在运行时生成图像，从低分辨率图像生成高分辨率图像，同时保持高帧率的技术，dlss3.5 是许多游戏玩家的福音，让英伟达许多中端显卡。此外还有一些新兴的抗锯齿技术如：TAA (Temporal Anti-Aliasing)、MLAA (Morphological Anti-Aliasing)

### 1.3 工作内容介绍

本文选择了传统抗锯齿技术、基于后处理的抗锯齿技术、基于深度学习的超采样技术的代表方法调用 pycharm 中的库或英伟达训练好的模型进行简单比较。准备一些适

合测试的图片、游戏截图，包括包含边缘和细节的图像，以便能够清楚地观察每种技术的效果。 本文将从实验准备工作、代码实现、实验结果与分析、结论等方面进行展开说明

## 2 实验准备工作

本章将讲述调用 `pycharm` 中相关库和训练好的模型对图片进行处理，比较图片的抗锯齿化后的效果。

### 2.1 实验环境和准备工作

编译器: `pycharm`

编程语言: `python`

相关库: `torch`、`torchvision`、`numpy`、`requests`、`os`

Dlss 模型:

```
url'https://github.com/xinntao/Real-ESRGAN/releases/download/v0.2.5.0/realesr-general-x4v3.pth'
```

```
model_path = 'realesr-general-x4v3.pth'
```

### 2.2 代码库作用

`torch` 和 `torchvision`:

`torch` 是一个深度学习框架，用于构建和训练神经网络模型。`torchvision` 提供了一些计算机视觉相关的功能，例如预训练模型、数据转换和数据加载等。

`numpy`:

`numpy` 是一个用于科学计算的核心库，提供了多维数组对象和一系列处理数组的函数。

`requests`:

`requests` 是用于发送 HTTP 请求的库，用于从网络上获取数据或与网络服务进行交互。本文为了 GitHub 下载 dlss 模型

`os`:

`os` 提供了与操作系统交互的功能，例如文件操作、路径操作等。

## 3 代码实现

### 3.1 MSAA 和 MFAA 实现

#### 3.1.1 MSAA

MSAA (apply\_msaa):

image.resize: 使用最近邻插值将图像放大 sample\_count 倍。模拟 MSAA 中将图像增大以增加采样点的效果。

msaa\_image.filter: 对放大后的图像应用高斯模糊。这一步用于混合多个采样点的颜色值，减少锯齿和颜色波动。

msaa\_image.resize: 最后，将图像缩小回原始尺寸，并使用 Lanczos 滤波器进行重采样。

下面函数相同但是参数不同。

#### 3.1.2 MFAA

image.resize: 直接使用 Lanczos 滤波器将图像放大 sample\_count 倍。模拟 MFAA 中增加多个样本点以提高图像质量的效果。

mfaa\_image.filter(ImageFilter.GaussianBlur: 对放大后的图像应用高斯模糊，类似于 MSAA 中的处理，以混合多个样本点的颜色。

mfaa\_image.resize: 最后，将图像缩小回原始尺寸，并再次使用 Lanczos 滤波器进行重采样。

### 3.2 SMAA 实现

Apply\_smaa 思路: 代码首先检查图像模式并转换，检查图像是否为 RGB 模式；如果不是，则将其转换为 RGB，然后获取图像的宽度和高度，以便在后续处理步骤中使用。RGB 通道分离将图像分离成单独的红色、绿色和蓝色组件。对 R、G、B 三个通道分别应用 process\_channel 函数。处理后，将单独的通道重新组合成一个 RGB 图像，函数返回处理过的 RGB 图像。

process\_channel(channel) 内部函数接受一个颜色通道作为输入。使用边缘增强滤镜对通道进行锐化处理。将锐化的图像转换为 numpy 数组，通过嵌套循环遍历通道的像素（不包括边界像素）。对于每个像素，它比较像素的强度与其直接邻居（上、下、左、

右) 的强度。如果当前像素的强度大于所有邻居的强度，它将被替换为上方像素的强度。这种操作不直接对应于 SMAA 技术，而是根据强度比较简化边缘，这可以以基本的方式帮助减少锯齿效应。

### 3.3 DLSS 实现

首先我们先简单定义了一个 CNN 模型 SimpleCNN，用于加载权重、下载模型和应用深度学习超分辨率（DLSS）技术来增强图像。

load\_weights 函数将预训练的权重加载到模型中。主要参数有 model：加载权重的模型实例。state\_dict：一个包含权重的字典，由 PyTorch 的 torch.load() 函数加载得到。

过程中我们遍历权重字典，如果存在于模型的状态字典中，使用 model.load\_state\_dict(model\_state\_dict) 来更新模型的权重。download\_model 函数从网上下载模型文件。model\_url：模型文件的下载 URL、model\_path：本地保存模型的路径。

最终 apply\_dlss 函数进行初始化 SimpleCNN 模型、加载模型权重，定义图像的预处理流程，包括调整图像大小和转换为张量。对图像进行处理后，将结果转换回图像格式。

```
def apply_dlss(image, model_path='realesr-general-x4v3.pth'):
    model = SimpleCNN()
    state_dict = torch.load(model_path, map_location=torch.device('cpu'))
    load_weights(model, state_dict)

    preprocess = transforms.Compose([
        transforms.Resize((32, 32)),
        transforms.ToTensor(),
    ])

    # Convert the image to RGB format
    image = image.convert('RGB')
    img_tensor = preprocess(image).unsqueeze(0)

    model.eval()
    with torch.no_grad():
        output = model(img_tensor)

    output_image = Image.fromarray((output.squeeze().cpu().numpy().transpose(1, 2, 0) * 255).astype(np.uint8))
    return output_image
```

#### 4.实验结果和结论。

各个方法最终得各个方法最终得到的图片大小和清晰度、对比如图












 dabing.png	2024/4/8 16:22	PNG 文件	937 KB
 dabing_dlss.png	2024/6/14 16:33	PNG 文件	227 KB
 dabing_mfaa.png	2024/6/15 10:38	PNG 文件	590 KB
 dabing_msaa.png	2024/6/15 10:38	PNG 文件	586 KB
 dabing_smaa.png	2024/6/14 16:36	PNG 文件	695 KB

图 4-1 效果对比图

通过牙齿毛发等细节可以看出，在相同分辨率下，不管是 MSAA 、MFAA、SMAA、DLSS 都可以对图片显示效果进行提升，同时降低文件的大小。

而对于三类不同的方法，其中基于深度学习的 DLSS 显示效果最好，文件也小，由此我们可以看出从硬件抗锯齿技术到 DLSS 方法，技术越来越先进，我们可以在获取更高的图像显示效果。

## 5 总结与展望

### 5.1 全文总结

MSAA 和 MFAA 适合于传统游戏和固定分辨率显示的需求，SMAA 则在平滑度和性能之间取得了一定的平衡，而 DLSS 则代表了未来抗锯齿技术发展的趋势，通过深度学习提供了更高质量的图像输出，是许多 3A 游戏的选择。选择合适的技术取决于应用的环境、性能需求以及用户对图像质量的要求。总的来说，MSAA 是一种较旧但仍广泛使用的抗锯齿技术，适合处理简单的几何边缘锯齿，但性能消耗较大。SMAA 提供了更好的性能优化，并能处理更多类型的边缘和场景复杂性，是一种更为现代且灵活的解决方案。DLSS 利用 AI 来改善图像质量和性能，适用于最新的游戏和高要求的图形应用，是最先进的选择之一

同时，作为一个学生，平时笔记本使用较多，也不会有太高端的显卡，相比于光线追踪技术，我觉得 dlss 这样的抗锯齿技术更能提高我们游戏与生活的质量，我们能在低端一些的显卡上（前提是得支持相关的技术，如 dlss3 只有 20 系及以上的显卡才支持）获得更好的游戏体验

## 参考文献

- [1] 图像缩放中抗锯齿单元的设计[J]. 郭潇蔚;王学水;于岗. 电脑知识与技术, 2012
- [2] MFAA 多帧采样抗锯齿简单测试[J]. 格伦. 个人电脑. 2014, 20(12) [J]. :91-93
- [3] 基于 MSAA 的控件识别应用研究[C]. 2010 国际计算机科学技术与应用论坛论文集, 西南财经大学信息技术应用研究所, 2010-12-10, 中国广西南宁. 信息科技, 计算机软件及计算机应用, TP311. 52.
- [4] 后处理反走样技术综述. 杜慧敏, 杜琴琴, 季凯柏, 蒋怵怵, 郭冲宇. 西安邮电大学学报, 21(01), 8-15, 2016. doi:10.13682/j.issn.2095-6533.2016.01.002.



