浙 江 理 工 大 学

毕业设计(论文)诚信声明

我谨在此保证：本人所写的毕业设计(论文)，凡引用他人的研究成果均已在参考文献或注释中列出。设计(论文)主体均由本人独立完成，没有抄袭、剽窃他人已经发表或未发表的研究成果行为。如出现以上违反知识产权的情况，本人愿意承担相应的责任。

声明人(签名)：

年 月 日

摘 要

随着网络技术的迅速发展，网络信息传输速度的进一步提高以及5G通信时代的到来，在游戏、购物等领域，单纯的网络二维交互性已经不能满足人们的需要了。近些年来，虚拟现实技术已经得到较广泛的应用。为此，提出构建基于WebGL的虚拟现实购物系统，在Web端进行商品的三维立体展示，模拟真实光照环境，展示商品的原貌，给用户以独特的视觉体验。

本文主要描述虚拟现实购物系统的搭建、模拟真实光照进行商品的三维立体展示以及商品模型的搭建和基于图像的后处理抗锯齿算法的实现与比较。对于虚拟现实系统的搭建主要采用Bootstrap前端框架，实现对网页的优化和美化。采用Blender建模软件构建商品模型，并进行纹理设置，导出OBJ、FBX等格式文件。Three.js则是通过对WebGL API进行封装与简化而形成的易用图形库。Three.js对WebGL 进行了良好的封装，简化了细节，却保留了WebGL良好的灵活性。通过Three.js 框架加载模型，完成灯光、相机设置，最后将商品模型渲染在网页上。对于绘制图像的边缘锯齿化现象，采取了两种不同的算法加以消除，实现FXAA算法和SMAA算法并加以比较。

关键词：WebGL；Three.js；3d模型；抗锯齿；FXAA；SMAA

**Abstract**

With the rapid development of network technology, the further enhancement of network information transmission speed and the arrival of 5G communication era, in the field of games, shopping and other fields, the simple network two-dimensional interaction cannot meet the needs of people. In recent years, virtual reality technology has been widely used. To this end, we propose to construct a virtual shopping system based on WebGL and carry out 3D display of goods on the Web side, simulate real illumination, and display the original appearance of commodities.

This paper focuses on the construction of the virtual reality shopping system, simulate the real light model for three-dimensional stereoscopic display of goods and goods of structures, and based on image post-processing the anti-aliasing algorithm implementation and comparison. The construction of virtual reality system mainly adopts the Bootstrap front-end frame to optimize and beautify the web page. Using Blender modeling software to build the commodity model, and the texture setting, export OBJ, FBX and other format documents. Three.js is an easy-to-use graphics library which is formed by encapsulation and simplification of WebGL API. Three.js makes good encapsulation of WebGL, simplifies the details, but retains the good flexibility of WebGL. Load the model through the Three.js frame, complete the lighting and camera Settings, and finally render the product model on the web page. In this paper, two different algorithms are adopted to eliminate the edge sawing phenomenon, and the FXAA algorithm and SMAA algorithm are implemented.

**Keywords:** WebGL；Three.js；3D Model；Anti-aliasing；FXAA；SMAA

**目 录**

摘 要

Abstract

[第1章 绪论 1](#_Toc387946909)

[1.1 研究目的与意义 …](#_Toc3302)

[1.2 国内外研究现状](#_Toc1946)

[1.2.1 基于显著性区域检测的图像分割算法 2](#_Toc20884)

[1.2.2 自适应分割算法 4](#_Toc22477)

[1.2.3图割算法 5](#_Toc22384)

[1.3 研究内容与论文结构](#_Toc24426)

[第2章 显著区域检测分割算法与自适应分割算法的原理](#_Toc12513)

[2.1显著区域检测分割算法](#_Toc3142)

[2.1.1基于频谱的FT算法 7](#_Toc16357)

[2.1.2基于直方图对比度的HC算法 8](#_Toc29713)

[2.1.3 基于区域对比度和图的RC算法 9](#_Toc18724)

[2.1.4 基于全局对比度的LC算法 10](#_Toc6420)

[2.1.5 基于剩余频谱的SR算法 11](#_Toc4092)

[2.2自适应分割算法](#_Toc14983)

[2.2.1最大直方图熵阈值分割法（ENT） 12](#_Toc722)

[2.2.2最大类间方差法（OSTU） 13](#_Toc12576)

[第3章 基于显著区域检测的分割算法的设计与实现](#_Toc13273)

[3.1显著区域检测算法的实现](#_Toc22837)

[3.1.1 FT算法 15](#_Toc7482)

[3.1.2 HC算法 17](#_Toc16079)

[3.1.3 RC算法 18](#_Toc13143)

[3.1.4 LC算法 2](#_Toc9047)0

[3.1.5 SR算法 22](#_Toc7739)

[3.1.6 五种显著性区域检测算法的对比 24](#_Toc29137)

[3.2自适应分割算法的实现](#_Toc31608)

I

[3.2.1最大直方图熵阈值法 25](#_Toc7611)

[3.2.2最大类间方差法 26](#_Toc21501)

[3.2.3两种图像分割算法的对比 28](#_Toc2564)

[3.3基于显著性区域检测算法的图像分割算法的实现](#_Toc3508)

[第4章 基于显著性检测的图像分割软件GUI设计与应用 3](#_Toc16581)1

[4.1基于显著性检测的图像分割GUI的设计 3](#_Toc7904)1

[4.2基于显著性检测的图像分割软件的应用](#_Toc14737)

[第5章 总结与展望](#_Toc21099)

[5.1 总结](#_Toc5236)

[5.2 展望](#_Toc29374)

参考文献…………………………………………………………………………………………….38

[致 谢 4](#_Toc1266)1

[附 录](#_Toc27461)

II

1. 绪论

## 研究目的与意义

随着网络信息技术的迅速发展，以及5G网络将在2020年的推广商用，在网络娱乐、购物等多领域，仅仅单纯的网络二维空间的交互性已经不能满足人们的需要。而就目前的网络购物而言，小到铅笔、牙刷，大到车辆、家用电器均可在各大网购平台购买。2017年十一月十一日狂欢购物节，开场11秒，交易金额超十亿，3分01秒成交额超破百亿。由此可见，网络购物已然成为趋势。网络购物，不仅简化了购物模式，也大大节省了买卖双方的时间和空间成本。目前的网络购物方便、快捷，但在购物的过程中，消费者只能通过商家的文字和图片信息来获取真实的商品信息，是无法直接感受和触摸到真实的商品，仅仅是意识上的接触，而且这些文字和图片信息是由商家提供的带有一定的诱惑性，从而导致买家无法获取商品的真实、完整的信息，而造成对商品的误判。由此导致的商品尺寸、颜色以及质感的差异，频繁的退还货等问题是目前网络购物无法避免的痛点问题，给买家和卖家都带来一定程度的损失。

因而，针对这一信息不对称问题，提出建立虚拟现实购物平台。平台旨在实现商品的三维立体展示，尽可能的模拟商品在真实环境下的状态，给人以身临其境的时空感；旨在更好的展现商品，提供视觉效果上的全部信息参数。使得顾客更好的了解商品，获取更好的视觉体验。从而减少因色差、尺寸问题导致的退换货发生率，减少买卖双方的时间和空间成本，给顾客以良好的购物体验。

虚拟现实技术，是一种创建和体验虚拟世界的计算机系统。它是利用计算机系统实现多元信息融合、可交互的三维动态实景和行为的系统仿真，使得用户沉浸到虚拟构建的环境中。正是基于这种视觉现实性的体验，提出构建虚拟现实购物平台，提高顾客的购物现实性体验，减少因信息不对称问题而引起的一系列问题。

## 国内外研究现状

虚拟现实（Virtual Reality, VR）技术是综合应用各种技术制造逼真的人工模拟环境，模拟人在自然环境中各种感知行为的高级人机交互技术。虚拟现实设备正成为重要的购物平台和社交平台，虚拟现实技术将成为下一场革命。2014年以来，国际互联网巨头纷纷涉猎虚拟现实领域。我国从20世纪90年代初才开始研究VR技术，与发达国家相比，起步较晚从而存在一定的差距。随着系统工程和图形学等计算机技术的快速发展。VR技术已得到国家和社会的高度重视。Web3D（网络三维），是一种在虚拟现实技术的基础上，将现实世界中有形的物品通过互联网进行虚拟的三维立体展示并可互动浏览操作的一种虚拟现实技术。相比起目前网上主流的以图片、FLASH、动画的展示方式来说，Web3D技术让用户有了浏览的自主感，可以以自己的角度去观察，还有许多虚拟特效和互动操作。

而网络三维立体展示系统仍存在很多的局限，比如现今Web3D依然没有统一的标准，每种方案都使用不同的格式和方法。插件也是一个问题，目前所有的Wed3D都需要插件的支持，因为标准的不同，各个厂商的插件也是不尽相同，从几百K到几十M不等。基于Web架构的3D内容渲染对本地计算机和带宽都有很高的要求，也是限制了用户的要求。

### 1.2.1 虚拟现实技术的应用现状

阿里巴巴是马云在1999年成立的网络科技公司，涉及网络购物、云计算服务平台等领域，2014年在纽约证券交易所挂牌上市，2016年成为全球最大的零售交易平台。作为站在时代前沿的网络零售公司，阿里巴巴在3月推出虚拟现实购物系统BUY+，11月正式上线。而作为全球第一个VR购物系统，BUY+实现了瞬间前往纽约，东京、悉尼的热门商场进行可视化的实景购物，从进店到付款的全部过程，通过眼球的运动实现所有的购物操作。BUY+的起点是一间普通的房间，布置有七张壁画。这些壁画的地点就是你将要去的购物目的地。而随着场景的变换，让你足不出户，却可以实现在全世界的购物体验。

我国中视典公司积极运用先进的图形以及相关技术，研发出三维虚拟现实平台软件，在一定程度上打破了国外垄断的局面，以性价比较高的优势受到一些客户的喜爱，在当前已成为市场中占有率较高的一款虚拟现实软件，不仅能够满足消费者的购物心理，还能为消费者带来较多的便利。

### 1.2.2 虚拟现实的技术现状

WebGL（Web Graphics Library）是一种3D绘图协议，这种绘图技术标准允许把JavaScript和OpenGL ES 2.0结合在一起，通过增加OpenGL ES 2.0的一个JavaScript绑定，WebGL可以为HTML5 Canvas提供硬件3D加速渲染，这样就可以借助系统显卡来在浏览器里更流畅地展示3D场景和模型了，还能创建复杂的导航和数据视觉化。显然，WebGL技术标准免去了开发网页专用渲染插件的麻烦，可被用于创建具有复杂3D结构的网站页面，甚至可以用来设计3D网页游戏等等。

Three.js框架是一款运行在浏览器中的3D引擎，采用JavaScript编写而成。随着Html5的普及应用，网页表现能力更加强大，浏览器为WebGL提供了可用接口，可以通过调用对应的API进行3D图形的绘制，而Three.js在基础接口之上进行了二次封装。因此它具有了掩盖3D渲染细节、面向对象、功能强大丰富、速度快、扩展性强以及同时支持Html5和Canvas。Three.js辅助我们导出了不同的模型数据，辅助生产材质，自由地配置灯光等。

### 1.2.3存在问题和局限

目前仍有许多因素限制网络三维展示的发展、应用。到目前为止Web3D依然没有统一的标准，不同的方案都使用不同的格式和方法。目前的Web3D依然需要插件的支持。正是由于标准的不同，不同的厂商生产的插件也是不尽相同，甚至是差异巨大，插件规模小到几百K到达到几十M皆有。基于Web架构的3D渲染对本地计算机和带宽都有很高的要求，也是限制了网络三维立体展示的推广应用。而WebGL和Html5的使用推广可以解决此类问题。

阿里推出的BUY+虚拟现实购物系统，必须使用穿着特定的

**研究内容与论文结构**

第1章主要介绍本次设计的研究目的与意义，并且介绍Web3D发展历程及分类。

第2章主要介绍基于WebGL虚拟现实购物展示系统的搭建。

第3章详细介绍FXAA和SMAA算法。

第4章主要介绍FXAA和SMAA算法的实现和改进。

第5章对本文进行总结，对SMAA算法展望。

1. 基于WebGL的虚拟现实购物展示系统的搭建

本章主要介绍使用three.js框架快速搭建3D展示系统。

## 2.1 虚拟现实系统的协议规则和框架

### 2.1.1 WebGL绘图协议

WebGL（Web Graphics Library）是一种网页端的3D绘图规范。它巧妙地将JavaScript和OpenGL ES 2.0结合在一起，添加一个OpenGL ES 2.0的JavaScript的绑定，实现为HTML5 Canvas提供硬件3D加速渲染。OpenGL是操作系统下的独立的三维图形程序函数库。实际开发人员就可以使用计算机系统显卡在网页端流畅的进行3D场景和模型的展示，还可以进行复杂的导航和数据的可视化。WebGL技术标准的优势之一便是不在需要开发网页渲染插件的开发，可直接在网页端进行复杂3D结构的展示开发，甚至是直接开发web端网页游戏。

而于2010年发布的WebGL主要有两大优势：第一，它通过HTML页面本身实现和Web交互式三维动态展示，无需任何浏览器插件的支持。第二，它是通过统一、标准以及夸平台的OpenGL接口来调用底层的图形硬件加速功能而进行3D图形的渲染，达到在Web上显示逼真的3维立体图像。

WebGL绘制主要是通过获取顶点坐标、图元装配（绘制三角形）以及光栅化。我们使用从三维建模软件中导出的数据文件，如OBJ文件、FBX文件等。我们从中获取到模型的顶点信息进行图元装配，即进行三角形构建。使用三角形覆盖进行模型表面构建。在这个过程中，顶点着色器会先对顶点坐标进行转换。而后GPU（Graphics Processing Unit，图形处理器）进行图元装配。在图元装配完毕后，我们需要给3D模型“上色”，而帮我们完成这些工作的是运行在GPU的“片元着色器”，它进行模型颜色、质地、光照效果以及阴影等处理。最后，WebGL进行光栅化处理，即根据每个已经确定好片元的颜色，以及根据深度缓存区来判断该片元是否被遮挡，最终将片元的全部信息存储到颜色缓存区完成整个模型的渲染。

### 2.1.2 运行在网页端的3D引擎 ------ Three.js

Three.js是运行在浏览器中的3D引擎，是使用JavaScript编写的第三方库。众所周知，JavaScript是运行在网页端的脚本语言，故Three.js也是运行在浏览器中的。OpenGL是操作系统下的独立的三维图形程序函数库。WebGL是基于OpenGL设计的面向Web的图形标准，它提供了一系列的JavaScript API，通过这些API利用计算机硬件进行渲染，获得较高的性能。而Three.js则是通过对WebGL API进行封装与简化而形成的易用图形库。Three.js对WebGL 进行了良好的封装，简化了细节，降低了初学者的学习成本，依然保留了WebGL良好的灵活性。

对于Three.js，它没有规定计量单位，本质上是无单位的，故使用的唯一要求是进行归一化处理或者使用统一的测量单位，最后按照比例进行绘制成最终展现在Web端的3D图像。Three.js是遵循右手坐标系的，即以屏幕中心为原点，水平方向为x轴，竖直方向为y轴，垂直屏幕为z轴。在对模型进行旋转操作时，rotation正值代表逆时针旋转，负值表示顺时针旋转。Three.js主要有三大组件，场景（scene）、相机（camera）和渲染器（renderer）。场景即是物体容器，编程开发人员将自己需要的模型、物体等放入场景中。而相机的主要功能是面对场景，娶一个合适的角度进行拍摄。而渲染器（renderer）的功能则是将相机拍摄的图片放在网页中进行展示。在计算机的世界中，一条弧线是由有限点构成的有限线段连接得到的。当线段的数量多到一定的程度的时候，在我们的视觉系统中，折线段也就变成了圆滑的曲线了。类似于数学中极限，当折线段的数量趋于无穷时，那展现给我们的就是圆滑的曲线。而在计算机中的3D模型也是采用此原理，通用的做法是3维立体模型采用三角形组成的网格模型来进行描述，业界称之为Mesh模型。

著名的斯坦福兔子。（图片）

正如所见，随着三角形数量的增加，斯坦福兔子的表面更加准确和光滑。在Three.js中Mesh的函数构造器是：Mesh（geometry, material）。其中，geometry是模型的形状，而material则是模型的材质。材质是模型表面除了形状之外的所有可视属性的集合，例如色彩、纹理、光滑度以及产生阴影等等。

### 2.1.3 虚拟现实购物平台的搭建

本课题构建的虚拟现实购物平台，主要采用Bootstrap前端框架进行样式格式控制。Bootstrap（Web框架）是目前最受欢迎的前端框架之一，是由Twitter设计师开发的CSS/HTML开源框架，是基于HTML、CSS和JavaScript，简洁、灵活，使得Web开发更加便捷。Bootstrap提供了优雅的HTML和CSS规范，提供了基本的样式、Web组件以及jQuery插件，这些插件为Bootstrap赋予了“生命”。

平台的采用延时加载，实时获取当前用户的操作，对页面进行分步加载，只有当用户浏览到当前网页位置时，进行图片等资源进行加载。

作为商品展示平台，采用图片加模型的展示方式，用户可在主页选中商品查看详情，在商品详情页面进行3D立体模型的展示。主页设计为标题栏、内容栏以及版权声明栏。标题栏主要是进行商品类别的选择，进入对应类别的商品展示页面。在商品展示页面，用户预览商品，获取商品图片、价格等信息。在商品的详情展示页面，用户对商品细节进行全方位的感知，获取最真实的商品感官数据，对商品做出最直接的判断，是否符合需要，避免盲目购买，从而减少退、换商品的发生，节约买卖双方的时间和空间成本。商品3D模型是按照商品的真实参数和真实环境下的光照条件下在3维建模软件中制作完成的。而平台使用建模软件导出的OBJ、FBX等格式的3D文件进行加载渲染在网页端。平台兼具购物功能，具有登陆、注册、浏览、加入购物车以及进行结算等购物平台的基本功能。

1. 基于图像的后处理抗锯齿技术FXAA和SMAA

本章主要阐述基于图像后处理抗锯齿技术FXAA和SMAA算法。走样是指图像在计算机中用离散量来表示连续量而引起的图像失真，变形现象。计算机图形走样主要发生在图形光栅化和纹理映射的过程中，称为几何走样和纹理走样。

而相对应地，采用一定的方法来消除走样现象的技术，则成为反走样技术。对于反走样技术的基本思想是探测到图像边缘，对图像边缘和边缘周围的像素点颜色进行混合运算，采用新生成的像素点取代原位置的像素点，以达到对实物边缘进行锯齿消除和柔滑处理。

当前计算机图形学中主要采用硬件方式、后处理方式以及混合方式三种。而对于硬件方式主要采用提高样点数目，提高采样频率以达到消除锯齿，圆滑边缘的目的。虽然可以很好的处理锯齿问题，但是对硬件资源的消耗比较大，并且对计算机的显示分辨率有一定的要求。后处理方式的基本思想是在完整图像绘制完毕之后，在屏幕显示之前进行边缘检测，对图像的边缘进行消锯齿操作，使之圆滑。

## 3.1基于图像的后处理抗锯齿技术FXAA算法

### 3.1.1 FXAA算法的优势与不足

2009年Inter工程师成功将形态学引入计算机图形学中，提出了计算机中具有代表性的形态学反走样技术（Morphological Antialiasing，MLAA）。MLAA是依据相邻元素之间的颜色插值进行边缘提取，对于取到的轮廓与模式进行匹配，而后计算权值进行颜色混合处理。优点是独立于图形处理流水线，在CPU上进行边缘像素的抗锯齿处理，算法易实现。而后，人们陆续提出了许多改进算法。

快速近似反走样技术（Fast Approximate Anti-Aliasing, FXAA）与MLAA类似，运行于渲染后期处理阶段，快速近似抗锯齿（Fast Approximate Anti-Aliasing, FXAA），对图像边缘进行抗锯齿处理，速度更快，不会造成镜面模糊和亚像素模糊（表面渲染不足一个像素时的闪烁现象）。由于该方法进行边缘检测时，需要对整幅画面进行搜索，并与临近元素进行比较，最后需要加以计算进行边缘的平滑处理，故该方法在计算和存储上对计算机资源消耗比较大。该方法中元素的偏移量计算复杂繁琐导致对于一些模型的边缘反锯齿绘制效果并不是很好。尤其对于文字的处理，会导致文字的边缘模糊，不能得到良好的抗锯齿锐化效果。

### 3.1.2 FXAA算法的实现与效果比较

本虚拟购物平台搭建使用的Three.js框架良好的兼容了自定义算法的实现。FXAA算法是以像素单元为基本单位的，利用像素单元的亮度进行边缘检测。对获取到的边缘提取梯度。获取边缘的梯度方向（函数值或灰度值的最大增长方向），并将子偏移量初始于梯度值小的方向从而计算出该位置的颜色值。得到该像素位置的计算颜色值，并利用当前像素的颜色值和相邻像素的颜色值进行混合，得到最终的颜色值。混合效果图如下：

（做一个混合效果图）

算法流程：

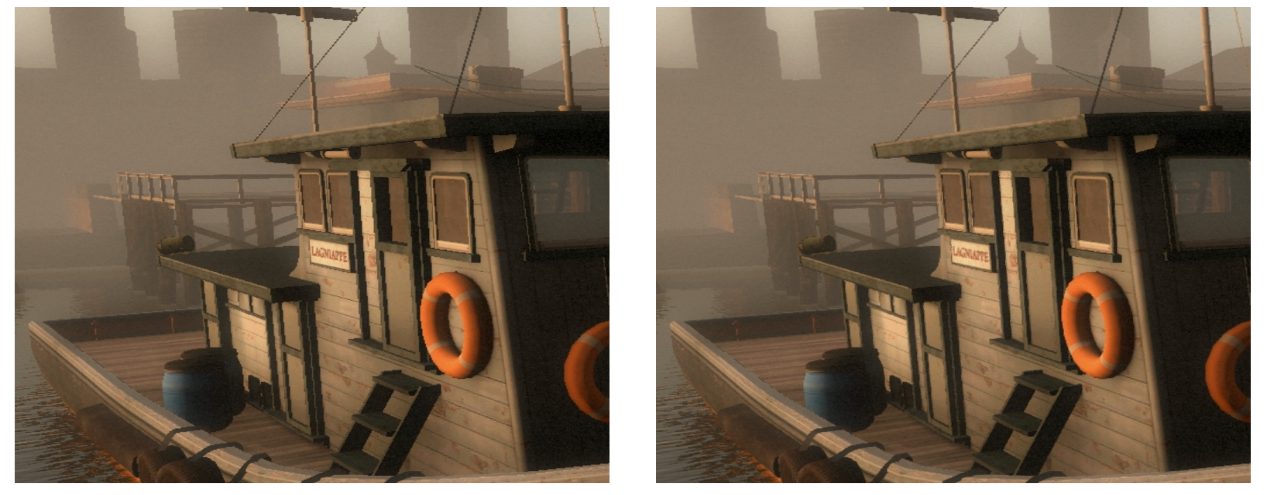
步骤一：模型加载，存储模型的顶点信息以及最后渲染绘制的纹理信息。

步骤二：判断是否为边缘，若是执行步骤三；否则执行步骤四。

步骤三：计算混合朝向和以及子偏移量，继而进行平滑处理，执行步骤四。

步骤四：绘制到计算机显示屏幕，程序结束。

通过上述步骤，借助Three.js框架将3维立体模型展示在屏幕，并通过FXAA算法对模型边缘进行了锯齿消除，使模型边缘平滑。效果如下图所示：



## 3.2基于图像的后处理抗锯齿技术SMAA算法

### 3.2.1 SMAA算法的优势和不足

增强子像素形态学反走样算法（Enhanced Subpixel Morphological Antialiasing, SMAA）【17】， SMAA算法是后处理抗锯齿技术的一种，它的处理流程是建立在优化的MLAA（计算机形态学抗锯齿）算法上的。而在SMAA算法之前，后期抗锯齿处理技术仍存在一定的缺陷，主要有：实物原本的形状信息没有得到很好的保留，甚至存在一些过度模糊的现象；忽略了对角线模式的检测，影响了抗锯齿的质量；基于颜色的边缘检测，导致高光和色彩上的锯齿没有很好的消除；局部对比度没有进行计算，无法准确评估可以被人眼识别的图像边缘，浪费了计算机的运算资源。而SMAA算法为解决这些问题应运而生。SMAA算法的边缘检测技术采用精确的距离搜索和增强了局部对比度，使得对图像的处理结果更加准确。与此同时，SMAA算法使用尖锐几何特征和对角线检测加以强化，可以使绘制的物体边缘更加准确，并获得良好的抗锯齿化效果。

SMAA算法相较于之前的后处理抗锯齿算法有了明显的提升，但算法本身仍有一定的不足和缺陷，如在绘制图像的边缘提取中，SMAA算法采用了固定阈值，针对与不同的模型特征，不能进行适应性更正，导致图像边缘的获取信息较少，与此同时SMAA算法存储了许多不必要的面积纹理，导致消耗计算机的许多存储资源。

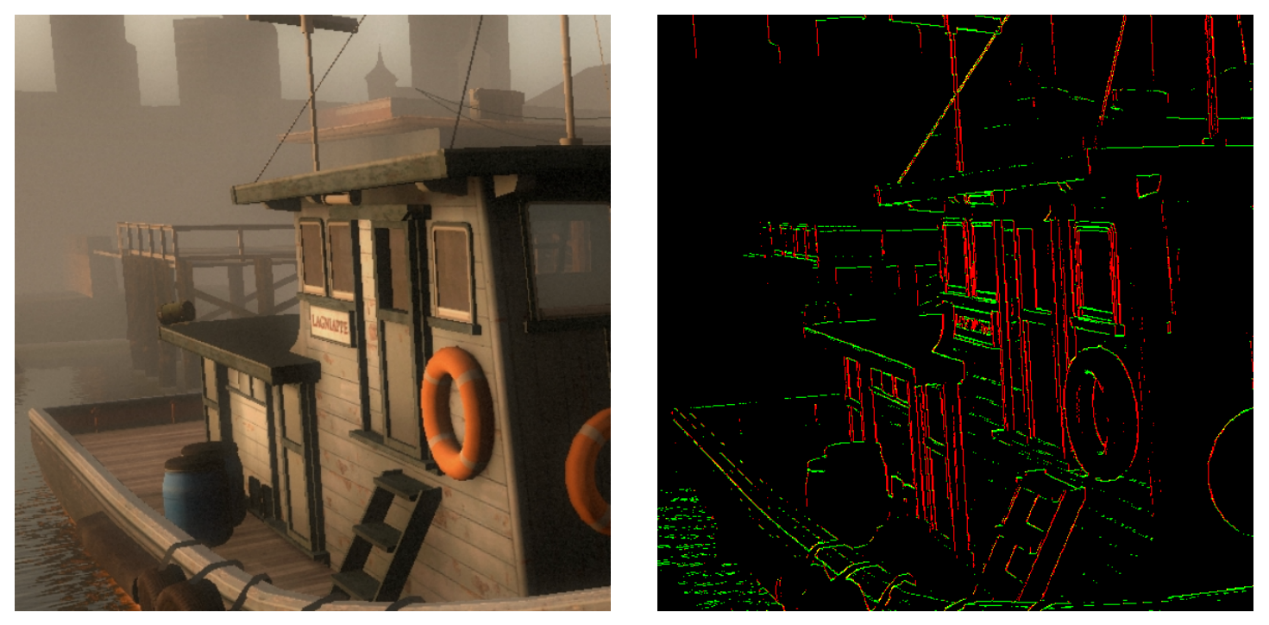
第4章 基于图像后处理抗锯齿算法SMAA的实现与改进

基于图像的后处理抗锯齿算法存在普遍的缺陷，主要是：由于算法的分析方式并不能准确的判定边界以及获取到准确的边界特征信息；缺少图像必要的场景集合信息，会使得绘制图像的结果出现边缘阶梯状严重以及过度模糊等现象，丧失了处理算法的优越性。故已实现的抗锯齿算法SMAA在一定程度上解决上述问题，得到较好的绘制图像，对图像边缘的锯齿消除有较好的效果。

## 4.1基于图像后处理抗锯齿算法SMAA的实现

本平台使用的基于Web GL的3D框架良好的性能借以实现SMAA算法。通过编写自定义SMAAShader.js进行实现算法，并进行性能测试。

算法概述：SMAA算法可分为边缘提取、权值计算和颜色混合等三个步骤。进行边缘提取时，主要有三种方式：亮度检测、颜色检测和深度检测。SMAA算法对边缘提取时采用固定阈值，导致对某些集合边缘的提取不甚有效。而后，将检测到的边缘与预设的几何形状进行匹配和颜色混合，得到最终图像的绘制效果图，并绘制到计算机屏幕中，加以显示。如图效果展示。



第5章 总结与展望

## 总结

虚拟现实技术是仿真技术和计算机图形学等多种技术的集合，是多领域的交叉科学，拉开了新世界的大门。本文介绍了基于WebGL的虚拟购物平台的搭建，以及围绕更好的进行3D展示做了一些工作。采用了比较流行的前端框架Bootstrap搭建显示页面，使用动态模态框展示商品的3D模型。采用轻巧、方便的Blender建模软件，导出指定格式的模型文件以便加载使用。在网页端，使用JavaScript语言编写的第三方开发库-----Three.js。在编程之前，比较详细地了解了WebGL的前身今生，找到比较良好的框架进行开发。在众多的抗锯齿算法中，选择了两种比较有特点的算法进行实现与比较，较好地消除了图像边缘的锯齿化现象，使渲染效果更加真实。

本次主要实现了基于虚拟现实技术的3D商品展示系统，建立模型，完成3D模型在网页端的展示，并实现基于图像的后处理抗锯齿算法FXAA和SMAA算法以及效果比较。本文的总结工作如下：

1. 介绍了虚拟现实技术的研究目的与意义，同时也对目前国内外的研究状况进行了总结，简介了虚拟现实技术的应用、技术现状以及存在的问题。
2. 介绍了本次设计与实现过程中涉及到的理论知识，介绍构建虚拟现实购物平台的使用的Web GL协议，以及为便捷开发而使用的3D框架---Three.js。为了提高图像以及阴影的成像效果，实现FXAA和SMAA算法，消除边缘锯齿化，平滑边缘。
3. 本次设计利用IDEA功能软件进行整体网站架构，使用Submit文本编辑器进行测试代码编写，使用Tomcat搭建本地服务器进行网络测试。

## 展望

当今网络技术的迅猛发展，极大的改变了人们的生活方式。虚拟现实技术的不断发展，带给人们新的体验，试图通过计算机等技术给人身临其境的真实体验感。WebGL协议的推广，实现了在浏览器内进行实物的三维立体展示，解除了对相应插件的依赖。开源的Three.js框架实现了的对WebGL的良好的封装，减轻了程序员的压力，又可以面向更多的普通人进行3D展示编程，虽然相关的文档还是比较稀少，相信伴随着该框架的不断发展，相关工作会逐步完善，更有利于Three.js的推广使用。由于我们计算机使用光栅化的图像显示技术，不可避免地产生走样问题，本体涉及的FXAA算法以及SMAA算法都是对渲染图像边缘锯齿化进行一定程度上的消除。随着计算机图像学的不断发展，FXAA算法以及SMAA算法会在继承的基础上加以改进和优化，越来越多的基于图像的后处理抗锯齿算法会相继问世，更好解决图像走样问题。

参考文献

1. 孙家广, 杨长青. 计算机图形学[M]. 北京：清华大学出版社, 1995.
2. Skolink M I. Radar handbook [M]. New York: McGraw-Hill, 1990.
3. 王鑫, 王斌, 张立明. 基于图像显著性区域的遥感图像机场检测[J]. 计算机辅助设计与图形学学报, 2012, (3):336-337.
4. Tnesman A M, Gelade G. A feature-integration theory of attention Cognitive Psychology[J]. 1980,12(1): 97-136.
5. 张佐光, 张晓宏, 仲伟虹, 等. 多相混杂纤维复合材料拉伸行为分析[A]. 见: 张为民编. 第九届全国复合材料学术会议论文集(下册)[C]. 北京: 世界图书出版公司, 1996. 410-416.
6. Odoni A R. The flow management problem in air traffic control[A]. In: Odoni A R, Szego G, eds. Flow Control of Congested Networks[C]. Berlin: Springer-Verlag, 1987. 269-298.
7. 钱丁丁. 图像显著性区域检测算法研究[D]. 济南：山东大学, 2012.
8. 金 宏. 导航系统的精度及容错性能的研究[D]. 北京: 北京航空航天大学, 1998.
9. GB/T 16159-1996, 汉语拼音正词法基本规则[S].
10. 姜锡洲. 一种温热外敷药制备方案[P]. 中国专利: 881056073, 1989-07-06.
11. 王明亮. 关于中国学术期刊标准化数据系统工程的进展[EB/OL]. http://www.cajcd.edu.cn /pub/ wm1.txt, 8-16/1998-10-04.

**说明：**

① 参考文献应是公开出版物，按在论著中出现的先后用阿拉伯数字连续排序。

② 参考文献中外国人名书写时一律姓前，名后，姓用全称，名可缩写为首字母(大写)，不加缩写点(见例2)。

③ 参考文献中作者为3人或少于3人应全部列出，3人以上只列出前3人，后加“等”或“et al”(见例3)。

**表1 参考文献类型及文献类型标识**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **型** | **专著** | **论文集** | **报纸文章** | **期刊文章** | **学位论文** | **报告** | **标准** | **专利** |
| 文献类  型标识 | M | C | N | J | D | R | S | P |

**表2 电子参考文献类型及其标识**

|  |  |  |  |
| --- | --- | --- | --- |
| **电子参考文献类型** | **数据库** | **计算机程序** | **电子公告** |
| 电子文献类型标识 | DB | CP | EB |

致 谢

经过大半年的学习研究，本科生的毕业设计已经基本完成。在这里我要感谢我的指导老师宋滢。从毕业设计的选题开始、开题答辩、中期答辩到最后的毕业论文的撰写以及毕业答辩，宋滢老师都细心地提供很多切实有效的指导，给予了我许多意见和帮助，尤其是在毕业设计后期遇到问题无法解决的时候，宋滢老师多次和我展开讨论，指出问题所在，给出解决问题的思路和可行的方案。正是宋滢老师的指导与帮助，才能让我一次次克服困难，最终完成毕业设计。宋滢老师面对问题的态度以及工作的认真以及严谨性都是我今后学习与工作的榜样。

与此同时，我要感谢我的母校，我的老师，我的同学。老师们的谆谆教导，给我今后的学习打下坚实的基础，让我有展望明天的自信。同学们的无私帮助，让我度过美好而快乐的大学四年。母校的无私培养让我能成长为一个不惧风雨自信自强的新世纪大学生。

最后感谢我的父母，感谢你们对我的无限包容与无私付出。你们是我人生的第一位导师，是你们的辛劳培养，才能成就今天的我。感谢我的父母，希望你们永远健康、快乐！

孙晓明

2018年4月30日

附录

1. FXAA算法测试程序

<!DOCTYPE html>

<html>

<head>

<title> FXAA 测试</title>

<meta charset="utf-8">

<style>

body {

color: #61443e;

font-family:Monospace;

font-size:13px;

text-align:center;

/\* background-color: #bfd1e5; \*/

background-color: #ffffff;

margin: 0px;

}

a { color: #a06851; }

canvas {

margin: 20px;

margin-top: 40px;

}

</style>

</head>

<body>

<div id="container"></div>

<script src="build/three.js"></script>

<script src="js/shaders/FXAAShader.js"></script>

<script src="js/shaders/SMAAShader.js"></script>

<script src="js/Detector.js"></script>

<script>

( function closure() {

'use strict';

if ( ! Detector.webgl ) {

Detector.addGetWebGLMessage();

}

var loader = new THREE.TextureLoader();

loader.load( "models/texture/fxaa\_scene.png", function onLoad( texture ) {

var image = texture.image;

texture.minFilter = THREE.LinearFilter;

texture.magFilter = THREE.LinearFilter;

var fxaaMaterial = new THREE.ShaderMaterial( THREE.FXAAShader );

fxaaMaterial.uniforms.tDiffuse.value = texture;

fxaaMaterial.uniforms.resolution.value.x = 1 / image.naturalWidth;

fxaaMaterial.uniforms.resolution.value.y = 1 / image.naturalHeight;

var basicMaterial = new THREE.MeshBasicMaterial( { map: texture } );

var plane = new THREE.PlaneBufferGeometry( 1, 1 );

var quad1 = new THREE.Mesh( plane, basicMaterial );

var quad2 = new THREE.Mesh( plane, fxaaMaterial );

var scene1 = new THREE.Scene();

var scene2 = new THREE.Scene();

scene1.add( quad1 );

scene2.add( quad2 );

var camera = new THREE.OrthographicCamera( -0.5, 0.5, 0.5, -0.5, -0.5, 0.5 );

var container = document.getElementById( 'container' );

var renderer1 = new THREE.WebGLRenderer();

renderer1.setPixelRatio( window.devicePixelRatio );

renderer1.setSize( image.naturalWidth, image.naturalHeight );

container.appendChild( renderer1.domElement );

var renderer2 = new THREE.WebGLRenderer();

renderer2.setPixelRatio( window.devicePixelRatio );

renderer2.setSize( image.naturalWidth, image.naturalHeight );

container.appendChild( renderer2.domElement );

renderer1.render( scene1, camera );

renderer2.render( scene2, camera );

});

})();

</script>

</body>

</html>