中国传媒大学研究生学位论文独创性声明

本学位论文作者声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得**中国传媒大学**或其他教育机构的学位或证书而使用过的材料。其他个人或组织对本研究所做的任何贡献，均已在论文中作了明确的说明并表示谢意。

作者签名：导师签名：

签字日期：年月日签字日期：年月日

学位论文版权使用授权书

本学位论文作者完全了解**中国传媒大学**有权保留并向国家有关部门或机构送交本论文的复印件和电子版，允许论文被查阅和借阅。论文作者授权**中国传媒大学**可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印、扫描或拷贝等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

作者签名：导师签名：

签字日期：年月日签字日期：年月日

基于像素分布匹配视觉模型的注视点渲染研究

摘要

随着虚拟现实（VR）和增强现实（AR）技术的不断发展，对于高质量视觉体验的需求越来越高。然而，提供高分辨率和高视觉保真度的实时视觉体验需要巨大的计算资源。本文进行了注视点渲染研究，利用人类视觉系统中注视点区域的敏感性和外围视野区域的模糊感知特性，通过创新地设计几何缓冲区的映射策略和帧间反走样算法，使得像素分布与人类视觉模型相匹配，同时减小了整个视野中的时间性不稳定。本文所介绍的方法在保证视觉效果的同时减少了计算资源的消耗，提升了渲染效率。实验结果表明，此方法能有效提高VR和AR等实时渲染系统的性能。

关键词：虚拟现实、注视点渲染、延迟渲染、图像映射、帧间反走样、视觉模型

Research on foveated rendering based on pixel-distributed matching visual model

**ABSTRACT**

As Virtual Reality (VR) and Augmented Reality (AR) technologies continue to evolve, there is an increasing demand for high-quality visual experiences. However, providing real-time visual experiences with high resolution and high visual fidelity requires huge computational resources. In this paper, we conduct a study on foveated rendering, which utilizes the sensitivity of the gaze point region and the ambiguity perception properties of the peripheral visual field region in the human visual system, and makes the pixel distribution match the human visual model by innovatively designing a geometric buffer mapping strategy and an inter-frame anti-alternation algorithm, while reducing the temporal instability throughout the visual field. The method presented in this paper reduces the consumption of computational resources and improves the rendering efficiency while ensuring the visual effect. Experimental results show that this method can effectively improve the performance of real-time rendering systems such as VR and AR.

**KEYWORDS: virtual reality; foveated rendering, deferred rendering, image mapping, temporal anti-aliasing, human visual model**

目录

[1 绪论 7](#_Toc4751)

[1.1 研究背景与意义 7](#_Toc30173)

[1.3 研究内容 8](#_Toc26324)

[1.4 论文结构 8](#_Toc25977)

[1.5 研究方法 9](#_Toc20066)

[2 注视点渲染系统的相关工作 9](#_Toc6346)

[2.1 视觉因素 10](#_Toc20246)

[2.1.1 视觉敏锐度 10](#_Toc15578)

[2.1.2 对比度 11](#_Toc3588)

[2.1.3 周边运动和闪烁感知 12](#_Toc9683)

[2.2 基于偏心率的视觉模型 12](#_Toc27765)

[2.2.1 双曲线模型 12](#_Toc7827)

[2.2.2 线性模型 13](#_Toc9624)

[2.2.3 对数模型 14](#_Toc27160)

[2.2.4 其他模型 15](#_Toc15070)

[2.3 注视点渲染空间 16](#_Toc11497)

[2.3.1 屏幕空间 16](#_Toc11295)

[2.3.2 对象空间 16](#_Toc26409)

[2.4 注视点类型 17](#_Toc8694)

[2.4.1 静态注视点 17](#_Toc11265)

[2.4.2 动态注视点 18](#_Toc9210)

[2.5 反走样 18](#_Toc26999)

[2.5.1 避免走样 19](#_Toc441)

[2.5.2 消除走样 19](#_Toc27636)

[3 方法理论与设计 21](#_Toc3241)

[3.1 基于矩形内部映射的图像变换 21](#_Toc25352)

[3.1.1 一维空间下的采样点分布 21](#_Toc29281)

[3.1.2 二维空间下的采样点映射 23](#_Toc6437)

[3.2 矩形与圆盘的区域映射 25](#_Toc16857)

[3.2.1 矩形内部映射存在的问题 25](#_Toc24175)

[3.2.2矩形-圆盘映射的作用与所需性质 26](#_Toc15242)

[3.2.3 基于费尔南德斯圆角矩形的映射 29](#_Toc26323)

[3.2.4 计算优化 34](#_Toc19358)

[3.3 帧间反走样 35](#_Toc7676)

[3.3.1背景介绍 36](#_Toc18067)

[3.3.2 方差采样 37](#_Toc18410)

[3.3.3 减少时间不稳定性 38](#_Toc1021)

[4 系统实现 39](#_Toc15152)

[4.1 延迟渲染管线 39](#_Toc4167)

[4.1.1 几何缓冲区的填充 39](#_Toc10783)

[4.1.2 基于PBR的延迟着色 41](#_Toc13741)

[4.2 几何缓冲区的映射 42](#_Toc23155)

[4.3 反走样的实现 42](#_Toc23997)

[4.3.1 抖动当前帧 42](#_Toc26427)

[4.3.2 解析历史帧 43](#_Toc6174)

[4.3.3 修正历史帧 45](#_Toc18185)

[4.4 采样密度的计算与可视化 46](#_Toc14460)

[5 实验 46](#_Toc24759)

[5.1 客观实验 46](#_Toc19643)

[5.2 用户实验与分析 46](#_Toc2883)

[5.2.1 实验概述 46](#_Toc30179)

[5.2.2 实验流程 46](#_Toc31114)

[5.2.3 实验人员 46](#_Toc16749)

[5.2.1 愉悦度 47](#_Toc22770)

[5.2.2 胜任感 47](#_Toc19913)

[5.2.3 努力感、重要性 47](#_Toc25151)

[5.2.4压力、紧张 47](#_Toc7264)

[6 结论 47](#_Toc22049)

[6.1 工作总结 47](#_Toc17504)

[6.2 未来工作展望 47](#_Toc27466)

[参考文献 48](#_Toc5385)

[附录 57](#_Toc8708)

[致谢 58](#_Toc27487)

# 1 绪论

## 1.1 研究背景与意义

虚拟现实和增强现实（VR 和 AR）将改变以计算机为媒介的通信方式，在科学、工程、医学、艺术、娱乐、商业和其他一些领域都有令人兴奋的应用。尽管目前的 VR 和 AR 系统给人留下了深刻印象，但在分辨率、视野和延迟等几个方面，它们还无法与我们在现实世界中体验到的视觉保真度相媲美。实时提供如此高质量的视觉体验需要巨大的计算资源，而最近用于渲染的硬件和软件系统都有所改进。

人类视觉系统（HVS）只能感知投射到人类视网膜中心--眼窝上的极小部分场景的最细节。视敏度是观察细节的能力，用光栅分辨率（周期/度）来衡量。研究表明，虽然 HVS 有很宽的视野（FOV），但视敏度最高的区域（也称为注视点区域）仅覆盖视野中央 2.5° [Levin 等人，2011 年]。眼窝是指在人类视觉系统中，视敏度随角度距离的增加而降低[Guenter 等人，2012]。注视点渲染利用了人类视觉系统的这一特点，有选择性地渲染图形帧的一小部分细节。注视点渲染使我们能够在现代 VR 和 AR 设备上协调高视觉逼真度、交互帧频和低功耗这几个相互冲突的目标。随着注视点渲染成为未来 AR/VR 设备不可或缺的一部分，了解该领域的现状和观察到的趋势非常重要。

注视点渲染的概念并不新鲜。视线相关显示的早期实现应用于建筑飞行模拟器[Murphy 和 Duchowski 2001]。不过，早期的方法主要是针对桌面显示器开发的，其视场大约为 43° × 33° [Murphy 和 Duchowski 2001]。这只是现代宽视场显示器的一小部分。Patney 等人[2016b]比较了从小视角智能手机屏幕到大视角 VR HMD 等各种设备外围区域像素的百分比，如图 1 所示。因此，与传统显示器相比，注视点渲染对于 VR 和 AR 头显变得更加重要。在 100° 宽 FOV HMD 中，只有 4% 的屏幕像素位于注视点区域，其余的都位于外围区域[Patney 等人，2016b]。当如此大比例的像素位于低视力的外围区域时，高效的渲染系统可以通过分配较少的资源来渲染外围区域，从而获得显著的计算增益。

直到最近，实时眼球跟踪技术在 VR 和 AR 头显中还不常见。不过，随着眼球追踪器在 VR 和 AR 头显中的商品化，注视点渲染技术可以利用注视点信息动态地以较高的质量渲染注视点区域，而以较低的质量渲染其他区域，同时不会降低图像的整体感知质量。随着未来 HMD 的 FOV 增加到与人类视觉相匹配，注视点渲染将成为大规模、宽 FOV 和高分辨率 VR 和 AR 显示屏实时渲染 3D 图形管道的重要组成部分。4D 光场是注视点渲染技术的另一个受益领域。4D 光场从多个摄像机位置表现一个场景，这会产生巨大的渲染成本。Sun 等人[2017]和 Meng 等人[2020b]提出了利用注视点渲染技术加速光场渲染过程的方法，从而实现实时光场渲染。

## 1.3 研究内容

本文针对虚拟现实（VR）和增强现实（AR）场景中实时渲染高分辨率和高保真度视觉体验所面临的挑战，研究了基于像素分布匹配的注视点渲染技术。研究内容主要集中在探索人类视觉系统的敏感区域特性，即注视点区域具有更高的视觉敏锐度，而外围视野的分辨率感知能力较低。通过这个特性，提出了一种方法以在不降低整体视觉质量的前提下减少所需的计算资源和提升渲染效率。研究内容包括对现有视觉模型的分析、基于偏心率的视觉因素考虑、针对性的映射策略，整体视野中的时间不稳定性的考虑、帧间反走样技术的设计与实现。

## 1.4 论文结构

本文共分为六个章节。每个章节的内容如下。

第一章为绪论，介绍研究背景、意义、内容、结构和方法。

第二章为注视点渲染系统的相关工作，回顾了视觉因素、基于偏心率的视觉模型、注视点渲染空间、注视点类型和反走样技术。

第三章为方法理论与设计，提出了基于矩形内部映射的图像变换理论，矩形与圆盘的区域映射及帧间反走样。

第四章为系统实现，详细描述了延迟渲染管线、几何缓冲区的映射及反走样的具体实现。

第五章为实验，进行了客观实验和用户实验与分析，评价了所提方法的性能。

第六章为结论，总结了工作内容和未来工作的研究方向。

## 1.5 研究方法

文献研究法：通过对现有文献的系统性回顾，梳理了视觉敏感度、对比度及注视点渲染等相关领域的理论和实践研究进展，确定了研究的理论基础和设计参考。

定性分析法：基于对人类视觉系统的认识及其在VR/AR中的应用，通过定性分析确定了适合于注视点渲染技术的视觉模型，包括对视觉敏锐度、对比度变化及外围视觉感知特性的理解。

实证研究法：设计了实验方案，包括注视点渲染系统的实现，以及对标注视点渲染技术效果的实证研究，通过客观实验和用户体验测试来评估所提出方法的效能和可行性。

定量分析法：在客观实验中，采集了性能数据，如帧率、延迟等指标，并运用定量分析方法对数据进行分析，从而确证注视点渲染技术对于提升渲染性能的贡献。

模型法：对所建立的视觉模型进行评估，通过变更参数设置，测试模型对视觉质量和渲染效率的影响，以评估模型的准确性和实际应用价值。针对所设计的帧间反走样算法进行评估，对算法在不同场景下的性能和效果进行了系统测试，并对结果进行了分析和讨论。

# 2 注视点渲染系统的相关工作

视杆细胞和视锥细胞在视网膜上的非均匀分布，以及它们对入射光的不同反应，导致了从眼窝到周边视力的一些变化。例如，对细节的感知随着偏心率的增加而降低，对相同物体的感知时间随着偏心率的增加而增加[11]。视杆细胞的空间辨别能力有限，而视锥细胞则具有出色的空间定位和颜色辨别能力[12]。然而，外周区域的运动检测能力比预期的要高，区分物体相对速度的能力和闪烁敏感度在整个视野中保持一致。此外，Patney等人[35]的研究表明，保留外围区域的对比度可以提高滤波图像的感知能力。研究还表明，虽然颜色感知会随着偏心率的增加而减弱，但这种减弱是渐进的，而且在高偏心率下颜色辨别力仍然存在[14]。Rosenholtz[15]认为，外围视觉并不仅仅是低分辨率版的眼窝视觉，而是负责一种类似纹理的表征，它保留了场景的汇总统计信息。外围区域需要保留的不同属性。注视点渲染可以利用这些随偏心率变化的人类视觉特性。

大多数实时渲染都使用光栅化管道。光线追踪技术能提供更逼真的图像，包括精确的反射、折射和阴影、折射和阴影提供更逼真的图像[16]。注视点渲染可同时应用于光栅化和光线追踪管道。实时光栅化的图形流水线通常由着色操作的计算复杂性所主导。着色操作的复杂性。可以通过以下方法降低着色成本简化操作，预先计算部分操作、或减少操作次数。注视点渲染试图通过减少每个像素的平均操作次数。灵敏度，从而减少每个像素的平均操作次数。接下来，我们将根据各种注视点渲染技术的特点，对其进行分类和比较，并讨论它们在实时渲染中的应用。

第2.1节介绍了人类视觉系统的各种视觉属性，这些属性在整个视野中存在空间上的变化。第2.2节讨论了表征视觉灵敏度空间差异的各种模型。第2.3节探讨了不同注视点渲染系统所发挥作用的不同阶段。第2.4节概述了与注视跟踪相关的技术。最后，第2.5节讨论了注视点渲染的反走样技术。

## 2.1 视觉因素

视觉的感知取决于各种因素，包括但不限于偏心、对比度、亮度（luminance）、明度（brightness）、颜色、形状、运动、固定和显著性。在本节中，我们将根据不同视觉因素对注视点渲染方法进行研究。

### 2.1.1 视觉敏锐度

视觉敏锐度是一种广泛使用的视觉指标，其定义为，在给定偏心率下，可分辨区域所占角度的倒数。视敏度在整个视野中都是不同的。早期研究[17]表明，随着偏心率的增加，视觉敏锐度会下降。因此，随着与注视点的角度距离的增加，分辨率也随之降低，这就是注视点渲染存在的原因。不同的注视点渲染方法，他们的主要区别就在于降低分辨率的方式。一种常见的方法是独立渲染多个离散层（后称偏心层），每个层以不同的比率采样[18][19]。这些偏心层的图层随后会按照屏幕分辨率进行上采样，然后合成在一起。由于着色是渲染管道中最耗时的操作，某些作品[20]改变了整个视野的着色率，在外围区域每隔几个像素着色一次。最近的一些作品[21][22][23]也展示了一种方式，能够更连续地改变分辨率。这些方法将视场映射到一个非线性空间，其像素分布与人类视觉的敏锐度相匹配。上述方法主要是利用光栅化管道开发的，而光线追踪方法自然可以在屏幕空间中实现平滑的非均匀采样。Weier等人[24]以及Fujita和Harada[25]在光线追踪中加入了注视点渲染，随着与注视点距离的增加，改变采样概率，其以此来实现稀疏采样。关于离散分布和连续分布之间权衡的更多讨论，最近关于视网膜显示的工作[26]做了详细的讲解。这些方法背后的核心理念，是要模仿与 HVS 视觉敏锐度密切相关的分布，他们以不同的方式构造非均匀采样分布函数，以最好地匹配整个视野的敏锐度。

此外，最近的研究[27][28]表明，在相同偏心率值下，横轴的视敏度高于纵轴。而且，低视场的视敏度更高。目前大多数的注视点渲染技术在渲染过程中都没有考虑到这种不对称，而是假设水平和垂直方向上的视敏度是一致的。

### 2.1.2 对比度

图像对比度是影响视觉感知的一个主要因素。检测到图案所需的最小对比度（对比度阈值），取决于图案的空间频率以及偏心率。对比度阈值的倒数称为对比度灵敏度，我们用对比度灵敏度函数（CSF）表示整个视野的对比度灵敏度。该函数描述的是人眼视觉系统识别图案差异的能力[29]。与空间敏锐度类似，给定空间频率的对比度敏感度在视野中心处最高，随着偏心率的增加而降低。对比度敏感度取决于多种因素，如目标图案的大小、对比度和观察角度，文献中提出了多种 CSF 建模函数[30][31]。

对比敏感度与空间频率之间的关系在整个视野中各不相同[32]。一般来说，随着空间频率的增加，在视野中心处，CSF的对比敏感度会逐渐下降，而视野外围处，CSF的对比敏感度则会突然下降。对比敏感度函数有一个截止空间频率，超过这个频率就无法辨别更高的频率。截止空间频率取决于视野中图像斑块位置的偏心率。此外，对比敏感度函数还与任务有关。检测任务和辨别任务的外围CSF 显示了高空间频率下的不同表现极限[33]。辨别任务的外围对比敏感度下降速度比检测任务快得多。检测任务和辨别任务的截止频率之间的差异表明，人类视觉系统可以感知外围的较高频率，但无法分辨细节。检测和辨别截止频率之间的频率范围被称为混叠区。根据分辨截止频率确定采样率的注视点渲染器比根据检测截止频率确定采样率的渲染器性能更快。不过，根据分辨截止频率渲染周边区域会降低频率处于混叠区的区域的对比度。为了保持检测能力，Patney等人[34][35]建议增强混叠区的外围对比度，因为过滤会降低混叠区的对比度，从而保持非注视点区域的感知质量。Tursun等人[36]利用了亮度对比对视觉感知的影响。他们观察到，虽然在高对比度区域降低一定的空间分辨率会降低感知质量，但在低对比度区域降低同样的分辨率却能保持感知质量。每个区域所需的最小分辨率是通过其对比度和偏心率估算出来的。除了亮度之外，对色彩的敏感度也会向周边区域降低。色调分辨率或每个RGB通道内可感知的灰度级数会随着偏心率的增加而减少[37]。Liu等人[38]的研究表明，随着偏心率从0°增加到30°，代表每个颜色通道的比特数会从8比特单调地减少到4比特。

### 2.1.3 周边运动和闪烁感知

有些任务在整个视野中的表现是一致的。例如，我们区分两个物体速度的能力在整个视野中是相当一致的[McKee 和 Nakayama，1984 年]。我们对闪烁的敏感度在整个视野中也是大致保持一致的[Kelly 1984；Koenderink 等人，1978b；Koenderink 等人，1978a；Koenderink 等人，1978c]。

一致的运动和时间敏感度要求注视点渲染器避免出现时间混叠等伪影，因为即使在外围也很容易察觉到这些假象。不考虑时间稳定性的注视点渲染器要么会破坏沉浸感，要么就必须在外围进行适当的处理，取得一个较为保守的结果。

## 2.2 基于偏心率的视觉模型

注视点渲染系统根据上述一个或多个视觉因素改变整个视野的分辨率或采样率。根据场景内容和任务，这些系统使用分析表达式来近似人类视野。这些近似值允许不同类型的像素分布。本节将讨论不同分布的异同。

### 2.2.1 双曲线模型

视觉敏锐度可以定量表示为最小分辨角的倒数[46]，即两点被感知为不同的最小角度。最小分辨角随着偏心率的增加而线性增加。视觉敏锐度随偏心率变化的一种可能的数学描述如下 ：

(1)

其中，相当于发生在眼窝处的最小可分辨角度分辨率（e = 0），m代表斜率。我们将这种分布称为双曲线模型，其中敏锐度的变化是1/eccentricity的函数。这种双曲线模型在低偏心率（小于8°）时，与真实视敏度近似[18]，之后视敏度会陡然下降。许多注视点渲染器[18][20][35][45][19][48]的采样分布都是基于这种双曲线式的视敏度下降。

Guenter等人[18]的开创性工作表明，采用注视点渲染进行光栅化可以显著提高性能。他们假设若干个离散层就足以模拟人类视觉系统中的敏锐度下降，他们使用了三个嵌套和重叠的矩形层，以不同的分辨率进行渲染。这三个层被称为偏心层，以注视点为中心。最内层即视野中心层是以最高分辨率（即显示器分辨率）渲染的。中间层比内层大，以相对较低的分辨率呈现。最外层覆盖整个屏幕，以更低的分辨率呈现。然后将所有三个图层插值到显示分辨率，并平滑混合。双曲线函数用于计算偏心层的大小和分辨率。参数取决于最小角分辨率斜率m，该斜率是根据用户研究估算的。该方法假定辐射视力下降是对称的，忽略了视野水平轴和垂直轴的差异。他们将自己的方法限制在三层，以近似人类视觉系统，尽管更多层次可以提供更好的近似效果，但代价是复杂性增加。因此，与以统一的高分辨率渲染整个画面相比，渲染像素的总数有所减少。

Stengel等人[45]对上述模型进行了扩展，纳入了平滑追逐眼动的效果。当移动物体吸引注意力时，会无意识地触发这种眼动。他们将聚焦区域建模为一条直线，通过整合连续帧上的注视位置获得。现在，偏心率是根据到这条直线的距离来测量的。因此，在最高分辨率下，渲染的是包裹直线的较大椭圆区域，而不是围绕单点的圆形区域。这种方法在延迟较高的显示器上性能更优。

Spjut等人[49]使用双曲线模型描述了HVS的敏锐度分布，并根据与该分布的匹配程度，对注视点渲染系统进行评估。对于偏心率小的情况，双曲线模型已被证明是准确的[18][45]。然而，通常情况下，偏心率超过30°的区域是以统一的分辨率渲染的，这可能会限制注视点渲染的性能。

### 2.2.2 线性模型

Weier等人[24]考虑的是随着偏心率的增加，敏锐度呈线性下降，而不是上述的双曲线下降。通过在光线跟踪管道中建立线性模型，采样概率会随着与注视点距离的增加而降低。与 Guenter等人[18]的研究类似，视野被分为三个区域。如图7所示，每个区域都有三个参数：r0、r1、pmin。内部区域（r0度范围内的区域）为中心区域，该区域以全分辨率渲染，因此采样概率为 1。最外围区域（距离注视点r1度以外的区域）的像素采样概率最小为pmin 。两层之间区域（r0 和r1之间）的像素根据线性方程采样：

(2)

其中，d(x)是像素x与视野中心的距离，单位为度。r0、r1 和 pmin 是用户自定义的参数。线性近似模型更为宽松，仅在偏心率值较小的情况下有效，偏心率值越大，区域渲染分辨率越低。

### 2.2.3 对数模型

这种对图像的对数极坐标映射[50]确保了感知灵敏度在视野中心较高，并随着与注视点距离的增加呈对数递减。与上述在渲染过程中使用多分辨率的模型不同，从笛卡尔空间到对数极坐标空间的映射与人类视觉敏锐度相匹配，可以使用单一的统一分辨率渲染，直接应用于转换后的空间。对数极坐标映射已被证明在计算机视觉、机器人、计算机制图和图像处理等许多领域非常有用，因为它能利用有限的计算资源提供足够的视觉细节[51]。、

Meng等人[22]利用人类视觉系统的对数极坐标映射为网格提供了一种注视点渲染方法。他们的系统引入了一种核函数对数极坐标映射技术，可以灵活地模拟与HVS相匹配的敏锐度下降。视敏度按对数递减，并取决于核函数。该技术的渲染加速是通过延迟着色实现的[52]，这是一种在实时渲染中广泛使用的技术。着色计算所需的每个表面的位置、法线、纹理和材质信息都会被渲染到几何缓冲区（G-buffer）中。几何缓冲区的内容从笛卡尔空间转换到对数极坐标空间。计算每个像素的直接和间接照明，并渲染到缩小分辨率的对数极坐标缓冲区。照明计算在缩小的对数极坐标空间中进行。应用反核函数对数极坐标映射将着色区域映射回笛卡尔屏幕空间。这种方法能够在对数极坐标空间中系统地连续改变采样率和采样分布。

Koskela等人[53]引入了从笛卡尔空间到视觉极坐标空间的类似映射。在视觉极坐标空间中对每个像素进行一次采样的路径追踪。极坐标空间经过修改，使采样分布与人类视敏度分布一致。为了与HVS相匹配，可以调整沿角度轴和径向轴的采样数量。他们观察到，改变沿角度轴的样本数量会导致周边区域出现伪影，而改变沿径向轴的样本数量则会导致注视点区域出现伪影。基于这些发现，他们的优化技术沿注视点角度轴改变分辨率，并重新调整周边区域的径向轴。以及对噪声扰动的路径追踪的视极空间图像进行去噪处理[54]。重建后的视觉极坐标空间图像通过反映射转换回笛卡尔坐标。他们的报告显示，与对数极坐标映射相比，视觉极坐标映射减少了造成干扰的伪影。

### 2.2.4 其他模型

保形渲染（Conformal rendering）[55]将视觉敏锐度建模为偏心率的非线性函数。该技术旨在利用屏幕与注视点距离的非线性映射，模仿从注视点到周边区域的平滑过渡。虚拟场景的投影顶点被扭曲成一个非线性空间，该空间与视网膜敏锐度和HMD镜头特性相匹配。然后以较低的分辨率对扭曲后的图像进行光栅化处理，再将其解压缩回笛卡尔空间。共形注视点渲染的复杂程度取决于场景的复杂程度。随着场景中顶点数量的增加，性能也会降低。与使用离散层来模拟敏锐度下降的方法相比，Meng等人[22]和Bastani等人[55]的方法使用非线性映射的连续平滑函数来模拟从注视点到周边区域的敏锐度下降。这种平滑过渡有助于减少视觉伪影。

Reddy[56]提出了一个视敏度模型，它是投射到视网膜上的刺激物角速度和偏心率的函数。视敏度的变化是偏心率的反二次函数[57]。Zheng等人[58]根据Reddy[56]的视敏度模型开发了一种注视点渲染方法，并相应地调整了细分级别。Friston等人[59]使用简单的径向幂折射函数p(x) = f(x2)，将注视点的距离映射为一种形变的距离。然后，像素位置会根据距离进行缩放。整体效果是放大了靠近注视点的区域，赋予了中心区域更多的重要性。他们假定形变函数是任意的，可以在每一帧自由变化，但要求该函数是可逆的，以便将有形变的图像映射回无形变的图像进行显示。Fujita和Harada等人[25]假设敏锐度下降为d(x)-2/3的函数，其中d(x)是与注视点的距离，并相应地定义了采样分布函数。最近，Li等人[60]提出了一种基于对数线性映射的注视点渲染方法，以模拟分辨率随偏心率增加而呈指数衰减的情况。

上述各种感知模型模拟了整个视野中HVS的视觉灵敏度。它们主要在原理、复杂性和样本分布上存在细微差别。最常用的双曲线模型模拟眼睛视网膜投射的早期操作，而对数模型则代表HVS的视觉大脑皮层部分。在实际应用中，因为需要实时渲染，计算复杂度是一个主要因素。与其他模型相比，最简单的线性模型能使外围区域的分辨率更高。对数模型提供了更可调的衰减分布。

这些模型主要在内围区域的分辨率要求上有所不同。双曲线函数会大幅降低分辨率，其次是核函数对数极坐标模型、反二次函数模型和线性下降模型。与双曲线衰减相比，核函数对数极坐标映射对内围区域的分辨率要求更高。在外围，所有模型都趋近于相似的分辨率。我们可以根据场景的复杂程度和要执行的任务，从这些近似值中选择一个分辨率分布函数。此外，我们还看到有多种视觉因素会影响场景的感知质量。大多数现有模型只考虑了其中的几个视觉因素。迄今为止，在权衡这些因素及其相互关系以建立整体感知模型方面的研究还不多。此外，视觉灵敏度会根据呈现给用户的外部刺激发生动态变化，这进一步增加了所需模型的复杂性。因此，在不影响视觉感知质量或性能的前提下，简化复杂模型、考虑视觉因素之间各种相互关系的技术非常重要。

## 2.3 注视点渲染空间

我们将注视点渲染技术分为基于屏幕和基于物体的方法，其依据是渲染管道中包含注视点概念的阶段：屏幕空间、模型空间。

### 2.3.1 屏幕空间

大多数注视点渲染技术会根据与注视点的距离改变屏幕空间的采样率。基于屏幕的注视点渲染方法涉及在显示之前对帧缓冲区内容进行操作，以降低整体着色率。Vaidyanathan 等人[48]提供了一种称为粗像素着色的架构，可对屏幕空间中的外围区域进行稀疏着色操作。Swafford等人[19]研究了注视点渲染对环境光遮蔽（AO）的影响，AO在现代实时渲染流水线上的计算成本很高。具体来说，他们介绍了屏幕空间环境光遮蔽（SSAO）在注视点和周边区域改变每像素深度样本的效果。他们的研究表明，在外围区域，由于敏锐度降低，因每像素样本数量较少而产生的条带效应不易察觉，从而提高了性能。另一种在屏幕空间做出改变的注视点渲染器是保形渲染[55]，其中投影顶点在光栅化之前被扭曲到非线性空间。然后以较低的分辨率对图像进行光栅化，最后将其变换到笛卡尔空间。

### 2.3.2 对象空间

基于对象的注视点渲染方法涉及在渲染之前对模型几何形状进行处理。Levoy 和Whitaker[61]最早的注视点渲染工作之一就是基于对象的方法。他们采用光线追踪方法进行体积渲染，根据像素与注视方向的角度距离，改变单位面积投射光线的数量和每条光线的采样密度。该系统使用三维mipmap对三维体积进行预过滤，并在外围区域使用较少的样本。一些早期的研究也是根据注视方向对物体进行建模。根据物体与注视点的距离来确定所需的物体细节级别。Swafford等人[19]开发了另一种物体空间技术。他们改变了注视点和周边区域的曲面细分级别。对于屏幕空间投影后落在注视点内的几何面片，使用较高的细分系数。对于那些位于外围区域的几何面片，则使用较低的细分系数。通过线性插值，决定位于注视点和外围区域之间的瓷砖的细分级别。他们的研究结果表明，与以均匀细分渲染整个场景相比，性能有所提高。与Swafford等人[19]的思路类似，Zheng等人[58]提出了一种基于给定偏心率下视觉灵敏度的细分方法。他们建议删除所有无法察觉的多边形，然后动态地对模型进行细分。屏幕空间中边长小于用户最小可感知长度的多边形不会被进一步细分。可感知多边形的细分等级由多边形在屏幕空间中的边长与最小可感知长度之比决定。

## 2.4 注视点类型

本节讨论眼动追踪在注视点渲染系统中的应用，并根据注视点的选取对其进行分类。

### 2.4.1 静态注视点

静态注视点渲染技术不依赖于眼动追踪设备，并假定用户的注意力集中在图像中的特定位置。用户注视特定区域的可能性会估算出该区域所需的敏锐度水平。许多早期的研究利用了静态注视点，即假设用户注视屏幕中心[62]，或使用基于内容的视觉注意力模型[63][64]。在眼球跟踪设备开发出来之前，头部的方向被认为是注视方向的良好近似值。大部分早期的注视点渲染工作都是基于这一近似值，并假设注视点位于屏幕中心。

静态注视点渲染的一个例子是Oculus开发的固定注视点渲染（FFR）[65]，它假定大多数用户的视线朝向显示屏中心。FFR采用基于平铺的方法。根据与中心的距离，图像被细分为不同敏锐度的区块。与较高分辨率区域相对应的区块位于中心位置，而靠近边缘的区块则与较低分辨率相对应。每个区块都以统一的空间分辨率进行渲染。静态注视点渲染要求用户将焦点保持在预定义的固定区域，主要是屏幕中心。静态注视点渲染技术的一个显著优势是，它不依赖于眼球跟踪设备，并且兼容所有现有设备。此外，静态注视点渲染方法还能随着视觉注意力和显著性模型的最新改进而得到进一步改善[66][40][67]。显著区域周围的区域总是以更高的分辨率进行渲染。然而，与动态方法相比，静态方法需要更大的高分辨率区域。所以其平均渲染成本可能会高于动态方法。

### 2.4.2 动态注视点

传统的注视点渲染是以眼球跟踪技术为前提的[56]。眼球跟踪设备可以实时确定用户的注视位置。有关头戴式眼动追踪器的研究可以追溯到20世纪60年代[68]。然而，近年来在计算能力、大规模并行图像处理、低成本和小型硬件方面取得的进步，使得在VR和AR中使用实时眼球跟踪成为可能。通过使用眼动跟踪工具主动跟踪用户的注视，注视点周围的一小块区域将以较高分辨率渲染，而周边区域则以较低分辨率渲染。大多数注视点渲染方法都是动态方法，依赖于眼动追踪设备的性能来获取注视点。除了基于硬件的眼动仪，人们对开发使用深度学习方法预测未来注视位置的方法也越来越感兴趣[69][70]。随着眼动追踪解决方案的精度和效率进一步提高，动态注视点渲染很可能会大大提升渲染性能和质量。

## 2.5 反走样

在注视点渲染中，由于外围区域的渲染分辨率较低，因此会出现明显的走样现象。注视点渲染产生的走样伪影可以是空间性的，也可以是时间性的。当虚拟世界的细节水平高于渲染分辨率时，就会出现空间性走样[72]。它们出现在物体层面，与任何运动无关。场景运动过程中的渐变会产生时间性走样。随着用户视图的变化，这些伪影会产生闪烁和闪烁效果，破坏用户在虚拟世界中的体验。研究表明，即使在偏心率较高的情况下，人类视觉系统对时间性走样也很敏感[71][72]。McKee和Nakayama[73]的研究表明，运动敏锐度从0°到10°会急剧下降，然后从10°到40°会更轻微地下降。人类视觉系统对外围区域运动的敏感性给所有注视点渲染技术带来了严峻挑战。Patney等人[35]的研究表明，要想获得有效的用户体验，就必须在注视点渲染中尽量减少时间性走样。

### 2.5.1 避免走样

确保时间和空间稳定性的一种方法是，在设计注视点渲染技术时，就防止走样现象的出现。这些技术一般会识别一些可能导致走样的特征，并在渲染管道中移除该特征或消除其影响[55]。注视点和周边区域会根据注视方向不断更新。一般来说，高分辨率注视点区域和低分辨率周边区域的渲染像素位置与显示坐标系一致。低分辨率的渲染像素随后会被上采样，以匹配原生显示器的分辨率。随着用户头部的旋转，每个渲染像素的值都会发生变化，而与场景内容无关，从而导致像素颜色的偏移和闪烁。这会在上采样的显示像素中产生时间性走样，从而破坏用户体验。Turner等人[74]观察到，视锥的适当角度对齐，可最大限度地减少，转头时的帧间闪烁效应。他们提出了一种相位对齐的注视点渲染系统，将低分辨率区域对齐到世界坐标系而非显示坐标系。低分辨率区域在世界空间中采样，然后向上采样并重新投影，以与显示坐标系对齐。现在，世界空间中采样的图像，不会随着转头而移动。因此转头时，图像时间性的变化会降到最低。相位对齐的注视点渲染大大降低了外围区域闪烁伪影的可感知性。不过，在渲染的画面中，注视点和周边区域的边界，在空间混叠时会很明显。注视点与周边区域边界的可辨识性，可以通过分辨率从注视点到周边区域的平滑过渡来降低[55]。Franke等人[75]和Mueller等人[76]利用时间一致性来减少着色次数。Franke等人[75]将前一帧的外围区域重新投影到当前帧，只渲染注视点像素和一致性较差的外围像素。他们提出使用片元的精确世界空间像素位置而不是深度值来避免重投影走样。

### 2.5.2 消除走样

大多数注视点渲染方法都会在后处理阶段降低伪影的可感知性。对低分辨率外围区域进行上采样时，插值技术的选择也会影响外围区域的走样。例如，简单的最近邻插值会放大时间性走样[71]，不过与双线性插值相比，它可以良好地保留对比度。

Guenter等人[18]结合多采样抗锯齿（MSAA）、时域重投影、空间采样点在时序上的抖动来减少空间和时间性走样。MSAA 通过提高这些区域的有效采样分辨率，减少剪影边缘的空间混叠。带有帧抖动的时域重投影可减少整个图像的伪影。时域抗锯齿策略能更好地减少，以较低采样率对高镜面材料进行采样时产生的锯齿。基于重投影的时域抗锯齿是一种常见的技术，它利用前一帧的信息来减轻时间性和空间性走样。当前帧被重投射到前一帧上，然后利用两帧的信息来计算最终色彩。然而，由于前一帧的细节可能会在正确投影后继续存在，因此时间抗锯齿可能会产生高频率的重影[35]。Karis[77]通过调节与当前帧样本一致的前一帧样本，减少了此类重影伪影。Meng等人[22]在屏幕空间中应用具有Halton采样的时间抗锯齿作为后处理方法，以减轻外围区域出现的走样。

Patney等人[35]除了使用时空抗锯齿方法外，还使用了预滤波器来缓解时空走样。他们引入了方差采样作为后处理图像增强技术。在每个像素周围，根据局部颜色分布构建一个可变大小的轴对齐包围盒。在定义的边界框内的上一帧图像的后投影和重采样信息与当前帧图像信息进行整合。作者的研究表明，通过明确使用局部色彩信息，重影走样现象得以减少。这种方法还进一步扩展到了眼球回旋运动。由于眼球的回旋运动，前一帧的周边区域可能成为当前帧的注视点区域。这种情况需要几帧的信息汇聚到注视点区域所需的细节水平。Patney等人[35]根据连续两帧的着色率加快了收敛速度。这就减少了眼球移动造成的模糊走样。

Weier等人[24]利用景深信息设计了一种后处理抗锯齿技术。聚焦物体时产生的景深效应可用作低通滤波器，以尽量减少外围区域的高频走样。首先对注视点渲染图像进行时间抗锯齿处理，以获得时间上平滑的样本。这些样本用于使用推拉插值法重建完整图像。然后使用低通景深滤波器减弱混叠走样。使用基于支持向量机的注视深度估算器估算注视点的大致深度，该估算器将各种深度测量值作为输入。根据注视深度和估计误差，计算出两个混淆圈，从而确定景深。根据景深模型设计了一个多层滤波器。对图像的过滤分层进行，最终以不同的权重进行混合，得到最终输出结果。

# 3 方法理论与设计

## 3.1 基于矩形内部映射的图像变换

### 3.1.1 一维空间下的采样点分布

在介绍基于二维矩形内部映射的图像变换之前，我们先从一维像素分布变换的概念开始。考虑由焦距为 f 的简单凸透镜成像的空间物体点。物体和图像距离透镜中心具有非线性关系，由原始的凸透镜方程给出。图像距离 di 随物体距离 do 的增加而增加；然而，如公式所示，di 的变化率是 f 和 do 的非线性函数。

注视点渲染方法一般会考虑渲染样本的非均匀分布，而传统渲染方法中的样本分布是均匀的。本文利用凸透镜方程来获得非均匀样本位置。本文考虑 do ∈ (0, f) 的情况，以确保映射函数的连续性和单调性。为了方便计算，我们重写等式。

随着 f 的减小，di 的分布更加集中在原点附近。假设显示屏的中心点、左边缘和右边缘分别位于原点、+1 和-1，则可以通过在原点周围分布更多样本、在图表, 直方图

描述已自动生成外围减少样本来实现有焦点的效果，然后再利用反函数将样本映射回规则空间。

图：不同的f下，像素的一维分布的映射变化

此时的像素分布已经初具了我们想要的性质——随着f的减小，像素在0点处的分布越密集。为了应用在图像区域的映射上，我们还需要将di的位置归一化到[-1,1]的区间内。

图表

中度可信度

图：归一化后的不同的f下，像素的一维分布的映射变化

### 3.1.2 二维空间下的采样点映射

现在，我们将这一想法扩展到二维笛卡尔坐标，并引入了基于矩形内部映射的注视点渲染框架。

这套映射算法需要结合延迟渲染管线来实现，该管道在生成几何图形缓冲区后执行一次光照计算。在传统延迟渲染管线和本注视点渲染管线中，顶点着色和片段着色阶段均保持不变。在本文的设计中，我们通过引入正向和逆向的矩形映射步骤来影响整个管线。

对于分辨率为 W×H 的全分辨率0号几何缓冲区，我们应用矩形内部映射图像变换，并将变换后的数据保存到分辨率为 w×h 的1号几何缓冲区中。为方便计算。对于给定注视点（x˚, y˚），我们将屏幕空间中笛卡尔坐标为 (x, y) 的每个像素，都映射到 (x’ , y’ )，其中

为简单起见，我们只列出 x > 0 和 y > 0 像素的映射方程。矩形映射通过方程 将0号几何缓冲区中的坐标 (x , y ) 转换为1号几何缓冲区中的坐标 (u, v)。

w

h

其中，fx 和 fy 分别控制像素在水平轴和垂直轴上的分布，NX () 和 NY () 是X 轴和 Y 轴的归一化函数。

负坐标像素的等式也同理可得。

在光照通道中，像素着色器使用1号几何缓冲区中的信息，计算每个像素的直接和间接光照，并将结果渲染到2号几何缓冲区中。屏幕的宽度（高度），与1、2号几何缓冲区的宽度（高度）之比为σ = W/w = H/h。着色成本与需要着色的像素数量呈线性关系。传统的延迟着色管道会着色 W × H 像素，而经过像素在矩形内部的映射，本渲染管线会着色 w×h 像素。因此，RMFR 在着色步骤中的理论提速 S 为

最后，渲染器将2号几何缓冲区中的渲染场景映射到全分辨率屏幕上，这正是上述过程的逆过程。具体来说，2号几何缓冲区中坐标为（u, v）的像素通过下面的公式转换回笛卡尔坐标中的（x, y）。

此时NU()和NV()的定义如下：

## 3.2 矩形与圆盘的区域映射

### 3.2.1 矩形内部映射存在的问题

上一节中的矩形内部映射，有若干出色的性质。其可以有效地将XY轴的操作解耦。因为其图像变换具有一定的保角性，可以很大程度上的避免锯齿的产生。

但是如果研究其变换后的图像，对原图像像素的采样密度。则会发现，相比于缓冲区的对角线方向，其在XY轴方向上有明显的过采样现象。如果我们在几何缓冲区上画一个以注视点为圆心的圆，作为辅助线。可以计算得出相同的偏心率、相同的fx与fy下，XY轴向的采样密度为0.6，而对角线方向的采样密度为0.29。这样的各向异性显然与任何一种视觉敏锐度都是不相符的，势必会因为局部过采样，导致全局的无效计算变多。

图表, 表面图

描述已自动生成图：矩形内部映射后愿图像像素的采样密度（临时）

图示, 示意图

描述已自动生成

图：人眼的视觉敏锐度模型

### 3.2.2矩形-圆盘映射的作用与所需性质

为了取得更好的效果，并且保持矩形内部映射的良好性质，本文的解决方法是，加入一个矩形-圆盘映射的中间模块，使变换后的几何缓冲区的采样密度，更符合真实人眼的视觉敏锐度。

瀑布图

描述已自动生成图：完整的几何缓冲区映射方案

如图，展示了加入矩形-圆盘映射后的延迟着色管线。矩形-圆盘映射，指的是正方形区域和圆盘区域的正向和逆向映射，在渲染管线中的体现是，几何缓冲区在矩形内部映射结束后，进行一次正向的矩形-圆盘映射。在着色阶段完成后，矩形内部映射前，再进行一次逆向的矩形-圆盘映射。因为着色器的处理单元是有待变换的每一个像素，所以，我们用关于x、y、u、v的解析表达式来描述，圆盘上的点(u,v)是如何被映射到正方形区域上的点(x,y)的。

使用这个模块，需要先讨论一些特性。在我们的映射中，我们希望尽量与原图相似，减少图像扭曲。微分几何中有两个度量标准是形状失真和尺寸失真。形状失真以变换前后的局部角度变化来衡量，而尺寸失真以变换前后的局部面积变化来衡量。在微分几何术语中，保持角度的变换特性成为保角性（conformal），保持尺寸的变换特性称为保面性（equiareal）。

图：保角映射与保面映射的例子。图示

中度可信度描述已自动生成

还有一个映射的理想特性是径向性。直观地说，就是点在映射过程中只能沿着圆盘中心的径向线移动。用数学语言来表达，即点（u,v）与 x 轴的夹角与点（x,y）的夹角相同。

在本文的场景中，我们希望所使用的变换具有良好的保形性和径向性，而在对角线方向和XY轴向上，其保面性应具有差异。这样才能在保持图像较小的扭曲的同时，对上一模块中过采样的局部区域进行压缩。

图片包含 体育, 运动, 游戏机

描述已自动生成图：矩形图表

描述已自动生成-圆盘映射效果图1

图：矩形-圆盘映射效果图2

### 3.2.3 基于费尔南德斯圆角矩形的映射

#### 3.2.3.1 圆角矩形的解析式

这里需要引入一种新的几何图形——圆角方形。用于表示介于圆形和方形之间的形状。

最经典的圆角方形是拉美曲线，是法国科学家拉梅（Gabriel Lame）首先发现和提出的，所以也叫拉梅曲线。拉梅曲线的数学方程如下：

用参数方程来表示则如下：

图表

描述已自动生成

图、不同n值（指数值）下的拉美曲线

但是求解析解十分困难，相比之下费尔南德斯圆角方形方程就更具有代数简洁性。他的方程包含一个参数 s，可用来平滑地混合圆形和方形。其方程表示如下：

下面简述其推导过程。首先观察到：

代表x=1和x=-1两条直线。可以使用两个二项式平方根项的乘积来表示正方形：

当 -r ≤ x ≤ r 且 -r ≤ y ≤ r 时，可以绘制成笛卡尔平面中的正方形。然后简化方程式，得到：

即：

当 -r ≤ x ≤ r 且 -r ≤ y ≤ r 时，这是笛卡尔平面中正方形的有效多项式表示。通过在方程中引入方形参数 s（取值区间为[0,1]），可以生成在正方形和圆形之间插值的形状：

即：

尽管莱姆曲线与费尔南德斯曲线在质量上有相似之处，但两者之间存在一个主要区别。

莱姆曲线只能逼近正方形。此外，由于莱姆曲线的多项式方程具有无界指数，因此它难以代数操作。相反，费尔南德斯曲线是低阶四次曲线。此外，由于其多项式方程中没有三次项或线性项，因此费尔南德斯曲线本质上是一条更简单的双二次曲线。这使得它更适合代数操作和简化。

#### 3.2.3.2 连续体之间的映射

对于上文得到的圆角矩形表达式，我们可以了令s = r ，可以得到:

在这个等式中。我们有一个参数s，可以控制圆角矩形的正方形程度和大小。从视觉上说，我们可以看到一系列不断“变大”“变方”的形状，它们在 s=0 时以一个点开始，在 s=1 时以一个正方形结束。

我们将这一系列变化的形状组成一个正方形区域。记为：

而对于我们需要的单位圆盘区域，可以看作是半径从0到1的同心圆组成的连续体，从而，可以将单位圆盘记为：

这里，t的意义为点（u,v）到原点的距离。

我们可以通过将单位圆盘内部的每一个圆形轮廓，映射到正方形内部的每一个圆角矩形轮廓，来建立单位圆盘和正方形区域之间的对应关系。我们令两组参数t相等，即可得到等式：

这个方程将作为求解圆盘-矩形映射的条件之一。

#### 3.2.3.3 圆盘到矩形的映射表达式的求解

求解圆盘-矩形映射表达式需要的另一个条件是径向约束：

将t代换，可以得到：

这就是从圆盘到矩形的映射表达式。具有良好的径向特性和保形性。

#### 3.2.3.3矩形到圆盘的映射表达式的求解

现在来推导逆向的映射表达式，即从矩形映射到圆盘。首先使用径向约束，我们可以得到一个x与y的关系式。

即：

将其带入另一个表达式条件：

整理成x的多项式即：

在这个关于x的四次方程中，没有三次项和线性项。所以可以作为关于的二元方程求解。

用径向约束条件的替换x，得到y的表达式：

### 3.2.4 计算优化

上一节中，我们在圆角矩形方程中，

令s=t，得到了描述如下的正方形：

而这只是一种特例，其绝非此类圆角矩形的唯一映射方式。

我们发现，费尔南德斯圆角矩形可以在圆形与正方形之间插值的关键是，在r接近0的时候，s也接近0；在r接近1的时候，s也接近1。得到这样性质的关键在于，是一条过（0,0）和（1,1）点的单调增函数。

令 s = ，此时的映射方程为：

将分离到等式的一边并开根号：

将t带入表达径向映射的参数方程：

化简可得u、v分别关于x、y的表达式，即正方形域到圆域的映射：

对于逆向的映射，与上一节中的计算类似，也是先得到一个以u、v为参数的，关于x的四次方程：

同样是没有三次项和

最终得到圆域到正方形域的映射表达式：

因为是上一节中圆域-正方形域映射方程的变形版本，所以其性质与结果类似，仅凭人眼无法直观地区分两种映射的效果，但是其解在计算上更简单。

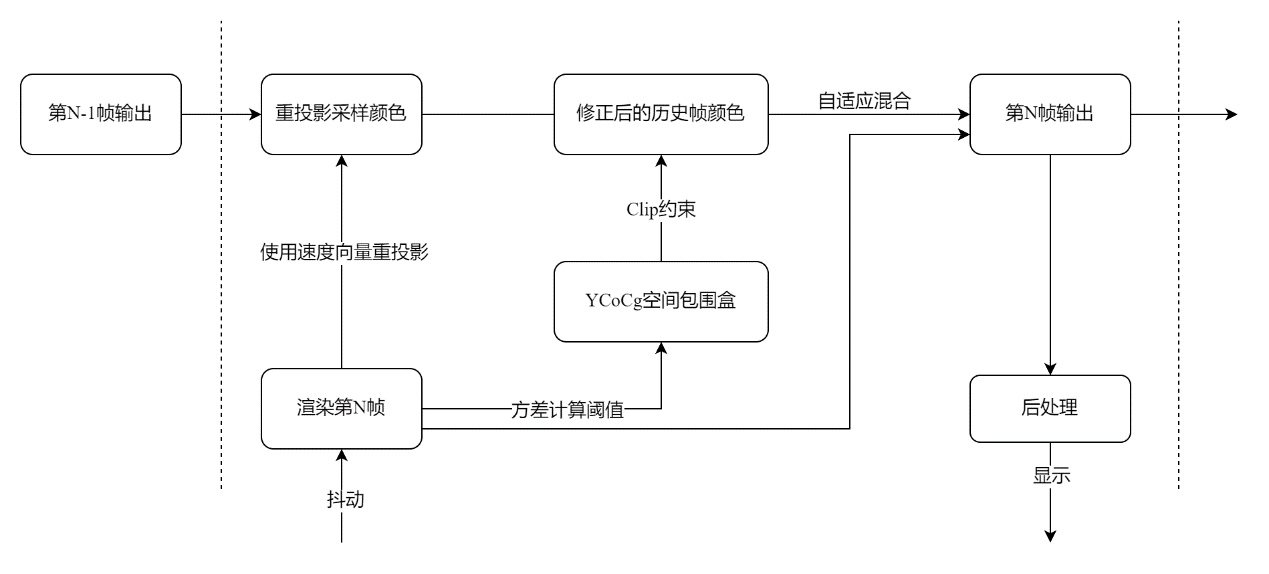
## 3.3 帧间反走样

粗糙的外围层的伪影，会在场景运动时产生时间走样，很容易令人分心。这些混叠着的伪影，如沿着锐利边缘或较薄的高光层移动，表现尤其糟糕，因为眼睛会被吸引过去。而增加采样密度这种暴力的解决方法，又会抵消我们先前通过对几何着色器做映射而带来的性能提升。除了需要高效之外，抗锯齿的方法还应具有通用性，只需很少或无需修改着色器或几何图形，从而使现有的图形应用软件更容易实现凹陷化。

所以，根据人眼对闪烁的高度敏感性，和人眼视觉外围对模糊的低敏感性，我们独特地设计了一种改进帧间凡走样算法，该算法通过使用方差采样及其它一些创新策略，提供了更好的视觉体验。

首先，高光闪烁问题的出现主要是由于光线在物体表面反射产生的亮斑，具有高频率和高对比度的特性。在光栅化显示时，因为较低的采样率（尤其是上一节中经过映射后的几何缓冲区，其使得外围像素对场景的采样率更低），所以会出现高光闪烁的现象。传统的各类反走样和滤波的方法提供了一种四舍五入的解决方案，这在消除锯齿和一些细微的闪烁上是有效的。但是，当应对更具有强对比度和高频率的闪烁问题时，其效果难以满足。对于高光闪烁，最大程度的消除闪烁效果是我们所需要的。

如图，我们所提出的TAA算法，通过当前帧与前一帧的图像信息计算，合并产生新的帧，独特的是，在标准TAA计算中并入了方差优化，尽可能地减少图像的尾影效果。其源自 Karis [2014] 所描述的算法，但引入了可变大小的采样和对历史帧更为严格的纠正。以此来消除高光闪烁。同时，利用人眼的生理特性，使得轻微的模糊在外围视觉中不会产生显著的不适感。

图：注视点渲染专用的帧间反走样流程图

### 3.3.1背景介绍

重复使用先前帧的样本需要进行时间重投影，以保持图像的稳定性。Guenter 等人的研究[2012]依赖于这种投影。最先进的时间抗锯齿（TAA）变体[Karis 2014；Jimenez 等人，2012]具有最小的计算和内存开销，并能显著改善图像质量。TAA 可以进行调整，以减少各种噪声和走样，包括高光走样、几何走样、随机噪声。作为减少混叠的交换条件，它往往会去除高频细节或引入重影，因为前一帧的细节在可以正确再投影后仍然存在。

持续整合前一帧的色彩样本是摊销超采样成本的常用方法[Yang 等人，2009]，但在使用陈旧样本时很容易产生重影走样。Karis [2014] 的 TAA 技术通过调节与当前帧样本不一致的前一帧色彩样本来减少重影。

首先，该方法在色彩空间中计算出一个轴对齐的边界框，其中包含当前帧中局部3×3像素窗口内的所有色彩样本。3×3 窗口以当前像素为中心。当前像素颜色记为。其次，将上一帧的反向投影和重新采样的颜色夹在该边界框中。从上一帧重新使用的数据如果被包围在边界框内，则不会受到影响。相反，经过截断操作后，重新使用的数据将位于边界框的边缘，从而与当前的色彩数据更加一致。最后，通过使用指数移动平均法对新的着色数据和截断后的颜色样本进行整合，实现时间超采样：

这里的参数α一般使用0.1。也就是说，当我们对上一帧的像素颜色没有做修正时，当前帧的像素颜色的90%来自上一帧。

### 3.3.2 方差采样

上述方法在游戏中已经得到了广泛的使用，但仍存在明显的重影走样问题。这是由于离群的色彩样本大大扩展了边界框，使其无法很好地呈现局部色彩分布。此外，以往的注视点渲染技术也不能很好地与这种方法配合使用，因为色彩样本会被复制到许多像素上，从而削弱了我们对局部色彩分布的感知。在更大的像素窗口上计算色彩空间边界框，能显著改善我们对局部色彩分布的了解，但成本高昂。

因此，我们引入了方差采样。

这样，我们就能得出分布均值：

和标准偏差 ：

从而生成一个以为中心的包围盒，包围盒的范围由确定。这种方法有两大好处。首先，我们的色彩边界框倾向于自然排除异常值，它能更好地包围当前帧色彩样本所在的色彩空间区域，与之前的方法相比，能有效减少重影走样，同时不增加计算和内存带宽成本。其次，分布的一阶原始矩是线性量，可以在大图像区域内进行预过滤。这一特性使得多尺度TAA对采样不足的图像区域，可以有效地确定其统计特性，并提高其时间稳定性。

图：包围盒对历史颜色的Clip约束和Clamp约束的区别图表, 折线图

描述已自动生成

### 3.3.3 减少时间不稳定性

TAA 的主要目标之一是减少帧的时间不稳定性，如高光闪烁或摩尔纹伪影。由于在每一帧中都会对视口应用不同的抖动偏移，因此即使摄像机保持静止不动，为像素计算的着色样本也会在每一帧中发生变化。通常情况下，样本累积步骤（当前帧与上一帧混合）会吸收这些差异，从而提供正确滤波的像素。但是，历史记录剔除或纠正算法，可能会因抖动引起的样本颜色变化，而错误地

建筑的摆设布局

中度可信度描述已自动生成图：注视点渲染尚存在的时间不稳定性问题

【此处应有图：Bloom下的高光的不稳定性】

使历史记录失效或截断历史记录。下图显示了这样一个例子：低频莫伊纹会导致某些像素附近的色域完全改变，从而迫使历史色彩在每一帧中都被clip或clamp到新的采样值，从而丢弃了历史结果。禁用或放宽历史校正可以避免闪烁（此处应有图，无闪烁的对比图），但可能会出现重影。事实上，在设计历史记录剔除或纠正算法时，避免重影和闪烁往往是相互矛盾的。

导致时间不稳定性的根本问题与前面介绍的问题的原因类似：采样不足的输入没有足够的信息来可靠地检测历史的有效性。有效历史被视为无效，会导致时间不稳定性；无效历史被视为有效，又会导致重影。

Karis在历史样本接近截断时减少混合因子α，以缓和截断发生后的时间变化。这种启发式的算法可以良好地减少时间不稳定性。

# 4 系统实现

## 4.1 延迟渲染管线

### 4.1.1 几何缓冲区的填充

这一节我们要完成填充Gbuffer的着色器。这里我们需要填充的既不是几何数据，也不是光照计算的结果，而是光栅化后，每个片元所需要的着色计算的“材料”。

在本文的实现中，我们的这个“材料”是四个缓冲区。因为Unity已经实现并封装了Multiple Render Target技术，我们只需要通过Unity Shader的SV\_Target语义来将结果输出到四个目标即可。

1. void frag (
2. v2f i,
3. out float4 GT0 : SV\_Target0,
4. out float4 GT1 : SV\_Target1,
5. out float4 GT2 : SV\_Target2,
6. out float4 GT3 : SV\_Target3
7. )
8. { … }

#### 4.1.1.1 几何缓冲区0

第一个 G 缓冲区用于存储漫反射反照率和表面遮挡。它是一个 ARGB32 纹理，就像常规帧缓冲区一样。反照率存储在 RGB 通道中，遮挡存储在 A 通道中。此时我们知道反照率颜色，我们可以使用它GetOcclusion来访问遮挡值。

1. output.gBuffer0.rgb = albedo;
2. output.gBuffer0.a = GetOcclusion(i);

我们可以使用Unity内置的帧调试器来检查第一个几何缓冲区的内容，以验证其是否被正确地填充。这将向您显示它的 RGB 通道。但是，未显示 A 通道。要检查遮挡数据，您可以将其临时分配给 RGB 通道。

#### 4.1.1.2 几何缓冲区1

第二个 G 缓冲区用于存储 RGB 通道中的镜面反射颜色以及 A 通道中的平滑度值。它也是一个 ARGB32 纹理。我们知道镜面反射色调是什么，并且可以用来GetSmoothness检索平滑度值。

1. output.gBuffer1.rgb = specularTint;
2. output.gBuffer1.a = GetSmoothness(i);

#### 4.1.1.3 几何缓冲区2

第三个 G 缓冲区包含世界空间法线向量。它们存储在 ARGB2101010 纹理的 RGB 通道中。这意味着每个坐标使用十位存储，而不是通常的八位，因为法线需要更加精确的数值。A 通道只有两位，所以总数又是 32 位，但我们不使用它，所以我们将其设置为 1。我们按常规法线贴图的格式进行编码。

1. output.gBuffer2 = float4(i.normal \* 0.5 + 0.5, 1);

#### 4.1.1.4 几何缓冲区3

最后的几何缓冲区用于存储场景的自发光。其格式取决于相机设置为 LDR 还是HDR。对于LDR，它是一个ARGB2101010纹理，就像法线的缓冲区一样。启用HDR后，格式为ARGBHalf，每个通道存储16位浮点值，总共64位，因为之后如果要支持一些高级光照特性比如Bloom ，会需要更高的精度。因此 HDR版本的缓冲区是其他缓冲区的两倍。仅使用RGB通道，因此可以将A通道再次设置为1。

如果我们不使用 A 通道，则意味着每像素 16 位未被使用。而每像素其实只需要 48 位，而不是 64 位。

我们使用 ARGBHalf 的原因是大多数 GPU 都使用四个字节的块。大多数纹理是每像素 32 位，对应于一个块。64 位需要两个块，所以这也可以。但 48 位对应 1.5 个块。这会导致错位，通过对 48 位使用两个块可以防止错位。这会导致每个像素有 16 位填充，与 ARGBHalf 相同。

存储法线的几何缓冲区2使用 ARGB2101010 的原因相同。两个未使用的位是起到填充的作用。RGB24 纹理通常也以 ARGB32 形式存储在 GPU 内存中。

### 4.1.2 基于PBR的延迟着色

## 4.2 几何缓冲区的映射

## 4.3 反走样的实现

### 4.3.1 抖动当前帧

帧间反走样的第一步是渲染带有抖动的场景。此步骤中渲染场景的方式与正常渲染场景的方式几乎相同。唯一的区别是场景在垂直和水平方向上略有移动，移动长度在一个像素长度之内，也就是所谓的抖动（Jitter）。

为此，顶点着色器稍作修改。在顶点着色器中，计算裁剪空间位置后，将抖动偏移添加到裁剪空间位置。

1. float4 worldPos  = modelMatrix \* float4(in.position, 1.0);
2. float4 clipPos = viewportParams.viewProjectionMatrix \* worldPos;
3. clipPos += viewportParams.jitter\*clipPos.w; // Apply Jittering
4. out.position = clipPos;

每个帧使用与其前一帧不同的抖动偏移。并且，在一帧内，整个场景都使用相同的抖动偏移进行渲染。

至于我们的抖动偏移量，有许多种选择。目前流行的方案是（x，y）的halton序列（2，3）。其被认为是一种低差异化序列。

1. **private** **static** **float** HaltonSeq(**int** prime, **int** index = 1/\* NOT! zero-based \*/)
2. {
3. **float** r = 0.0f;
4. **float** f = 1.0f;
5. **int** i = index;
6. **while** (i > 0)
7. {
8. f /= prime;
9. r += f \* (i % prime);
10. i = (**int**)Mathf.Floor(i / (**float**)prime);
11. }
12. **return** r;
13. }

我们用halton序列生成每帧的抖动。Halton序列的数字范围在0到1之间。为了使用这些数字作为抖动的偏移量，我们需要调整范围，使位置在正向和负向都抖动，并且抖动不超过大小像素。我们可以使用以下代码来更改范围。

1. **for** (**int** i = 0, n = seq.Length / 2; i != n; i++)
2. {
3. **float** u = HaltonSeq(2, i + 1) - 0.5f;
4. **float** v = HaltonSeq(3, i + 1) - 0.5f;
5. seq[2 \* i + 0] = u;
6. seq[2 \* i + 1] = v;
7. }

【此处应有图，采样图】

### 4.3.2 解析历史帧

为了在动态场景上使用 TAA，我们需要生成另一个存储每个像素的运动的缓冲区。其每个纹素存储的是每个像素的当前帧位置和前一帧位置之间的增量。该数据使我们能够计算前一帧中像素的位置。

为什么我们需要这些数据？这是因为场景每一帧都在移动。在解析颜色的步骤中，我们需要从历史纹理中获取前一帧中像素的颜色。为此，我们需要知道像素在前一帧中的位置。我们可以使用该速度数据来计算像素之前的位置。

为了计算像素间的运动，我们需要前一帧的观察矩阵和投影矩阵。最重要的是，我们还需要前一帧中每个模型的世界矩阵。我们暂且不考虑使用骨骼蒙皮的对象。

利用前一帧的世界、观察、投影矩阵，我们可以在顶点着色器中计算当前帧像素和其在前一帧的像素在裁剪空间中的顶点位置，如下所示。这种计算称为“重投影”。

1. OUT.cs\_pos = mul(mul(STEREO\_INDEX(\_CurrVP), \_CurrM), ws\_pos\_curr);
2. OUT.ss\_pos = ComputeScreenPos(OUT.cs\_pos);
3. OUT.ss\_pos.z = -mul(mul(STEREO\_INDEX(\_CurrV), \_CurrM), ws\_pos\_curr).z - occlusion\_bias;// COMPUTE\_EYEDEPTH
4. OUT.cs\_xy\_curr = OUT.cs\_pos.xyw;
5. OUT.cs\_xy\_prev = mul(mul(STEREO\_INDEX(\_PrevVP), \_PrevM), ws\_pos\_prev).xyw;

现在，我们有顶点着色器为每个顶点生成当前帧位置和前一帧位置，我们可以使用像素着色器中的位置信息来生成每像素速度数据。我们可以对历史纹理坐标进行简单检查，看看它是否超出边界(0,1)。如果是，我们就在这里停止并返回过滤后的源样本。

1. float4 frag(v2f IN) : SV\_Target
2. {
3. float2 ss\_txc = IN.ss\_pos.xy / IN.ss\_pos.w;
4. **float** scene\_d = depth\_sample\_linear(ss\_txc);
6. // discard if occluded
7. clip(scene\_d - IN.ss\_pos.z);
9. // compute velocity in ndc
10. float2 ndc\_curr = IN.cs\_xy\_curr.xy / IN.cs\_xy\_curr.z;
11. float2 ndc\_prev = IN.cs\_xy\_prev.xy / IN.cs\_xy\_prev.z;
13. // compute screen space velocity [0,1;0,1]
14. **return** float4(0.5 \* (ndc\_curr - ndc\_prev), 0.0, 0.0);
15. }

分别获得当前帧与对应历史帧的像素：

1. float4 texel0 = sample\_color(\_MainTex, ss\_txc);
2. float4 texel1 = sample\_color(\_PrevTex, ss\_txc - ss\_vel);

### 4.3.3 修正历史帧

我们使用方差计算aabb盒的大小：

1. float oneDividedBySampleCount = 1.0 / 9.0;
2. float gamma = 1.0;
3. float3 mu = m1 \* oneDividedBySampleCount;
4. float3 sigma = sqrt(abs((m2 \* oneDividedBySampleCount) - (mu \* mu)));
5. float3 minc = mu - gamma \* sigma;
6. float3 maxc = mu + gamma \* sigma;
8. historySample = clip\_aabb(minc, maxc, clamp(historySample, neighborhoodMin, neighborhoodMax));

对于历史帧的修正，我们使用inside团队在GDC2016中的方案：

1. float4 clip\_aabb(float3 aabb\_min, float3 aabb\_max, float4 p, float4 q)
2. {
3. float4 r = q - p;
4. float3 rmax = aabb\_max - p.xyz;
5. float3 rmin = aabb\_min - p.xyz;
7. const float eps = FLT\_EPS;
9. if (r.x > rmax.x + eps) r \*= (rmax.x / r.x);
10. if (r.y > rmax.y + eps) r \*= (rmax.y / r.y);
11. if (r.z > rmax.z + eps) r \*= (rmax.z / r.z);
12. if (r.x < rmin.x - eps) r \*= (rmin.x / r.x);
13. if (r.y < rmin.y - eps) r \*= (rmin.y / r.y);
14. if (r.z < rmin.z - eps) r \*= (rmin.z / r.z);
16. return p + r;
17. }

## 4.4 采样密度的计算与可视化

# 5 实验

## 5.1 客观实验

## 5.2 用户实验与分析

### 5.2.1 实验概述

### 5.2.2 实验流程

### 5.2.3 实验人员

### 5.2.1 愉悦度

### 5.2.2 胜任感

### 5.2.3 努力感、重要性

### 5.2.4压力、紧张

# 6 结论

本研究成功提出并实现了一套基于像素分布匹配的注视点渲染研究框架。通过创新性地结合了视觉模型与映射技术，并设计了一套有效的帧间反走样算法来处理高光闪烁等问题，显著提升了渲染性能。实验结果验证了方法的有效性，并展现了该方法在节省计算资源和提升视觉体验方面的潜力。未来工作将聚焦于进一步优化性能、增强用户体验以及探索新的应用场景。

## 6.1 工作总结

①分析现有视觉模型：对视觉敏锐度、对比度、外围运动和闪烁感知等视觉因素进行深入研究，并分析了基于偏心率的视觉模型，探索适合于注视点渲染的模型。

②设计适应性映射算法：基于矩形内部映射的图像变换理论进行映射设计，进而研究矩形与圆盘的区域映射策略，并对映射过程的计算性能进行优化。

③设计帧间反走样处理：提出并设计帧间反走样技术，用以解决因采样率变化产生的摩尔纹和高光闪烁等问题。

④实现完整的注视点渲染系统：在延迟渲染管线上实践上述理论与设计，实现几何缓冲区的映射和注视点渲染技术，并针对性地对反走样算法进行集成和优化。

⑤进行效果测试和性能分析：通过实验验证注视点渲染技术的有效性，包括客观实验和用户实验，以及对实验结果进行分析和讨论。

## 6.2 未来工作展望

# 参考文献

1. A
2. A
3. A
4. A
5. A
6. A
7. A
8. A
9. A
10. A
11. Sun, Xuetong and Amitabh Varshney. “Investigating perception time in the far peripheral vision for virtual and augmented reality.” Proceedings of the 15th ACM Symposium on Applied Perception (2018): n. pag.
12. Dale Purves, George J Augustine, David. Fitzpatrick, Lawrence C Katz, Anthony-Samuel LaMantia, James O McNamara, and S Mark Williams. 2001. Neuroscience. 2nd edition. Sunderland (MA): Sinauer Associates. https://www.ncbi.nlm.nih.gov/ books/NBK10799/
13. Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016b. Towards Foveated Rendering for Gazetracked Virtual Reality. ACM Trans. Graph. 35, 6, Article 179 (Nov. 2016), 12 pages. https://doi.org/10.1145/2980179.2980246
14. Thorsten Hansen, Lars Pracejus, and Karl R. Gegenfurtner. 2009a. Color perception in the intermediate periphery of the visual field. Journal of Vision 9, 4 (04 2009), 26–26. https://doi.org/10.1167/9.4.26 arXiv:https://arvojournals.org/arvo/content\_public/journal/jov/933534/jov9-4-26.pdf
15. Ruth Rosenholtz. 2016. Capabilities and Limitations of Peripheral Vision. Annual Review of Vision Science 2, 1 (2016), 437–457. https://doi.org/10.1146/annurevvision-082114-035733 arXiv:https://doi.org/10.1146/annurev-vision-082114-035733 PMID: 28532349.
16. Holger Ludvigsen and Anne Cathrine Elster. 2010. Real-Time Ray Tracing Using Nvidia OptiX. In Eurographics 2010 - Short Papers, H. P. A. Lensch and S. Seipel (Eds.). The Eurographics Association. https://doi.org/10.2312/egsh.20101049
17. S.M. Anstis. 1974. A chart demonstrating variations in acuity with retinal position. Vision Research 14, 7 (1974), 589–592. <https://doi.org/10.1016/0042-6989(74)90049-2>
18. Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. 2012. Foveated 3D Graphics. ACM Trans. Graph. 31, 6, Article 164 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366183
19. Nicholas T. Swafford, José A. Iglesias-Guitian, Charalampos Koniaris, Bochang Moon, Darren Cosker, and Kenny Mitchell. 2016. User, Metric, and Computational Evaluation of Foveated Rendering Methods. In Proceedings of the ACM Symposium on Applied Perception (Anaheim, California) (SAP ’16). ACM, New York, NY, USA, 7–14. <https://doi.org/10.1145/2931002.2931011>
20. Yong He, Yan Gu, and Kayvon Fatahalian. 2014. Extending the Graphics Pipeline with Adaptive, Multi-rate Shading. ACM Trans. Graph. 33, 4, Article 142 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601105
21. Matias Koskela, Atro Lotvonen, Markku Mäkitalo, Petrus Kivi, Timo Viitanen, and Pekka Jääskeläinen. 2019b. Foveated Real-Time Path Tracing in Visual-Polar Space. In Eurographics Symposium on Rendering - DL-only and Industry Track, Tamy Boubekeur and Pradeep Sen (Eds.). The Eurographics Association. https://doi.org/10.2312/sr. 20191219
22. Xiaoxu Meng, Ruofei Du, Matthias Zwicker, and Amitabh Varshney. 2018. Kernel Foveated Rendering. Proc. ACM Comput. Graph. Interact. Tech. 1, 1, Article 5 (July 2018), 20 pages. <https://doi.org/10.1145/3203199>
23. Okan Tarhan Tursun, Elena Arabadzhiyska-Koleva, Marek Wernikowski, Radosław Mantiuk, Hans-Peter Seidel, Karol Myszkowski, and Piotr Didyk. 2019. Luminancecontrast-aware Foveated Rendering. ACM Trans. Graph. 38, 4, Article 98 (July 2019), 14 pages. <https://doi.org/10.1145/3306346.3322985>
24. Martin Weier, Thorsten Roth, Ernst Kruijff, André Hinkenjann, Arsène Pérard-Gayot, Philipp Slusallek, and Yongmin Li. 2016. Foveated Real-Time Ray Tracing for Head-Mounted Displays. Comput. Graph. Forum 35, 7 (Oct. 2016), 289–298. https: //doi.org/10.1111/cgf.13026
25. Masahiro Fujita and Takahiro Harada. 2014. Foveated real-time ray tracing for virtual reality headset. Light Transport Entertainment Research (2014).
26. Josef Spjut and Ben Boudaoud. 2019. Foveated Displays: Toward Classification of the Emerging Field. In ACM SIGGRAPH 2019 Talks (Los Angeles, California) (SIGGRAPH ’19). ACM, New York, NY, USA, Article 57, 2 pages. https://doi.org/10.1145/3306307. 3328145
27. Jared Abrams, Aaron Nizam, and Marisa Carrasco. 2012. Isoeccentric locations are not equivalent: The extent of the vertical meridian asymmetry. Vision Research 52, 1 (2012), 70–78. https://doi.org/10.1016/j.visres.2011.10.016
28. Antoine Barbot, Shutian Xue, and Marisa Carrasco. 2021. Asymmetries in visual acuity around the visual field. Journal of Vision 21, 1 (01 2021), 2–2. <https://doi.org/10.1167/jov.21.1.2> arXiv:https://arvojournals.org/arvo/content\_public/journal/jov/938512/i1534- 7362-21-1-2\_1609762263.01982.pdf
29. David R. Bull. 2014. Chapter 2 - The Human Visual System. In Communicating Pictures, David R. Bull (Ed.). Academic Press, Oxford, 17 – 61. https://doi.org/10.1016/B978- 0-12-405906-1.00002-7
30. Peter G. J. Barten. 2003. Formula for the contrast sensitivity of the human eye. In Image Quality and System Performance, Yoichi Miyake and D. Rene Rasmussen (Eds.), Vol. 5294. International Society for Optics and Photonics, SPIE, 231 – 238. <https://doi.org/10.1117/12.537476>
31. J.S. Pointer and R.F. Hess. 1989. The contrast sensitivity gradient across the human visual field: With emphasis on the low spatial frequency range. Vision Research 29, 9 (1989), 1133 – 1151. https://doi.org/10.1016/0042-6989(89)90061-8
32. Larry N. Thibos, David L. Still, and Arthur Bradley. 1996. Characterization of spatial aliasing and contrast sensitivity in peripheral vision. Vision Research 36, 2 (1996), 249 – 258. https://doi.org/10.1016/0042-6989(95)00109-D
33. R. S. Anderson, M. B. Zlatkova, and R. O. Beirne. 2002. The contrast sensitivity function for detection and resolution of blue-on-yellow gratings in foveal and peripheral vision. Ophthalmic and Physiological Optics 22, 5 (2002), 420–426. https://doi.org/10.1046/j.1475-1313.2002.00068.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1475-1313.2002.00068.x
34. Anjul Patney, Joohwan Kim, Marco Salvi, Anton Kaplanyan, Chris Wyman, Nir Benty, Aaron Lefohn, and David Luebke. 2016a. Perceptually-based Foveated Virtual Reality. In ACM SIGGRAPH 2016 Emerging Technologies (Anaheim, California) (SIGGRAPH ’16). ACM, New York, NY, USA, Article 17, 2 pages. https://doi.org/10.1145/2929464. 2929472
35. Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016b. Towards Foveated Rendering for Gazetracked Virtual Reality. ACM Trans. Graph. 35, 6, Article 179 (Nov. 2016), 12 pages. <https://doi.org/10.1145/2980179.2980246>
36. Okan Tarhan Tursun, Elena Arabadzhiyska-Koleva, Marek Wernikowski, Radosław Mantiuk, Hans-Peter Seidel, Karol Myszkowski, and Piotr Didyk. 2019. Luminancecontrast-aware Foveated Rendering. ACM Trans. Graph. 38, 4, Article 98 (July 2019), 14 pages. <https://doi.org/10.1145/3306346.3322985>
37. Thorsten Hansen, Lars Pracejus, and Karl R. Gegenfurtner. 2009b. Color perception in the intermediate periphery of the visual field. Journal of Vision 9, 4 (04 2009), 26–26. https://doi.org/10.1167/9.4.26 arXiv:https://arvojournals.org/arvo/content\_public/journal/jov/933534/jov9-4-26.pdf
38. Sheng Liu and Hong Hua. 2008. Spatialchromatic Foveation for Gaze Contingent Displays. In Proceedings of the 2008 Symposium on Eye Tracking Research & Applications (Savannah, Georgia) (ETRA ’08). Association for Computing Machinery, New York, NY, USA, 139–142. <https://doi.org/10.1145/1344471.1344507>
39. Leonard A Levin, Siv FE Nilsson, James Ver Hoeve, Samuel Wu, Paul L Kaufman, and Albert Alm. 2011. Adler’s Physiology of the Eye E-Book: Expert Consult-Online and Print. Elsevier Health Sciences.
40. Youngmin Kim, Amitabh Varshney, David W. Jacobs, and François Guimbretière. 2010. Mesh Saliency and Human Eye Fixations. ACM Trans. Appl. Percept. 7, 2, Article 12 (Feb. 2010), 13 pages. <https://doi.org/10.1145/1670671.1670676>
41. Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. 2005. Mesh Saliency. In ACM SIGGRAPH 2005 Papers (Los Angeles, California) (SIGGRAPH ’05). Association for Computing Machinery, New York, NY, USA, 659–666. https://doi.org/10.1145/ 1186822.1073244
42. Ran Song, Yonghuai Liu, Ralph R. Martin, and Paul L. Rosin. 2014. Mesh Saliency via Spectral Processing. ACM Trans. Graph. 33, 1, Article 6 (Feb. 2014), 17 pages. <https://doi.org/10.1145/2530691>
43. Shivanthan Yohanandan, Andy Song, Adrian G. Dyer, and Dacheng Tao. 2018. Saliency Preservation in Low-Resolution Grayscale Images. In Computer Vision – ECCV 2018, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.). Springer International Publishing, Cham, 237–254.
44. Yong He, Yan Gu, and Kayvon Fatahalian. 2014. Extending the Graphics Pipeline with Adaptive, Multi-rate Shading. ACM Trans. Graph. 33, 4, Article 142 (July 2014), 12 pages. <https://doi.org/10.1145/2601097.2601105>
45. Michael Stengel, Steve Grogorick, Martin Eisemann, and Marcus Magnor. 2016. Adaptive Image-Space Sampling for Gaze-Contingent Real-time Rendering. Comput. Graph. Forum 35, 4 (July 2016), 129–139. <https://doi.org/10.1111/cgf.12956>
46. Frank W. Weymouth. 1958. Visual Sensory Units and the Minimal Angle of Resolution\*. American Journal of Ophthalmology 46, 1, Part 2 (1958), 102 – 113. https://doi.org/ 10.1016/0002-9394(58)90042-4
47. Yong He, Yan Gu, and Kayvon Fatahalian. 2014. Extending the Graphics Pipeline with Adaptive, Multi-rate Shading. ACM Trans. Graph. 33, 4, Article 142 (July 2014), 12 pages. <https://doi.org/10.1145/2601097.2601105>
48. K. Vaidyanathan, M. Salvi, R. Toth, T. Foley, T. Akenine-Möller, J. Nilsson, J. Munkberg, J. Hasselgren, M. Sugihara, P. Clarberg, T. Janczak, and A. Lefohn. 2014. Coarse Pixel Shading. In Proceedings of High Performance Graphics (Lyon, France) (HPG ’14). Eurographics Association, Goslar Germany, Germany, 9–18. http://dl.acm.org/ citation.cfm?id=2980009.2980011
49. J. Spjut, B. Boudaoud, J. Kim, T. Greer, R. Albert, M. Stengel, K. Akşit, and D. Luebke. 2020. Toward Standardized Classification of Foveated Displays. IEEE Transactions on Visualization and Computer Graphics 26, 5 (2020), 2126–2134. https://doi.org/10. 1109/TVCG.2020.2973053
50. H. Araujo and J. M. Dias. 1996. An introduction to the log-polar mapping. In Proceedings II Workshop on Cybernetic Vision. 139–144.
51. Marco Antonelli, Francisco D. Igual, Francisco Ramos, and V. Javier Traver. 2015. Speeding up the log-polar transform with inexpensive parallel hardware: graphics units and multi-core architectures. Journal of Real-Time Image Processing 10, 3 (01 Sep 2015), 533–550. <https://doi.org/10.1007/s11554-012-0281-6>
52. Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. 1988. The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics. SIGGRAPH Comput. Graph. 22, 4 (June 1988), 21–30. https://doi.org/10. 1145/378456.378468
53. Matias Koskela, Atro Lotvonen, Markku Mäkitalo, Petrus Kivi, Timo Viitanen, and Pekka Jääskeläinen. 2019b. Foveated Real-Time Path Tracing in Visual-Polar Space. In Eurographics Symposium on Rendering - DL-only and Industry Track, Tamy Boubekeur and Pradeep Sen (Eds.). The Eurographics Association. https://doi.org/10.2312/sr. 20191219
54. Matias Koskela, Kalle Immonen, Markku Mäkitalo, Alessandro Foi, Timo Viitanen, Pekka Jääskeläinen, Heikki Kultala, and Jarmo Takala. 2019a. Blockwise Multi-Order Feature Regression for Real-Time Path Tracing Reconstruction. ACM Transactions on Graphics (TOG) 38, 5 (June 2019). https://doi.org/10.1145/3269978
55. Behnam Bastani, Eric Turner, Carlin Vieri, Haomiao Jiang, Brian Funt, and Nikhil Balram. 2017. Foveated Pipeline for AR/VR Head-Mounted Displays. Information Display 33 (11 2017), 14–19 and 35. <https://doi.org/10.1002/j.2637-496x.2017.tb01040.x>
56. Martin Reddy. 2001. Perceptually Optimized 3D Graphics. IEEE Comput. Graph. Appl. 21, 5 (Sept. 2001), 68–75. <https://doi.org/10.1109/38.946633>
57. Martin Reddy. 1998. Specification and evaluation of level of detail selection criteria. Virtual Reality 3, 2 (1998), 132–143.
58. Zipeng Zheng, Zhuo Yang, Yinwei Zhan, Yuqing Li, and Wenxin Yu. 2018. Perceptual Model Optimized Efficient Foveated Rendering. In Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology (Tokyo, Japan) (VRST ’18). ACM, New York, NY, USA, Article 109, 2 pages. https://doi.org/10.1145/3281505. 3281588
59. Sebastian Friston, Tobias Ritschel, and Anthony Steed. 2019. Perceptual Rasterization for Head-mounted Display Image Synthesis. ACM Trans. Graph. 38, 4, Article 97 (July 2019), 14 pages. <https://doi.org/10.1145/3306346.3323033>
60. David Li, Ruofei Du, Adharsh Babu, Camelia D. Brumar, and Amitabh Varshney. 2021. A Log-Rectilinear Transformation for Foveated 360-degree Video Streaming. IEEE Transactions on Visualization and Computer Graphics (2021).
61. Marc Levoy and Ross Whitaker. 1990. Gaze-directed Volume Rendering. SIGGRAPH Comput. Graph. 24, 2 (Feb. 1990), 217–223. <https://doi.org/10.1145/91394.91449>
62. Thomas A. Funkhouser and Carlo H. Séquin. 1993. Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (Anaheim, CA) (SIGGRAPH ’93). ACM, New York, NY, USA, 247–254. <https://doi.org/10.1145/166117.166149>
63. Eric Horvitz and Jed Lengyel. 1997. Perception, Attention, and Resources: A Decisiontheoretic Approach to Graphics Rendering. In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (Providence, Rhode Island) (UAI’97). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 238–249. http: //dl.acm.org/citation.cfm?id=2074226.2074255
64. Hector Yee, Sumanita Pattanaik, and Donald P. Greenberg. 2001. Spatiotemporal Sensitivity and Visual Attention for Efficient Rendering of Dynamic Environments. ACM Trans. Graph. 20, 1 (Jan. 2001), 39–65. https://doi.org/10.1145/383745.383748
65. Oculus. 2018. Oculus Go: Fixed Foveated Rendering. https://developer.oculus.com/ documentation/unreal/latest/concepts/unreal-ffr
66. L. Itti, C. Koch, and E. Niebur. 1998. A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 11 (1998), 1254–1259.
67. Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. 2005. Mesh Saliency. In ACM SIGGRAPH 2005 Papers (Los Angeles, California) (SIGGRAPH ’05). Association for Computing Machinery, New York, NY, USA, 659–666. https://doi.org/10.1145/ 1186822.1073244
68. Viviane Clay, Peter König, and Sabine Koenig. 2019. Eye Tracking in Virtual Reality. Journal of Eye Movement Research 12 (04 2019). <https://doi.org/10.16910/jemr.12.1.3>
69. Z. Hu, S. Li, C. Zhang, K. Yi, G. Wang, and D. Manocha. 2020. DGaze: CNN-Based Gaze Prediction in Dynamic Scenes. IEEE Transactions on Visualization and Computer Graphics 26, 5 (2020), 1902–1911.
70. Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. 2018. Gaze Prediction in Dynamic 360° Immersive Videos. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
71. David Hoffman, Zoe Meraz, and Eric Turner. 2018a. Limits of peripheral acuity and implications for VR system design. Journal of the Society for Information Display 26, 8 (2018), 483–495. https://doi.org/10.1002/jsid.730 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jsid.730
72. David M. Hoffman, Zoe Meraz, and Eric Turner. 2018b. 65-2: Sensitivity to Peripheral Artifacts in VR Display Systems. SID Symposium Digest of Technical Papers 49, 1 (2018), 858–861. https://doi.org/10.1002/sdtp.12261
73. Suzanne P. Mckee and Ken Nakayama. 1984. The detection of motion in the peripheral visual field. Vision Research 24, 1 (1984), 25 – 32. https://doi.org/10.1016/0042- 6989(84)90140-8
74. E. Turner, H. Jiang, D. Saint-Macary, and B. Bastani. 2018. Phase-Aligned Foveated Rendering for Virtual Reality Headsets. In 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). IEEE Computer Society, Los Alamitos, CA, USA, 1–2. <https://doi.org/10.1109/VR.2018.8446142>
75. Linus Franke, Laura Fink, Jana Martschinke, Kai Selgrad, and Marc Stamminger. 2021. Time-Warped Foveated Rendering for Virtual Reality Headsets. Computer Graphics Forum 40, 1 (2021), 110–123. https://doi.org/10.1111/cgf.14176 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14176
76. Joerg H. Mueller, Thomas Neff, Philip Voglreiter, Markus Steinberger, and Dieter Schmalstieg. 2021. Temporally Adaptive Shading Reuse for Real-Time Rendering and Virtual Reality. ACM Trans. Graph. 40, 2, Article 11 (April 2021), 14 pages. <https://doi.org/10.1145/3446790>
77. Brian Karis. 2014. High-quality temporal supersampling. Advances in Real-Time Rendering in Games, SIGGRAPH Courses 1 (2014), 1–55.

# 附录

# 致谢

衷心感谢我的导师黄石老师在我撰写本篇硕士论文过程中所给予的悉心指导和宝贵意见。黄老师的严谨学风和深厚学识不仅在学术上给予了我极大的帮助，也在人生观和价值观的形成上产生了深远的影响。

同时，我也要特别感谢费广正老师和韩红雷老师。通过他们的教诲，我学习到了许多专业技能，也领悟了学术探究的真谛。两位老师认真负责的工作态度和对待学生的关爱，将永远激励我在今后的道路上不断前进。

我还要感谢我的师兄周威，他在我遇到困难和挑战时总是不吝啬给予我指导和建议，帮助我渡过难关。他的经验和智慧对我的研究进程有着不可忽视的促进作用。

对我的同学高越、徐啸迪和李斯濠，我同样充满感激之情。在这段求学之旅中，与他们的互帮互助和共同讨论不仅丰富了我的学术视角，也让我在精神上得到了极大的支持和鼓舞。我们彼此分享的知识和快乐时光，将成为我珍贵的记忆。

我还要感谢动画学院提供的学习平台和资源，在这样的环境下，我有机会接触到最前沿的知识和技术，满足我对知识的求索和对艺术的追求。

最后，我要向在我学业和生活各个阶段始终支持和鼓励我的父母表示最诚挚的感谢。没有他们无私的奉献和耐心的支持，就没有我今天的成就。他们不仅是我的坚强后盾，更是我前进道路上的指南灯。

再次感谢所有帮助、关心和支持我的人。是你们的帮助和鼓励，让我在学术海洋中乘风破浪，勇往直前。