ICS 35.240.99
CCSA L60/69

团　　体　　标　　准

T/INFOCA 6—2024

基于端云协同的全景视频实时渲染系统设计指南

Design guidelines for real-time rendering system of panoramic video based end-cloud collaboration

（征求意见稿）

2024-xx-xx 发布　　xxxx-xx-xx 实施

中关村现代信息消费应用产业技术联盟 发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由xxx提出并归口。

本文件起草单位：中国传媒大学、河南广播电视台、哈尔滨工业大学、中国科学院大学、中兴通讯股份有限公司、广东南方新媒体股份有限公司

本文件主要起草人：裘初、高鹏东、齐全、王博、卢慎勇、陈纳新、李昕、范晓鹏、王兴涛、张新峰、安泓宇、李欢洋、史美康、刘群、周宇阳、王兵、罗泽文、丁凤

引　言

全景视频作为一项新兴技术，已经成为许多国家在科技创新领域争相布局的战略方向。全景视频主要涉及360度视频拍摄、合成、播放、传输等技术，为用户带来沉浸式视觉体验，被广泛地应用于旅游、教育、房产、新闻报道、体育赛事、虚拟现实（VR）和增强现实（AR）等多个领域。实时渲染系统将计算机图形（CG）软件制作的虚拟场景转换成逼真画面，是全景视频制作中不可缺少的重要组成部分。

VR/AR等移动设备的普及应用，对全景音视频的生成、分发和调度机制提出了交互性和实时性的新要求。伴随着终端设备计算能力的增强以及5G等新一代通讯技术的发展，全景音视频传统独立统一处理的模式，具备了向端云协同生产转变的条件。为指导实时渲染系统向端云协同架构的转变，特编制本文件。

基于端云协同的实时渲染系统设计，充分考虑了终端计算和云计算的特点，减轻了终端的计算压力，提高了整个系统部署的灵活性。一方面可以使用云端的强大计算能力，确保背景图像生成的质量和效率；另一方面，确保终端有更多的计算能力用于处理和提高全景视频中交互相关内容的渲染质量和效率，保障交互的流畅性，为用户提供更好的使用体验。

基于端云协同的全景视频实时渲染系统设计指南

1 范围

本文件给出了基于端云协同的全景视频实时渲染系统（以下简称“端云系统”）设计的设计原则和系统架构的设计指导性建议。

本文适用于指导基于端云协同的全景视频实时渲染系统的设计和开发。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 5271.13-2008 信息技术词汇 第13部分：计算机图形

3 术语和定义

下列术语和定义适用本文件。

3.1

云边协同 cloud-edge collaboration

将边缘计算和云计算相结合相互配合完成某项计算任务。

3.2

渲染 rendering

将景物的几何形状、着色、纹理构成、亮度和其他的特性转换成显示图像的过程。

[来源：GB/T 5271.13-2008，13.02.21]

3.3

实时渲染 real-time rendering

以每秒30帧以上的速度完成渲染的过程，需要在GPU上执行。

3.4

渲染管线 rendering pipeline

一种实现渲染计算的方法，它将整个渲染计算从模型顶点到像素分为若干个按照流水线方式执行的阶段，每个阶段需要完成特定的计算任务。

3.5

向前渲染 forward rendering

一种基本的实时渲染流程，为待渲染几何体→顶点着色→片元着色→渲染目标。

3.6

延迟渲染 deferred rendering

一种基于屏幕空间着色的实时渲染流程，为待渲染几何体→顶点着色→二维缓存→片元着色→渲染目标。

4 缩略语

下列缩略语适用于本文件。按照英文顺序调整

API：应用编程接口（Application Programming Interface）

GPU：图形处理器（Graphics Processing Unit）

UDP：用户数据报协议（User Datagram Protocol）

5 设计原则

5.1 宜采用模块化的思路，合理进行系统架构分层。

5.2 宜充分考虑时延是决定端云系统性能的核心要素，在各个环节都要考虑如何降低时延。

5.3 宜充分考虑如何在云边之间划分待渲染内容。

5.4 宜充分考虑全景视频图像的特点，根据内容与用户视点关系对待渲染场景采取不同的处理方式。

5.5 宜充分考虑在模块之间实现功能的并行处理。

6 系统设计建议

6.1 概述

系统设计宜考虑系统架构以及构成端云系统的各个子系统设计两部分。

6.2 系统架构

6.2.1 系统架构图

图1给出了基于端云协同的全景视频实时渲染系统的系统架构。从整体上而言，端云系统宜由终端子系统、传输子系统、云端子系统三部分组成，其中终端子系统和云端子系统负责协同地完成场景渲染工作并最终呈现给用户，传输子系统负责在两个子系统之间传输场景、控制参数、渲染结果等数据。

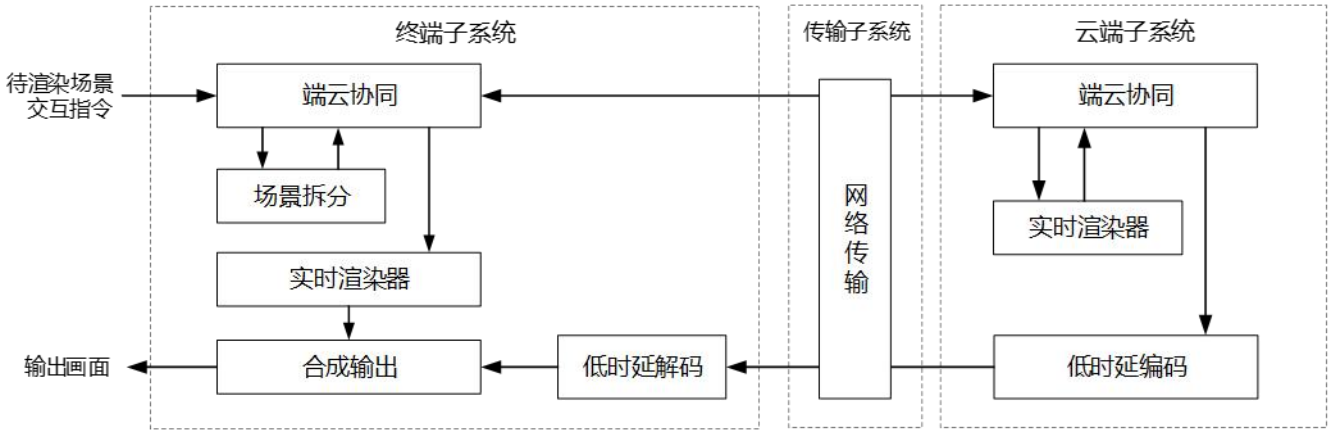

图1 端云系统的系统架构图

6.2.2 系统架构描述

6.2.2.1 终端子系统

终端子系统负责接收待渲染场景并采集用户交互指令，将场景拆分成本地部分和云端部分，其中云端部分传输到云端子系统进行计算,本地部分调用终端计算资源直接完成计算；最后接收来自云端子系统的渲染结果，进行解码后与本地部分进行合成，呈现给用户。

终端子系统宜由端云协同、场景拆分、实时渲染器、合成输出和低时延解码五部分组成：

1) 端云协同模块的核心功能是与云端子系统的端云协同模块一起，完成端云协同渲染的任务，具体功能包括以下几个方面：
   a) 接收待渲染场景和交互指令，调用场景拆分模块将场景拆分成本地和云端部分，将本地部分分配到本地实时渲染器进行渲染，将远端部分发送到云端子系统进行计算。
   b) 对场景、视点等的变化进行预测，提前向云端子系统发送渲染指令进行预渲染及预提取。
   c) 与云端子系统的端云协同模块进行通信，发送前述功能涉及到的数据及指令，并接收反馈消息。
2) 场景拆分模块用于将待渲染场景拆分成本地和云端部分，并将结果返回给端云协同模块。
3) 实时渲染器用于从端云协同模块接收并执行本地部分场景的渲染计算任务。
4) 合成输出模块用于融合本地和云端部分的渲染结果，向用户输出最终画面。

5) 低延时解码模块负责接收来自云端子系统的经过编码压缩的渲染结果，进行解码后发送给合成输出模块进行最终输出。

6.2.2.2 传输子系统

传输子系统是终端子系统与云端子系统之间的通信单元，主要承担终端子系统和云端子系统之间的数据传输功能，数据包括场景数据、渲染结果、交互指令、控制指令等等。

6.2.2.3 云端子系统

云端子系统负责接收来自终端子系统的场景数据和交互指令，完成渲染计算，并将渲染结果进行编码压缩后返回给终端子系统进行合成。

云端子系统宜由端云协同、实时渲染器和低延时编码三部分组成。

1) 端云协同模块的核心功能是终端子系统的端云协同模块一起，完成端云协同渲染的任务，具体功能包括以下几个方面：
   a) 与终端子系统的对应模块进行通信，接收相关数据和指令，发送反馈信息。
   b) 对场景进行分块，按照一定优先级依次调用相应模块进行渲染、编码、传输工作。
2) 实时渲染器用于从端云协同模块接收并执行云端部分场景的渲染计算任务，并返回结果。
3) 低延时编码采用简化的压缩算法对渲染结果进行编码压缩，节省编码时间，并将结果发送到终端子系统。

6.2.3 各个子系统设计建议

6.2.4 终端子系统设计建议

6.2.4.1 端云协同

6.2.4.1.1 宜充分考虑本地和云端之间的性能差异与全景图像的特点，通过预加载和视点变化预测等手段，在三个子系统之间实现本地渲染、云端渲染、传输、呈现的并行处理，从而降低时延，原理如下图所示：

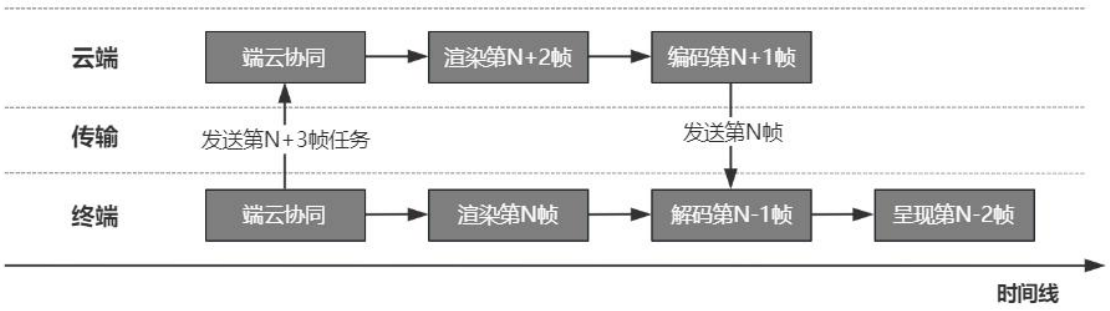

图 2 三个子系统之间并行处理原理图

6.2.4.1.2 宜考虑对视点变化进行预测，提前发送变化信息到云端子系统进行预渲染并预提取回终端子系统。

6.2.4.1.3 宜考虑对场景变化进行预测，将未来一段时间内待渲染内容提前传输到云端子系统。

6.2.4.1.4 宜考虑对于分配到云端子系统的待渲染内容，只传输相对于已分配内容变化的部分。

6.2.4.2 场景拆分

6.2.4.2.1 合理的场景拆分对确保端云系统的性能至关重要，场景拆分宜考虑以下原则：

a) 根据场景中内容变化频率进行拆分：将变化频率高的内容分配到终端子系统进行渲染，变化频率低的分配到云端子系统进行渲染；

b) 根据场景中内容与观察相机之间的距离进行拆分：将离相机近的内容分配到终端子系统进行渲染，离相机远的物体分配到云端子系统进行渲染。

c) 在制作待渲染场景时，就标记好哪些内容在终端子系统渲染，哪些内容在云端子系统渲染。

6.2.4.2.2 宜支持根据网络状况动态调整分配到云端子系统的待渲染内容量。

6.2.4.3 实时渲染器

6.2.4.3.1 宜支持 Windows 系统和移动端。

6.2.4.3.2 宜支持 OpenGL ES 3.0+、Vulkan 1.*、DirectX 12、Metal 等图形 API。

6.2.4.3.3 宜支持可扩展渲染管线，以向前渲染为主要渲染管线。

6.2.4.3.4 宜支持分块渲染和多分辨率渲染。

6.2.4.4 合成输出

6.2.4.4.1 合成之前，宜考虑采用合适的策略解决两个子系统渲染结果帧编号之间的不一致，确保结

果的同步。

6.2.4.4.2　宜考虑缓存云端子系统传回的渲染结果，在网络通信不佳的情况下，宜考虑使用旧的结果进行合成。

6.2.4.4.3　宜支持渲染结果的畸变校正。

6.2.4.5　低延时解码

6.2.4.5.1　宜采用与云端子系统的编码器对应的解码算法。

6.2.5　传输子系统设计建议

6.2.5.1　宜考虑在端云系统运行期间，在边缘子系统与云端子系统之间保持长链接，避免频繁重连对端云系统性能造成影响。

6.2.5.2　宜考虑采用基于 UDP 的传输协议，最大限度地利用带宽。

6.2.5.3　宜考虑根据传输内容采用不同的丢包重传机制，对不重要的数据可以容忍更高的丢包率。

6.2.5.4　宜考虑支持毫秒级别的网络状态上报，例如网络吞吐、丢包率等等，帮助终端子系统和云端子系统做到编解码率/策略的调整，渲染分配的调整。

6.2.6　云端子系统设计建议

6.2.6.1　端云协同

6.2.6.1.1　宜将待渲染场景划分成多个区域，分区域完成渲染、编码和传输，实现并行处理，从而降低时延，如下图所示。

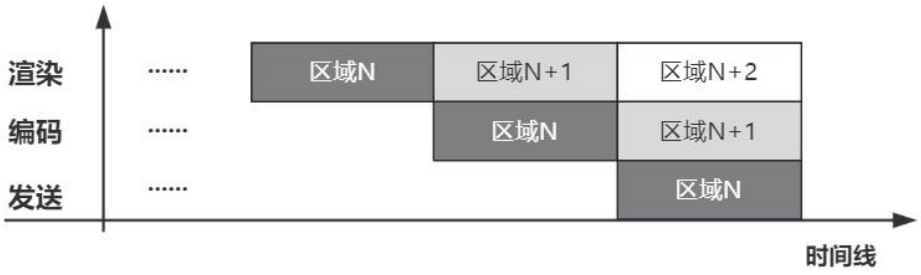

图 3 端云系统实现分块作业的并行原理图

6.2.6.1.2　宜考虑全景图像的特点，按照内容与视点远近关系来设置分区的先后计算次序，甚至在不同分区之间采用不同的分辨率来处理。

6.2.6.1.3　在带宽不足或是网络状况不佳时，宜考虑支持先完成低质量的全场景渲染，发送给终端子

系统来完成最基本的画面呈现后，再进行高质量的增量更新。

6.2.6.1.4 宜考虑定时向终端子系统发送状态信息，包括各模块正在处理的帧编号、负载等等。

6.2.6.2 实时渲染器

6.2.6.2.1 宜支持 Vulkan 1.*、DirectX 12 等图形 API。

6.2.6.2.2 宜支持可扩展渲染管线，支持延迟渲染。

6.2.6.2.3 宜支持分块渲染和多分辨率渲染。

6.2.6.2.4 宜支持通过深度学习渲染来减少计算量。

6.2.6.2.5 宜考虑在集成 Nvidia RTX 系列、AMD Radeon 系列等高性能 GPU 的硬件系统上运行。

6.2.6.3 低延时编码

6.2.6.3.1 宜考虑简化编码算法来换取编码时间，实现快速编码，可以考虑采取以下简化手段:

a) 简化运动搜索算法；

b) 减少帧内预测；

c) 降低帧间预测算法的复杂度；

d) 减少编码单元划分层次。

6.2.6.3.2 宜根据网络情况调整编码码率。

6.2.6.3.3 宜考虑采用 GPU 对算法进行加速。