|  |  |
| --- | --- |
| 鸿蒙OS的跨设备空间计算软件Kit任务技术文件 | |
| [课题编号: ]  [文档标识号: TN/x-DO-DS-V{N.xx}]  [修订日期: 2010-XX-XX] | |
| 空间计算  软件总体技术方案 |
| 2024年12月31日 |

[1. 简介 3](#_Toc186417469)

[1.1 项目背景 3](#_Toc186417470)

[1.2 竞品分析 4](#_Toc186417471)

[1.2.1 ARKit 4](#_Toc186417472)

[1.2.2 ARCore 7](#_Toc186417473)

[1.2.3 总结 7](#_Toc186417474)

[1.3 文档结构说明 8](#_Toc186417475)

[2. HarmonyOS操作系统介绍 8](#_Toc186417476)

[2.1 HarmonyOS技术特点 8](#_Toc186417477)

[2.1.1 分布式架构的设计优势 8](#_Toc186417478)

[2.1.2 生态统一性 9](#_Toc186417479)

[2.2 HarmonyOS下AR系统的独特优势 10](#_Toc186417480)

[3. 系统框架设计 12](#_Toc186417481)

[3.1 系统总体架构图 12](#_Toc186417482)

[3.2 模块划分与职责说明 15](#_Toc186417483)

[3.2.1模块定位 15](#_Toc186417484)

[3.2.2模块之间的主要数据流向 17](#_Toc186417485)

[3.3 目录结构说明 19](#_Toc186417486)

[3.4 模块间的接口定义 22](#_Toc186417487)

[3.4.1 ARSession管理 22](#_Toc186417488)

[3.4.2 数据处理及分布式 25](#_Toc186417489)

[3.4.3 接口之间的协作方式（数据流） 33](#_Toc186417490)

# 简介

## 项目背景

近年来，随着计算机视觉（CV）、人工智能（AI）以及光学芯片技术的进步，增强现实（AR）和虚拟现实（VR）技术（以下统称为XR技术）和应用迅速发展。由于开发XR应用存在对空间定位、环境理解、空间交互等空间计算能力的需求，业界相继推出了提供XR能力的SDK。例如：苹果推出的支持iOS和visionOS的ARKit；谷歌推出的支持多种开发环境（Android、iOS、Unity、WebXR、OpenXR等）的手机AR应用SDK。

目前，XR SDK，以提供3D渲染和空间计算等能力为核心，正处于蓬勃发展的阶段，在未来，XR有潜力成为操作系统（OS）的一部分原生能力，帮助设备实现对周围环境的精确感知和自然交互，提高用户人机交互体验。苹果、谷歌等企业已经在探索XR设备的操作系统以及XR SDK在专用设备上的扩展。

在WWDC 2023上，苹果推出了Apple Vision Pro和visionOS，ARKit in visionOS针对用户需求和硬件差异，对iOS原有能力进行大幅度的更改，为系统提供核心能力，是visionOS实现沉浸式场景的重要支撑。visionOS是第一个为VR眼镜设计的操作系统，为它提供基础能力的ARKit，对于开发跨平台提供XR能力的SDK具有较大的参考价值。

近期，谷歌发布了Android XR OS作为Android平台和生态系统在XR领域的扩展。Android XR OS提供的Android XR SDK提供了多种将安卓APP移植到Android XR的方法，如使用与Android Jetpack类似的Jetpack XR SDK开发XR应用等。同时，ARCore也增加了为Jetpack XR SDK设计的扩展，用于在Android XR设备上提供空间计算能力。

随着HarmonyOS的发展，华为公司也推出了AREngine等在HarmonyOS上完成空间计算和3D渲染等XR任务。为了解决未来的跨设备和多设备问题，有必要将空间计算的部分能力嵌入系统服务，实现在不同形态设备上的流畅运行、多设备协作时数据的稳定传输，为原生应用和第三方应用提供更强的空间计算能力，为开发者提供更丰富和稳定的XR服务接口。

本项目旨在开发适用于HarmonyOS的支持跨设备的AREngineKit，具体开发内容包括以下几个方面：

#### Kit开发

实现系统AR功能服务化，完善进程管理、资源管理和Session生命周期管理、算法特性调度等功能。实现Kit内的数据传输通路，以确保传感器与Kit间的连接、传感器数据低时延下发给算法。

#### API开发

提供API接口，以便上层XR应用集成使用。

#### Sample开发

基于Kit API构建典型场景XR Sample，展示空间计算能力。

## 竞品分析

### ARKit

ARKit最初设计为iOS系统提供AR能力，又为了提供能力给visionOS，将ARKit分为了两部分：ARKit in iOS和ARKit in visionOS。

#### ARKit in iOS

在2017年WWDC会议上，苹果公司发布了ARKit in iOS，经过多次改进扩展，为其增加了3D模型加载、多设备协同、QuickLook等能力。ARKit in iOS是业界较成熟的AR SDK，附表的api in ios部分中总结了ARKit in iOS主要能力和API，对API进行整理，可以看到其API主要分为四类：Session、View、Anchor、ARConfiguration。根据其API和文档，可以总结出如下图所示的系统架构。

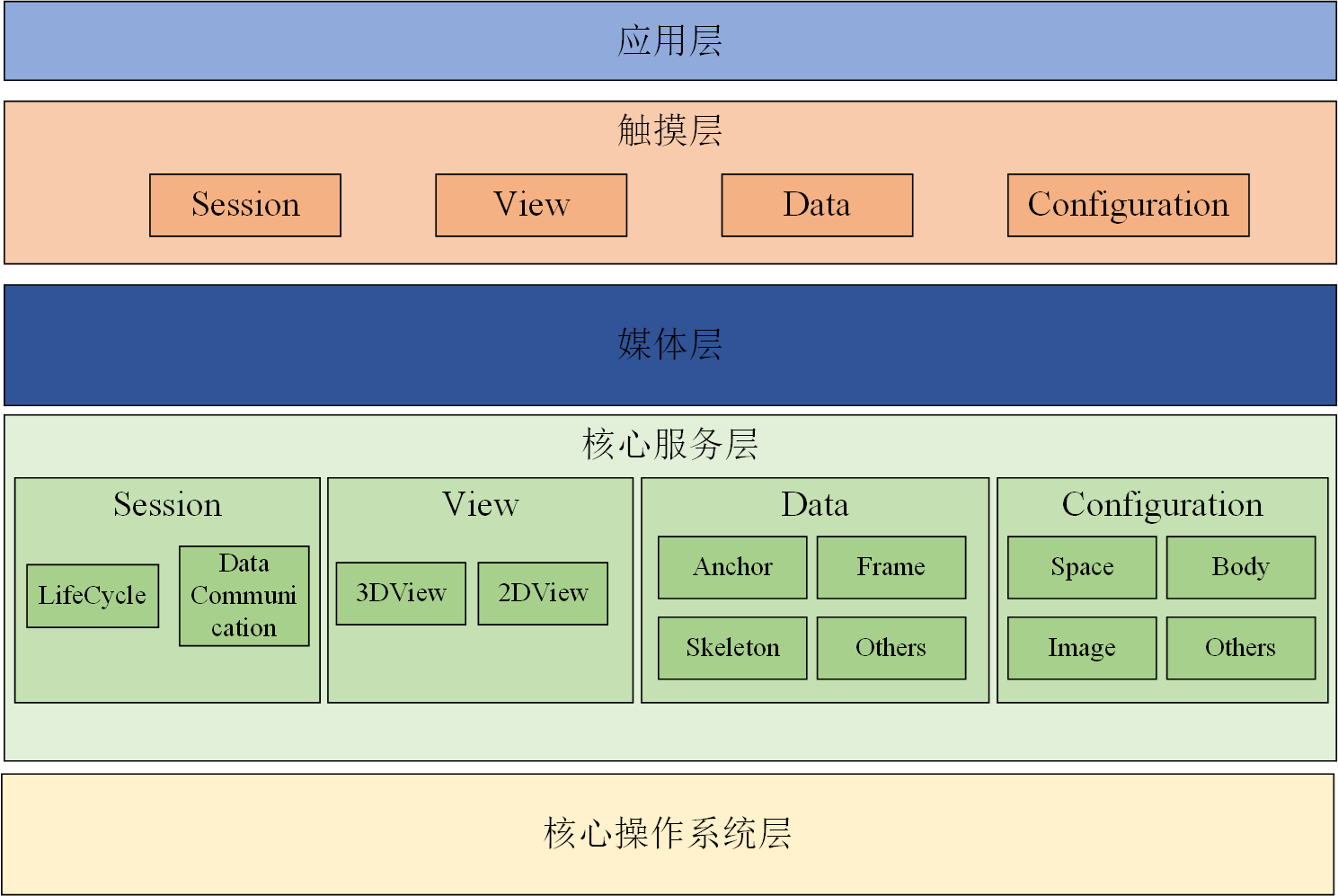


图 1.1 ARKit in iOS系统架构图

其中，Session用于管理生命周期，协调View和Configuration的执行, 控制View和Configuration之间的数据传输；View负责2D、3D视图界面的显示； Configuration包含世界追踪、肢体检测等能力的配置；Data是Configuration对环境进行分析得到的数据。

#### ARKit in visionOS

ARKit in iOS虽然功能较强，但是由于它最初是为iOS专门设计的，并没有考虑在其它设备上的使用；为了更符合Apple Vision Pro的开发需求，苹果对ARKit in visionOS提供的能力种类进行了较大幅度的调整。而且在visionOS中，ARKit为核心操作系统提供能力中，和SwiftUI、RealityKit一起支撑起visionOS的运行，推断依据将在后面给出[1]。

附表的api in visionos中总结了ARKit in visionOS主要能力和API，对API进行整理，可以看到其API主要分为三类：Session、Anchor、Dataprovider。根据其API和文档，可以总结出如下图所示的系统架构。

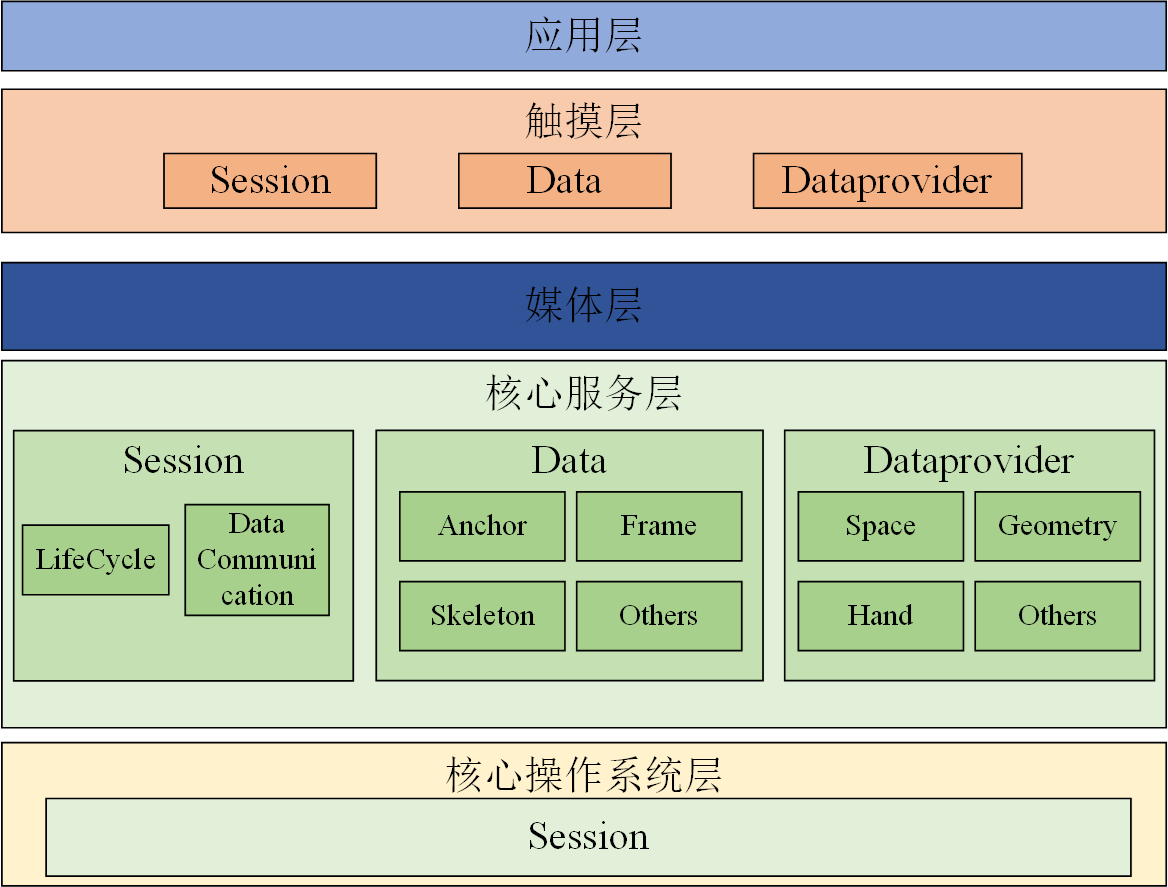


图 1.2 ARKit in visionOS系统架构图

其中，Session用于管理生命周期，控制Dataprovider的运行，将Dataprovider自动的传递到视图进行显示； Data是ARKit in visionOS对环境进行分析得到的数据，包括：锚点、视频帧等；Dataprovider用于提供世界追踪、平面检测等能力。

苹果官方文档提到大多数软件不需要直接调用ARKit即可实现，我们对比了ARKit in iOS和ARKit in visionOS二者，ARKit in visionOS在前者基础上经过很大的调整，在架构上，移除了View部分，不再使用ARKit内部的ARSCNView等显示界面，更换为SwiftUI和RealityKit协同的方式，即使使用了ARKit，结果也只以自动的方式显示出来，大多数情况下为通过SwiftUI和RealityKit隐式的使用ARKit[2]。

另外，还有一些差异体现在二者的API上，下表为Session的对比，ARKit in visionOS中对外开放API明显少于iOS，只保留了核心的run、stop、Event等。同样的ARConfiguration、Anchor等都被简化，保留了核心的接口，如Dataprovider的isSupported、State等。这一方面是简化了接口的使用，另一方面减少了ARKit使用者对系统层的接触。

表1.1 ARKit Session对比

|  |  |
| --- | --- |
| ARKit in iOS | ARKit in visionOS |
| run | init() |
| identifier:UUID | run |
| ARSession.RunOptions | stop |
| configuration | ARKitSession.Error |
| pause | requestAuthorization |
| delegate | ARKitSession.AuthorizationType |
| delegateQueue | queryAuthorization |
| ARSessionDelegate | ARKitSession.AuthorizationStatus |
| ARSessionObserver | events:ARKitSession.Events |
| add | ARKitSession.Events |
| remove | ARKitSession.Event |
| getCurrentWorldMap | description:String |
| createReferenceObject |  |
| setWorldOrigin |  |
| raycas |  |
| trackedRaycas |  |
| getGeoLocation |  |
| currentFrame |  |
| ARFrame |  |
| captureHighResolutionFrame |  |
| update |  |
| ARSession.CollaborationData |  |
| ARSessionProviding |  |

除了作为企业API的BarcodeDetectionProvider[3]，Dataprovider相比于ARConfiguration，看似能力更顶层化了，如RoomTrackingProvider、SceneReconstructionProvider等，但这些大致都是构成visionOS共享空间（共享空间相关概念见[1]中A spectrum of immersion部分）所需的能力。

表1.2 ARKit AR能力对比

|  |  |
| --- | --- |
| ARKit in iOS | ARKit in visionOS |
| ARWorldTrackingConfiguration | CameraFrameProvider |
| ARGeoTrackingConfiguration | PlaneDetectionProvider |
| AROrientationTrackingConfiguration | WorldTrackingProvider |
| ARPositionalTrackingConfiguration | HandTrackingProvider |
| ARBodyTrackingConfiguration | SceneReconstructionProvider |
| ARFaceTrackingConfiguration | ImageTrackingProvider |
| ARImageTrackingConfiguration | EnvironmentLightEstimationProvider |
| ARObjectScanningConfiguration | ObjectTrackingProvider |
|  | RoomTrackingProvider |

另外， RealityKit以ARKit为基础，能够提供图像渲染、相机特效、动画、物理特效等能力，使得ARKit in visionOS中一些非核心能力可以剪除，如Quick Look、HitTest、ObjectScanning等。

### 1.2.2 ARCore

#### 1) ARCore for Android

ARCore是Google推出的用于设计AR应用、进行空间计算的SDK，主要在 Android和iOS上提供服务。

附表api in android中总结了ARCore for Android主要能力和API，对API进行整理后，可以看到其API主要分为：Session、Data、EnvironmentEstimation三种。根据其API和文档，可以总结出如下图所示的系统架构。

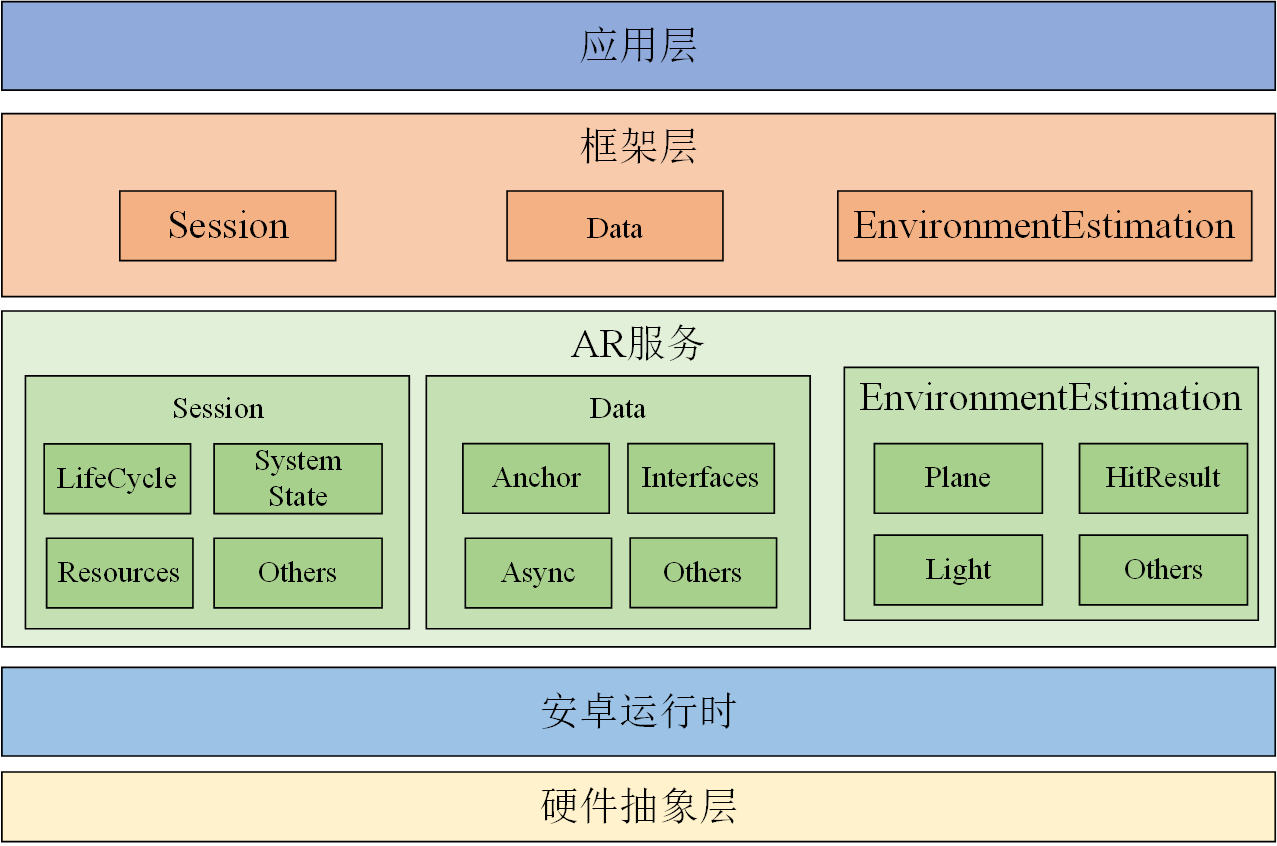


图 1.3 ARCore for Android系统架构图

其中，Session用于管理生命周期，控制EnvironmentEstimation中API的运行； Data是ARCore for Android对环境进行分析得到的数据包括Anchor、Future、Geometry等；EnvironmentEstimation中包含平面检测、光线估计等能力的配置。

#### 2) ARCore for Jetpack XR

近期，谷歌发布了Android XR OS作为Android平台和生态系统在XR领域的扩展。Android XR OS提供的Android XR SDK提供了多种将安卓APP移植到Android XR的方法，如使用与Android Jetpack类似的Jetpack XR SDK开发XR应用等。同时，ARCore也增加了为Jetpack XR SDK设计的扩展，用于在Android XR设备上提供空间计算能力，其提供的空间计算能力较少：包括平面检测和HitTest两种。

附表api in AndroidXR中总结了ARCore for Jetpack XR主要能力和API，对API进行整理后，可以看到其API主要分为三类：Session、Data、Perception、EnvironmentEstimation。根据其API和文档，可以总结出如下图所示的系统架构。

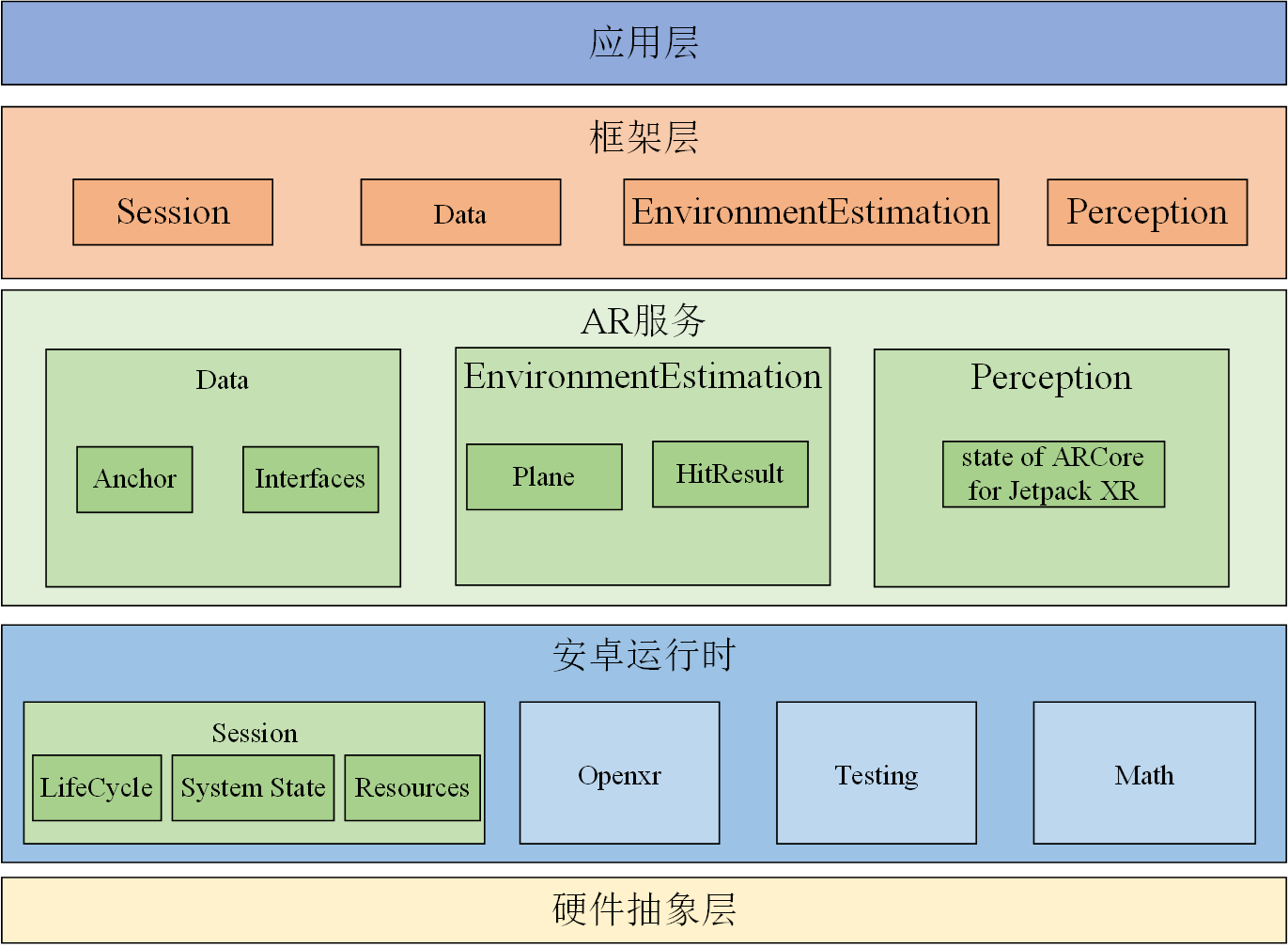


图 1.4 ARCore for Jetpack XR系统架构图

其中，Session用于管理生命周期，控制EnvironmentEstimation中API的运行； Data是ARCore for Jetpack XR对环境进行分析得到的数据包括Anchor、Future、Geometry等；EnvironmentEstimation中包含平面检测、命中测试等能力的配置、Perception代表 ARCore for Jetpack XR 在特定时间点的状态。

与ARCore for Android的api相比，ARCore for Jetpack XR，大幅度减少了提供的API接口和能力数量，详见附表api in AndroidXR和api in android，目前仅保留了Plane和HitTest两种空间计算能力，并且将Session转移到AndroidXR的runtime中运行[4]；另外由SceneCore针对空间计算能力的API进行了扩展，并且ARCore和SceneCore共用了runtime中的Session[5]；而Jetpack XR的其他部分Compose、View等承担显示界面视图的任务，其中Compose是基于SceneCore构建而成的[6]。

### 1.2.3总结

ARKit和ARCore是业界内较为完善的提供空间计算能力的SDK。随着XR技术和XR设备的发展，都推出自己的XR操作系统、设备和SDK扩展，二者都对原有SDK进行了修改，裁剪封装了原来用于手机应用开发的SDK，并且将能力加入到系统服务，以提供适配眼镜设备的SDK。根据1.2.1和1.2.2两节可以总结二者以下异同点：

表1.3 ARKit ARCore对比

|  |  |  |
| --- | --- | --- |
| 差异 | ARKit in visionOS | ARCore for Jetpack XR |
| 删改较轻，保留了较多XR能力 | 删改较重，只保留了Plane、HitTest |
| Session仅ARKit使用 | Session放入runtime与SceneCore共用 |
| 独占式，共享空间下，App不能使用ARKit | 可共享的 |
| 相同 | 基于ARKit/ARCore使用其它SDK提供额外的空间计算能力 | |
| 基于ARKit/ARCore为UI相关SDK扩展XR系统显示能力 | |
| 手机和眼镜上两套SDK相差较大 | |

根据二者的异同点，我们想要设计一套不同设备共用的AREngineKit，其主要特点为：不同设备通用，AR能力可共享的。为满足以上特点，我们预期对已有的AREngine进行改进，鉴于与ARKit in visionOS的Session API有较高的相似性，可以Session API不变，保留已有的AR能力，并下沉到系统服务层，提供可共享使用的Session及AR能力，同时保证了AR能力API以及UI相关SDK的可拓展性。在改进过程中，针对XR场景下需要，为应对启动速度和资源占用问题，可能需要对Session和AR能力进行优化。

## 文档结构说明

本文档共分为3节，第1节为项目简介，主要介绍了项目的背景，对XR SDK的主要竞品进行了分析，提出我们要设计支持跨设备开发、将空间计算能力嵌入到系统能力的ARKit; 第2节为HarmonyOS操作系统的介绍，主要介绍了HarmonyOS分布式架构和生态统一性的技术特点，还介绍了HarmonyOS下AR系统的独特优势以及HarmonyOS SA开发介绍；第3节为系统框架设计，主要介绍了本项目ARKit预期中的系统总体架构、模块划分与职责、AREngine执行流程、代码目录结构以及模块间接口定义。

**附录**

1. <https://developer.apple.com/visionos/> Apple frameworks — extended for spatial computing部分

1. <https://developer.apple.com/documentation/visionos/bringing-your-arkit-app-to-visionos> Replace your ARKit code
2. <https://developer.apple.com/documentation/visionos/building-spatial-experiences-for-business-apps-with-enterprise-apis>
3. <https://developer.android.com/reference/androidx/xr/runtime/package-summary>
4. <https://developer.android.com/develop/xr/jetpack-xr-sdk/check-spatial-capabilities?hl=zh-cn>
5. <https://developer.android.com/develop/xr/jetpack-xr-sdk/develop-ui-views?hl=zh-cn>