*基于视线预测资源预加载*

基于视线预测资源预加载

姚以真, 符殊源

摘 要: 用户在观看VR应用中的全景场景时，往往只会注意到处于当前视角之内的部分，因此可以采用对用户未来视角进行预测的方式，对资源进行动态加载，以节约带宽和能耗。本文实现了一个基于视线预测资源预加载系统，系统包含一个模拟VR应用的3D全景场景，以及一个基于LSTM神经网络的视线预测模型，可以通过对用户下一关键帧的视角进行预测，来对资源进行预加载。系统为客户端和服务端分离架构，客户端与服务端之间通过Socket通信。本文提出的系统不仅能节约能耗，也能给用户带来良好的观看体验，具有良好的应用性和可移植性。

**关键词：视线预测，神经网络，预加载**

Title: Pre-loading of Resources Based on Gaze Prediction

Yizhen Yao, Shuyuan Fu

**Abstract**: When watching panoramic scenes in VR applications, users often only pay attention to the parts that are within the current viewport. Therefore, the future viewport of the user can be predicted and then be used to dynamically load resources to save bandwidth and power. This report implements a resources pre-loading system based on gaze prediction. The system includes a 3D panoramic scene like VR applications and a gaze prediction model based on LSTM neural network. It can pre-load resources by predicting the user’s viewport of the next key frame. The system is a separated architecture of client and server, which communicate through Socket. The system proposed in this report can not only save bandwidth and power, but also bring a good viewing experience to users, with good applicability and portability.

**Key word：Gaze Prediction, Neural Networks, Pre-loading**

# **简介与意义**/Introduction

## 项目意义和依据/Significance

虚拟现实 (Virtual Reality, VR) 系统为用户提供了在沉浸式环境中探索新世界的机会。随着VR技术的发展，近年来市场上诞生了许多VR应用，其中，大多数应用都包含360度场景，比如360度全景照片、360度全景视频等，用户可通过佩戴的头戴式设备自由地调整观看角度，产生不同的视觉体验。但是，为了提供良好的用户体验，360度场景往往会消耗大量的带宽。然而，由于用户可以灵活地选择观看方向，只有处于用户视角部分的场景（Field of View, FOV）会被用户观看到。研究表明，由于用户只会注意到视角之内的场景，大约有80%的带宽在360度视频传输的过程中被浪费[1]。

为了减少带宽浪费，提升360度场景传输的质量，可以仅将预测的视角内的帧以较高质量进行传输，其余部分以较低质量进行传输，以此节省带宽。这要求建立模型对用户的视线进行预测。现有视线预测主要对用户的眼睛运动进行追踪。常见的眼动追踪手段依赖于眼动仪来完成，眼动仪可以装配在头戴设备上，然而，眼动仪往往造价高昂，且不易于使用。另一种想法是对用户的头部运动进行预测。通过用户佩戴的头戴式设备，可以采集到用户的头部运动数据，基于这些数据可以对用户的头部运动位置进行预测，再利用数学公式建立用户头部位置与眼睛位置之间的联系，将用户的头部位置和眼睛位置关联起来，从而得到预测的视线。无论是对哪部分运动进行预测，都需要建立一个预测模型。现有的预测模型多采用线性回归或机器学习方法，

基于运动轨迹和画面内容进行预测。尽管当前已有很多视角预测方面的研究，但缺少对预测结果的进一步应用。如何将预测结果结合到VR应用中，更好地发挥视线预测的价值，是该领域内值得关注的一个问题。

基于以上思想，本小组实现了一个基于视线预测资源预加载的系统。系统中包含一个模拟VR应用的3D全景场景，以及一个基于LSTM神经网络的视线预测模型，可以对用户的视线进行预测，并利用预测结果对场景的资源视线预加载，从而达到节约能耗的目的。该系统可以得到比较准确的视线预测结果，且能将预测结果进行很好的应用，给用户带来流畅的观看体验，具备很强的应用性和可移植性。

本文将分为以下五部分内容进行阐述。第1部分介绍项目的意义与系统框架，第2部分介绍相关领域内的研究工作，第3部分阐述本小组的研究内容与方法，第4部分给出实验结果并分析，最后总结本研究的特色与创新。

## 本系统框架/Article Structure

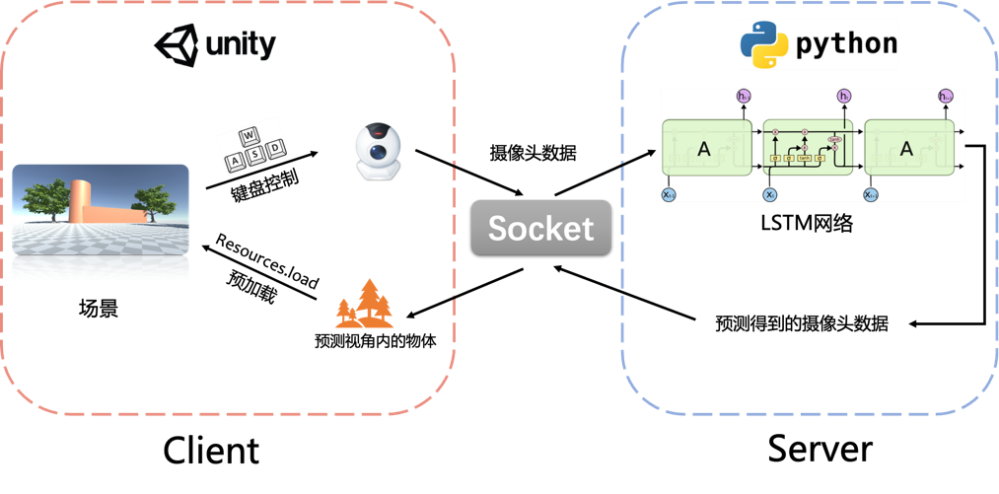


图1 系统框图

本文实现了一个基于视线预测进行资源预加载的系统，系统框架如图1所示，其中，Unity项目作为客户端，Python项目作为服务端，客户端与服务端之间通过Socket进行通信。在客户端，本文建立了一个包含建筑物、树木、马路等物体的学校图书馆主馆场景，用户可以通过键盘移动摄像头，系统记录摄像头的数据，发送给服务端。在服务端，将当前时刻之前的摄像头历史移动记录输入到LSTM神经网络中，输出预测的用户视角，并发送给客户端。客户端接收到服务端传来的预测视角后，对在预测视角内的物体进行资源预加载。该系统部署在笔记本电脑上，不需要额外的硬件设备。

# **相关工作**/Related Works

## 序列预测

视角预测以及相关的移动性预测问题都属于序列预测问题，即，从给出的序列预测下一个值。可以将序列预测问题的研究方法分为传统的机器学习方法和深度学习方法两类。

在传统机器学习领域，随机决策树和随机森林已被证明在许多序列预测任务中具有快速有效的性能[6][7]。其中，袋装决策树（Bootstrap-aggregated decision trees, BT）[8]是一种有效的方法。在深度学习领域，递归神经网络（Recurrent Neural Networks, RNN）及其变体LSTM网络以及门控循环单元（Gated Recurrent Units, GRU）已被证明能成功地解决序列预测任务[9][10]。除了RNN及其变种之外，还有一些研究使用深度神经网络方法来解决序列预测问题，比如深度置信网络（Deep Belief Networks, DBN）[11]和堆叠稀疏自编码器（Stacked Sparse Autoencoders , SAE）[12]。在这些方法中，LSTM循环神经网络因具有学习训练数据之间的时间依赖性的记忆，在捕捉人类运动的转换规律方面表现出良好的潜力。

## VR领域中的视角预测

视角预测在VR领域中的相关研究根据预测基于的内容可以分为两类：基于轨迹的视角预测，以及基于内容的视角预测。根据预测结果可以分为两类：对视线的预测，以及对头部位置的预测。接下来阐述视角预测在VR领域方面的相关工作。

文献[13]使用缩略图、运动矢量和导航轨迹等特征在视频中执行注视预测。文献[14,15,16]使用先进的机器学习技术，包括神经网络在内的各种监督学习方法来提高固定检测中的特征提取和预测精度。文献[15]建立了一个CNN架构，并使用残差运动作为预测视频显着性的特征。文献[14]观察注视图与时空邻居之间的相关性，提供了一种量化观看者对视频的关注的方法。文献[16]采用静态图片中的显着性信息，并将相机运动考虑在内以进行动态视频中的显着性预测。文献[2]使用线性回归和3层多层感知器（Multilayer Perceptron, MLP）预测用户未来视角的中心位置。文献[3]提出了一种注视预测网络，同时利用过去的视角位置和视频的内容来预测未来几帧的视角轨迹或基于视频单元的注视概率图。文献[4]利用LSTM网络来对视角扫描的历史路径进行编码，并将隐藏态的特征与视觉特征相结合，进行最多1秒的预测。文献[5]提出了两种深度强化学习模型，一种是离线模型，仅根据视觉特征估计每一帧的潜在视角的热力图，另一种是在线模型，根据过去的头部位置和离线模型得到的热力图预测头部运动。

# **研究内容与方法(或算法)**/Contnts and Methods(or Algorithm)

## 场景构建

首先，本小组尝试搭建Unity的场景，作为项目的客户端。该场景不必特别复杂，但是需要具备预加载模型功能，以通过对比发现模型预加载所带来的改善。我们设计了两个版本的模型，最终采用了后一种版本。

第一个版本的场景如图2所示。本小组沿z轴方向每隔10m设定一个里程碑，里程碑上面记录了当前所行走的里程数，并在里程碑的右方设置了一个圆球。然而该场景存在着一些问题。比如，当用户控制角色时是随机移动的，并不一定会只沿着z轴的方向行走，因此很多里程碑可能并不会被用户看到。而没被看到的里程碑没有加载的需要，无法体现出预加载的优势。此外，该场景中的物体过于单一，不适用于预加载。本小组认为这个场景不能很好地具备预加载的需要，最终放弃了该版本场景。

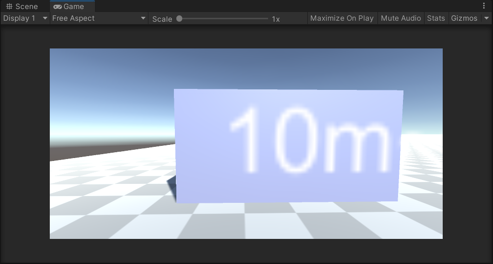
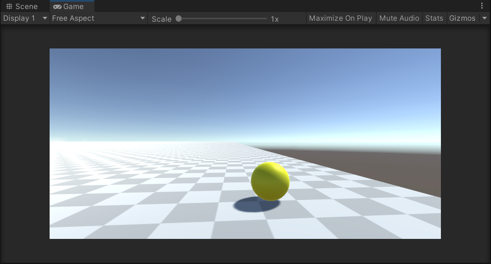
 

图2 场景版本一

考虑到版本一的不足，本小组进行了一系列改进，基于上海交通大学图书馆主馆及其周边景色设计了第二个版本的场景，如图3所示。在版本二中，本小组有意添加了一些模型，如马路、树木、建筑物等，这些模型相比于版本一而言更为复杂，如果不进行预加载，会给用户带来相对较差的观看体验。在本文的第4部分展示了预加载给用户体验带来的优化。

在版本二场景中，存在一些树木模型，这些树木会在地面投射出影子。如果等到用户的视线旋转到树木的位置才加载模型而不是预先进行预加载的话，当用户旋转到一定位置时，会明显感觉到画面上突然出现了树木的影子，从而给用户带来不友好的观看体验。在该场景中，本小组还设计了一条较长的马路。如果不使用预加载技术，那么只有当用户的视线旋转到马路正中间时，马路才会被加载到画面上。这会导致之前没有显示马路的地方突然多出了马路，同样给用户造成不友好的观看体验。本小组认为，版本二的场景具备研究本课题所需的条件，并且模型相较版本一而言更为复杂，为真实生活中复杂场景应用提供了借鉴意义。

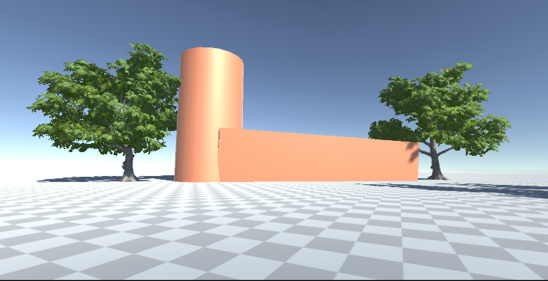
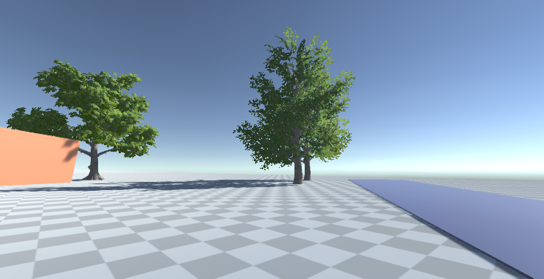
 

图3 场景版本二

## 摄像头数据采集

构建完毕场景后，需要提取用户的观看行为数据，以用于后续的预测研究。本小组将摄像头的移动数据作为用户的观看行为数据。摄像头数据的采集在Unity中完成。本小组在Main Camera上绑定CameraControllerWithKey.cs脚本和CameraAngle.cs脚本，分别用于控制用户视角变化和收集摄像头数据。

### 控制用户视角变化

在本系统中，用户通过键盘来分别控制摄像头的上下左右移动。在CameraControllerWithKey.cs中，通过调用Input.GetAxis()方法获得指定轴上的位置偏移量，值位于-1~1之间，平滑变化。之后，再将得到的值乘上一个浮点数，利用transform.rotate()方法实现摄像头的转动。键盘上下键和左右键操作分别对应的坐标轴参数为Vertical和Horizontal。本小组认为，通过调用Input.GetAxis()来使移动值平滑变化，有利于预测模型对用户突然改变方向的行为进行学习。

### 收集摄像头数据

由于无法找到能完全适用于本小组研究的开放数据集，本小组采用自行采集摄像头数据的方式来获得训练数据集，采集过程实现在CameraAngle.cs脚本中。在该脚本中，首先在Start()函数中创建一个csv格式的文件，用于记录摄像头的数据。文件的表头为Rotation\_x和Rotation\_y，分别表示摄像头x轴和y轴的旋转角度。在Update()函数中，设置一个frameCount自增变量，以记录当前帧数。由于摄像头转动角度在相邻几帧的变化不大，本小组设置每70帧采集一次摄像头数据，即，当frameCount值为70时。摄像头转动的角度通过this.transform.rotation.eulerAngles获得，将其对应的x轴、y轴角度写入到csv文件中，并将frameCount值设为0，为下一次采集做准备。在实际操作时，本小组随机调整摄像头转动角度，得到一定规模的摄像头数据集，用于后续研究。采集得到的摄像头数据集示例如表1所示。

表1 摄像头数据集示例

|  |  |
| --- | --- |
| **Rotation\_x** | **Rotation\_y** |
| 358.0996 | 22.01567 |
| 358.0231 | 22.09371 |
| 358.0231 | 22.54645 |
| 357.8503 | 23.63998 |
| 357.2317 | 24.4834 |

## 视线预测模型

收集到用户视角移动的数据后，本小组对项目的服务端进行构建。项目的服务端基于Python构建，包含一个视线预测模型，核心是通过收集到的摄像头数据训练一个LSTM网络。训练好的网络可以在前后端交互时进行实时预测，服务端将预测的摄像机视角信息传回给客户端。

### LSTM网络介绍

循环神经网络（Recurrent Neural Network, RNN）是一种用于处理序列数据的神经网络。不同于普通的CNN网络，RNN的输入不仅包含当前数据的x值，还包含了前几次输入的x值，如图4所示。将这些输入通过一些方式加权到一起后，才传给网络进行训练。这就意味着，网络的输出不仅受当前输入的影响，还受到前几次输入的影响。RNN网络的特性十分契合一些与上下文有关联的应用场景，比如本课题的研究问题。在本课题中，为了预测下一帧的用户视角信息，需要将前几帧的视角信息作为输入。但是，普通的RNN往往面临着无法处理较长序列的问题，存在梯度消失和梯度爆炸的现象。

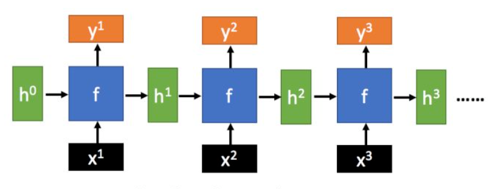


图4 RNN网络模型

长短期记忆（Long Short-Term Memory, LSTM）是一种特殊的RNN，它解决了普通RNN的问题，可以处理更长的序列，是当前比较流行和成熟的网络模型。LSTM与普通的RNN相比，多了三个控制器：遗忘门，输入门，输出门。这三个门可以让网络动态决定以前的信息遗忘多少，从而使更重要的信息被记住。LSTM网络模型结构如图5所示。

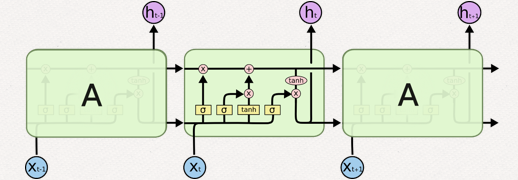


图5 LSTM网络模型

### 数据处理

如3.2.2节中所述，本小组在Unity中每隔70帧采集一次角色的视角信息，然后写入csv文件中，完成对摄像头数据的采集。得到数据集后，首先需要对数据进行处理。由于本小组采用的数据集是通过自行采集得到的，所以脏数据较少，处理也较为简单。接下来描述对数据的处理过程。

将数据读入Python程序后，首先对一些值为空或为负数的数据进行去除。接着，采用min-max的归一化方法，将每个轴的视角值归一化到0~1之间。然后，利用归一化后的序列数据构造网络输入和标签。由于本小组的网络是利用前5帧的信息作为输入来预测下一帧的信息的，所以每一个输入是5个连续时间的坐标信息，标签是这5个帧的后一帧的坐标信息。通过序列数据构造好网络输入和对应标签后，本小组将数据集划分成了训练集和测试集，划分比例为7：3。经过以上步骤处理，即可得到LSTM网络所需使用的训练集和测试集。

除此之外，本小组还对x轴的旋转数据进行了一定的改变。本小组发现，Unity中对x轴旋转角度的记录与常识不同。在Unity中，x轴不旋转时对应角度是0/360度。当x轴从水平开始进行顺时针旋转时，变化的角度在0~90度范围；进行逆时针旋转时，变化的角度在360~270度范围。这就导致x轴旋转角度的数据经常会在0与360之间跳变。考虑到这一问题，本小组对x轴的旋转做了一定的平移。当x轴在0~90度时，将其加上90度，使其范围变成90~180度。当x轴在360~270度时，将其减去270度，使其范围变成90~0度。这样一来，x轴的值就是连续变化的了，更有利于网络的训练。

### 训练网络及测试网络

处理好数据之后，需要搭建神经网络并进行训练。本小组使用Pytorch框架下的LSTM来实现LSTM网络。经过对实验结果的比对，本小组最终使用了一个3层的LSTM，然后连接到线形层。此外，本小组认为，用户视角在x轴上的旋转行为和在y轴上的旋转行为具有独立性，即摄像头的x轴旋转方向和角度与y轴无关。基于这一设定，本小组分别对x轴和y轴使用了独立的LSTM网络。数据流图如图6所示。

图示

描述已自动生成

图6 数据的划分与数据在网络中的流向

在训练阶段，本小组使用MSELoss作为损失函数，学习率为0.0002。训练了200epoch之后，loss下降至0.0002，说明在训练集上的效果十分优秀。当本小组将训练好的网络应用到测试集上进行测试时，发现同样能取得比较好的效果，这可能是因为预测的数据比较相关。有关训练结果的分析将在第4节中进行详细讨论。

## Unity与Python交互

设计好项目的前后端之后，接下来需要考虑如何实现前后端数据的传递。在本系统中，客户端每隔一段时间将前5帧的用户视角信息传递给服务端，服务端需要根据这5帧的数据，预测下一帧的用户视角位置，并传递回客户端。经过本小组的调研，发现有两种方案可以实现前后端之间的数据互传。一种是直接在Unity中调用Python脚本，另外一种是借助Socket通信。本小组对这两种方案都进行了实现，并对两种方案的效果进行了比较，最终选择了方案二作为本系统的前后端数据传递方式。下面将对这两种方案进行详细介绍。

### 直接在Unity中调用Python脚本

因为Python脚本可以直接在命令行中以“Python3 test.py”的形式执行，所以最直观的办法就是写一个Python脚本，该脚本可以创建LSTM网络，加载训练好的网络参数，然后将传入的参数作为网络输入，将网络输出打印出来。每当Unity需要得到预测值时，只需调用该Python脚本即可。在Unity中调用Python的主要思想是在C#中将Python解释器的路径、Python脚本的路径以及要传递的5个参数编码成字符串，然后创建一个进程，该进程直接将字符串作为命令在terminal中执行，最后得到terminal中输出的结果。

本小组实现了这种方案后，发现使用该方案会导致显著的卡顿，影响用户的观看体验。产生卡顿的原因主要是每次调用Python脚本都会重新加载预测模型，第4节中对此进行了详细讨论。

### 借助Socket，实现Unity项目和Python程序通信

在方案一无法奏效后，本小组使用了另一个性能良好的解决方案，即，采用Socket进行前后端通信。这种方法完美解决了方案一的问题。使用Socket通信，只需在最开始加载好预测模型和网络相关参数即可，不必每次预测都重新创建网络。同时，只需对每个Socket创建一个进程，减少了进程资源的浪费。具体的Socket通信方式如图7所示。

图示

描述已自动生成

图7 Socket通信方式

本小组基于Python搭建了服务端，借助了Python中的Socket库实现了server.py脚本。在该脚本中，首先创建LSTM网络，并加载训练好的网络参数。接着创建套接字，绑定IP地址和端口号。这里由于是测试环境，绑定的IP地址是127.0.0.1，端口号是5000。然后设置成监听状态。一旦有客户端的连接请求，就执行accept，获得客户端的相关信息并为该客户端服务。使用recv函数接收客户端发送过来的字节流，将字节流编码成字符串后获得当前5帧的视角信息。将视角信息放入网络中运行，得到的预测结果再通过send函数返回给客户端。

本小组基于Unity搭建了客户端，借助了C#中的System.Net.Sockets库实现了client.cs脚本。在Start()函数中，该脚本首先创建套接字，向服务端发送连接请求并且判断是否成功连接。在Update()函数中，该脚本每隔70帧记录一次关键帧的摄像头信息。关键帧的信息都被保存在buffer数组中，该数组只会保存最近5个关键帧的视角信息。每当新记录了一个关键帧，该脚本就调用Send()函数向服务端发送buffer中的数据，并且调用Receive()函数接收服务端返回的预测值。得到服务端返回的预测值后，即可判断物品是否会出现在用户将来的视角中了。

通过对方案二的实现，本小组发现，基于Socket通信的方法能起到很好的效果，不会产生卡顿，而且适用性很广。当需要在大型项目中进行资源预加载时，前后端通信保证了客户端的稳定流畅。相关的结果将在第4节中进行详细展示。

## 资源预加载

客户端接收到服务端传来的预测用户视角后，需要将即将出现或已经出现在用户视角内的资源加载到画面上。该部分分为两个步骤来完成，首先判断哪些资源需要进行预加载，其次再采用动态加载的方法对资源进行预加载，将物体渲染到画面上。

### 判断需要预加载的资源

根据本系统的设计宗旨，系统启动时，不对处于用户视角之外的资源进行加载，运行时再判断哪些资源即将出现在用户视角内，将这些资源加载到画面中。根据3.3节中提到的视线预测模型，已经可以得到预测的用户视角数据，因此只需采用一定的方法判断哪些资源处于预测的视角内即可。

本小组设置了两个摄像头来完成对预加载资源的判断，如图8所示，黄色为主摄像头视锥边框，红色为副摄像头视锥边框。主摄像头即Main Camera，为用户实际操控的摄像头，表示用户当前的视角；副摄像头Pre Camera所处位置和主摄像头一致，仅角度不同，将副摄像头的角度设为预测的用户视角对应的角度。

图片包含 桌子, 街道, 水, 交通

描述已自动生成

图8 主副摄像头示意图

设置好两个摄像头后，即可对需要进行预加载的资源进行判断。如果某个物体位于副摄像头的视角中，那么该物体就是即将出现在用户视角中的物体，系统需要对其进行预加载。由于预测会产生不可避免的轻微误差，为了避免判断错误，系统也要同时对出现在主摄像头视角中的物体进行加载。因此，判断哪些资源需要进行预加载的策略是：如果资源位于主摄像头或副摄像头的视角内，则该部分资源需要被加载；否则，不对其进行加载。

对物体是否出现在主副摄像头视角中的判断分别通过函数IsInView()和PredInView()来实现。IsInView()函数判断物体是否出现在主摄像头视角中，如果物体在主摄像头前面且物体的视口坐标在(0,0)和(1,1)之间，则说明物体出现在主摄像头视角中，返回True，否则返回False；PredInView()函数判断物体是否出现在副摄像头视角中，首先通过GeometryUtility.CalculateFrustumPlanes计算副摄像头的视景并返回它的六个面，如果对每个面而言，物体都在其正方向上，则说明物体出现在副摄像头视角中，返回True，否则返回False。对场景中的每一个物体，分别绑定一个Showxxx.cs脚本，每个脚本中都实现IsInView()和PredInView()函数。

### 动态加载资源

完成对需要加载的资源的判断后，系统对该部分资源进行加载。在Unity中，可以通过资源文件夹和资源包两种方式来在运行时动态加载资源。

资源文件夹是包含在构建的Unity播放器中但不一定链接到Inspector中的对象集合，通过在Project视图中创建一个名为“Resources”的文件夹来实现。将资源放入Resources文件夹，每当需要对资源进行加载时，调用Resources.Load()即可。

资源包是外部资源的集合。资源包通常位于Web服务器上，供用户动态访问。通过在Editor脚本中调用BuildPipeline.BuildAssetBundles()并指定一系列对象以及其他相关参数即可完成对资源包的创建。运行时，通过调用AssetBundle.LoadAsset()来对资源进行动态加载。

本文选择创建资源文件夹的方式来对资源进行动态加载。在每一个GameObject所绑定的Showxxx.cs脚本中，除了Start()、Update()和上文提到的IsInView()、PredInView()函数之外，另外实现Check()函数。首先在Start()函数中使用Resources.Load()来加载预设体资源，从资源文件夹中下载预设体并存放在GameObject中。此时资源还未进行实例化，故不会出现在场景中。在Update()函数中，调用IsInView()函数，如果IsInView()返回True，则说明资源出现在主摄像头视角范围中，对资源进行实例化。同理，在Check()函数中，调用PredInView()函数，如果PredInView()返回True，则说明资源出现在副摄像头范围中，需要对资源进行实例化。对预设体资源的实例化通过调用MonoBehaviour.Instantiate()函数来实现，该函数的实质是从预设体资源中克隆出一个属性与预设体完全相同的对象，来加载到当前场景中。

在Showxxx.cs脚本中实现好资源加载的相关方法后，需要在上层对其进行调用。Check()函数的调用发生在客户端接受到服务端传来的预测数据之后。在client.cs脚本中，如果成功接收到服务端传来的数据，则分别对每个对象的Check()函数进行调用。Update()函数在系统运行时逐帧自动调用。完整的动态加载资源流程示意图如图9所示。

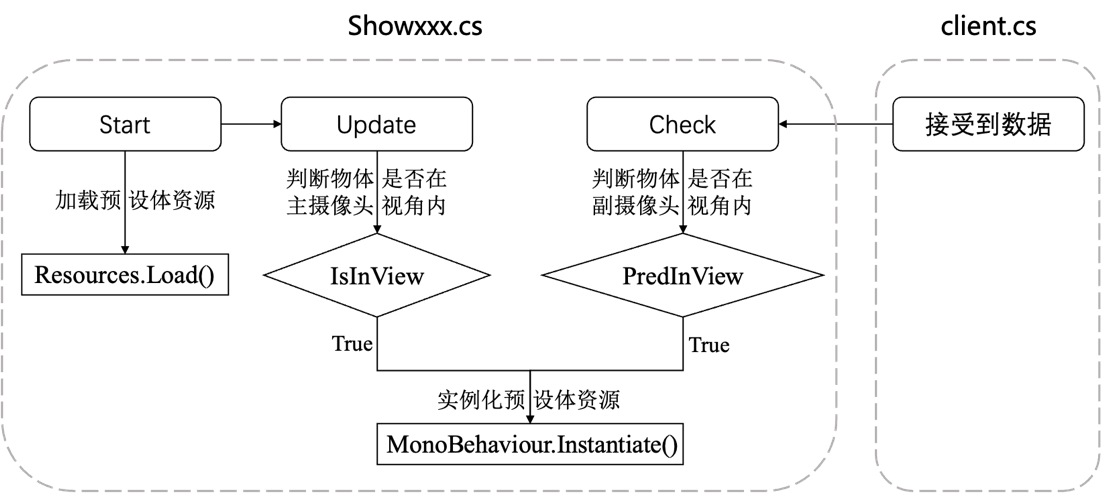


图9 动态加载资源流程图

# **实验结果与分析**/Experiment Results and Analysis

## 不同的数据采集方式对结果的影响

在准备工作阶段，本小组搭建了Unity场景，并且在该场景中收集了用户的视角信息。本小组认为，采集到的数据是否具备多样性会对网络的训练产生重要的影响。这里的多样性包括：需要向多个方向进行旋转；既要旋转x轴，也要旋转y轴；需要具备长时间静止不动的数据。一个直观的结论就是，如果训练数据越多样，那么网络最终的泛化能力就会越好。因此，本小组采用了两种不同的数据收集方式，产生了两组数据序列，并对相关结果进行了简单对比。

第一种采集方式是只将摄像头沿y轴进行顺时针旋转。第二种采集方式包括了一段时间的y轴顺逆时针旋转、一段时间的x轴顺逆时针旋转、一段时间的x，y轴同时旋转以及一段时间的静止不动，这些时间段都交错在一起，增加了数据集的复杂性。

将构建好的LSTM网络基于通过这两种采集方式得到的数据集进行训练，表现如图11所示。可以看到LSTM网络都已训练完成。

图表, 直方图

描述已自动生成 图表

中度可信度描述已自动生成

a) 方式一，y轴训练结果 b) 方式一，x轴训练结果

图表, 折线图

描述已自动生成 图表, 直方图

描述已自动生成

c) 方式二，y轴训练结果 d) 方式二，x轴训练结果

图10 不同数据采集方式得到的实验结果对比

接下来，将两种采集方式得到的网络应用到项目中进行实时预测。本小组发现，第一个网络对于y轴顺时针旋转表现较好，但是当摄像头沿着y轴逆时针旋转时，网络的预测就会有明显的滞后，无法做到提前预测用户视角。而且，当沿着x轴进行旋转时，网络无法产生预测结果。此外，当摄像头静止不动时，网络仍在产生移动的预测结果，需要经过较长的时间才能预测得到静止的数据。

使用基于第二个数据集训练的神经网络进行实时预测，本小组发现，第二个网络的预测表现比较均衡，对各个情况都能得到比较好的预测结果。而且，本小组发现，网络对于某个方向预判的好坏与否，与该方向的数据在数据集中占比有关。该方向的数据在数据集占比越多，则在实时预测时对于该方向的预测就会得到更敏捷准确的结果。

## LSTM结果分析

神经网络是本文工作的核心，它关乎到系统是否能够正确地预加载资源。本小组选用了LSTM神经网络，其在序列预测应用方面的优势已在许多相关研究中被证实。下面将从LSTM网络的结果和参数调整两方面，给出本小组在本项目中使用LSTM的经验。

在本系统中，本小组最终选用了回看数5，隐藏数12，层数3作为通用型LSTM的参数。经过200个EPOCH的训练后，预测y轴的net的loss下降到了0.0077，预测x轴的net的loss下降到了0.0001。最后几次epoch的loss变化如图11所示。可以看到，x轴的训练效果十分良好，这与本文在3.3.2小节中对x轴的旋转角度信息进行了平移有关。平移消除了x轴突变的情况，使网络更能捕获到关键特征。

文本

描述已自动生成 文本

描述已自动生成

a) y轴训练的最后几个epoch的loss变化 b) x轴训练的最后几个epoch的loss变化

图11 最后五个epoch的loss变化

从最后的结果可以看出，本文实现的网络可以获得较为出色的表现，能够较为准确地预测用户未来视角的位置。在用户观看方向突然发生改变时，网络也能够做出及时地应对。

在训练LSTM网络的过程中，本小组也收获了一些经验。首先要确定回看的个数。这个数字通常是根据具体问题来确定的。比如对于本课题的问题，预测的下一帧一般只与前面几帧的移动趋势有强相关性，所以回看数可以不用设置得很大。而如果是NLP类型的问题，理解某个句子的关键信息可能出现在句子开头，这种回看数就要设置大一点。确认完回看个数后，可以大致确定隐藏数和LSTM网络层数。本小组在调参的过程中发现，当隐藏数在回看数的2~3倍，网络层数在回看数的0.5~1倍时，通常能够得到不错的结果。而且由于网络层数对LSTM网络训练时间的影响特别大，可以先确定电脑可接收的网络层数，再调整隐藏数。本小组还对帧的采样率进行了调整。经过比较，本小组最终采用了每70帧采集一次数据的方式。该采样率能在保证数据与数据之间有明显差异的同时，还能具有较好的连续性。

## Unity与Python两种交互方式的结果对比

Unity与Python的交互关乎到预测的实时性。如果交互方式不得当，交互过程浪费大量时间，那么即使预测模型再优秀，也难以实现资源的预加载。本小组实现了两种Unity和Python交互的方案，并最终采用了使用Socket进行通信的方案。从结果上来看，通过Socket通信的方式得到的表现远超于直接调用Python脚本得到的表现。下面将展示两种交互方式的结果并进行比较，分析原因。

在方案一中，通过在Unity中直接调用Python脚本来进行Unity和Python之间的交互。虽然该方案实现起来较为简单而且容易理解，但是产生的结果十分卡顿，甚至导致项目无法流畅运行，严重影响用户的观看体验。尽管本小组尝试了一些改进方法，比如减少调用次数，减少不必要的赋值和循环等，但最终效果仍不理想。经过不断地调试和测量，本小组发现，每当Unity中调用Python脚本时，就会产生2~3s的卡顿，此时不管输入什么，Unity界面都会保持不动。这是因为在调用Python脚本时，当前Unity的进程被悬挂了。如果不悬挂Unity进程，那么等到Python脚本返回结果时，用户在Unity中的移动早已经过预测点。本小组认为这其中有两点原因。第一，这种方案每次预测都需要创建一个新进程，由于Unity请求预测特别频繁，会使操作系统产生极大的开销。第二，每次调用Python脚本时，都需要创建神经网络，重新加载网络参数的整个过程，这使得每次执行该Python脚本都需要较长的时间，最终导致Unity运行时有明显的卡顿感。

方案二通过Socket通信的方式能够得到明显优于方案一的表现。由于系统是在本机上运行，网络的延迟等瓶颈可以不用考虑。而且，由于服务端的Python进程已经创建好了网络并加载好了网络参数，唯一限制Unity和Python交互效率仅为神经网络进行计算所需的时间。从结果上可以看出，通过Socket通信的方式几乎感受不到Unity运行的卡顿，并且总是能及时地将预测值输出。除此之外，本小组还发现，通过简化LSTM中保存的数据量，或者使用性能更好的电脑，均可以减少延迟，使客户端更为及时地得到预测值。

## 视线预测对用户观看体验的影响

本系统对场景的资源采用了动态加载的方式，本小组希望在这种方式下产生的效果与在场景启动时加载全部资源无异。在实验中，本小组分别测试了系统在添加与不添加预测功能下的观看体验差异。当不添加预测功能时，系统不具备资源预加载过程，只有当物体实际出现在用户视角中时才被加载到画面上。在这种实现方式下，用户可以很明显地感知到画面上物体出现的延迟。以马路为例，从用户视角旋转到马路区域到马路出现在画面上，这中间相差近1s的时间。而添加预测功能后，系统具备预加载过程，如图12所示，黄色视锥为用户当前视角，红色视锥为系统预测视角，马路已被检测位于红色视锥区域内并被预加载，当黄色视锥移动到马路区域时，马路已被提前加载到画面上，延迟的问题即可被解决。经过多次反复测试，本小组随机尝试了多种摄像头运动方式，均能得到流畅的观看体验，说明视线预测结果较为准确，且能在节约能耗的同时使用户的观看体验与非动态加载资源无异。

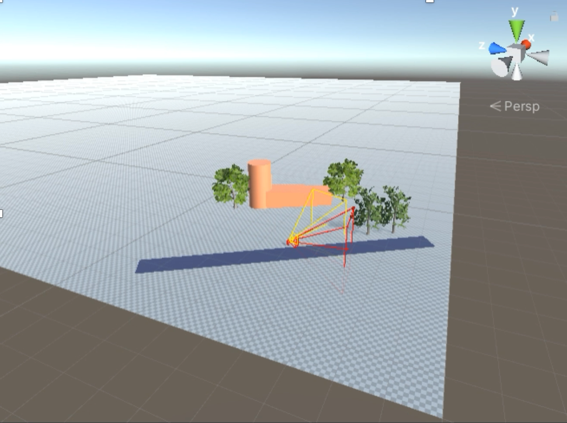


图12 添加预测功能后的马路观看效果

# **特色与创新**/ Distinctive or Innovation Points

本文实现了一个基于视线预测的资源预加载系统，通过对用户未来的视线进行预测动态加载资源，而不是在一开始就加载出所有资源，以此来达到节约能耗的目的。系统分为客户端和服务端两部分。客户端由Unity实现，建立了一个与生活现实息息相关的3D全景场景，场景中的资源在最初是没有被加载的。用户可通过键盘控制摄像头，调整观看方向。系统按一定时间间隔记录摄像头数据，并传输给服务端。服务端由Python实现，包含一个基于LSTM网络的视角预测模型，通过学习客户端传来的摄像头历史数据来对用户下一帧的视角进行预测。服务端得到预测视角后，将其传输给客户端，客户端进一步判断场景中有哪些对应在预测的视角上的物体，对这些物体资源进行提前加载。通过该系统，可以得到较为准确的用户视角预测结果，有效地对资源进行预加载，且服务端与客户端之间的通信速度很快，可以给用户带来流畅的体验，资源的预加载过程对于用户来说是无法感知的。本系统部署在单台笔记本电脑上，不需要额外的硬件设备，具有良好的应用性和可移植性。未来可以为系统设置不同的全景场景来测试系统的泛用性，并将系统连接到VR设备上实现完整的VR应用。

此外，本文还在数据采集、Unity和Python交互方面进行了多种方案的尝试，并对预测模型的不同参数进行了大量实验，通过对比和分析得到了宝贵的经验和结论，具有一定的借鉴意义和研究价值。

References:

1. Park S, Bhattacharya A, Yang Z, et al. Advancing user quality of experience in 360-degree video streaming[C]//2019 IFIP Networking Conference (IFIP Networking). IEEE, 2019: 1-9.
2. Bao Y, Wu H, Zhang T, et al. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos[C]//2016 IEEE International Conference on Big Data (Big Data). IEEE, 2016: 1161-1170.
3. Fan C L, Lee J, Lo W C, et al. Fixation prediction for 360 video streaming in head-mounted virtual reality[C]//Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video. 2017: 67-72.
4. Xu Y, Dong Y, Wu J, et al. Gaze prediction in dynamic 360 immersive videos[C]//proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 5333-5342.
5. Xu M, Song Y, Wang J, et al. Predicting head movement in panoramic video: A deep reinforcement learning approach[J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 41(11): 2693-2708.
6. Kim T, Yue Y, Taylor S, et al. A decision tree framework for spatiotemporal sequence prediction[C]//Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 2015: 577-586.
7. Yan C, Zhang Y, Xu J, et al. A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors[J]. IEEE Signal Processing Letters, 2014, 21(5): 573-576.
8. Breiman L. Random forests[J]. Machine learning, 2001, 45(1): 5-32.
9. Alahi A, Goel K, Ramanathan V, et al. Social lstm: Human trajectory prediction in crowded spaces[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 961-971.
10. Liu J, Shahroudy A, Xu D, et al. Spatio-temporal lstm with trust gates for 3d human action recognition[C]//European conference on computer vision. Springer, Cham, 2016: 816-833.
11. Mohamed A, Dahl G E, Hinton G. Acoustic modeling using deep belief networks[J]. IEEE transactions on audio, speech, and language processing, 2011, 20(1): 14-22.
12. Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. science, 2006, 313(5786): 504-507.
13. Mavlankar A, Girod B. Video streaming with interactive pan/tilt/zoom[M]//High-Quality Visual Experience. Springer, Berlin, Heidelberg, 2010: 431-455.
14. Alshawi T, Long Z, AlRegib G. Understanding spatial correlation in eye-fixation maps for visual attention in videos[C]//2016 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2016: 1-6.
15. Chaabouni S, Benois-Pineau J, Amar C B. Transfer learning with deep networks for saliency prediction in natural video[C]//2016 IEEE International Conference on Image Processing (ICIP). IEEE, 2016: 1604-1608.
16. Nguyen T V, Xu M, Gao G, et al. Static saliency vs. dynamic saliency: a comparative study[C]//Proceedings of the 21st ACM international conference on Multimedia. 2013: 987-996.

时间安排与分工统计表

|  |  |  |  |
| --- | --- | --- | --- |
| **组员信息**（含组长） | | | |
| 学生姓名 | 姚以真（组长） | 学 号 | 518030910185 |
| 项目分工 | 项目选题及分工，方案调研，系统实现，小组成果汇报及展示，报告撰写 | | |
| 学生姓名 | 符殊源 | 学 号 | 517030910124 |
| 项目分工 | 项目选题，方案调研，小组成果汇报及展示，报告撰写 | | |
| **时间安排/**  **Schedule** | 第8周~第9周 项目选题  第9周~第11周 方案制定，场景构建，摄像头数据采集，项目成果汇报  第11周~第13周 场景完善，预测模型构建，项目成果汇报  第13周~第15周 数据处理，前后端交互，实验及分析，撰写总结报告  第15周~第16周 项目完善，撰写总结报告 | | |