OpenCV 图像捕捉与 Mediapipe 与 Unity 引擎的结合

罗登仁 [1*] 李易朗 [2*]
(四川大学 软件学院 四川省 成都市 610065)
(1* 452966434@qq.com，2* 935937451@qq.com)

摘 要: 这个项目旨在探索如何结合 Python 中的 OpenCV 图像捕捉技术和 Mediapipe 库，实现手势和人体动作的实时检测与识别。OpenCV 作为一个强大的计算机视觉库，能够处理图像和视频流，而 Mediapipe 则提供了优秀的机器学习模型和工具，适用于关键点检测、手部追踪和姿势估计等任务。通过这两个工具的结合，我们可以利用摄像头捕捉到的视频流，进行手势和人体动作的识别。这对于实时交互应用、体感游戏或者虚拟现实场景具有重要意义。我们的目标是深入了解这些技术的实际运用，从而将识别出的手势或人体动作信息传输至 Unity 引擎中，以实现角色或物体的实时运动。本次项目目标是探索处理图像数据、运用机器学习模型进行关键点检测和姿势识别的方法，同时了解如何在 Python 环境中处理数据流，并将识别结果有效地传递到 Unity 引擎中，以便在虚拟场景中实现手部和人体模型的实时运动。这将为探索基于计算机视觉和深度学习的实时动作捕捉技术提供宝贵的实践经验和理论支持。

关键词: OpenCV；Mediapipe；Unity

Military education cultivates and promotes the collectivist spirit of college students

Luo Dengren[1*] Li Yilang[2*]
(College of Sichuan, Sichuan University, Sichuan Chengdu 610065, China)

Abstract: This project aims to explore how to combine OpenCV image capture technology in Python with the Mediapipe library to achieve real-time detection and recognition of gestures and human movements. OpenCV, as a powerful computer vision library, can handle image and video streams, while Mediapipe provides excellent machine learning models and tools suitable for tasks such as keypoint detection, hand tracking, and pose estimation. By combining these two tools, we can use the video stream captured by the camera to recognize gestures and human movements. This is of great significance for real-time interactive applications, sensory games, or virtual reality scenes. Our goal is to gain a deeper understanding of the practical applications of these technologies, in order to transmit the recognized gestures or human motion information to the Unity engine for real-time motion of characters or objects. The goal of this project is to explore methods for processing image data, using machine learning models for key point detection and pose recognition, while understanding how to process data streams in a Python environment and effectively transmit recognition results to the Unity engine, in order to achieve real-time motion of hands and human models in virtual scenes. This will provide valuable practical experience and theoretical support for exploring real-time action capture technologies based on computer vision and deep learning.

Keywords: OpenCV; Mediapipe; Unity

0 引言

在当今数字化时代，科技的迅猛发展为人们的日常生活注入了更多的便利与创新。随着计算机视觉和人工智能技术的不断演进，图像处理领域也取得了巨大的进展。本文将介绍一项令人振奋的技术，即如何运用 Python 编程语言结合 OpenCV 图像捕捉与 Mediapipe 库的强大功能，实现对使用者手势和人体动作的高效检测与识别。

在这个信息爆炸的时代，图像处理不再是一项陌生的技术，而是与我们生活息息相关的重要组成部分。通过 OpenCV，我们能够借助计算机对图像进行高效处理，为视觉数据提供丰富的信息。而 Mediapipe 库，则作为一个强大而灵活的工具，进一步拓展了图像处理的边界，为我们提供了便捷的人体姿态和手势识别功能。

随着虚拟现实（VR）和增强现实（AR）技术的崛起，人机交互方式也在不断演进。本文所介绍的技术不仅能够在二维图像上准确捕捉手势和人体动作，更具有将这些信息同

作者简介: 罗登仁 (2003—)，男，海南，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；
李易朗 (2003—)，男，四川，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；

步至 Unity 引擎的能力，实现了与虚拟环境的无缝融合。这项技术的引入不仅为游戏开发、虚拟现实应用等领域提供了全新的可能性，同时也丰富了人机交互的方式，使用户能够以更自然的方式与计算机进行沟通与互动。

1 相关技术介绍

1.1 Mediapipe

[1]MediaPipe 是一款由 Google Research 开发并开源的多媒体机器学习模型应用框架。其提供了一系列工具和机器学习模型，包括姿态估计、手部追踪和面部检测等，可精准地捕捉人体动作、识别手势和检测面部特征。例如，姿态估计模型能够准确追踪人体关键点，实现运动分析和姿势识别；手部追踪模型能捕捉手部动作，用于交互式应用；而面部检测则可以辨识面部特征，支持面部表情分析等。此框架跨平台支持，可在各种操作系统上运行，为开发者在计算机视觉、增强现实和虚拟现实等领域提供了高效、可靠的多媒体处理解决方案。

1.2 OpenCV 图像捕捉技术

[1]OpenCV 作为一款广泛应用的计算机视觉库，提供了强大的图像捕捉技术。通过使用 cv2.VideoCapture()函数初始化摄像头设备并调用 read()函数捕获图像帧，开发者能够轻松地从摄像头或视频文件中获取图像数据。这些数据可以被用于各种应用，包括实时监控、人机交互中的手势识别、医疗影像分析以及自动驾驶中的图像处理等领域。OpenCV 的图像捕捉功能为图像处理和计算机视觉领域的开发提供了便捷且高效的工具和接口

2 项目实现过程

2.1 手部特征点捕获

一个简单的实时手部追踪程序，它使用 cvzone 库，其中包含了一个 HandTrackingModule 模块来检测和跟踪手部。同时利用了 OpenCV 来捕获摄像头的视频流，并使用 Socket 来发送手部坐标数据。

```
pTime = 0

cap = cv2.VideoCapture(0)
cap.set(3, 1280)
cap.set(4, 720)
success, img = cap.read()
h, w, _ = img.shape
detector = HandDetector(detectionCon=0.8, maxHands=2)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
serverAddressPort = ("127.0.0.1", 5052)
```

图 1 捕获摄像头的视频流，并使用 Socket 来发送手部坐标数据

初始化摄像头和手部检测器，并开启 UDP 套接字传输，设定了检测手部的置信度阈值为 0.8，并限制最多检测 2 只手。

```
while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)  # with draw
    data = []

    if hands:
        hand = hands[0]
        lmList = hand["lmList"]
        for lm in lmList:
            data.extend([lm[0], h - lm[1], lm[2]])

        sock.sendto(str.encode(str(data)), serverAddressPort)
```

图 2 特征点的坐标转换

使用 OpenCV 的 cap.read()方法读取摄像头的一帧图像。success 会返回一个布尔值，表示读取图像是否成功，捕获到图像数据。detector.findHands(img)会将图像传递给手部检测器 detector，该检测器使用 cvzone 中的手部追踪模块，用于识别图像中的手部，并返回手部信息列表 hands 和被标记过的图像 img。

如果检测到手部捕获条件成立，取出第一只检测到的手，并获取手部关键点的坐标列表 lmList。然后遍历这些坐标点，将每个坐标点的 x、y、z 值组成的列表（[lm[0], h - lm[1], lm[2]]）添加到 data 列表中。这里 h - lm[1]将 y 轴坐

*作者简介: 罗登仁 (2003—)，男，海南，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；
李易朗 (2003—)，男，四川，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；

标翻转，使其符合常见的图像坐标系。在这部分只用一只手作为演示，如果添加第二只手可以另外定义一组手部特征点序列通过 UPD 传输到 unity 实现，同时可以扩展到全身的特征点识别，只需对每一组特征点进行恰当的捕获和传输。

之后使用 OpenCV 的 cv2.imshow()显示当前帧的图像窗口，并通过 cv2.waitKey(1)等待 1 毫秒，以便处理键盘输入。这样可以持续显示摄像头捕获到的图像，并实时更新手部追踪的效果。

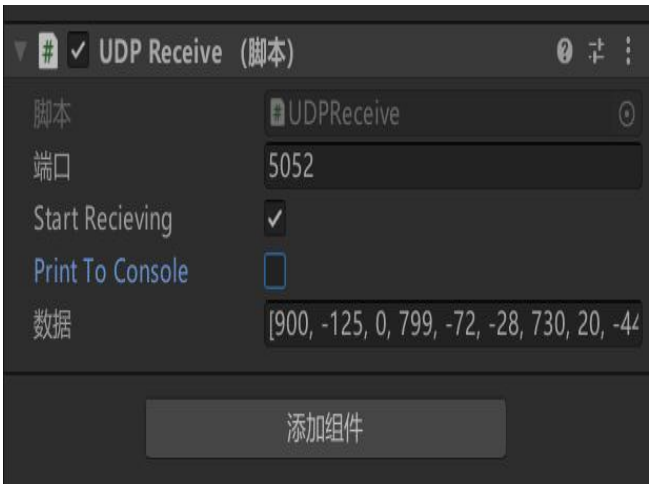

图 3 实时更新的坐标点位

在 unity 对手部坐标数据的处理过程中，只需对数据的坐标系进行可能的调整，适应 Unity 场景中的坐标系统和摄像机位置，消除位置偏移，保证手和肢体的姿态和视角处于正确的状态

2.2 特征点测试

关键点数量。MediaPipe 手部检测模型通常可以识别手部的关键点，一般是 21 个关键点。这些关键点覆盖了手掌、手指和手腕等部位，用于描述手部的姿势和形状。

关键点表示：每个关键点有自己的坐标位置（x，y，

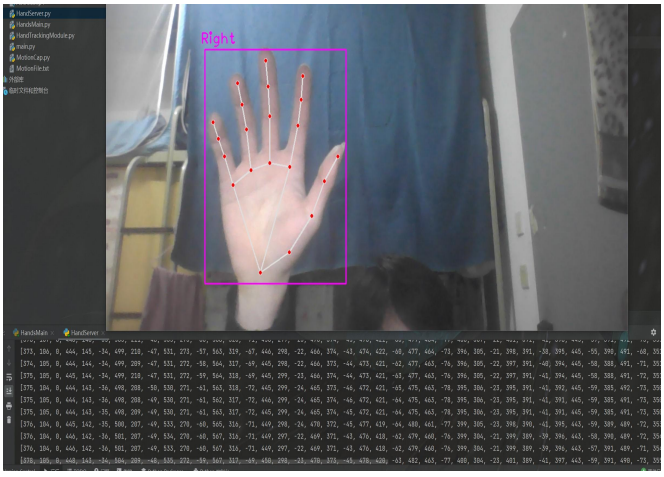

图 4 特征点测试

对 Mediapipe 手部特征点识别和读取摄像头流式数据，通过 socket 进行流式传输的数据测试。

这些坐标的相对位置与 Unity 摄像机的关系是实现基于相机视角的手部追踪和互动的关键部分。Unity 中的摄像机视角用于观察和渲染场景，因此，关键点的坐标需要转换为摄像机的视角空间，以便正确地在场景中显示手部模型或实现互动效果。

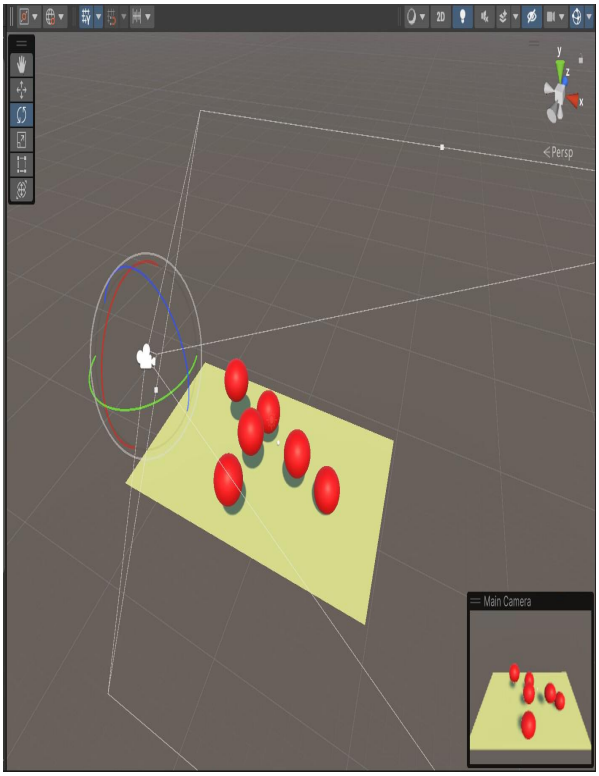

图 5 Unity 摄像机视角转换

通过将关键点坐标从世界空间或其他坐标空间转换到摄像机的局部空间，可以使得这些坐标相对于摄像机的位置和朝向进行描述，从而进行相机视角转换。

这种转换通常需要使用 Unity 中的 Camera 对象的方法来将世界空间坐标转换为视口坐标或屏幕坐标。

关键点的相对位置也可以用于手势识别、交互和渲染等功能。比如，可以通过判断关键点的相对位置来识别特定手势动作，或者根据手部位置与摄像机的关系在场景中实时渲染手部模型。如下通过手势识别到特定操作从而操控鼠标。

*作者简介: 罗登仁 (2003—)，男，海南，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；
李易朗 (2003—)，男，四川，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；

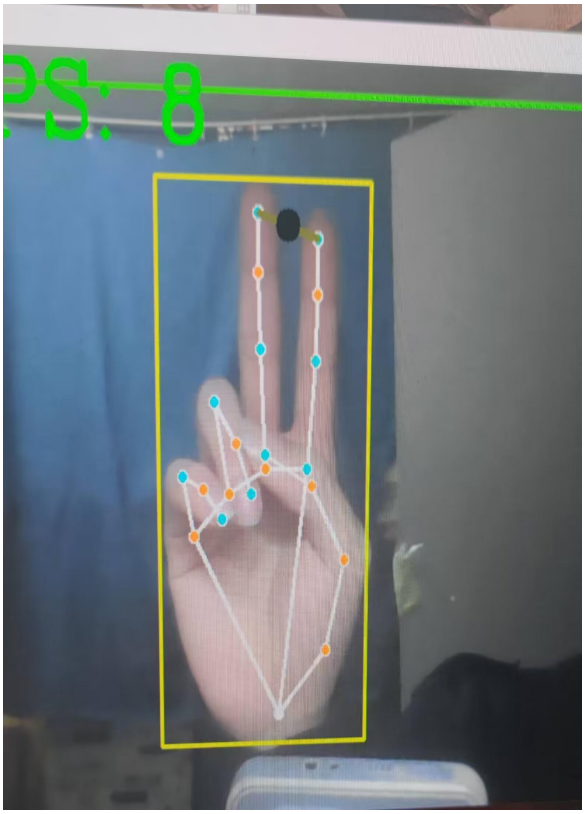

图 6 手势识别

2.3 Unity 引擎结合

2.3.1建立模型

首先在 Unity 中搭建一个手部的建议模型。根据手部特征点图（图 1）的提示，我们创建了 20 个球体对象来作为手部的特征点，同时创建了 21 个名为 Line 的空对象，并给 Line 挂载上 Line Renderer 组件来为特征点之间连线。

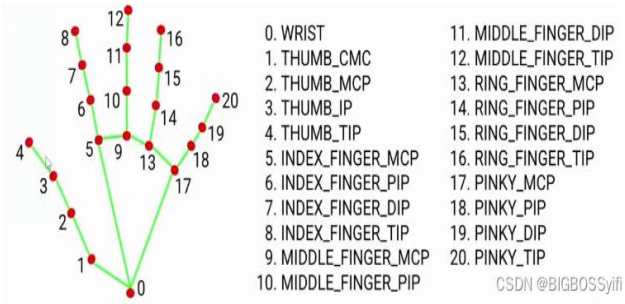

图 7 手部特征点图

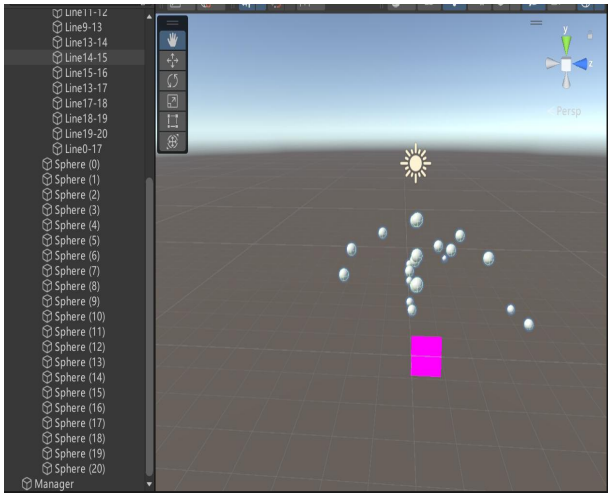

图 8 创建的 20 个球体和 Line 对象

2.3.2编写脚本

首先编写 Line 脚本

```
public class Line : MonoBehaviour
{

    LineRenderer lineRenderer;

    public Transform origin;
    public Transform destination;

    void Start()
    {
        lineRenderer = GetComponent<LineRenderer>();
        lineRenderer.startWidth = 0.2f;
        lineRenderer.endWidth = 0.2f;
    }

    void Update()
    {
        lineRenderer.SetPosition(0, origin.position);
        lineRenderer.SetPosition(1, destination.position);
    }
}
```

图 9 Line 脚本

Line 脚本中的功能简要说明如下：

LineRenderer 设置：在 Start 函数中，通过 GetComponent<LineRenderer>() 来获取当前物体上的 LineRenderer 组件，并设置了线的起始宽度和结束宽度，使用了 lineRenderer.startWidth 和 lineRenderer.endWidth。

更新线的位置：Update 函数中，通过 SetPosition 方法不断更新线的端点位置。它将线的第一个端点（索引为 0）设置为 origin 的位置，将线的第二个端点（索引为 1）设置为 destination 的位置。

*作者简介: 罗登仁 (2003—)，男，海南，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；
李易朗 (2003—)，男，四川，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；

将 Line 挂载到每一个 Line 对象上，同时确定其 origin 和 destination 是哪两个球体对象。这样就可以在两个特征点之间生成连线，且随着特征点的变化而变化。

然后创建一个 Manager 空对象，挂载 UDPReceive 脚本和 HandTracking 脚本。

UDPReceive 目的是实现 UDP 协议的数据接收功能。它使用了一个后台线程来持续监听指定端口（默认为 5052）上的 UDP 数据包。通过 UdpClient 建立了一个 UDP 客户端，一旦接收到数据，就会将字节数据转换为 UTF-8 编码的字符串，并将其存储在名为 data 的变量中。同时，该脚本提供了可选功能，可以控制是否将接收到的数据打印输出到 Unity 编辑器的控制台上，这对于调试和验证接收到的数据非常有用。

```
public class UDPReceive : MonoBehaviour
{

    Thread receiveThread;
    UdpClient client;
    public int port = 5052;
    public bool startRecieving = true;
    public bool printToConsole = false;
    public string data;


    public void Start()
    {

        receiveThread = new Thread(
            new ThreadStart(ReceiveData));
        receiveThread.IsBackground = true;
        receiveThread.Start();
    }


    private void ReceiveData()
    {

        client = new UdpClient(port);
        while (startRecieving)
        {

            try
            {
                IPEndPoint anyIP = new IPEndPoint(IPAddress.Any, 0);
                byte[] dataByte = client.Receive(ref anyIP);
                data = Encoding.UTF8.GetString(dataByte);


            }
            catch (Exception err)
            {
                print(err.ToString());
            }
        }
    }
```

图 10 UDPReceiver 脚本

HandTracking 是一个 Unity 中的脚本，用于接收通过 UDP 协议传输的手部追踪数据。它通过 UDPReceive 组件获取数据，并在每一帧中更新。首先，从 UDP 接收器中获取字符串形式的数据，然后对数据进行处理，去除可能的特殊字符。接着，将字符串分割为手部关键点的坐标信息，这些信息被假设为以一定格式顺序存储在 UDP 接收到的字符串中。随后，使用 float.Parse()转换数据为浮点数，并将这些坐标应用到 handPoints 数组中对应的球体对象的位置属性上，从而实现手部关键点的位置更新，可能用于手势追踪或交互功能。

```
public class HandTracking : MonoBehaviour
{
    public GameObject gameObject;
    public UDPReceive udpReceive;
    public GameObject[] handPoints;
    void Start()
    {
        udpReceive = gameObject.GetComponent<UDPReceive>();
    }

    void Update()
    {
        string data = udpReceive.data;

        data = data.Remove(0, 1);
        data = data.Remove(data.Length - 1, 1);
        string[] points = data.Split(',');


        for (int i = 0; i < 21; i++)
        {

            float x = 7 - float.Parse(points[i * 3]) / 100;
            float y = float.Parse(points[i * 3 + 1]) / 100;
            float z = float.Parse(points[i * 3 + 2]) / 100;

            handPoints[i].transform.localPosition = new Vector3(x, y, z);

        }
    }
}
```

图 11HandTracking 脚本

在处理实时数据时，特别是涉及到位置坐标等信息时，测试过程变得至关重要。针对这段代码中的 for 循环，我们进行了一些测试来确保它能够正确地解析并应用手部关键点的位置信息到相应的 handPoints 数组中的对象。

测试过程包括以下步骤：

步骤一：数据接收和处理

数据接收验证： 确保 UDPReceive 组件能够成功接收并存储 UDP 数据。检查数据是否正确、是否有特殊字符需要去除。

数据解析验证： 在接收的数据中手动输入一些已知的数据，然后运行代码，检查 data 变量是否正确解析为字符串数组 points，并且每个关键点的坐标值是否正确解析。

步骤二：手部关键点坐标计算

坐标转换验证： 用已知的测试数据替换实际接收到的数据，并检查计算出的 x、y、z 坐标值是否正确。

对比数据验证： 将测试数据手动转换为期望的坐标，然后对比计算出的坐标值和期望的值，确保计算逻辑正确。

步骤三：游戏对象更新

游戏对象位置更新： 确保 handPoints 数组中的对象正确地获取并更新了位置信息。我们在 Unity 编辑器中手动修改一些位置值，然后观察是否与实时更新的位置一致。

位置范围验证： 确保转换后的坐标值适用于场景。可能需要进行一些限制或调整，以确保手部关键点在游戏场景中的位置正确映射。

*作者简介: 罗登仁 (2003—)，男，海南，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；
李易朗 (2003—)，男，四川，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；

这些测试确保代码能够正确地解析和应用从 UDP 接收到的手部关键点的位置信息，并将其实时应用于 Unity 中的游戏对象。

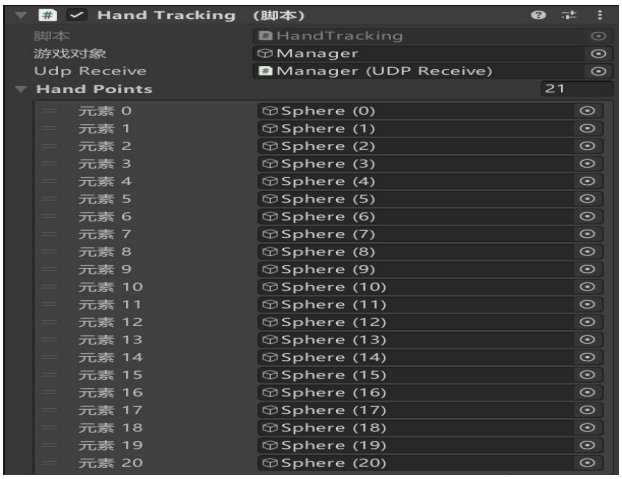

图 12 HandPoints 对象

综上便可实现手部模型在 unity 中实时运动

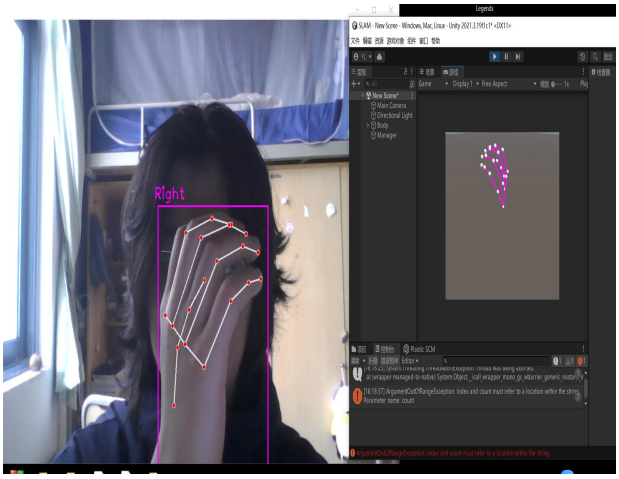

图 13 效果展示

3 项目未来发展和前景

因团队自身 unity 技术不足，预期目标没有完全实现。但该项技术在游戏开发领域具备引人瞩目的前景。借助 Python 结合 OpenCV 进行图像捕捉，结合 Mediapipe 库实现手势和人体动作的高效检测与识别，并将识别结果同步至 Unity 引擎，游戏体验将焕发新生。未来，玩家可通过自身动作和手势参与游戏，创造出更沉浸、真实的虚拟环境。这一技术也将拓展[1]游戏教育应用，激发学生学习兴趣，同时在康复领域发挥作用，辅助康复训练。这种技术的发展不仅将提升游戏互动性，更将深刻影响游戏开发，成为游戏制作中的重要创新方向，为游戏体验带来更多个性化和深度参与的可能性。

3.1 与 AR 技术的结合

这种技术在[2]AR 领域的应用结合能够产生非常独特的魅力和新奇的视觉效果，同时对很多行业的发展都有其独特的和广泛的应用领域。如虚拟物体操作、远程教育和合作等。

通过在远程位置使用摄像头捕获手部动作，利用类似之前提到的方法，使用 MediaPipe 或其他手部追踪技术，将手部姿态的坐标数据发送到 Unity 中。借助 AR 技术，可以在远程端捕获到的手部数据基础上，在 AR 环境中呈现虚拟的手部模型，实现远程手势和动作的跟踪。

利用远程手部追踪，用户可以在 AR 场景中操控虚拟物体，例如移动、旋转或放置。通过手势识别和 Unity 中的物理引擎，可以实现对虚拟物体的实时操作，使远程用户能够共同探索和互动。

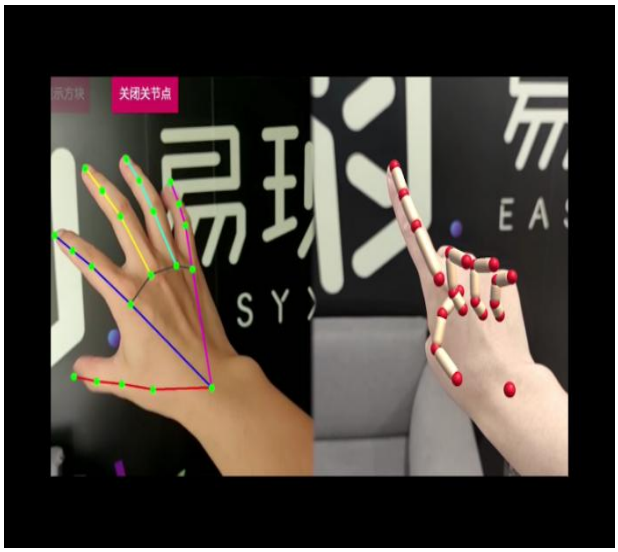

图 14 示例

在教育领域，可以利用远程手部追踪和 AR 技术，实现远程教学和合作。教师或专家可以在 AR 场景中展示知识或技能，学生或合作者可以通过远程手部追踪参与互动，进行实时的学习或合作。

3.2 实时远程互动

得益于关键点识别将现实同步到虚拟的操作，这样的应用模式使得近些年的元宇宙构想有了新的想象力。如远程手部追踪、共享虚拟空间等。

通过在远程位置使用摄像头捕获手部动作，利用类似之前提到的方法，使用 MediaPipe 或其他手部追踪技术，将手部姿态的坐标数据发送到 Unity 中。借助 AR 技术，可以在远程端捕获到的手部数据基础上，在 AR 环境中呈现虚拟的手部模型，实现远程手势和动作的跟踪。

在 AR 环境中创建一个共享的虚拟空间，使远程用户能够在同一场景中进行互动。Unity 与 AR 技术的结合可以实

*作者简介: 罗登仁 (2003—)，男，海南，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；
李易朗 (2003—)，男，四川，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；

现共享的虚拟环境，让远程用户可以看到彼此的虚拟模型或角色，并通过手势进行互动。

3.3 问题和挑战

网络延迟和同步。在远程互动中，网络延迟是一个重要的问题，会影响手部追踪数据的实时性和准确性。因此，需要针对网络延迟设计同步策略，比如在本次项目中所采用的 UDP 传输方式有效地保证了传输的实时性。但在真实场景使用中要确保安全性和实时性还需要更加高效的传输协议支持，以保持远程互动的流畅性。

精确性和用户体验。手部追踪的精确性和实时性是提供出色用户体验的关键。随着技术的进步，需要不断改进手部追踪技术和 AR 渲染技术，以提高精确度和用户体验。

结合 Unity 和 AR 的远程虚拟互动有着巨大的潜力，可以用于教育、娱乐、协作和远程工作等多种场景。但是仍然需要在技术和用户体验上不断进行改进和创新，以实现更加丰富、流畅和真实的远程互动体验。

参考文献

[1] 谢鹏.基于 OpenCV+MediaPipe 实现运动姿态 AI 检测在体育训练中的应用[J].无线互联科技,2023,20(18):100-104.

[2] 窦朝勋,左志斌.基于 Unity3D 和 MediaPipe 的虚拟人物驱动系统[J].河南科技,2023,42(21):18-22.DOI:10.19968/j.cnki.hnkj.1003-5168.2023.21.004

*作者简介: 罗登仁 (2003—)，男，海南，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；
李易朗 (2003—)，男，四川，无职称，四川大学本科软件工程在读，否 CCF 会员，主要研究方向：软件工程；