虚拟化身流：元宇宙体验的降本方案

Joseph Changjosephjbc@gmail.com

arXiv:2304.01443v1 [cs.HC] 4 Apr 2023

摘要

通过 VR 头显体验元宇宙是一个迅速发展的理念，但目前较高的入门成本限制了许多用户的使用。本项目旨在利用 Web 技术，为用户提供体验沉浸式元宇宙的便捷入口。所开发的平台允许用户仅使用网页浏览器、麦克风和网络摄像头，即可与渲染出的虚拟化身互动。应用借助 Google 的 WebGL 和 MediaPipe面部追踪人工智能模型，为用户实时生成三维人脸网格。平台通过客户端到客户端流媒体集群建立连接，并由客户端通过 WebRTC 协商 SRTP协议，以实现直接数据流传输。此外，本项目还通过无服务器、分布式、自动扩缩容、高韧性且安全的架构，应对后端挑战。该平台提供了一种可扩展且无需专用硬件的方案，让用户体验近乎沉浸式的元宇宙，并有望在未来与游戏服务器集群集成。本项目为打造更具包容性、让更广泛受众都能体验的元宇宙迈出了重要一步。

关键词 元宇宙 ·人脸网格 ·流媒体

1 引言

元宇宙是一个旨在通过技术 [22]复制物理世界的数字宇宙。随着虚拟现实（VR）技术的发展，VR头显已成为访问元宇宙的主要接口。VR头显带来的沉浸式体验有望彻底改变人际交互。这些设备可以采集用户的上半身运动数据并创建虚拟化身，让用户通过移动自己的身体，在虚拟数字世界中行动。此外，VR头显配有两个屏幕和扬声器。屏幕根据虚拟世界中物体的距离产生视差，模拟人类在现实世界中的三维视觉感受。扬声器则通过立体声模拟现实世界中的三维空间感和声音方向。通过将身体运动数据与数字界面相结合，VR头显创造出一种虚拟现实体验，模拟物理世界的空间环境。这项技术使用户无论身处何地，都能在虚拟但具有空间感的环境中彼此互动。它的应用场景十分广泛，包括虚拟会议、游戏和教育。

然而，高昂的入门成本阻碍了消费者投资这项技术，尤其是那些不了解或尚未体验过沉浸式元宇宙潜力的人。在此，我们提出一种入门级沉浸式元宇宙体验方案，结合人脸跟踪、三维网格渲染和直播技术。此外，我们的方案还可应用于虚拟 Omegle 聊天和虚拟 YouTuber 配置。前者在虚拟化身网格上呈现面部动作和表情，为用户与随机对象进行一对一视频聊天提供安全空间。后者利用软件和网络摄像头跟踪并驱动角色动画，无需昂贵的追踪头显。

为了提供部分沉浸式元宇宙体验，我们提出了一种解决方案，用以替代 Meta 的三维会议通话功能，该功能需要昂贵的 VR头显 [6]。简而言之，我们使用人工智能人脸网格模型追踪用户的面部表情、位置和朝向，再通过 WebGL 渲染管线将三维网格数据渲染到二维屏幕上，让用户仅凭网络摄像头、麦克风和网页浏览器即可体验模拟的三维会议。当前的实现支持两位用户进行虚拟视频通话，并显示各自的虚拟形象面部。

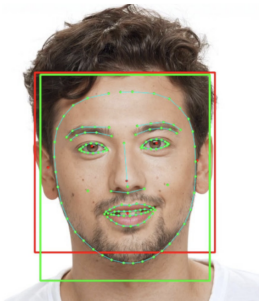


图 1：Face Detection Model（人脸检测模型）的输出示例，改编自 BlazeFace [5]。绿色线条表示人脸网格流水线中使用的输出矩形。

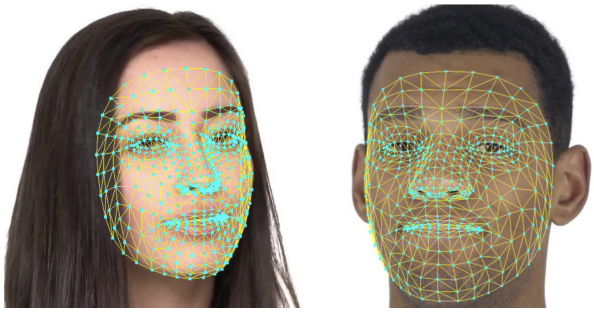


图 2：Face Landmark Model（人脸关键点模型）的输出示例，改编自 [19]。

总之，我们展示了 Meta 等公司如何通过门户网站提供入门级元宇宙体验，而无需消费者投资购买昂贵的硬件。通过提供这些易于使用的应用，我们的解决方案有望缩小消费者体验沉浸式元宇宙之间的差距。

2 人脸跟踪

本产品采用 Google MediaPipe 库中的 AI 模型，具体来说是 MediaPipe 人脸网格模型。该模型以图像为输入，并为用户面部的每一帧生成 468 个特征坐标作为输出 [12]。该模型由两个深度神经网络子模型组成：人脸检测模型和人脸关键点模型。人脸检测模型采用 BlazeFace 模型 [5] ，用于定位、裁剪视频中的人脸并预测其旋转角度。随后，将裁剪后且带有旋转角度的人脸变换为 128 或 256 像素见方的平直正向矩形，并输入人脸关键点模型以生成三维关键点向量 [19]。这些向量可用于计算预测的三维面部关键点坐标，并映射回裁剪后且带有旋转角度的人脸。人脸检测模型采用卷积神经网络架构 [5], ，而人脸关键点模型采用 ResNet，后者同样是一种卷积神经网络架构 [19]。两个模型都针对 GPU 加速进行了优化，以支持在个人设备上运行应用程序 [5,19]。图 1 和图 2 展示了这些模型的输出示例。系统从用户的网络摄像头实时捕获图像，并使用人脸网格模型进行处理以获取面部关键点，随后利用这些关键点进行渲染。

3 渲染

3.1 图形流水线

该平台使用 WebGL 进行图形渲染。WebGL 是 OpenGL 的 JavaScript 实现。OpenGL 是一种广泛使用的图形 API，可帮助开发者构建 GPU 加速渲染流水线。这些流水线由多个阶段组成，包括变换、顶点着色、片段着色和透视投影。变换阶段涉及一系列操作，这些操作由公式 1 至 5 表示。齐次坐标是一种用于三维计算的 4×4 矩阵表示，所有三维操作都使用齐次坐标来简化平移。

$$Homogeneous\ coordinates \qquad P_{init} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

$$Scale \qquad S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$Rotation \qquad R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$Translation \qquad T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$Transformation \qquad P_{after} = M_{trans} \times P_{init} \tag{5}$$

不过，透视投影并不需要齐次坐标，因此可以将投影矩阵简化为 3×3 矩阵，以降低计算开销。定义摄像头和投影屏幕后，即可使用公式 6 和 7 实现透视投影算法。这种方法能够更高效地渲染三维场景。

$$\begin{bmatrix} f_x \\ f_y \\ f_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{e_x}{e_z} \\ 0 & 1 & \frac{e_y}{e_z} \\ 0 & 0 & \frac{1}{e_z} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & sin\theta_x \\ 0 & -sin\theta_x & cos\theta_x \end{bmatrix} \begin{bmatrix} cos\theta_y & 0 & -sin\theta_y \\ 0 & 1 & 0 \\ sin\theta_y & 0 & cos\theta_y \end{bmatrix} \begin{bmatrix} cos\theta_z & sin\theta_z & 0 \\ -sin\theta_z & cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \right) \tag{6}$$

$$\begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} f_x / f_w \\ f_y / f_w \end{bmatrix} \tag{7}$$

p： $The\ coordinates\ of\ a\ vertex$
c: $The\ coordinates\ of\ the\ camera$
e: $The\ display\ surface\ of\ the\ projection\ screen$
f: $Homogenous\ coordinates\ of\ 2D\ point$
b: $2D\ display\ coordinates$

3.2 顶点与校准

人脸网格的顶点通过将特征点分组成三角形来计算，如图3所示。顶点通过顺时针旋转和逆时针旋转生成，从而使网格的两侧都被填充。因此，即使将网格旋转180度，它也不会呈现透明状态。

为确保用户的面部与三维渲染的虚拟形象面部完成校准，需要进行空间平移和旋转。系统假定用户在初始帧中面部居中且保持直立，并变换虚拟形象的面部，使初始帧中的鼻子居中，同时使由鼻子和两只睫毛构成的三角形位于图形空间中的一个平面上，且其法线朝向屏幕外侧。校准条件由公式8、9、10和11表示。用于构建人脸网格的面部关键点如图3所示。

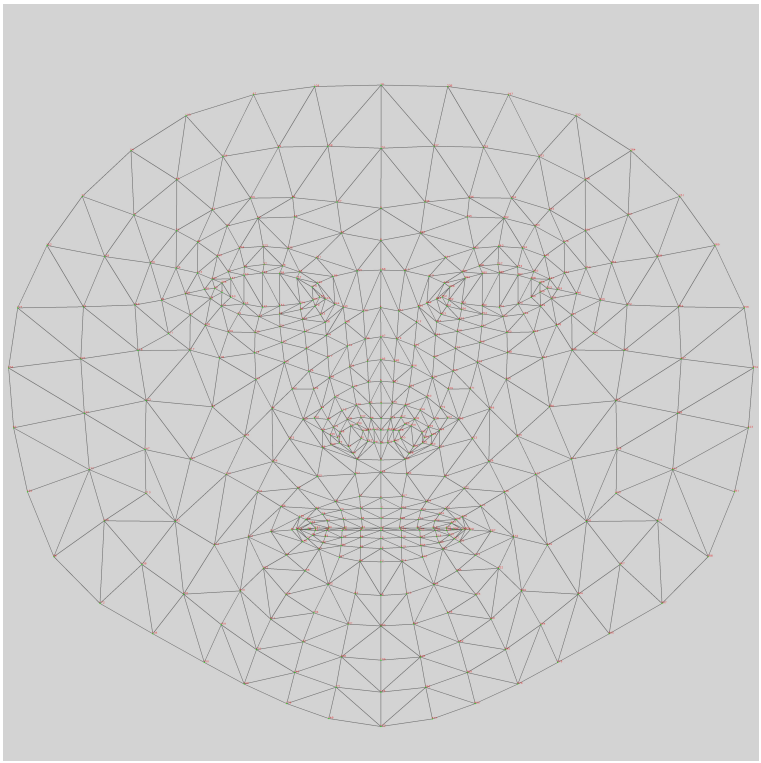


图3：带有特征点编号标签的人脸网格，改编自 [27]。

$$landmark_5 \Leftarrow \begin{bmatrix}0\\0\\0\end{bmatrix} \tag{8}$$

$$\vec{up} = (\frac{landmark_{52} - landmark_{282}}{2}) - landmark_1 \tag{9}$$

$$\vec{right} = landmark_{282} - landmark_{52} \tag{10}$$

$$\hat{up} \times \hat{right} \Leftarrow \begin{bmatrix}0\\0\\1\end{bmatrix} = \hat{z} \tag{11}$$

$$\begin{aligned} landmark_5 &: center\ nose \\ landmark_1 &: lower\ nose \\ landmark_{52} &: left\ eyelash \\ landmark_{282} &: right\ eyelash \end{aligned}$$

为满足公式 11 给出的条件，校准算法会测量向上和向右向量的偏角，并使用四元数角将人脸网格按这些偏角旋转。校准算法通过执行公式 12、13 和 14，确保满足方向条件。

$$Q_{up} = QuaternionAngle(\hat{up}, \hat{y}) \tag{12}$$

$$Q_{right} = QuaternionAngle(\hat{right}, \hat{x}) \tag{13}$$

$$Q_{FaceMeshRotation} = Q_{up} \times Q_{right} \tag{14}$$

3.3 四元数

四元数是一种四维系统，使用 $\vec{x}$、 $\vec{y}$、 $\vec{z}$和 $\vec{w}$ 表示旋转。另一种表示旋转的方法是使用 ϕ、 θ和 ψ 欧拉角，并采用 XYZ 等旋转顺序来实现 $roll$、 yaw和$pitch$。使用四元数系统是因为它可以避免万向节锁定。XYZ 旋转顺序下欧拉旋转的矩阵表示，以及由欧拉角推导四元数的方法，见公式 15 和 16，改编自 [17]。

$$\begin{aligned}\begin{bmatrix} x_{after} \\ y_{after} \\ z_{after} \end{bmatrix} &= R_z(\psi)R_y(\theta)R_x(\phi)\begin{bmatrix} x_{init} \\ y_{init} \\ z_{init} \end{bmatrix} \\ &= \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}\begin{bmatrix} x_{init} \\ y_{init} \\ z_{init} \end{bmatrix}\end{aligned} \tag{15}$$

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ Q_w \end{bmatrix} = \begin{bmatrix} sin\frac{\phi}{2}cos\frac{\theta}{2}cos\frac{\psi}{2} - cos\frac{\phi}{2}sin\frac{\theta}{2}sin\frac{\psi}{2} \\ cos\frac{\phi}{2}sin\frac{\theta}{2}cos\frac{\psi}{2} + sin\frac{\phi}{2}cos\frac{\theta}{2}sin\frac{\psi}{2} \\ cos\frac{\phi}{2}cos\frac{\theta}{2}sin\frac{\psi}{2} - sin\frac{\phi}{2}sin\frac{\theta}{2}cos\frac{\psi}{2} \\ cos\frac{\phi}{2}cos\frac{\theta}{2}cos\frac{\psi}{2} + sin\frac{\phi}{2}sin\frac{\theta}{2}sin\frac{\psi}{2} \end{bmatrix} \tag{16}$$

4 流

4.1 基础设施

该平台使用 AWS（亚马逊云科技）进行部署，所述架构是 AWS 建议的最佳实践集群部署 [1]。应用程序资源被分组在一个 VPC（虚拟私有云）中，该 VPC 由公有子网和私有子网组成。ALB（应用程序负载均衡器）部署在公有子网中，用于接收来自公共网络的流量并将其分发到集群。ECS（弹性容器服务）集群和 Aurora 数据库部署在私有子网中，因此二者都无法从公共网络直接访问。冗余子网可确保应用程序在可用区故障时仍能正常运行。

该系统采用分布式架构并支持水平扩展，通过监控多个实例的 CPU 和内存使用量来实现弹性和自动扩缩容。Terraform 是一种基础设施即代码软件工具 [16], ，用于管理部署流程，从而实现稳健的基础设施管理。该架构用途广泛，可支持多种应用，包括 Discord 等热门服务或在线游戏，这些应用需要同时处理数百万用户。架构图见图 4。

4.2 协议

在本系统中，使用 WebSocket（一种基于 TCP 的数据流协议）建立并维持前端与后端之间的连接。该流协议对于帮助前端客户端相互发现并协商点对点协议至关重要。由于协商需要双方反复交换数据，直到达成一致，因此必须采用流传输。由于系统采用分布式架构，具有相同唯一房间 ID 的客户端不一定会连接到同一个集群实例。如果客户端在不同的分布式集群实例上打开 WebSocket，实例会在数据库中查询私有 IP，并在实例之间建立代理连接，以便交换数据。

WebRTC 是一种 SRTP协议，可直接在客户端之间传输数据 [10], ，用于避免后端处理用户语音、人脸网格等敏感数据。前端通过后端协商 WebRTC 的提议、应答和 ICE 候选项；使用 WebRTC 建立点对点连接后，客户端便开始向另一客户端发送音频和人脸网格数据，包括网格平移和旋转。语音通过媒体轨道发送，人脸网格则通过数据通道发送。

4.3 压缩

当网速较慢或客户端之间距离较远时，数据通道可能会出现网络拥塞。解决方法是将关键点数据压缩为 16 位浮点数，即半精度浮点数。每帧的有效载荷大小压缩至 2838 字节，最大帧率设为 30，因此数据速率约为 85 KB/s。数据速率

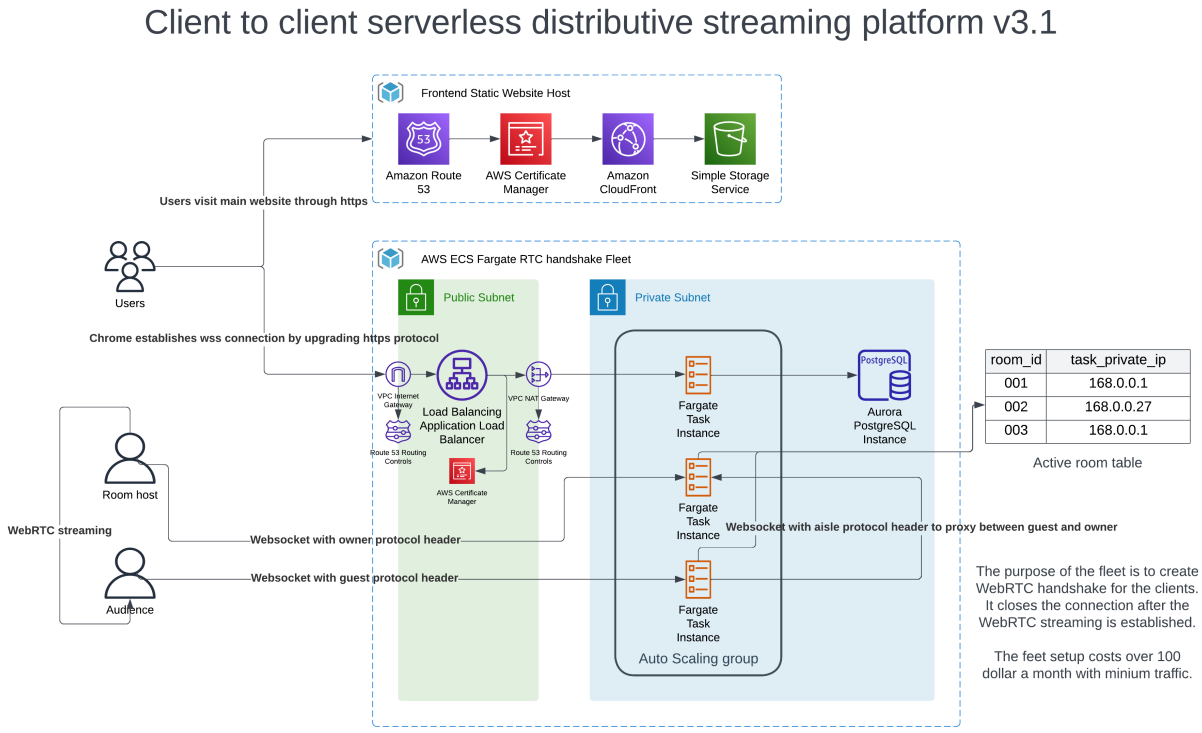


图4：虚拟化身流架构图

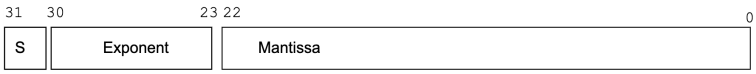


图5：IEEE单精度浮点数（32位浮点数）标准，改编自 [2]。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	E					T									

图6：IEEE E 半精度浮点数（16 位浮点数）标准改编自 [3]。这里的“T”指尾数。

对于典型的 YouTube 视频，分辨率为 1080p、帧率为 30帧/秒，采用 H.264 压缩，压缩比为 2000:1 [26]，时，数据速率为 93 KB/s，这表明 85 KB/s 的平台数据速率是可行的。H.264 视频数据速率的推导见公式17。

$$1080 \times 1920 \times 3\frac{bytes}{pixel} \times 30fps \times \frac{1}{2000} = 93KB/s \tag{17}$$

图5和图6分别展示了 IEEE 标准中的 32位浮点数和 16 位浮点数。将 32位浮点数转换为 16 位浮点数时，符号位保持不变，尾数被截断，并将 32位浮点数最左侧的 10 位存入 16 位浮点数的尾数中。同时，指数按公式18进行算术运算，但必须考虑溢出和下溢等特殊情况。

$$E_{float16} = E_{float32} - 0x7F + 0xF = E_{float32} - 127 + 15 \tag{18}$$

5 项结果

该项目成功展示了如何利用大多数客户已有的传统硬件，在 Web 上提供 VR头显元宇宙体验的潜力。它结合使用了开源库和 Web

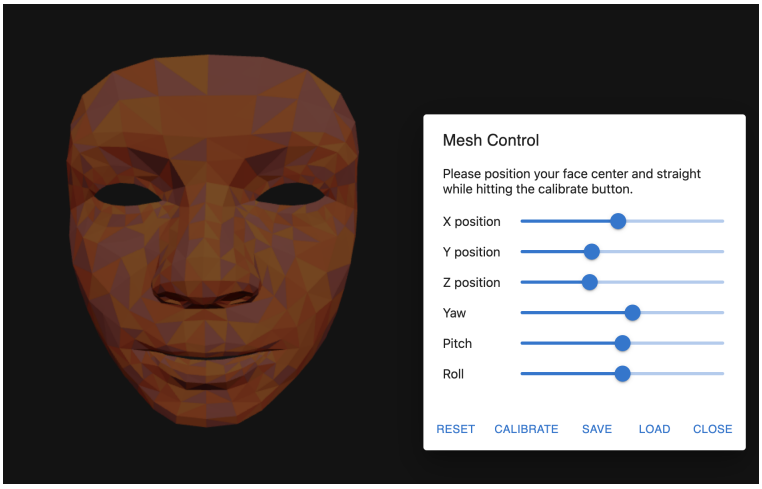


图 7：用于校准的虚拟化身流式网格控制面板

技术，包括 Google MediaPipe、ThreeJS、WebSocket、WebRTC 和 AWS。最终产品可在网站 virtualavatar.trip1elift.com 上体验。源代码托管于代码仓库 github.com/Trip1eLift/virtual-avatar，用户指南见 github.com/Trip1eLift/virtual-avatar/tree/main/userguide。
首次访问时，系统会提示用户授权浏览器访问摄像头和麦克风。摄像头用于采集用户的面部图像，再由 MediaPipe 的 3D人脸追踪AI模型处理，以获取面部关键点。随后，使用 ThreeJS 库将这些特征点渲染为带有随机颜色的 2D图形。用户与其他用户进行房间聊天时，麦克风会用于采集音频数据。网站会将用户的鼻子和睫毛与空间中的一个平面进行对齐，该平面的法向量指向屏幕外。用户可以手动修改、保存和加载校准设置。图 7 展示了校准界面。https://www.overleaf.com/project/6408de81f53a680b6c52b908

配置好校准设置后，用户即可开始流式传输。系统支持两类用户：房主和访客。房主可以点击流媒体控制面板中的“创建房间”按钮来创建房间。流媒体集群会生成一个唯一的房间 ID，房主必须通过外部渠道将其分享给访客。由于网站没有用户账户系统，因此不支持好友通话。房主会与后端保持 WebSocket 连接，并等待访客加入。缩容期间，房主和访客可能会被分配到同一个分布式集群实例，使前端能够在该实例上协商 WebRTC 协议。流量扩缩容期间，房主和访客可能会被分配到不同的分布式集群实例。访客所在的实例会使用房间 ID，从数据库中获取房主所在实例的私有 IP；访客实例再将 WebSocket数据代理到房主所在的实例。双方通过代理协商 WebRTC 协议。

前端之间建立 WebRTC连接后，双方都会断开与后端集群的 WebSocket连接。随后开始传输网格数据流和音频数据流。网格数据经过压缩，传输速率为 43 KB/s，以防止网络拥塞。图 8 展示了流媒体界面。该产品展示了如何在无需 VR头显的情况下，将使用 VR头显的沉浸式元宇宙体验带到网页端。

该产品的前端使用 ReactJS [28] 开发，并辅以各种 API文档资源 [21, 11, 14, 23, 25, 10]。此外，前端还使用了多个 npm软件包 [29, 13, 8, 24,20, 30]。后端使用 Go语言 [4] 构建，并采用开源软件包 [7, 9]。系统使用 Docker [18] 进行容器化，并借助 Terraform [16]部署到 AWS [1] 。该产品使用 Postgres [15] 作为数据库系统。

6 讨论

当前产品表明，借助传统硬件，可以在基于网络的平台上替代 VR 头显的面部动作和表情。不过，该产品的潜力远不止于目前所提供的功能。市面上许多 VR 头显都配有 VR控制器，允许用户在虚拟世界中控制手部动作。虽然 Google MediaPipe 支持上半身、脊柱、手臂和手指追踪，但该

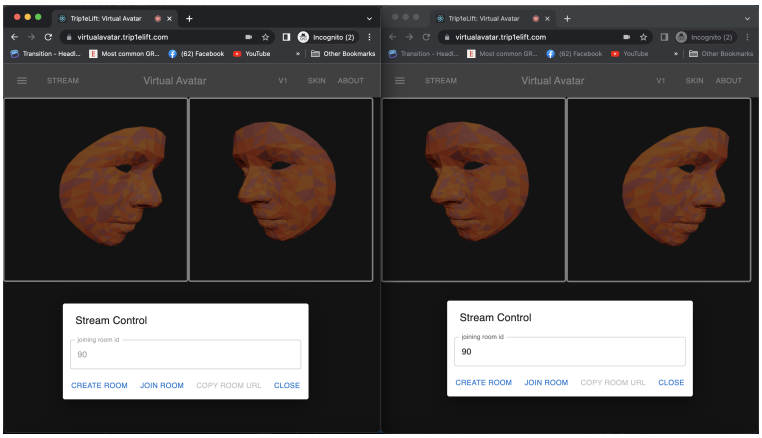


图8：两位用户通过虚拟化身流在90号房间进行虚拟视频通话

平台受其基础图形引擎的限制，不支持这些身体部位。要呈现手臂和手指动作的细节，需要更先进的物理引擎，以避免网格裁剪或其他图形故障。

另一个可以改进的方面是面部网格渲染。元宇宙允许用户变成自己想要的任何形象，例如长着猫脸或狗脸的类人形体。实现此功能的挑战在于图形引擎。需要更先进的物理/图形引擎来支持带权重的顶点映射，将人脸网格映射到狗脸上，同时插值处理面部表情。

此外，由于输出接口不同，该平台与运行在 VR 头显上的元宇宙在用户视觉感知方面存在差异。在 VR 头显中，两个镜片会产生视差，以模拟三维可视化效果，因此用户可以看到三维内容。本项目使用传统电脑屏幕作为输出接口；尽管数据是在三维空间中采集的，但为了向用户展示，三维数据会被投射到二维空间中。使用红蓝立体图像渲染即可解决这一限制，任何彩色二维显示器都能输出这种图像。纸板制红蓝立体眼镜的成本极低，因此只需升级图形引擎并付出少量成本，用户便能体验只有在 VR 头显上才能看到的三维视觉效果。

当前平台的另一个局限是仅支持一对一视频聊天，且不提供三维空间音效，而这却是大多数元宇宙平台的标准功能。不过，未来可以扩展本产品，以支持多人会议室。实现三维空间音效的方法包括调整左右音轨的声音偏移，并根据平方反比定律所描述的能量衰减来降低声音强度。

7 结论

在本项目中，我们展示了利用普通摄像头和麦克风打造沉浸式元宇宙体验的潜力。我们构建了一个可扩展的客户端到客户端流媒体后端，可支持数百万用户，并可扩展到其他应用。本产品利用 Google MediaPipe 的三维面部跟踪模型，将网络摄像头数据转换为面部网格关键点，再通过 ThreeJS 将其渲染为三维计算机图形。我们使用 WebSocket 和 WebRTC 协议，在两位用户之间直接传输人脸网格和音频数据，以实现虚拟化身视频通话。我们的压缩技术可防止网络拥塞，并将流媒体质量维持在可接受的水平。

虽然我们的产品目前提供的体验不如VR头显那样身临其境，但它为初学者提供了入门级体验，帮助他们理解元宇宙的概念。随着进一步开发，产品有望得到提升，提供几乎可媲美VR头显的体验。我们相信，我们的平台可以成为元宇宙领域未来创新的基石。

参考文献

[1] 亚马逊。简介 介绍 - 亚马逊弹性容器服务。2023年 网址： https://docs.aws.amazon.com/AmazonECS/latest/bestpracticesguide/intro.html（访问于 03/02/2022）。

[2] Arm。6.1. 浮点数基础与 IEEE-754 标准。2023。网址：https://developer.arm.com/ documentation / den0042 / a / Floating - Point / Floating - point - basics - and - the - IEEE - 754 - standard（访问于 2022年3月8日）。[3] Arm。半精度浮点数格式。2023。网址：https : / / developer . arm . com / documentation/100067/0607/Other-Compiler-specific-Features/Half-precision-floating-point-number-format（访问于 2022年3月8日）。[4] Go 作者团队。文档。2009年。网址：https://go.dev/doc/（访问于 03/02/2022）。[5] Valentin Bazarevsky 等。“BlazeFace：在移动 GPU 上实现亚毫秒级神经网络人脸检测”。载于：arXiv 预印本 arXiv:1907.*05047 [*cs.CV*]*（2019年7月）。[6] Meghan Bobrowsky。“Meta Quest Pro VR头显降价，应对 TikTok 母公司竞争”。载于：《华尔街日报》（2023年3月）。DOI：https://www.wsj.com/articles/meta-slashes-price-for-quest-pro-vr-headset-months-after-launch-d61f8e30。[7] Gary Burd 等。Gorilla WebSocket。2022年。网址：https://github.com/gorilla/websocket（访问于 03/02/2022）。[8] Ricardo Cabello 和 Michael Herzog。three.js。2023年。网址：https://www.npmjs.com/package/three（访问于 03/02/2022）。[9] Jack Christensen。pgx - PostgreSQL 驱动程序和工具包。2023年。网址：https://github.com/jackc/pgx（访问于 03/02/2022）。[10] Google。WebRTC 入门。2019年5月。网址：https://webrtc.org/getting-started/overview（访问于 03/02/2022）。[11] Google。Google/mediapipe：面向实时媒体和流媒体的跨平台、可定制机器学习解决方案。2019年。网址：https://github.com/google/mediapipe（访问于 03/02/2022）。[12] Google。MediaPipe Face Mesh。2023年。网址：https://google.github.io/mediapipe/solutions/face_mesh.html（访问于 03/02/2022）。[13] Google。MediaPipe JavaScript 解决方案。2023年。网址：https://www.npmjs.com/package/@mediapipe/ pose（访问于 03/02/2022）。[14] Khronos Group。WebGL：面向 Web 的 2D 和 3D 图形 - Web API：MDN。2011年。网址：https://developer. mozilla.org/en-US/docs/Web/API/WebGL_API（访问于 03/02/2022）。[15] PostgreSQL 全球开发组。PostgreSQL 14.7 文档。2023年2月。网址：https://www. postgresql.org/docs/14/index.html（访问于 03/02/2022）。[16] HashiCorp。AWS Provider。2023年。网址：https://registry.terraform.io/providers/hashicorp/aws/ latest/docs（访问于 03/02/2022）。[17] D. M. Henderson。“航天飞机计划：欧拉角、四元数与变换矩阵的关系”。载于：NASA 技术报告服务器（1977年）。DOI：https : / / ntrs . nasa . gov / citations / 19770024290。[18] Docker 公司。Docker 概览。2023年3月。网址：https://docs.docker.com/get-started/overview/（访问于 03/02/2022）。[19] Yury Kartynnik 等。“基于单目视频的移动 GPU 实时面部表面几何重建”。载于：arXiv 预印本 arXiv:1907.*06724 [*cs.CV*]*（2019年7月）。[20] Moz Morris。React Webcam。2022年。网址：https://www.npmjs.com/package/react-webcam（访问于 03/02/2022）。[21] MUI。概览 - Material UI。2023年。网址：https://mui.com/material-ui/getting-started/overview/（访问于 03/02/2022）。[22] Stylianos Mystakidis。“元宇宙”。载于：《百科全书》2.1（2022年），第486–497页。ISSN：2673-8392。网址：https: // www.mdpi.com/2673-8392/2/1/31。[23] Three.js 组织。创建场景。网址：https : / / threejs . org / docs / index . html # manual / en / introduction/Creating-a-scene（访问于 03/02/2022）。[24] Nikhil Saraf 等。@react-three/fiber。2023年。网址：https://www.npmjs.com/package/@react- three/ fiber（访问于 03/02/2022）。[25] 开源。简介。2023年。网址：https : / / docs . pmnd . rs / react - three - fiber / getting - started/introduction（访问于 03/02/2022）。[26] RGB Spectrum。H.264 配置文件。2023年。网址：https://www.rgb.com/h264- profiles#:~:text=High% 20Profile-,H.,ratio%20of%20about%202000%3A1.（访问于 2022年3月10日）。[27] Google MediaPipe 团队。canonical-face-model-uv-visualization.png。2020年。网址：https://github.com/ google/mediapipe/blob/master/mediapipe/modules/face_geometry/data/canonical_face_model.fbx（访问于 2022年3月7日）。

[28] Jordan Walke。《*React* —— 用于构建用户界面的 *JavaScript* 库》。网址：https://reactjs.org/（访问日期：03/02/2022）。[29] Albert Yu 等。《*@mui/material*》。2023年。网址：https://www.npmjs.com/package/@mui/material（访问日期：03/02/2022）。[30] Matt Zabriskie 和 Samuel Reed。《*React-Draggable*》。2022年。网址：https://www.npmjs.com/package/react-draggable（访问日期：03/02/2022）。