Real-Time Head Pose Synchronization for Multiplayer Virtual Presence

Ryan Zmuda — University of Dayton — zmudar1@udayton.edu
Ju Shen — University of Dayton — jshen1@udayton.edu
Vijayan Asari — University of Dayton — vasari1@udayton.edu

University of Dayton

Executive Summary

This research focuses on integrating Gaussian Splatting, AI-based head pose estimation, and multiplayer networking to create an immersive and scalable multi-user racing simulation. In this environment, each player's real-world head pose is captured via webcam and synchronized across the network, enhancing immersion and interactivity between participants. We successfully implemented a complete end-to-end system that satisfies real-time constraints, achieving low latency and stable pose estimation accuracy. This project demonstrates the effectiveness of combining pose-driven interaction with Gaussian Splatting for simulation and game environments, highlighting its potential for future research and real-world applications.

What is Gaussian Splatting?

Gaussian Splatting is a modern 3D rendering technique that represents objects as a large set of tiny, semi-transparent 3D Gaussians (ellipsoids), rather than explicit polygonal meshes or textured surfaces. Each Gaussian is parameterized by a mean position and covariance, forming a continuous volumetric representation:

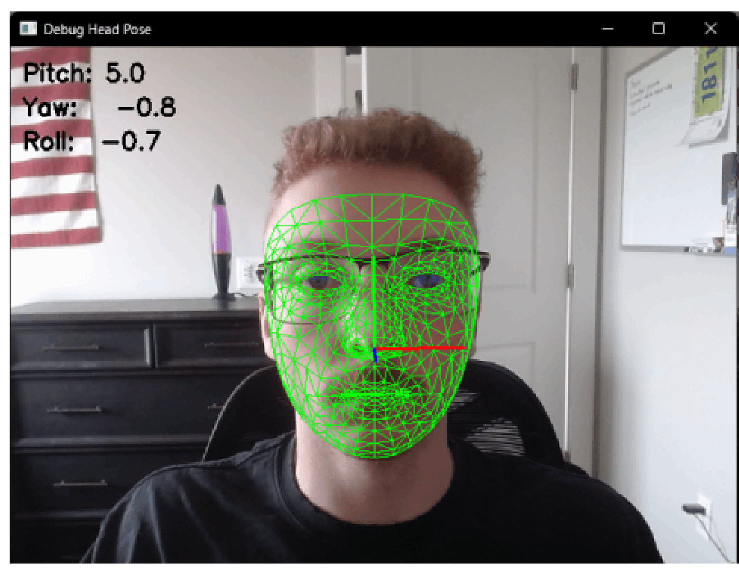
$$G(\mathbf{x}) = \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

where μ defines the spatial center and Σ encodes scale, orientation, and anisotropy.

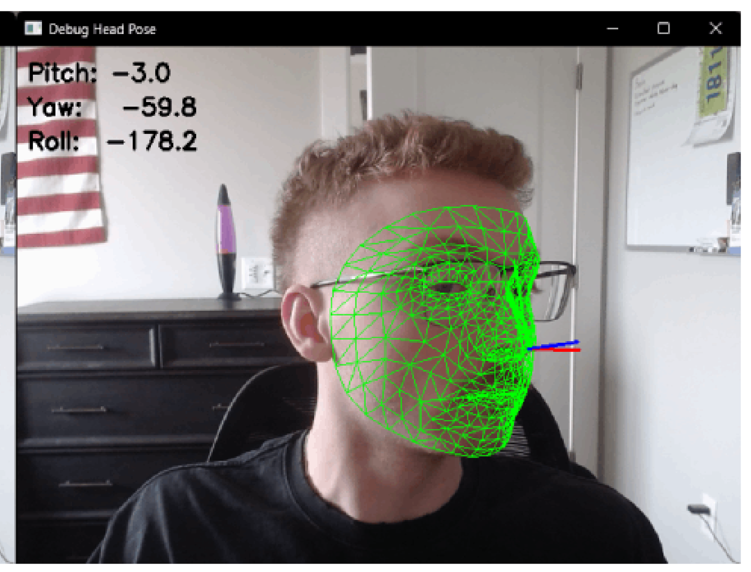
- Photorealistic graphics
- Extremely fast on consumer hardware

A Gaussian Splat of Ryan's head.

Head-Angle Pose Detection



Player looking straight ahead.



Player looking to his left.

We utilize OpenCV and MediaPipe to leverage existing AI facial feature extraction and pose detection.

> Head pose is estimated using **nose tip**, **chin**, **left eye**, **right eye**, and **mouth corners**.

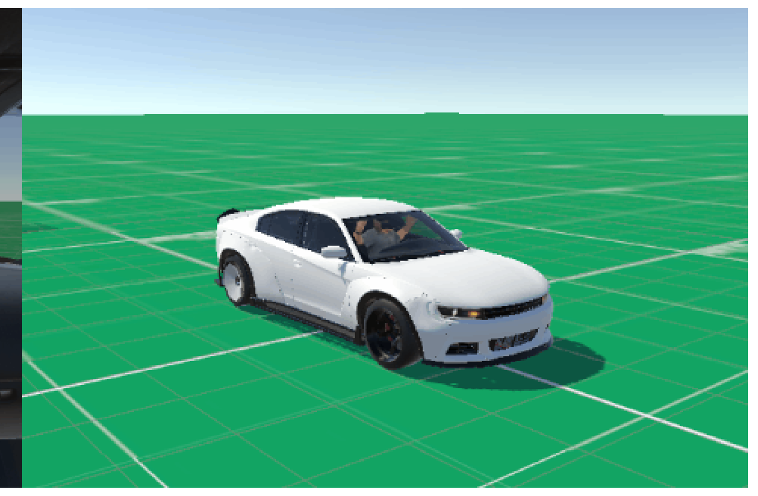
We approximate camera intrinsics using image dimensions and assume lens distortion to be negligible. The PnP (Perspective-n-Point) algorithm is used to estimate the rotation and translation vectors based on the 2D landmarks. The rotation vector is converted to a rotation matrix, and then Euler angles (pitch, yaw, roll) are extracted using **cv2.decomposeProjectionMatrix**.

Interfacing with the simulation

Estimated head pose values are pushed over the network via UDP towards localhost, for later consumption by the Unity engine (and Editor).

Multiplayer Racing Game

Player's first-person view.



External view of the player's car.

In our simulation, players can drive race cars around in first person together. In order to build the simulation, we used existing open-source Gaussian Splatting implementations and a popular Unity networking solution, paired with our custom-built racing game.

Integration

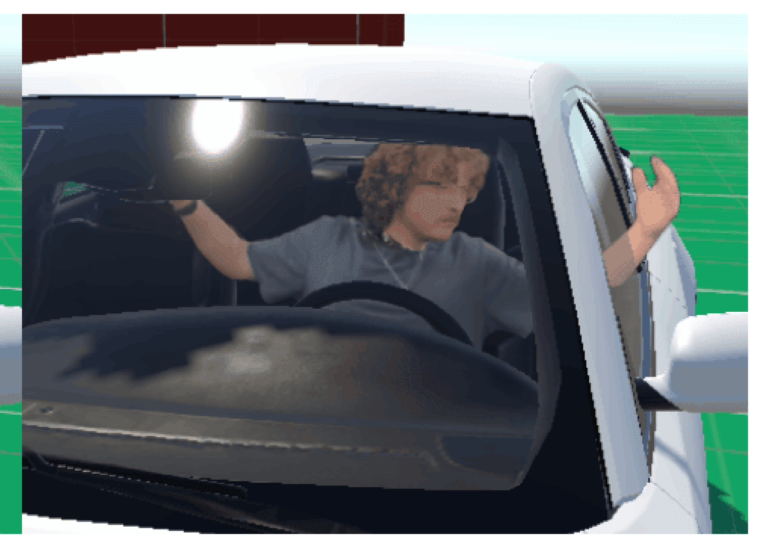
1. Imported open-source Unity Gaussian Splatting by Aras Pranckevičius into Unity 6
2. Scripted the first-person racing simulation car controls
3. Integrated Photon Fusion networking for cars

Many modifications were made to the game source code and game objects to accommodate network synchronization, but yielded a final result where two players could drive around in a synchronized multiplayer environment, without head-angle pose reflection.

Network Pose Mirroring

A listener module (implemented in **C#**) is embedded within the Unity client to continuously receive streamed head-pose data from the local webcam interface via a socket connection. The received pose is parsed in real time and used to update the corresponding player avatar's head transform, aligning the virtual orientation with the user's estimated physical head angle. This updated transform is then propagated across the network using Photon Fusion's built-in state synchronization, enabling consistent replication of head orientation across all connected clients.

Remote player looking straight ahead.

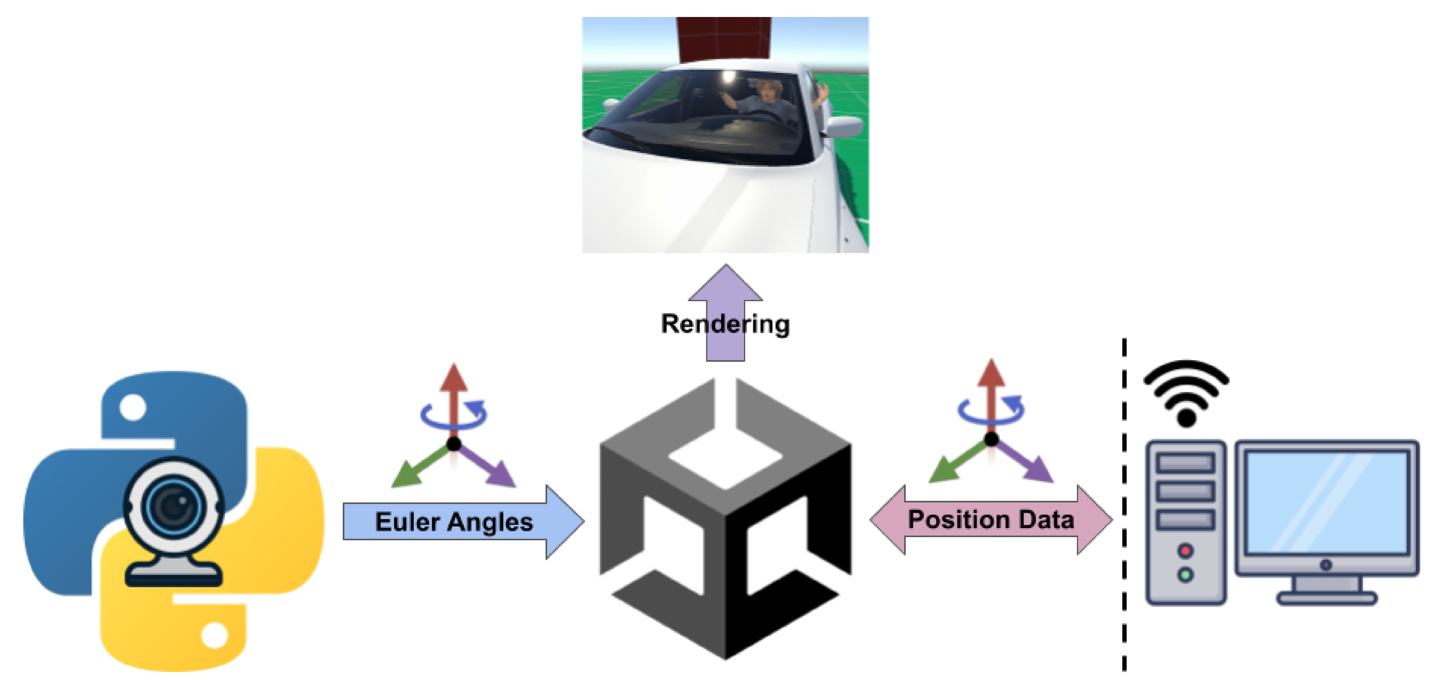


Remote player looking to his left.

By synchronizing the player's physical head pose with their corresponding in-game avatar ("digital twin"), the system enhances embodiment, improving perceived presence and immersion. This tight coupling between real-world motion and virtual representation enables more natural interaction dynamics and a more coherent user experience. This approach has broader applicability beyond gaming, including interactive simulation and training environments where realistic embodiment and responsive avatar behavior are critical.

Full System Architecture

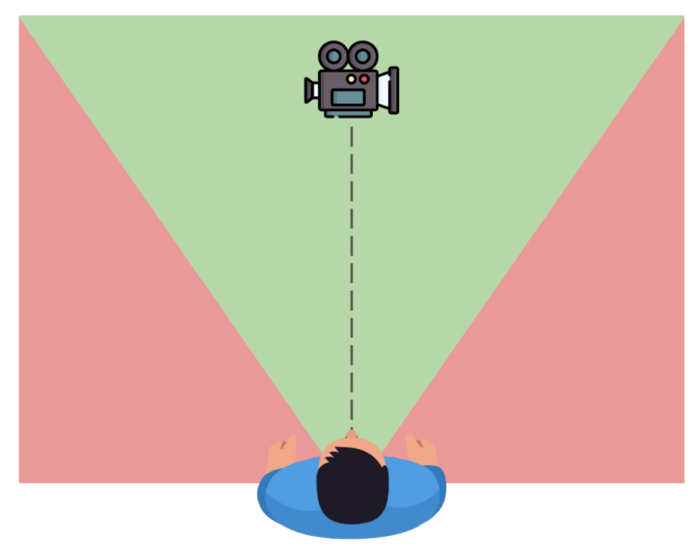
Each client runs both the Unity game application and a local webcam-based pose estimation module. The pose estimation module transmits the user's estimated Euler angles to the Unity client via UDP for real-time integration. The Unity client then updates the local avatar's head orientation accordingly. The client further synchronizes both head pose and vehicle state across the network, broadcasting these values to all connected peers while simultaneously receiving corresponding updates from remote clients. These incoming states are used to update remote avatars, enabling consistent real-time replication of both motion and interaction within the shared environment.



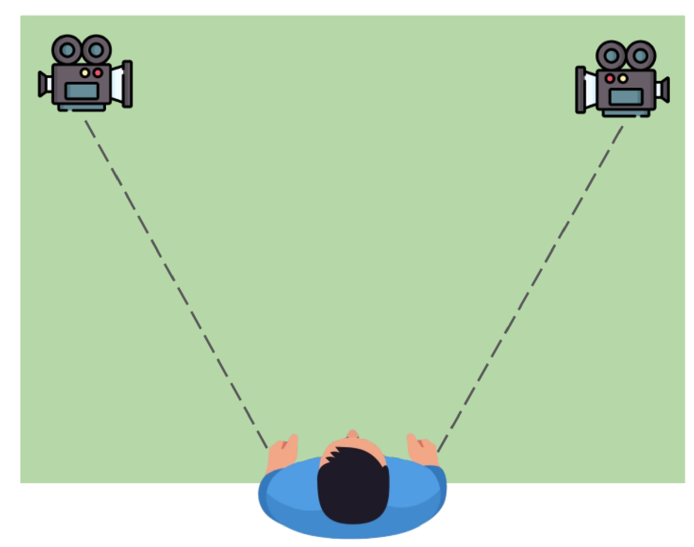
The game client reads rotation data from the local webcam server, and mirrors this to both the local and remote player.

Future Work

A limitation of single webcam-based pose detection is that beyond certain angles, the accuracy of the pose estimation greatly degrades. When approaching 180° angles, the current implementation of the head pose analyzer fails to return a coherent pose consistently, introducing a lot of noise and jitter into the final positions. To address this, we recommend that future research include temporal considerations in the pose detection to correct for jitter and explore multi-camera setups, as having additional perspectives could aid the accuracy and working range of the head-angle pose detection.



Current (limited) range of detection (approx).



Theoretical valid range of detection with multiple cameras.

Acknowledgement

This research was supported by the Science and Engineering Catalyst Center Summer Fellowship Program.