Software and Hardware Architecture Proposal for "Using Augmented Reality to Experience Prosopagnosia"

Introduction

This proposal outlines a comprehensive software and hardware architecture to implement an augmented reality (AR) system that simulates the subjective experience of prosopagnosia (face blindness). The system aims to manipulate live video feeds by altering facial features in real-time, based on neural models of face recognition, specifically those in the fusiform face area (FFA). The primary goal is to create an immersive experience that accurately reflects the perceptual challenges faced by individuals with prosopagnosia.

Overview of the Proposed Architecture

The system comprises both on-device and server-side components to achieve real-time performance with minimal latency. It utilizes advanced computer vision and machine learning algorithms to detect, model, and alter faces within a live video stream. The architecture is designed to be scalable, modular, and optimized for real-time processing, ensuring an immersive and seamless user experience.

1 Software Architecture

1.1 Software Components Overview

- **Face Detection and Tracking Module**
- **Face Representation and Modeling Module**
- **Face Distortion Module**
- **Rendering Module**
- **User Interface Module**
- **Networking Module (if offloading processing)**
- **Latency Optimization Module**

1.2 Data Flow and Processing Pipeline

1. **Video Capture**: The AR headset's camera captures live video feeds.
2. **Face Detection and Tracking**: Real-time detection of frontal faces and tracking their positions in successive frames.
3. **Face Representation and Modeling**: Generation of 3D face models and extraction of a 50-dimensional eigenvalue representation based on FFA neural models.
4. **Face Distortion**: Alteration of facial features by modifying the eigenvalues to simulate prosopagnosia.
5. **Rendering**: Overlay of the distorted faces onto the original video feed in real-time.

6. **Display**: Presentation of the augmented video feed to the user through the AR headset.
7. **User Interaction and Feedback**: Optional modules for user input and experience surveys.

1.3 Detailed Component Description

1.3.1 Face Detection and Tracking Module

Function: Detects faces in real-time from the video feed and tracks them across frames.
Technologies:

- **OpenCV**: For initial face detection using Haar Cascades or DNN-based detectors.
- **Dlib**: For facial landmark detection.
- **TensorFlow Lite** or **ONNX Runtime**: For optimized neural network inference on mobile devices.

Algorithms:

- **Multi-task Cascaded Convolutional Networks (MTCNN)** for face detection.
- **Kalman Filters** or **Optical Flow** for tracking.

1.3.2 Face Representation and Modeling Module

Function: Generates 3D face models and computes the eigenvalue representation of facial features. [**chang˙2017**]
Technologies:

- **Eigenfaces Method**: For face representation using Principal Component Analysis (PCA).
- **3D Morphable Models (3DMM)**: To create and manipulate 3D face geometry.

Algorithms:

- **PCA**: To extract the 50-dimensional eigenvalue representation.
- **Non-linear optimization**: For fitting the 3D model to the detected face.

1.3.3 Face Distortion Module

Function: Alters facial features by modifying the eigenvalues to simulate the perceptual distortions experienced in prosopagnosia.
Technologies:

- Custom algorithms based on FFA neural encoding studies.

Algorithms:

- **Eigenvalue Manipulation**: Adjusting eigenvalues to distort facial features while maintaining realism.
- **Neural Style Transfer**: As an alternative to apply perceptual changes.

1.3.4 Rendering Module

Function: Integrates the distorted faces back into the video feed and renders the augmented reality scene.
Technologies:

- **Unity3D** or **Unreal Engine**: For AR rendering.
- **ARCore** or **ARKit**: Depending on the AR headset compatibility.

Algorithms:

- **Shader Programming**: For real-time rendering effects.
- **Texture Mapping**: To overlay the altered face onto the original.

1.3.5 User Interface Module

Function: Manages user interactions and displays contextual information.
Technologies:

- Cross-platform UI frameworks such as **Qt** or **Unity UI**.

Features:

- Menu systems for settings.
- Visual cues or instructions for participants.

1.3.6 Networking Module (If Offloading Processing)

Function: Handles data transmission between the AR headset and the local server.
Technologies:

- **WebSockets** or **gRPC**: For real-time communication.
- **UDP Protocol**: For low-latency data transfer.

Security:

- **Encryption**: SSL/TLS protocols to secure data transmission.

1.3.7 g. Latency Optimization Module

Function: Ensures that the total processing time remains within the 20 ms target.
Technologies:

- **Multi-threading**: To parallelize tasks.
- **GPU Acceleration**: Utilizing shaders and compute shaders.

Strategies:

- **Algorithm Optimization**: Simplifying models without significant loss of quality.
- **Asynchronous Processing**: Decoupling processing steps to prevent bottlenecks.

1.4 Latency Reduction Techniques

- **On-Device Processing**: Utilize the AR headset's GPU for parallel computations.
- **Model Compression**: Use quantization and pruning on neural networks.
- **Efficient Algorithms**: Implement lightweight versions of face detection and recognition algorithms.
- **Edge Computing**: Deploy edge servers close to the user to reduce network latency if offloading is necessary.

2 Hardware Architecture

2.1 AR Headset Specifications

Device Requirements:

- High-resolution camera(s) for video capture.
- Powerful GPU for real-time rendering and processing.

- Sufficient RAM and storage.

Suggested Devices:

- **Microsoft HoloLens 2**
- **Magic Leap 2**
- **Meta Quest Pro**

2.2 Processing Capabilities

2.2.1 On-Device Processing

Advantages:

- Reduced latency due to elimination of network delays.
- Increased privacy as data is processed locally.

Limitations:

- Limited computational resources compared to servers.
- Potential thermal constraints.

2.2.2 Offloading to Local Server

Server Specifications:

- **GPU**: NVIDIA RTX series or equivalent for deep learning tasks.
- **TPU**: Tensor Processing Units if compatible.
- **CPU**: Multi-core processors for handling multiple threads.

Network Architecture:

- High-speed local network (Ethernet or Wi-Fi 6).
- Dedicated router to prioritize traffic.

2.3 Networking Considerations

Latency:

- Target end-to-end latency of less than 20 ms.
- Use of **Quality of Service (QoS)** settings to prioritize AR data.

Bandwidth:

- Sufficient to handle high-resolution video streams.

Reliability:

- Redundant network paths if possible.

Security:

- Secure Wi-Fi networks with strong encryption.

2.4 Integration of Hardware Components

Camera Systems:

- Integrated with the AR headset for seamless video capture.

Sensors:

- **IMUs** (Inertial Measurement Units) for motion tracking.
- **Depth Sensors**: Optional, for improved 3D modeling.

Wearable Computing:

- Lightweight and ergonomically designed for comfort during extended use.

3 Implementation Considerations

3.1 Scalability and Flexibility

- **Modular Design**: Allows for components to be updated or replaced without overhauling the entire system.
- **Cross-Platform Compatibility**: Development using platforms that support multiple devices (e.g., Unity3D).
- **Future-Proofing**: Designing with future technological advancements in mind.

3.2 Real-Time Processing Challenges

- **Data Throughput**: Ensuring the system can handle the data rates required for high-resolution video.
- **Synchronization**: Keeping all processing modules in sync to prevent visual artifacts.
- **Error Handling**: Robust methods to handle failures without disrupting the user experience.

3.3 Privacy and Ethical Considerations

- **Data Management**: Ensuring that facial data is not stored unnecessarily.
- **User Consent**: Informing participants about the data processing involved.
- **Compliance**: Adhering to data protection regulations (e.g., GDPR).

4 Exhibit Setup Considerations

4.1 Participant Experience Design

- **Controlled Environment**: A designated area where lighting and background conditions are optimal.
- **Wearables**: Provision of uniform robes and headgear to minimize recognition based on non-facial features.
- **Rotation System**: Scheduling to ensure participants experience both known and unknown faces.
- **Data Gathering**: Data will be gathered to measure the participants' expriences using psychometrics such as [**degutis˙2014**] [**duchaine˙2006**]

4.2 Hardware Deployment

- **AR Headsets**: Multiple units for simultaneous use by small groups.
- **Local Servers**: Positioned to minimize network latency, possibly within the exhibit space.
- **Networking Equipment**: High-performance routers and switches dedicated to the exhibit.

4.3 Support Infrastructure

- **Technical Staff**: On-site personnel to assist with equipment and troubleshoot issues.
- **Educational Materials**: Contextual placards or videos explaining the experience.
- **Feedback Mechanisms**: Tablets or kiosks for participants to complete surveys post-experience.

5 Conclusion

The proposed software and hardware architecture leverages cutting-edge technologies in computer vision, machine learning, and augmented reality to create an immersive simulation of prosopagnosia. By carefully balancing on-device processing with potential server-side support, the system is designed to meet the stringent latency requirements essential for a seamless user experience. The modular design ensures scalability and adaptability, allowing for future enhancements and the potential incorporation of additional perceptual disorder simulations.

This architecture not only serves the immediate goals of the project but also lays the groundwork for future research and development in assistive AR technologies and educational tools that foster empathy and understanding of perceptual diversity.