UsAR Mirror

Authors: Steven Lee, Anna Paek, Shengxi Wu
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—This project introduces an AR mirror that enhances the user experience by displaying multiple viewpoints of the user's face. In addition to overlaying 3D filters, the mirror incorporates two webcams on either side of the display, allowing users to view their left and right profiles. This feature is particularly useful for makeup application, as users can see how their makeup looks from different angles, especially the side profile. This addition provides a more versatile and accurate reflection compared to existing AR mirrors.**

***Index Terms*—AR/ VR System, Computer Vision, Sensor Fusion, Face Reconstruction, Gesture Recognition**

1 INTRODUCTION

Augmented Reality (AR) mirrors have seen a significant rise in retail, particularly in boutiques and jewelry stores. These mirrors enhance the shopping experience by allowing customers to virtually try on makeup, clothing, and accessories with convenience and efficiency. Compared to traditional fitting rooms, AR mirrors enable customers to try on **4x as many products** while also increasing engagement—users are **9x more engaged** than with traditional video ads [19]. Retailers also benefit, experiencing **3x higher foot traffic** when AR mirrors are placed in storefront windows to attract passersby. Additionally, AR mirrors can operate for long periods of time, providing an interactive and personalized shopping experience at any time.

Despite these advantages, traditional AR mirrors still face challenges. **Accuracy and rendering speed** remain key issues, particularly in aligning virtual elements with a user's reflection when they move. Tilting the head or stepping away from the mirror can cause virtual overlays to remain fixed in place, disrupting the immersive experience. This misalignment is especially problematic for applications requiring precision, such as virtual clothing try-ons and makeup simulations [13]. Additionally, concerns over **privacy** due to facial recognition technology and **high costs** associated with software, hardware, internet connectivity, and maintenance, especially for high-quality cameras, pose significant barriers to widespread adoption [13].

Our product aims to create a more cost-effective and efficient AR mirror. By integrating multiple cameras, LED lighting, and advanced display technologies, our solution will provide users with **multiple viewpoints** of themselves for a more immersive experience. **Real-time** interaction will be achieved through efficient algorithms and specialized GPU hardware, while an extensive library of AR filters and overlays will further **enhance user engagement**. Additionally, we plan to incorporate smaller, low-cost displays and gesture recognition technology as an **affordable alternative** to expensive touchscreen interfaces.

2 USE-CASE REQUIREMENTS

The UsAR Mirror has four primary use-case requirements. These requirements ensure that the UsAR Mirror delivers a responsive, intuitive, and immersive AR experience. To make sure the AR mirror system is sustainable, it should be able to run continuously for greater than 2 hours straight.

2.1 Accurate Real-Time User Interaction

The UsAR Mirror must provide smooth and responsive interactions. Camera movement delay shall not exceed **200 milliseconds (ms)** when adjusting the viewing angle. A **3D face model** must be generated within **1 second (s) per user**. AR filters shall be displayed at a minimum of **15 Frames Per Second (FPS)** to ensure fluid visuals. AR filter alignment should have no more than **5% pixel deviation** from the user's head position. Fast rendering and accurate AR localization are essential to provide matching experiences as looking into a physical mirror. Low camera movement delay is also important for natural and immersive interaction.

2.2 User-Controlled Camera Movement

Conventional mirror that only provides the front view of the user, so we want our system to allow users to also be able to view the side of their face. Users shall have full control over their viewing angle. Camera adjustments will be managed by **Arduino-controlled stepper motors**, allowing **vertical movement up to 11.8 inches (in)** (matching the display width) and **panning/rotation up to 90°**. The camera shall maintain an accuracy of **±5°** from the desired angle.

2.3 Screenshot Capture and Storage

Users shall be able to capture and save images of themselves directly from the mirror display. A **Capture** option will be available for taking screenshots, which should be saved automatically to either a **default directory** or a **user-selected directory**. The supported image formats should include **PNG** and **JPEG** .

2.4 Gesture-Based Navigation

Users shall navigate menus and make selections using **hand gestures (swipe up, down, left, and right)**. The system must respond to gestures within **200 ms**. Gesture detection shall operate within a **0.5 to 2-meters(m) range** to accommodate different user positions. The gesture recognition accuracy shall be at least **90%** to ensure a seamless user experience. We prioritized gesture recognition reaction speed and reliability to prevent frustration from misinterpretation. In terms of social factor consideration, the ability to interact with the AR mirror without touching it directly effectively prevents the spread of bacteria among users.

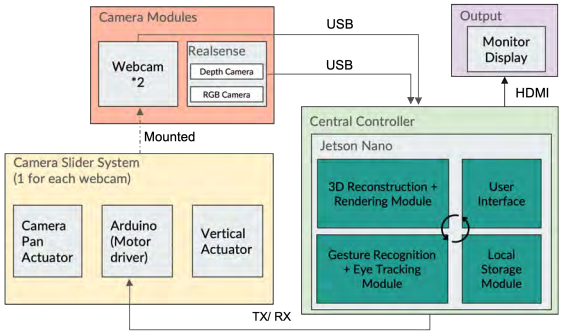

Figure 1: Block diagram of overall system design. (See Fig. 6 for enlargement).

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

The system architecture described in Fig.1 consists of a Jetson Nano as the central processing unit, interfacing with multiple cameras and a motorized camera rig. A depth camera and two webcams are connected to the Jetson Nano via USB, providing visual and depth input for real-time processing. The Jetson Nano serves as the central controller of the system, handling real-time processing through multiple software modules and renders processed AR overlays and outputs them to a display via HDMI. Additionally, the Jetson Nano communicates with an Arduino via TX/RX to control stepper motors that adjust the camera rig's position and rotation angles.

This integrated hardware-software approach enables low-latency, high-accuracy interactions, addressing alignment and rendering speed challenges faced by traditional AR mirrors.

3.1 Computer Vision

The system integrates multi-camera fusion to combine RGB and depth images, improving pose estimation and 3D reconstruction. Depth-based segmentation allows for accurate separation of the user from the background, ensuring precise overlay placement. Additionally, GPU acceleration on the Jetson Nano facilitates real-time rendering, reducing latency and enhancing the smoothness of AR interactions.

The RGB input from sensor is used as input of machine learning models for gesture recognition, allowing users to interact with the AR interface using hand gestures instead of physical buttons. Eye-tracking algorithms estimate eye position which makes it possible to adjust frame position relative to the position of the eyes.

3.2 Actuator Control

The motorized camera rig operates based on instructions from the Arduino, which receives commands from the Jetson Nano. The camera's movement is driven either by user gestures for horizontal panning or by an initial eye-position-based adjustment to ensure the camera aligns with the user's eye level for an optimal viewing experience.

To maintain smooth operation, the pan and vertical actuators follow controlled motion profiles, minimizing abrupt shifts that could disrupt user interaction. Additionally, the seamless integration of embedded systems allows the Jetson Nano to communicate efficiently with the Arduino-controlled motor system, enabling precise and responsive camera positioning.

4 DESIGN REQUIREMENTS

The design requirements specify the necessary hardware, software, and performance criteria to ensure the use-case requirements are met.

4.1 Hardware Requirements

- **Camera System:** The system shall incorporate a camera sensor with a minimum capture rate of **60 FPS** for real-time tracking and rendering. The display must support a refresh rate of at least **60 Hz** for seamless interaction.
- **Camera Positioning:** An **Arduino-driven stepper motor** shall control vertical movement (**up to 11.8 inches**) and **panning/rotation up to 90°** with an angular accuracy of **±5°**. The width of the display is around 11.8 inches, so we want to capture as much distance as possible. We want up to 90° so that we can capture the user's side profile.
- **Computing Hardware:** The **Jetson Orin Nano** shall be used for GPU-accelerated processing, handling real-time **image processing, gesture recognition, and AR filter rendering** while maintaining low latency.
- **Thermal and Power Stability:** The system must operate continuously for $\geq$ **2 hours** without overheating or performance degradation, which are important factors to make sure system is sustainable for long period of time without malfunctioning. 2 hours

is the estimated time duration a user would spend using the AR mirror.

4.2 Software Requirements

- **Gesture Recognition:** The system shall use a **computer vision algorithm** to detect swipe gestures (left, right, up, down) with an accuracy of $\geq$ **90%**. The gesture recognition pipeline must provide real-time feedback with an end-to-end latency of $\leq$ **200 ms**. This is to make sure the user experience isn't impaired with slow framerate or input latency.
- **3D Face Model Generation:** To ensure realistic AR filter alignment, the system must reconstruct a **3D face model within 1 second** using efficient mesh reconstruction techniques. The model must have an average **RMSE** $\leq$ **5 mm** when compared to the full point cloud generated from depth data. Although a high start up cost per user, moving the longest compute to the start prevent disrupts after AR overlay is activated, thus ensures the user experience would not be delayed or interrupted as AR filter is on.
- **AR Filter Rendering:** The system shall use **OpenGL shaders** for real-time texture mapping and rendering. The rendering engine must maintain a frame rate of $\geq$ **15 FPS** to ensure smooth application of AR effects.
- **Filter Tracking Accuracy:** The system shall maintain AR filter alignment with an accuracy of $\leq$ **5% deviation** from the user's head movement and a drift of $\leq$ **10 pixels** per frame at **1920 $\times$ 1080 resolution**. This fulfills the requirements of accurate real-time rendering and minimizes perceptible drift, preventing noticeable misalignment that could disrupt the user's experience with the AR filter.
- **Camera Control and UI Navigation:** The system shall allow users to select their desired camera view and filter options via gesture input. The camera angle selection must have an accuracy of $\geq$ **95%**, with a deviation of no more than **5°** from the intended orientation. We give ourselves a 5% room for error in accuracy to account for jitters and abrupt movement. Theoretically, it should be greater than 90% since the camera control is relatively precise with mathematical equations (trigonometry) as its algorithm.
- **Screenshot Capture and Storage:** Users shall be able to capture and store screenshots with a **100% success rate**. The saved images must retain their quality in **PNG or JPEG format**, and file storage shall be verified to prevent corruption. This is to ensure quality of experience and make sure the app isn't frustrating to use for the end user. PNG ad JPEG format are high-quality and the most widely used image format.

5 DESIGN TRADE STUDIES

5.1 Microcontroller Choice for Camera Control System

When choosing the right microcontroller for our camera control system, there are several important factors and trade-offs we considered, including clock speed, I/O pin configuration, memory (flash and SRAM), the number of PWM pins, size, USB interface, and power supply.

While the Arduino Nano offers a built-in USB port and more analog pins, the Arduino Pro Mini is better due to its higher power and energy efficiency. Since the camera will need to operate for extended periods of time, particularly when the user is interacting with the mirror, the ability to minimize power consumption is essential. Additionally, while the Nano's built-in USB port could be convenient, we can instead use an external USB-to-serial adapter for uploading code, which eliminates the need for the built-in USB port [5, 20].

The Arduino Mega 2560 has more I/O pins and larger memory (256 kB of flash memory and 8 kB of SRAM). However, our mirror does not require a that many I/O pins or extensive memory capacity. Our system will only have a few electronic components, such as OLED display, stepper drivers, switch, and rotary encoder push buttons, and our code does not need such large memory resources [2, 7].

While the ESP32 offers substantial processing power with its 240 MHz dual-core processor and built-in Wi-Fi and Bluetooth, these features are unnecessary for our application. The mirror does not require wireless connectivity or extreme processing power, so the ESP32 would be overkill for this project, both in terms of complexity and power consumption [21].

Finally, the Raspberry Pi Zero W and Teensy 4.0 are also too complex for the relatively simple task of controlling the camera. These platforms, while more powerful, would introduce unnecessary complexity and consume more power than what is required [8, 18].

5.2 Pre-defined Facial Priors for 3D Reconstruction

One key trade-off in 3D face reconstruction is between the speed of constructing the 3D face model with the accuracy of the final reconstruction. Method that includes high detail such as iterative volumetric fusion (e.g., Dynamic Fusion) [14], offer high flexibility but come at the cost of increased computational overhead and processing time. Instead, our system leverages pre-defined facial priors, wrapping the captured face data onto a structured facial template using key facial landmarks [11]. This approach ensures that the 3D model remains structured but also computationally efficient, meeting the real-time processing constraints necessary for AR applications.

Additionally, by directly aligning facial landmarks to the pre-modeled face structure, our method eliminates the need for Iterative Closest Point (ICP) registration when

mapping the reconstructed face onto a rendering model. This is very important as with experiments, the full point cloud reconstruction takes far more than 1 second to compute, while the landmark-based reconstruction generates a 3D landmark model within 1 second, satisfying the design requirement.

Although some details on the face may be lost, this does not significantly impact our application, as the primary purpose of the 3D face model is to serve as a reference for rendering overlays. Since we apply an additive layer on top of the camera output, minor depth offsets have minimal effect on the final rendered AR output [6]. The overlay remains visually accurate, ensuring that the AR elements align correctly with the user's facial features while maintaining real-time performance.

5.3 Motion Tracking for AR Alignment

One initial approach considered was performing full 3D optical flow tracking on the entire face mesh. This method estimates dense motion across depth variations, allowing for precise tracking of both rigid and non-rigid facial movements. While highly accurate, this makes it extremely difficult to meet the design requirement of 15 FPS rendering with the GPU resources we have.

Another approach we explored was taking advantage of the facial landmarks we extracted and do pose estimation using the Perspective-n-Point (PnP) [17] method. This method estimates rigid head motion by tracking facial keypoints such as eyes, nose, and mouth corners, solving for 3D transformations.

While PnP requires less computation, it comes at the cost of reduced accuracy. One major limitation is that it treats the head as a rigid body, whereas facial expressions can cause certain landmarks to shift without actual head movement, leading to miscalculations. Additionally, rapid head movements and partial occlusions further impact accuracy, as landmark detection may become unreliable in these scenarios. These factors introduce errors in head pose estimation, making PnP alone insufficient for precise AR filter alignment in dynamic environments.

To improve accuracy, we adjust for these limitations by categorizing facial landmarks into sub-regions, distinguishing between rigid and non-rigid motion. The nose, facial contour points, and the outer corners of the eyes and mouth are used for rigid head motion estimation, ensuring stable tracking of overall head position. Meanwhile, non-rigid facial motions, such as eye blinking or mouth opening, are accounted for separately to prevent misalignment caused by expression changes. This hybrid approach allows for more precise head pose estimation, making the system more robust to different head motions while maintaining real-time performance.

5.4 Dear ImGui for the UI

Dear ImGui is an efficient and flexible immediate-mode graphical user interface (GUI) framework, making it ideal for real-time applications. We chose Dear ImGui for its low power consumption and performance efficiency. It only processes UI components displayed on the screen, which conserves resources for more demanding tasks like image processing, machine learning, and camera control [1, 9]. This is crucial for the mirror, as users will interact with it for extended periods.

Unlike other GUI frameworks such as Qt or GTK, which maintain a static UI state and require complex setup, Dear ImGui renders only the actively displayed elements. This immediate-mode approach reduces overhead, ensures efficient resource usage, and enables quick rendering with low-latency updates, essential for real-time feedback on camera angle controls and filter selection [22].

Fully compatible with Linux-based systems like the Jetson Nano, Dear ImGui integrates seamlessly with libraries such as X11, SDL, and OpenGL, avoiding the need for complex configurations or heavy dependencies [15, 23]. This reduces system strain and simplifies maintenance for system integration.

Additionally, Dear ImGui speeds up UI development, allowing for rapid prototyping and testing, which is critical for integrating and optimizing various subsystems within the camera control and gesture recognition systems [16].

6 SYSTEM IMPLEMENTATION

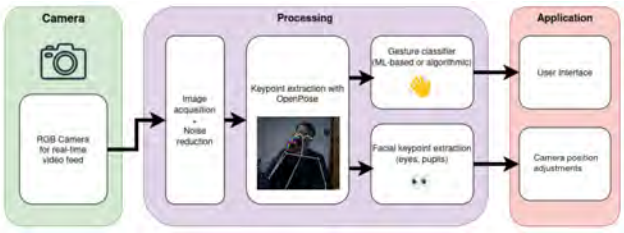

Figure 2: Overview of gesture recognition and eye tracking system. (See Fig. 7 for enlargement).

6.1 Gesture Tracking System

The gesture tracking system allows the user to control the application in front of the mirror without having to use a controller or touchscreen-based input. This system will comprise of a computer-vision based pipeline which transforms a real-time camera feed of a person to application input commands (such as filter adjustments, camera adjustments), as detailed in Fig. 2. To achieve this, **OpenPose** is utilized for body keypoint extraction and a custom gesture recognition algorithm is written for application input [3].

Image processing, image acquisition, denoising, and body keypoint extraction will be handled exclusively by OpenPose's C++ API (provided via a dynamic library, which can be dynamically linked to the main application binary). Using these keypoints, a rolling history of 3D pose

data (e.g. relative positions of the joints) is stored and a gesture recognition algorithm can be evaluated on the stored pose history to classify which gesture input it belongs to at the current frame. The gesture algorithm will be rule-based and inspect relevant features such as angular velocity of different bones in the estimated pose. For pipeline output, this system will provide an interface to other parts of the application (in particular, the UI) in the form of an API where users can access the gesture at the current frame.

To meet performance requirements, OpenPose's CUDA backend will be used on the Jetson Nano to take advantage of GPU acceleration and reduce keypoint estimation latency incurred by the underlying ML-based vision models that the library uses. Moreover, this project opts to use C++ over Python, as the Jetson Nano is a constrained device, and using a well-optimized compiled language over an inefficient interpreted one will make it easier to meet latency requirements.

Figure 3: Overview of the UI [4].

6.2 User Interface (UI)

Users can navigate the horizontal menu of filters by swiping left or right, with arrows indicating the direction of selection. Vertical swipes determine whether they are adjusting the camera angle or changing the filter.

Since the camera angle selection is at the top of the screen, users swipe up to adjust the angle. To change the filter, users swipe down, as the filter menu is located at the bottom.

For angle control, swiping left rotates the view to the left. Swiping right rotates the view to the right. In other words, 0° indicates that the user is looking at the front view of their face, -30° indicates that the user is looking 30° to the left side of their face, and +30° indicates that the user is looking 30° to the right of their face.

This intuitive gesture-based system allows seamless interaction without the need for physical buttons or controllers.

6.3 Eye tracking system

The eye tracking system of the project will make sure that the camera rig is consistent and level with the user's eyes to ensure a comfortable and accurate viewing experience. To achieve this, this system will work alongside the gesture recognition pipeline as both will use data from OpenPose library. In the case of the eye tracking system, keypoints for the eyes will be extracted from the real-time camera feed via OpenPose, then projected onto screen-space coordinates. A PID algorithm will be used to make sure appropriate feedback is provided to the camera rig so that the eye positions, in the image received by the camera, stay at some preset level along the height of the screen.

In addition, the eye tracking system will play a part in correcting for camera distortion, and transforming the webcam feed into a perspective which is mirror-like and comfortable to view for the user.

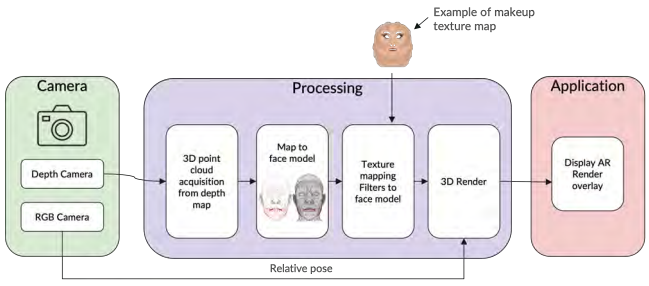

Figure 4: Overview of depth reconstruction and rendering system. (See Fig. 8 for enlargement).

6.4 Face Reconstruction System

The 3D face reconstruction and rendering process is responsible for generating a structured 3D face model of the user in real time as described in Fig. 4. This process utilizes the Intel Realsense camera and takes input from the co-located RGB and depth camera within the Realsense module, which are then transmitted via USB to the Jetson Nano for processing.

To extract key facial features, **dlib** [11] is used for facial landmark detection, using the RGB input from the Realsense camera to identify key facial features and contours.

These detected landmarks are then projected onto the corresponding depth image, ensuring precise feature alignment between 2D and 3D data. The conversion from 2D image coordinates to real-world 3D coordinates is performed using ray marching, leveraging the camera's intrinsic and extrinsic parameters for accurate depth-to-3D mapping.

For visualization and point cloud processing, **Open3D** is used to render the reconstructed 3D face model, integrating the extracted 3D landmarks into a cohesive structure. The Jetson Nano SDK with **CUDA** acceleration is employed to optimize depth processing and real-time rendering, utilizing GPU parallelization to enhance computational efficiency and ensure a smooth user experience.

6.5 AR Overlay Rendering System

The 3D facial landmarks from the prior section are warped onto a 3D face model, aligning with predefined facial structures. This ensures that the virtual elements conform accurately to the user's facial geometry, providing a natural and realistic AR experience.

To maintain accurate overlay positioning, the 3D face model undergoes a coordinate transformation to align with the webcam's perspective, the calibration between different hardware components are done using **OpenCV calibration** and **Apriltag**, and changes is motion during the process is tracked by operation log of motors. Additionally, real-time head motion tracking is incorporated to ensure that the AR filter remains correctly positioned on the user's head. This is achieved using **PnP** [17], which analyzes the motion patterns of key facial features—such as the eyes, nose, and mouth endpoints—to compute both the 3D rigid motion of the head and the non-rigid transformations of facial sub-regions. By doing so, the AR filters remain properly positioned, even as the user moves within the camera's field of view.

OpenGL shaders handle real-time texture blending between the predefined texture map of makeup effect selected by the user and the webcam feed, with opacity control to blend naturally with the live video. This process ensures that AR effects appear seamlessly integrated with the user's face, avoiding visual artifacts or misalignment.

6.6 Camera Control System

The camera control system allows users to select and adjust their viewing angles in real-time, enhancing their interaction with the AR mirror. The system consists of two webcams, each mounted on a separate camera rig controlled by stepper motors. Each camera rig is equipped with two stepper motors: one for enabling up/down motion and the other for rotation [12].

The Jetson Nano serves as the main processing unit, determining the appropriate movement based on user input (i.e. angle selection) or through automated tracking algorithms. The system operates in four modes: pan, rotate, pan & rotate, and track object. To execute these movements, it utilizes five key parameters: distance, travel direction, rotation angle, rotation direction, and duration. Distance specifies how far the camera moves up or down. Travel Direction determines whether the camera moves up or down. Rotation Angle defines the degree of camera rotation. Rotation Direction specifies whether the camera rotates clockwise or counterclockwise. Duration controls how long the selected movement persists.

The Jetson Nano communicates movement commands to an Arduino via I2C or UART, which then controls the stepper motors accordingly. The object tracking mode uses computer vision techniques, such as face or body tracking, to automatically adjust the camera angle in real-time, ensuring users remain properly framed.

For optimal performance, the system will require calibration to align movement parameters with the physical constraints of the rig. Additionally, maximum rotation limits and speed caps will be implemented to prevent mechanical strain and excessive motion, ensuring smooth and reliable operation.

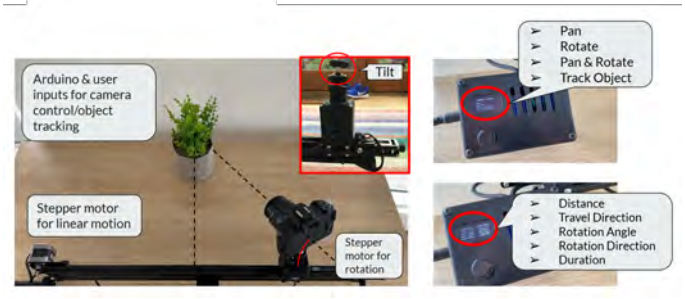

Figure 5: Overview of camera control system [12].

7 TEST & VALIDATION

These tests ensure that the system meets real-time performance constraints, maintains accurate tracking, and provides a reliable user experience. Through risk management strategies, we mitigate potential hardware failures, computational delays, and user interaction inconsistencies, ensuring a robust and efficient AR system. We will be conducting a 2 hour stress test to ensure that the mirror is functional for the two full hours.

7.1 Tests for Accurate Interaction in Real-Time

One of the critical aspects of our system is ensuring accurate and responsive real-time interaction. We will test this by using high-speed logging scripts to get the exact time stamps.

There will be continuous depth map feed and user head motion as test inputs. The expected outputs are less than a 1 second delay in generating 3D face model per user. Furthermore, the 6DoF head transform must be computed in $\leq$ 200 milliseconds, enabling accurate alignment of AR overlays. The 3D reconstruction error is expected to have

an average RMSE threshold of $\leq$ 5 mm (after transform from camera coordinates to world coordinates), to ensure the generated 3D model to the full point cloud of head reconstructed from depth map.

To evaluate the performance of AR filter rendering, there will be continuous real-time camera feed and filter selection as input. The system must maintain a minimum frame rate of $\geq$ 15 FPS to ensure smooth and immersive visual effects. Furthermore, to measure the stability of the filter in natural head movement, $\leq$ 10 pixels of drift (for 1920 * 1080 resolution) should be guaranteed for motion within the system's field of view, and the overlay should maintain a deviation of $\leq 5^\circ$ compared to the orientation of the detected head.

7.2 Tests for Screenshots and Photos

We aim for a 100% success rate of screenshots saved and to verify file creation and image quality. To do so, we will take multiple screenshots all at once (simulating it as if the user is having a one-time experience).

7.3 Tests for Selecting Target Views

We want a 95% accuracy of camera orientation and frame stabilization. Additionally, we want at most 5° deviation from the desired angle. We can test this by moving the camera in all directions and getting exact camera degree from Arduino print statements. In terms of time it takes to react to selections made by user, with continuous real-time camera feed as input, we want to guarantee the camera reacts within a 200 millisecond delay.

7.4 Tests for Gesture Recognition & Navigating Menus

We want the mirror to clearly distinguish between the up, down, left, right swiping motion. Tests will be manually performed, where one repeatedly makes a series of gestures. The expected time for the system to identify and react to user motion in UI should be less than 200 millisecond end to end. Classification accuracy will be measured via a confusion matrix. Moreover, we will perform these tests at various distances away from the mirror to ensure that our system is tolerant under diverse conditions. We want a detection range of 0.5-2 meters, and a 90% accuracy in gesture cue recognition. We will use a series of hand gestures under various conditions, including reverse action.

8 PROJECT MANAGEMENT

8.1 Schedule

The schedule is shown in Fig. 9.

8.2 Team Member Responsibilities

Steven will be responsible for the development and implementation of the gesture recognition and eye tracking systems for the AR mirror. His primary task will be designing and optimizing algorithms that enable the mirror to accurately detect and interpret user gestures in real-time. Additionally, Steven will oversee the integration of eye tracking technology to track the user's gaze and focus, ensuring that the AR content is dynamically adjusted based on the user's perspective. This functionality will enhance the user experience by providing intuitive control and ensuring the AR display responds accurately to gestures and eye movements.

Anna will manage the camera control system and develop the UI for the AR mirror. Her responsibilities will include overseeing the configuration and calibration of the camera system to ensure optimal image capture and tracking. She will also be in charge of designing and implementing a user-friendly interface, ensuring that interactions with the AR mirror are intuitive and visually appealing. Anna will work closely with the rest of the team to ensure that the camera system integrates seamlessly with the gesture recognition, eye tracking, and AR filters, and that the UI provides clear, interactive elements for the user to engage with.

Shengxi will be responsible for 3D reconstruction and the development of AR filters for the AR mirror. Her role will involve creating accurate 3D models of the user's environment and incorporating them into the AR system to enhance the realism of the AR experience. She will develop algorithms for real-time 3D scanning and rendering, ensuring that virtual objects or effects blend seamlessly with the physical environment. In addition to 3D reconstruction, Shengxi will design and implement AR filters that provide users with customizable and interactive visual effects. Her work will be critical to ensuring the AR mirror delivers a polished and immersive augmented reality experience.

8.3 Bill of Materials and Budget

Please refer to Table 1 for a full list of items and its associated costs. The capacitors will be supplied by the ECE receiving offices, therefore amounting to $592.80.

8.4 TechSpark Usage Plans

Techspark will be 3D printing the camera tripod shoes, motor coupling, and timing belt clamp for the camera control system.

8.5 Risk Mitigation Plans

To ensure the system operates reliably under real-world conditions, we have identified potential risks and developed mitigation strategies.

8.5.1 Head Motion Tracking and 3D Reconstruction

To mitigate inaccuracies in head motion tracking and 3D reconstruction, we introduce motion smoothing techniques to prevent jitter, ensuring stable tracking results. Additionally, fallback strategies are implemented for cases where facial landmarks are occluded or missing, allowing the system to compensate for data loss and maintain tracking accuracy. To ensure a consistent response to user engagement, a gesture cue is incorporated to trigger face model reconstruction, reducing errors caused by unexpected system delays.

8.5.2 AR Filter Stability and Rendering Performance

To maintain stable AR filter placement and rendering performance, GPU-based rendering pipelines are optimized to sustain consistent frame rates for real-time interaction. Adaptive resolution scaling is also employed to dynamically balance performance and rendering quality, preventing processing overload while preserving visual fidelity. Furthermore, motion-based correction methods are used to minimize AR filter drift when users move, ensuring that overlays remain accurately positioned throughout interactions.

8.5.3 Screenshot Capture and Image Storage

To address potential failures in screenshot capture and image storage, robust error-handling mechanisms are implemented to detect and correct file I/O issues. Additionally, real-time feedback warnings are integrated to alert users of storage or saving failures, preventing unexpected data loss and enhancing the reliability of the system's screenshot functionality.

8.5.4 Camera Orientation and Stabilization

To improve camera positioning accuracy, an automatic feedback loop is introduced to detect and correct mechanical or software-based misalignment, ensuring precise orientation adjustments. Additionally, manual user controls are provided to allow fine-tuning of the camera position, giving users greater control over their preferred viewing angles while maintaining stability.

8.5.5 Gesture-Based Navigation

To improve the robustness of gesture recognition, extensive testing is conducted under various lighting conditions and distances to ensure reliable operation across different environments. Additionally, a fallback mechanism using mouse control is incorporated to provide an alternative interaction method in cases where gesture recognition fails. To enhance gesture detection accuracy, the system is tested under multiple conditions, including reversed actions, ensuring that hand motions are consistently and correctly interpreted.

9 RELATED WORK

Recent advancements in computer vision have significantly improved the quality and effect of Augmented Reality (AR) filter overlays, enabling more immersive and realistic visual effects for many applications. Both academic research and industry innovations have focused on real-time face tracking, 3D reconstruction, and AR-based visualization, which closely align with the technicals and goals of our project.

One widely adopted application of AR filters is seen in mobile applications like Instagram filters, which apply 2D overlays to users' faces in real-time. Similarly, dedicated applications like Sephora Virtual Artist [10] dedicates specifically at virtual makeup try-ons, allowing users to preview cosmetic products through AR-enhanced facial mapping. While these applications offer real-time virtual effects, they primarily rely on single-camera setups with limited range, whereas our system aims to provide a more immersive multi-view AR mirror experience.

For integrating these AR effects into a hardware-based interactive system, Wang et al. [24] introduced the Perspective-Aligned AR Mirror with Under-Display Camera, an AR mirror system designed to dynamically adjust the user's viewpoint for an enhanced immersive experience. Their approach leverages real-time computer vision techniques to refine AR overlay accuracy, similar to our system's focus on precise filter alignment and real-time responsiveness.

10 SUMMARY

In brief, UsAR Mirror is an augmented reality system that enhances makeup application by displaying multiple viewpoints of a user's face. Unlike traditional mirrors or existing AR solutions, this system incorporates two webcams on either side of the display and allows users to adjust their viewing angle, enabling them to see their left and right profiles in real time. Real-time AR overlays allow filters and virtual makeup to be applied to the user's face with minimal deviations. Users will be able to control the interface entirely through intuitive hand gestures, handling operations including capture images of themselves, switching between filters and adjust camera angles.

Our team faces several implementation challenges ahead. Regarding performance of our application, maintaining less than 200 ms of latency for camera movements while simultaneously processing depth data, facial landmark detection, and AR overlay rendering on the Jetson Nano will be computationally demanding. Generating a 3D face model within one second per user while maintaining high fidelity for the AR overlay requires careful performance fine-tuning of the face reconstruction pipeline. Achieving 90 % accuracy in gesture recognition across various lighting conditions and distances (0.5 to 2 m) will require robust algorithm development and testing.

Coordinating multiple subsystems to work together seamlessly will be the our most significant challenge, especially given the accuracy and real-time performance requirements. The careful consideration of design trade-offs and risk mitigation strategies should help address these challenges as implementation proceeds.

Glossary of Acronyms

Include an alphabetized list of acronyms if you have lots of these included in your document. Otherwise define the acronyms inline.

- AR - Augmented Reality
- VR - Virtual Reality
- I2C - Inter-Integrated Circuit
- I/O - Input/Output
- PRWM - Pulse Width Modulation
- SRAM - Static Random Access Memory
- TX/RX - Transmit/Receive
- UART - Universal Asynchronous Receiver Transmitter

References

[1] K. Akira. "A study on real-time performance of Immediate Mode GUI (ImGui) framework". In: *Journal of Real-Time Software Engineering* 15.2 (2020), pp. 85–95. DOI: DOIlink.

[2] Arduino. *Arduino Mega 2560*. Accessed: 2025-03-02. 2024. URL: https://www.arduino.cc/en/Main/ArduinoBoardMega2560.

[3] Z. Cao et al. "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[4] Jun-Yan Chang et al. *Photorealistic Image Synthesis for Unreal Engine*. 2018. URL: https://gfx.cs.princeton.edu/pubs/Chang_2018_PAS/index.php.

[5] PCB Copy. *Arduino Pro Mini vs Nano: A Comprehensive Comparison*. https://pcb-copy.com/arduino-pro-mini-vs-nano-a-comprehensive-comparison/. 2024.

[6] echo3D. *Unity AR Foundation Face Makeup Demo*. Accessed: March 2, 2025. 2025. URL: https://github.com/echo3Dco/Unity-ARFoundation-echo3D-demo-Face-Makeup.

[7] Arrow Electronics. *Arduino Uno vs Mega vs Micro*. https://www.arrow.com/en/research-and-events/articles/arduino-uno-vs-mega-vs-micro?. Accessed: 2024-03-02. 2024.

[8] Raspberry Pi Foundation. *Raspberry Pi Zero W*. Accessed: 2025-03-02. 2024. URL: https://www.raspberrypi.org/products/raspberry-pi-zero-w/.

[9] Dear ImGui. *Dear ImGui: Bloat-free Graphical User Interface for C++ with minimal dependencies*. https://github.com/ocornut/imgui. Accessed: 2025-03-02. 2024.

[10] Business Insider. *Sephora's Virtual Artist App Feature Teaches How to Apply Makeup Using AI*. Accessed: March 2, 2025. 2017. URL: https://www.businessinsider.com/sephora-visual-artist-app-feature-teaches-how-to-apply-makeup-using-ai-photos-2017-3.

[11] Davis E. King. *Dlib C++ Library: Machine Learning and Computer Vision*. Accessed: March 2, 2025. 2025. URL: https://dlib.net/.

[12] Michael Klements. *DIY Motorised Camera Slider with Object Tracking*. Accessed: March 2, 2025. 2024. URL: https://www.the-diy-life.com/diy-motorised-camera-slider-with-object-tracking/.

[13] Moody Mattan. *AR Mirror Technology in 2024: Transforming the Retail Landscape*. 2024. URL: https://www.brandxr.io/ar-mirror-technology-in-2024-transforming-the-retail-landscape.

[14] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. "DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 343–352. DOI: 10.1109/CVPR.2015.7298631. URL: https://doi.org/10.1109/CVPR.2015.7298631.

[15] O. O'Cornut. *Dear ImGui: A Bloat-Free GUI for Real-Time Applications*. https://www.siggraph.org/. 2018.

[16] O. O'Cornut. *Dear ImGui and Its Application in Embedded Systems*. https://www.siggraph.org/. Accessed: 2025-03-02. 2024.

[17] OpenCV. *solvePnP: Camera Calibration and Pose Estimation*. Accessed: March 2, 2025. 2025. URL: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html.

[18] PJRC. *Teensy 4.0*. Accessed: 2025-03-02. 2024. URL: https://www.pjrc.com/teensy/teensy40.html.

[19] *Reflecting Trends: AR Mirrors in Marketing and Retail*. 2024. URL: https://blog.lenslist.co/2024/02/29/reflecting-trends-ar-mirrors-in-marketing-and-retail/.

[20] SparkFun Electronics. *Arduino Comparison Guide: Mega's, Arms, YNsoh, My*. https://learn.sparkfun.com/tutorials/arduino-comparison-guide/megas. 2024.

[21] Espressif Systems. *ESP32 Technical Specifications*. Accessed: 2025-03-02. 2024. URL: https://www.espressif.com/en/products/socs/esp32.

[22] Unknown. *Immediate Mode GUI: A simplified UI design for real-time applications*. https://www.ics.uci.edu/~eppstein/161/IMGUI/. Accessed: 2025-03-02. 2024.

[23] Unknown. *SDL – Simple DirectMedia Layer*. https://www.libsdl.org/. Accessed: 2025-03-02. 2024.

[24] Jian Wang et al. "Perspective-Aligned AR Mirror with Under-Display Camera". In: *ACM Transactions on Graphics (TOG)* 43.6 (Nov. 2024). ISSN: 0730-0301. DOI: 10.1145/3687995. URL: https://doi.org/10.1145/3687995.

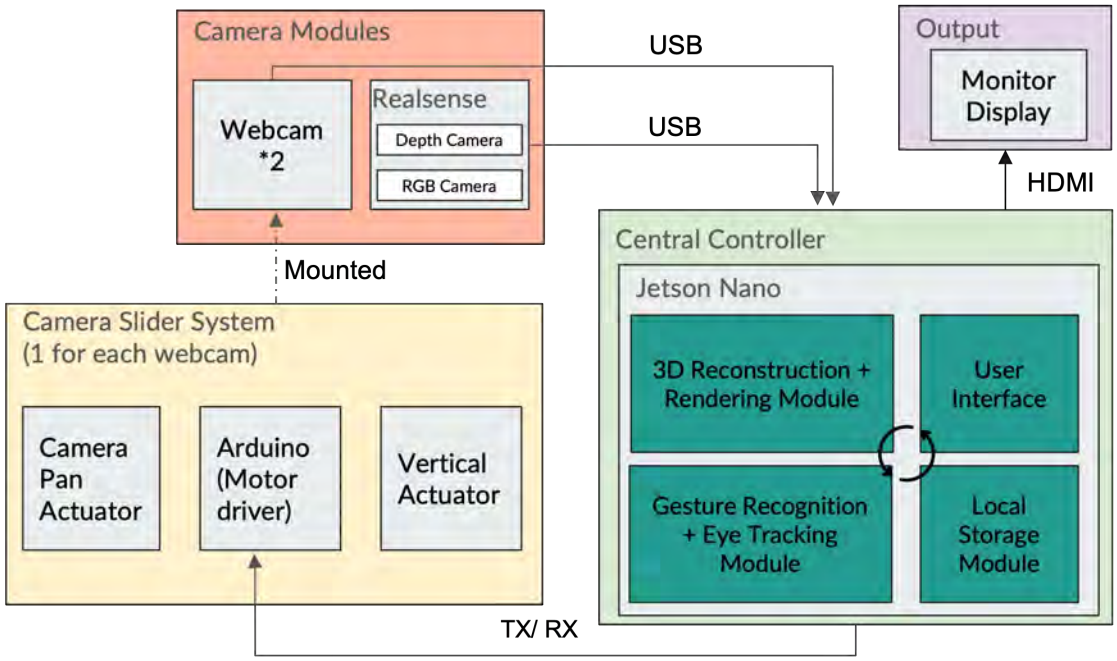

Figure 6: Enlargement of Figure 1.

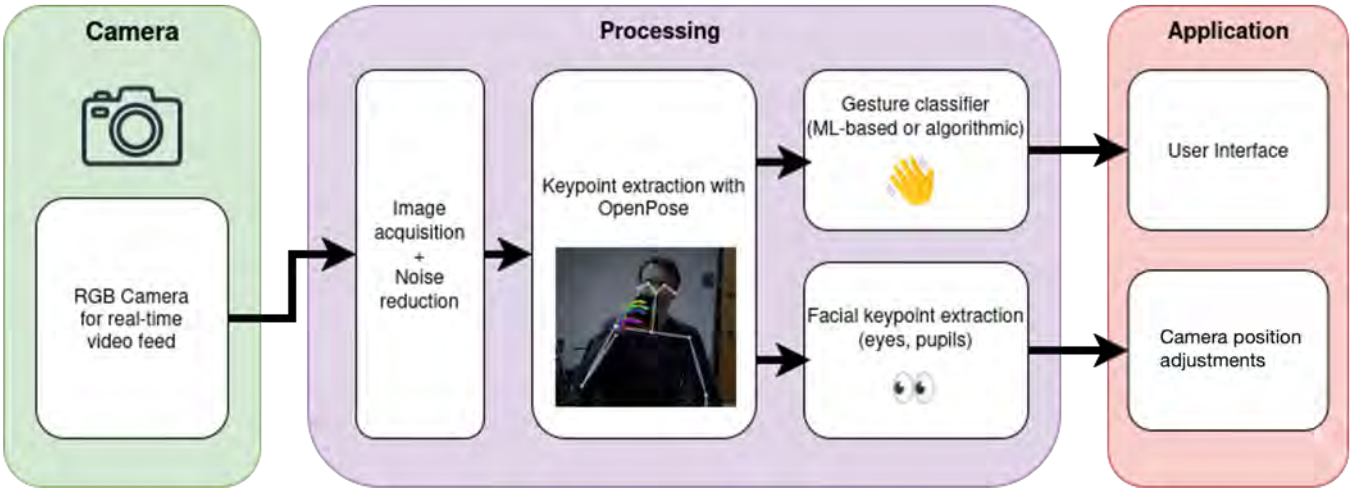

Figure 7: Enlargement of Figure 2.

Table 1: Bill of Materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Pin Header Strip for Arduino	LJQ-40P-ZPZ-PM-TC-2.54-20	CHANZON	1	$9.99	$9.99
Rotary Encoder Code Switch Digital Potentiometer	a19012600ux0593	uxcell	1	$8.89	$8.89
UCTRONICS 0.96 Inch OLED Module 12864	U602602	Broadcom	1	$6.99	$6.99
3D Printer Stepper Motor Driver	TMC2209 V1.3	BIGTREETECH Direct	1	$22.99	$22.99
Swivel Mini Ball Head Screw Tripod Mount	4332050683	AKOAK	2	$6.97	$13.94
M5 Slide in T Nut Tee Sliding Nut	BR-TN-0015	Boeray	1	$7.99	$7.99
Stainless Steel Nuts and Bolts	N/A	Shenzhen Hongkangmin Wujin Youxian Gongsi	1	$18.99	$18.99
V Gantry Plate 3D Printer Parts with Mini V-Wheels	N/A	3Dman	2	$9.99	$19.98
Belt Pulley Wheel Aluminum	TBL6MM205	WINSINN Technology Ltd	1	$6.99	$6.99
GT2 Open Timing Belt	ZR-011101-ca	3D Printer Belt 10m	1	$13.99	$13.99
2PCS 300mm 2040 V Slot Aluminum Extrusion	N/A	Iverntech	1	$16.99	$16.99
X-Axis Synchronous Belt Stretch Straighten Tensioner	AE19600	AEDIKO	2	$7.99	$15.98
Nema 17 Stepper Motor	N/A	YEJMKJ	1	$39.99	$39.99
Arduino Pro Mini	03-01-2003	HiLetgo	1	$13.99	$13.99
PCB	W984022AS5D2	PCBWay	1	$26.91	$26.91
100 microFarad Capacitors	N/A	N/A	4	$5.48	$0.00
HiLetgo FT232RL Mini USB to TTL Serial Converter Adapter Module	3-01-0661-1	HiLetgo	2	$6.49	$12.98
Camera Tripod Shoes	N/A	Techspark	2	$5.19	$10.37
Motor Coupling	N/A	Techspark	2	$2.02	$4.04
Timing Belt Clamp	N/A	Techspark	2	$0.24	$0.48
Monitor	S2425HS	Dell	1	$106.00	$106.00
Realsense Camera	D455	Intel	1	$0.00	$0.00
Realsense Cable	U3S1A01C11-010	Newnex	1	$45.87	$45.87
Jetson Orin Nano	945-137766-0000-000	NVIDIA	1	$0.00	$0.00
Webcam	C922	Logitech	2	$79.99	$159.98
DP to HDMI Cable	HL-007263	Amazon Basics	1	$8.49	$8.49
				Total	$592.80

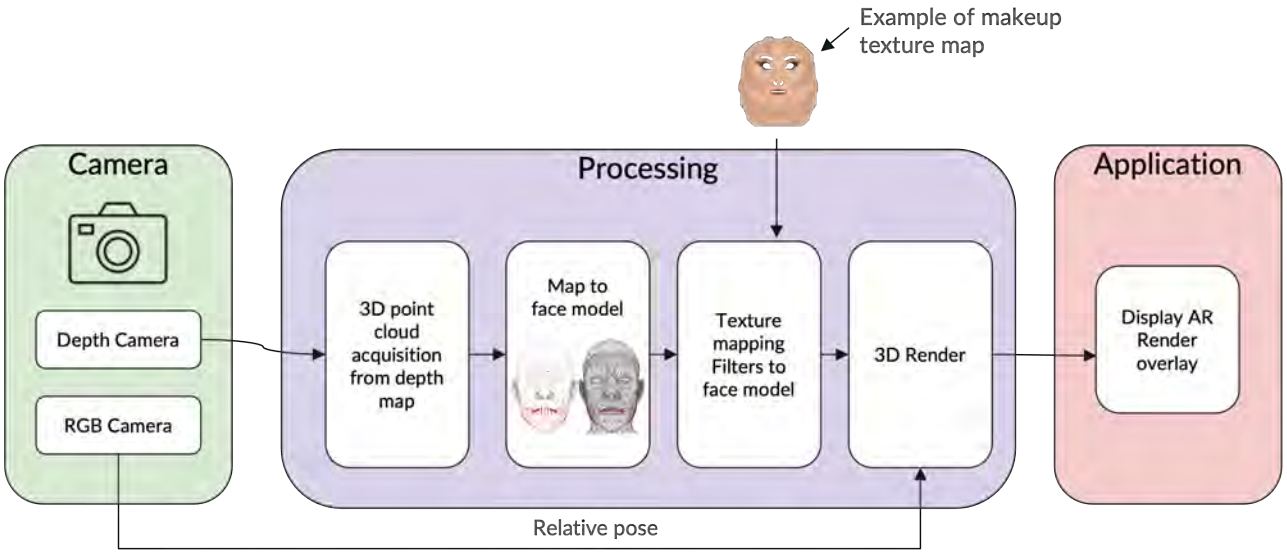

Figure 8: Enlargement of Figure 4.

Figure 9: Enlargement of Schedule.