Assignment Report

IMPLEMENTATION OF HAND-BASED IMAGE INTERACTION INTERFACE USING ML5.JS

Abed Al Rahman | 57744270 | CS3483 | 26/11/2024

Dedicated to: Prof. Hau Sau Wong

Table of Contents

Overview

This report details the implementation of an interactive image viewing system using P5.js and the ml5.js Handpose and COCOSSD models. The system allows users to engage with images using finger tracking and gesture detection.

The project folder contains the following***: sketch.js; index.html; style.css; myImage.jpg;*** and ***replaed_image.jpg***.

Development Journey and Implementation

The development process began with the challenge of creating an interactive image viewing system. The first task was to establish a proper foundation using p5.js and ml5.js libraries.

STAGE 1: SETTING UP THE FOUNDATION

At first, I planned to create two separate canvases in p5.js—one for showing a static image and another for the webcam feed. However, this approach caused an unexpected issue: during the initial implementation, I ran into a major DOM rendering conflict. The two canvases were competing for z-index positioning, which resulted in the webcam feed overlapping the image display area.

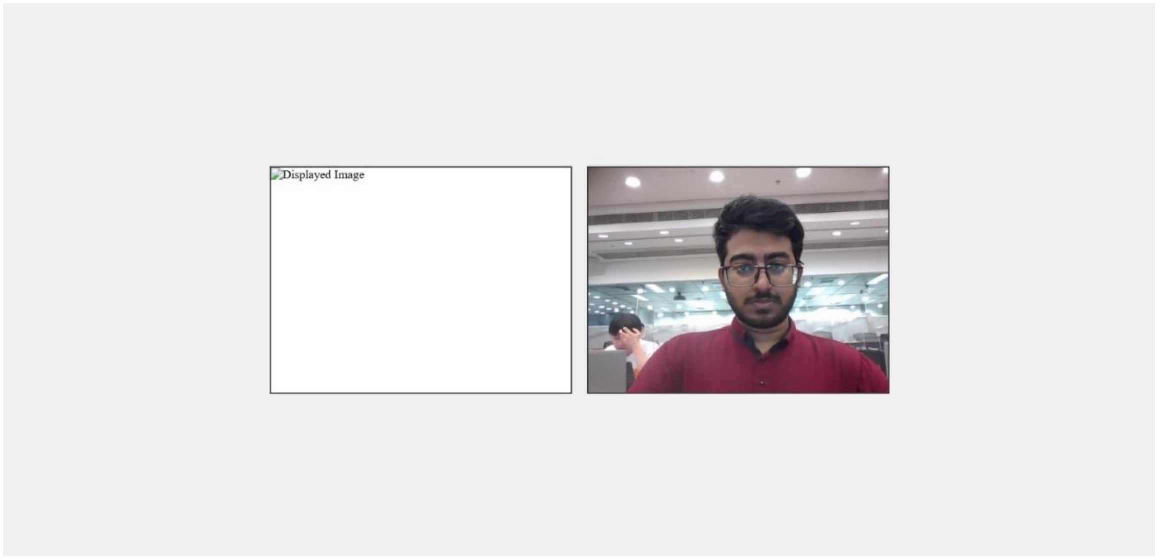


Figure: Initial error state showing canvas overlap

To resolve this, I had to completely restructure the canvas initialization process. By implementing proper event queue management through p5.js's preload() function, I ensured that resources loaded in the correct sequence. After this modification, both canvases displayed correctly.

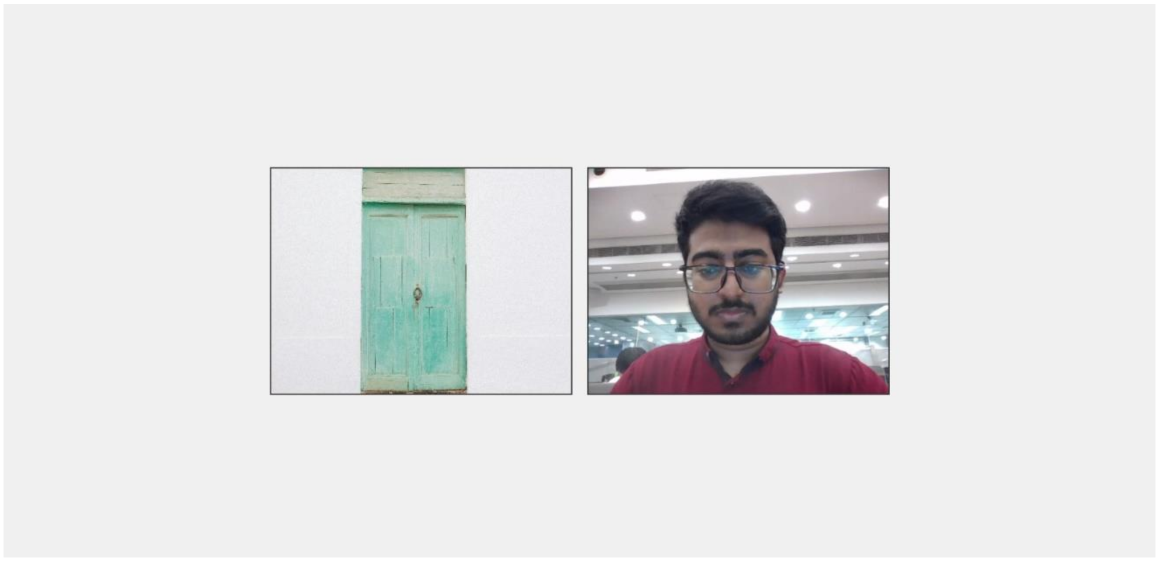


Figure: Fixed canvas layout showing green door image

STAGE 2: WEBCAM INTEGRATION CHALLENGES

This phase brought some unexpected complications with the webcam implementation. While testing the live feed, I discovered that the MediaStream API was failing to maintain consistent access: The camera feed would initially start but then terminate unexpectedly.

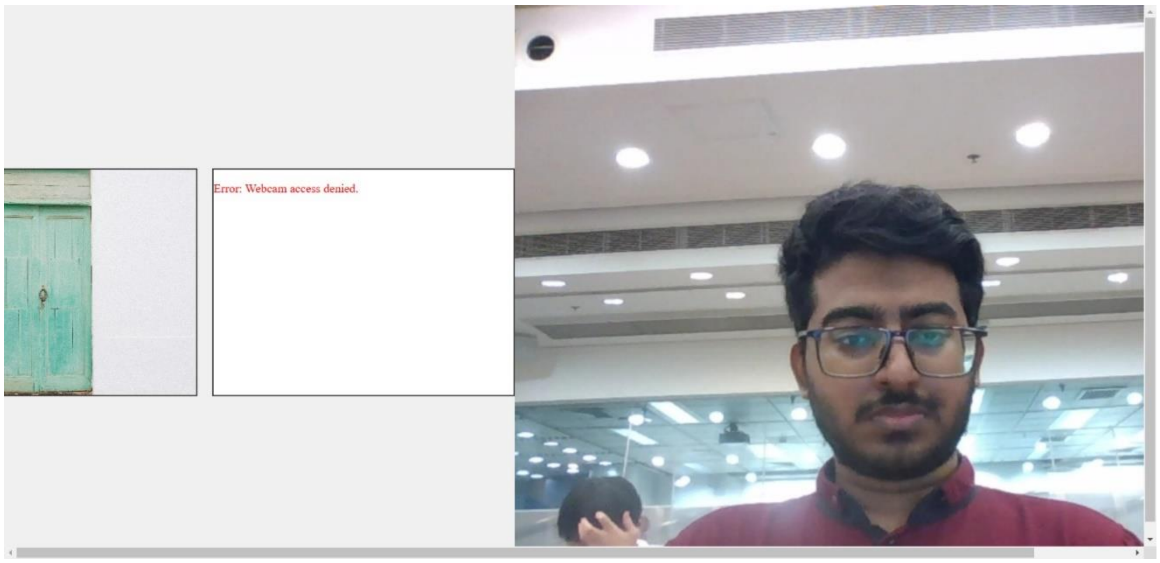

Error: Webcam access denied.

Figure: Camera feed error message

Through console debugging, I identified that this was happening due to asynchronous promise resolution conflicts. The solution required implementing a robust error handling system.

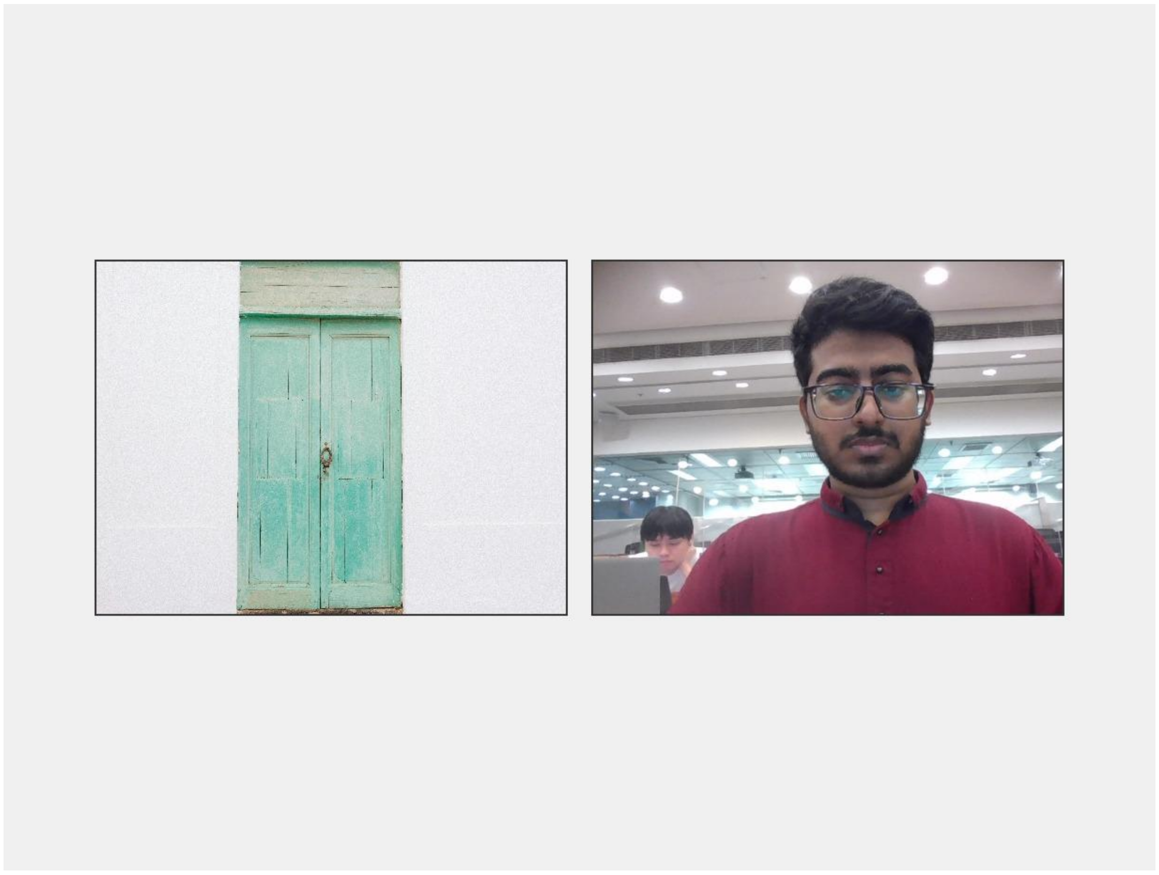


Figure: Successfully implemented webcam feed (after debugging program errors)

STAGE 3: HAND DETECTION EVOLUTION

After carefully reading the ml5 documentations, I understood that the handpose model had to be integrated with some important features, using functionalities derived from the P5.js library.

I drew a red ellipse to indicate the region where the index finger was being detected in the live camera feed (as shown on the right canvas). However, Initial tracking showed significant latency and position inaccuracies. Additionally, the tracking point would often jump erratically, making precise interaction difficult.

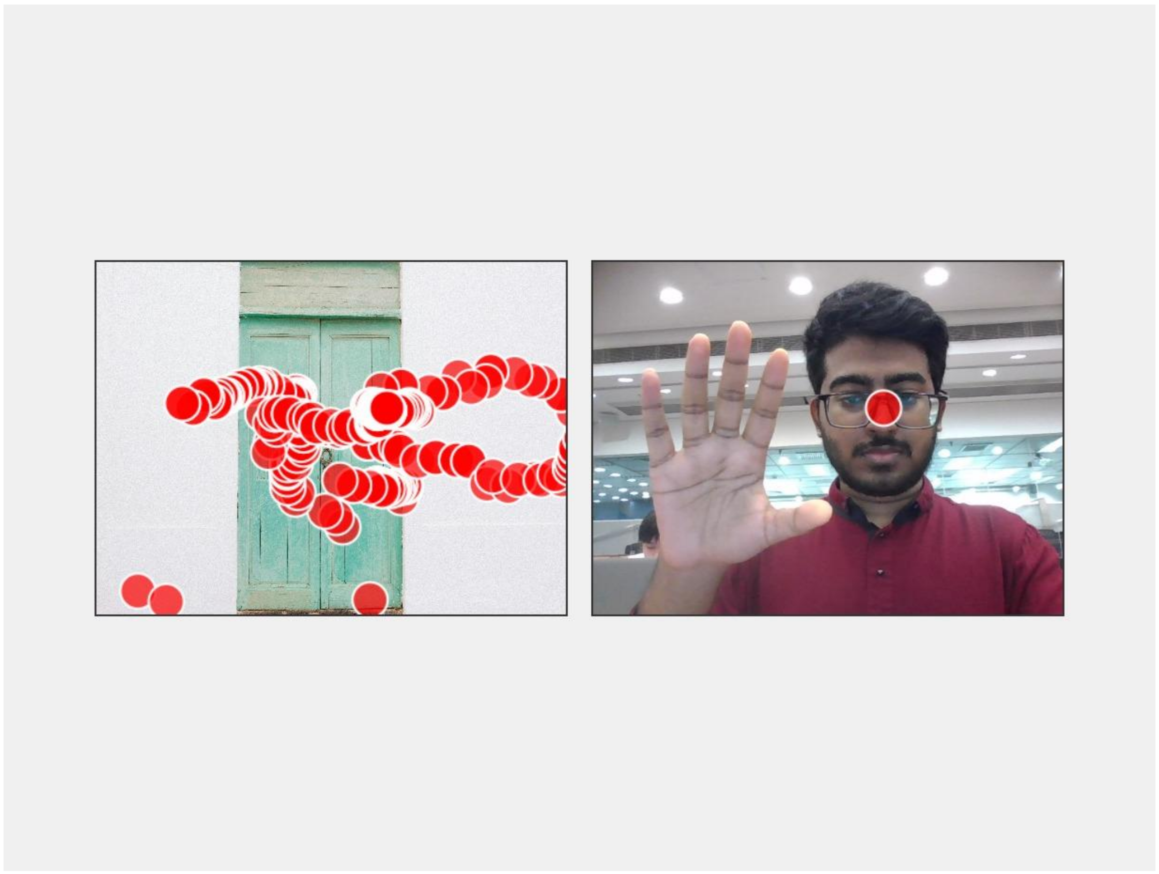


Figure: Initial finger tracking with noticeable lag

To address this, I developed a weighted smoothing algorithm: By implementing an exponential moving average with a carefully tuned smoothing factor, I achieved a balance between responsiveness and stability. This significantly improved the user experience by eliminating jitter while maintaining accurate tracking.

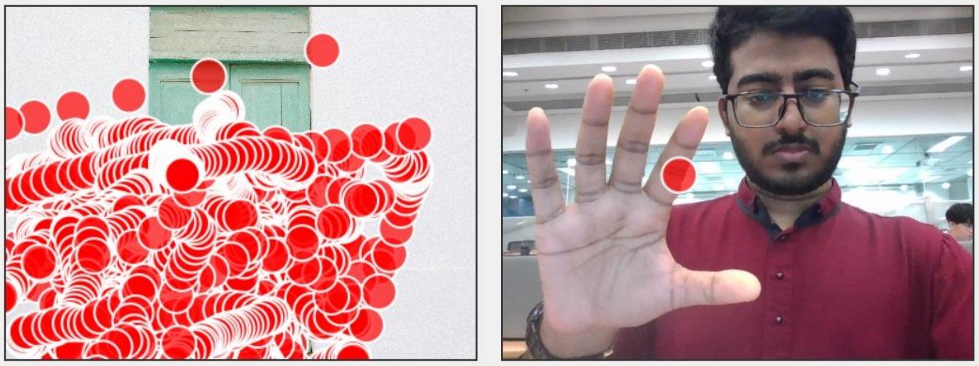
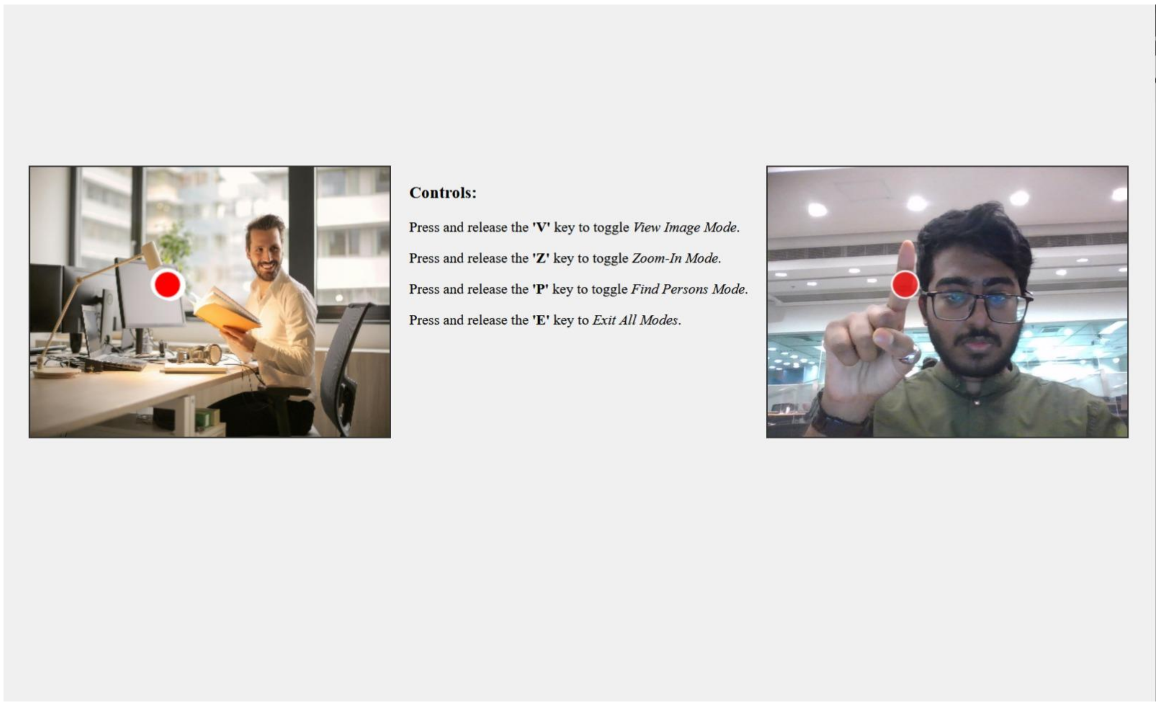


Figure: Improvised finger tracking

I have also documented that the red circle tracker was leaving traces of its movement on the corresponding image, making it look as if there are multiple indicators on the loaded image. I kept this on my mind and fixed the issue towards the end of the development of this program.



Controls:

Press and release the **'V'** key to toggle *View Image Mode*.

Press and release the **'Z'** key to toggle *Zoom-In Mode*.

Press and release the **'P'** key to toggle *Find Persons Mode*.

Press and release the **'E'** key to *Exit All Modes*.

Figure: Single red sphere corresponding to the live location of the index finger (tested with “myImage.jpg”, at a later stage of the development)

STAGE 4: IMPLEMENTING VIEW MODE (V KEY)

When first implementing the blur functionality in my Sketch.js file, I noticed considerable frame rate drops. The continuous redrawing of the entire blurred canvas was creating visible stuttering, especially when moving the index finger rapidly across the screen.

To solve this problem, I developed a two-layer approach: Rather than repeatedly applying blur filters, I maintained a constantly blurred base layer with a dynamic clear rectangle. This significantly improved performance while maintaining the desired visual effect. The clear rectangle now smoothly follows the index finger position, creating a “**window**” into the unblurred image.

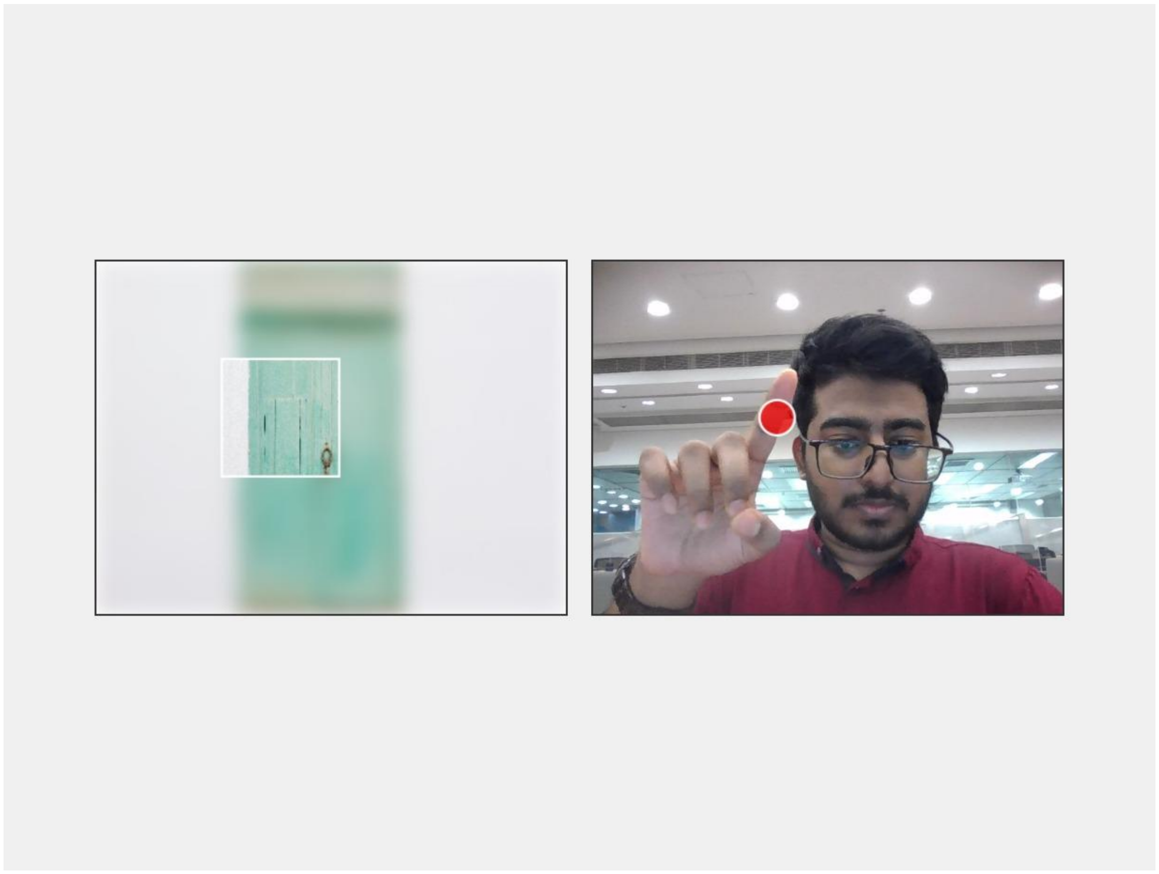


Figure: Blur effect with clear rectangle following finger (tested on the "replaced_image.jpg")

STAGE 5: DEVELOPING ZOOM FUNCTIONALITY (Z KEY)

To develop the zoom functionality that would be induced by clicking the z button, I implemented a dual-point tracking system in my sketch.js program, as outlined in the assignment instructions.

I introduced separate tracking indicators - **red for index finger** and **blue for thumb finger**. The zoom functionality, was then activated by **pressing the "z" button**, and it works by simultaneously tracking both the index and thumb finger positions in real-time.

The core function (drawZoomedImage) determines the midpoint between the two fingers and measures their separation distance. This distance is then used to calculate the zoom level through the **formula currentZoom = 1 + (distance * zoomSensitivity),** where the zoom sensitivity is a small constant (0.002) that controls how quickly the zoom responds to finger movement.

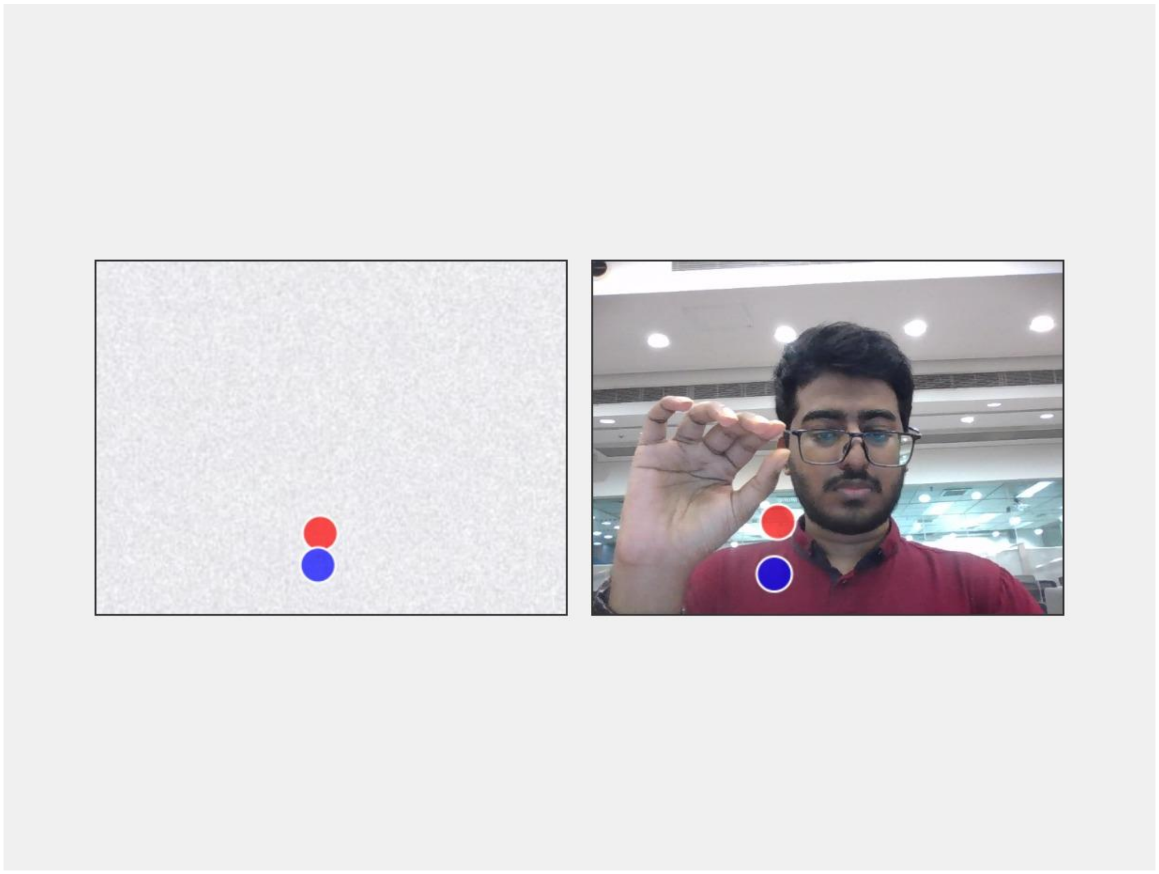


Figure: Demonstration of the "Z" mode (tested on the "replaced_image.jpg")

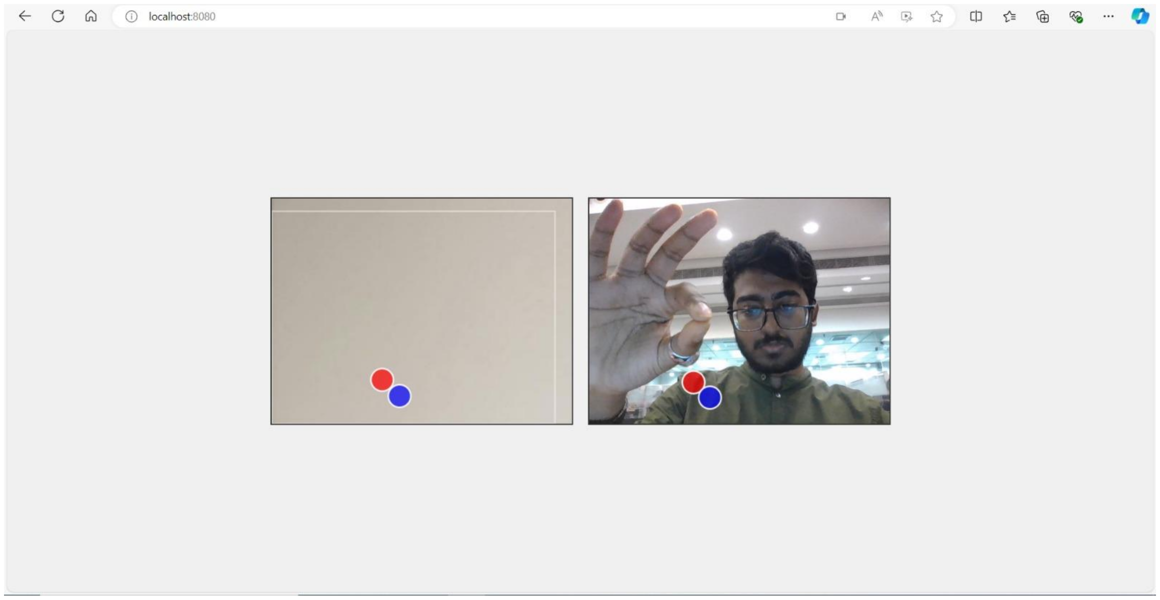


Figure: Demonstration of the "Z" mode (tested with "myImage.jp")

To maintain context, the system displays a faded version (0.3 opacity) of the full image in the background while showing the zoomed portion at full opacity. The system is bounded by minimum (1x) and maximum (2x) zoom levels to prevent excessive magnification, and includes bounds checking to ensure the zoomed view never extends beyond the image edges.

STAGE 6: EXIT FUNCTIONALITY IMPLEMENTATION (E KEY)

At this stage, my implementation could toggle modes on and off (by pressing the v or z button). Then, I wrote codes for the development of a comprehensive exit protocol that properly clears all active states and returns the image canvas to its default configuration when the “e” button on the keyboard is pressed.

This straightforward solution ensures users can reliably reset the system at any point during interactions.

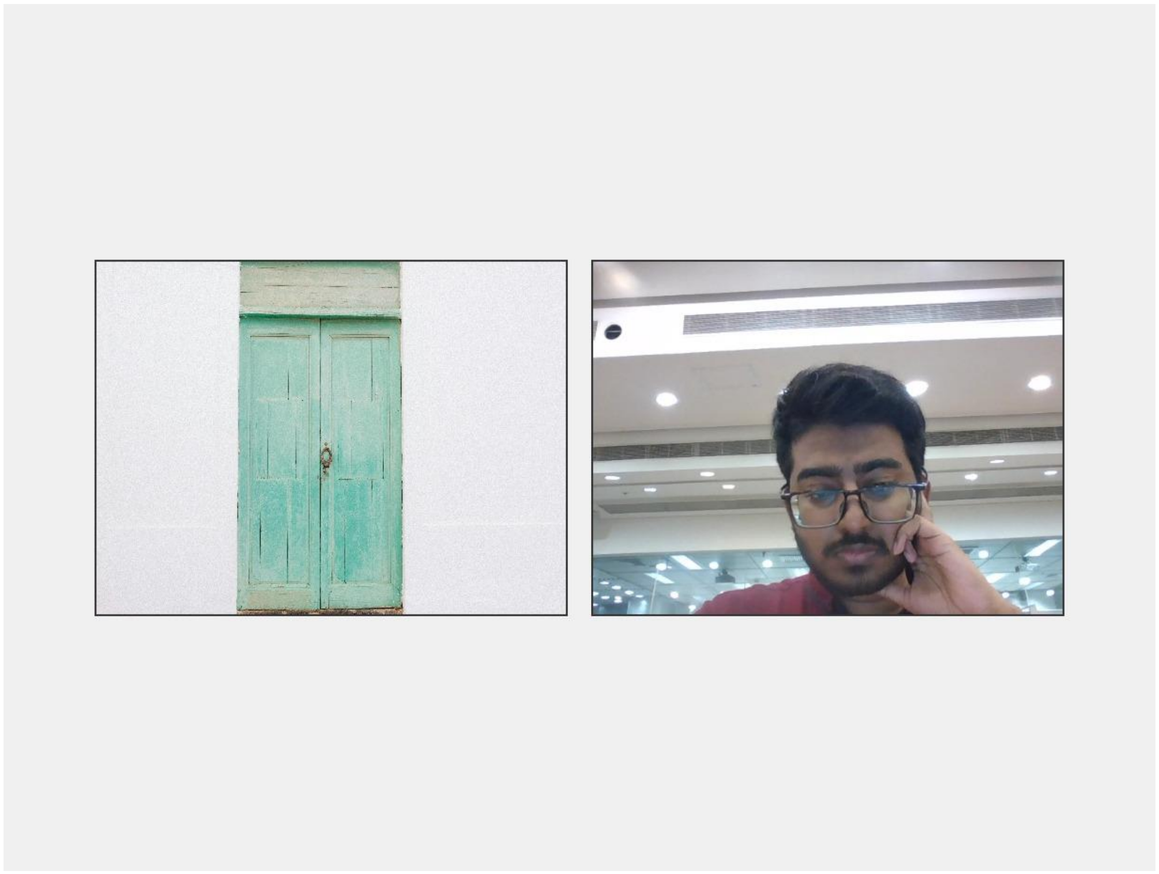
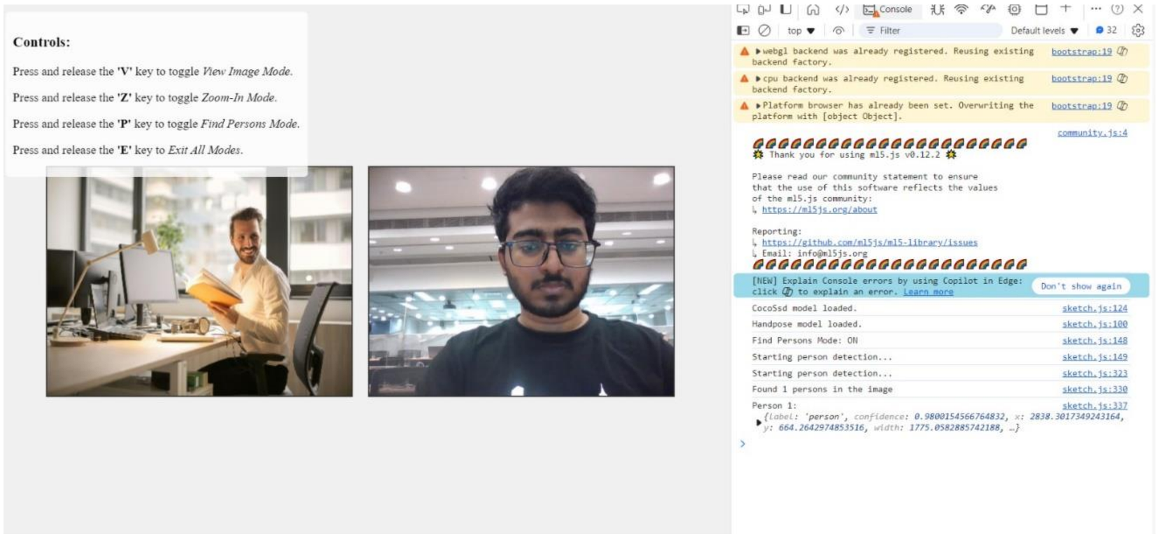


Figure: Demonstration of a smooth blur-to-clear transition (using the “replaced_image.jpg”)

STAGE 7: PERSON DETECTION IMPLEMENTATION (P KEY), ENLARGEMENT, & PAGE LAYOUT

In this final stage of the program, I wrote a function that could load the COCOSSD model and detect a potential person on the left canvas. To test my implementation, I first needed to replace my test image (green door) with an image containing people.

During this process, I also realized that users had no way of knowing the available controls without prior instruction. To address this usability concern, I implemented an instructions panel using HTML. My first attempt involved adding a simple div element with control descriptions.



Controls:

Press and release the **'V'** key to toggle *View Image Mode*.

Press and release the **'Z'** key to toggle *Zoom-In Mode*.

Press and release the **'P'** key to toggle *Find Persons Mode*.

Press and release the **'E'** key to *Exit All Modes*.

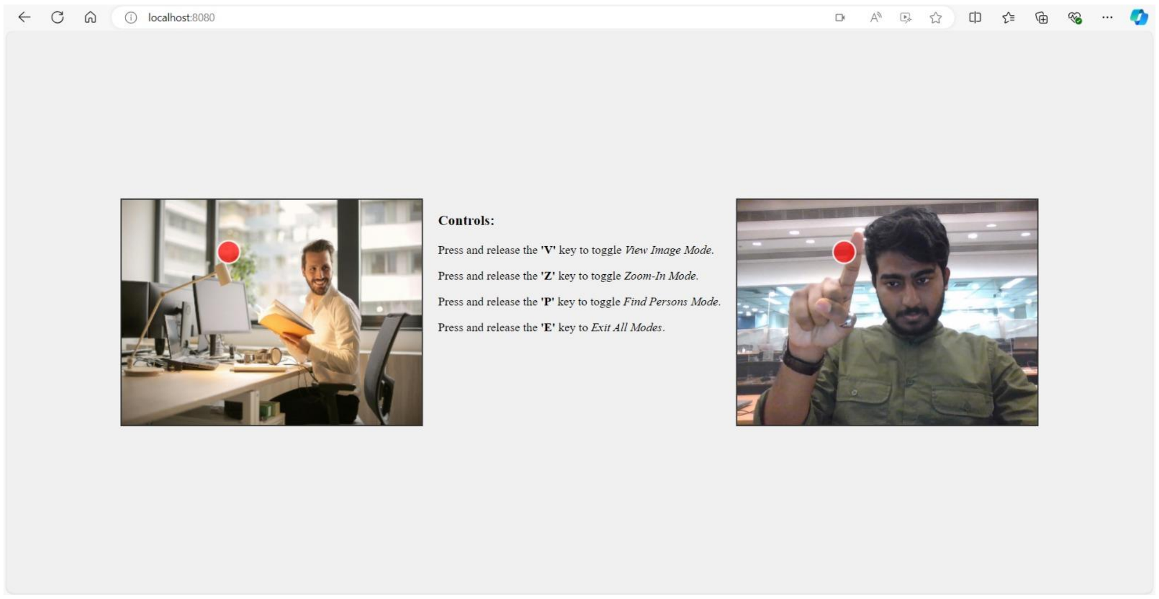
```
▶webgl backend was already registered. Reusing existing    bootstrap:19
backend factory.
▶cpu backend was already registered. Reusing existing      bootstrap:19
backend factory.
▶Platform browser has already been set. Overwriting the    bootstrap:19
platform with [object Object].
                                                           community.js:4
🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈
🌟 Thank you for using ml5.js v0.12.2 🌟

Please read our community statement to ensure
that the use of this software reflects the values
of the ml5.js community:
↳ https://ml5js.org/about

Reporting:
↳ https://github.com/ml5js/ml5-library/issues
↳ Email: info@ml5js.org
🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈
[NEW] Explain Console errors by using Copilot in Edge:
click to explain an error. Learn more
CocoSsd model loaded.                                      sketch.js:124
Handpose model loaded.                                     sketch.js:100
Find Persons Mode: ON                                      sketch.js:148
Starting person detection...                               sketch.js:149
Starting person detection...                               sketch.js:323
Found 1 persons in the image                               sketch.js:330
Person 1:                                                  sketch.js:337
▶{label: 'person', confidence: 0.9800154566764832, x: 2838.3017349243164,
  y: 664.2642974853516, width: 1775.0582885742188, …}
```

Figure: Successful detection of a person & instruction panel (tested on the "myImage.jpg")

However, this implementation had an unexpected layout issue: The instructions panel created a positioning conflict, overlapping with the image canvas.

To solve this problem, the program required a careful restructuring of the CSS layout. Rather than allowing the instructions to float freely, I implemented a flex-based positioning system. By modifying the container CSS to use flex-box properties and adding specific positioning rules, I created a clean layout where the instructions naturally fit between the two canvases without interfering with their functionality.

The result was a much more intuitive user interface that maintained the system's visual clarity while providing necessary guidance.



Controls:

Press and release the **'V'** key to toggle *View Image Mode*.

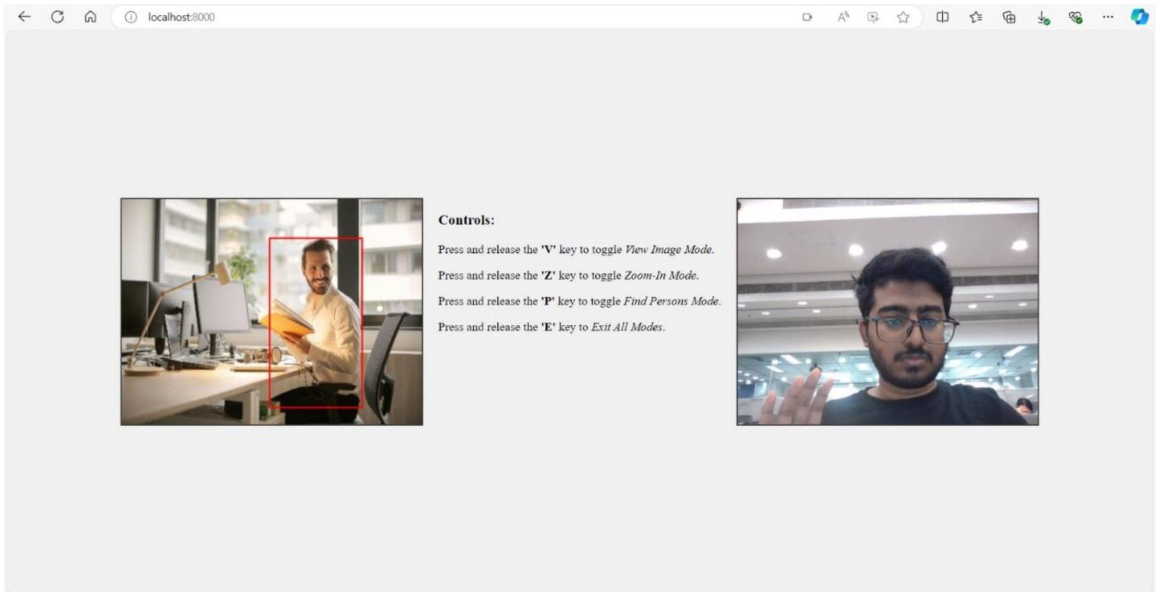
Press and release the **'Z'** key to toggle *Zoom-In Mode*.

Press and release the **'P'** key to toggle *Find Persons Mode*.

Press and release the **'E'** key to *Exit All Modes*.

Figure: Clean, responsive alignment of the "Controls" panel

Moving forward, I wrote code to ensure that a red rectangle was drawn on the detected person-object during the P mode (when the system detects the movement of the index finger). The model was successfully loaded and the person was also detected on the image.



Controls:

Press and release the **'V'** key to toggle *View Image Mode*.

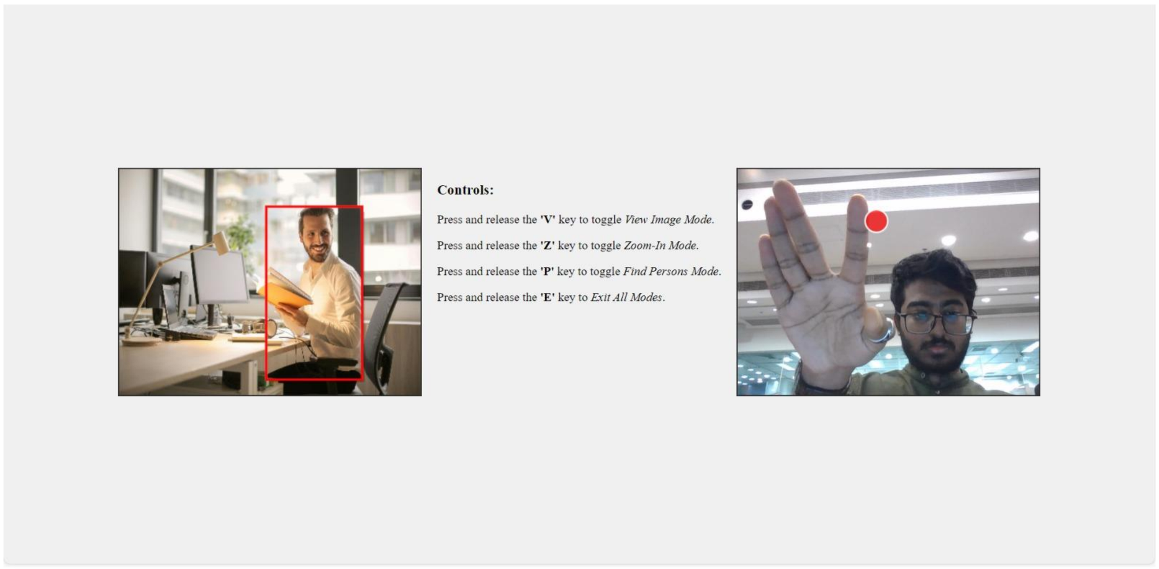
Press and release the **'Z'** key to toggle *Zoom-In Mode*.

Press and release the **'P'** key to toggle *Find Persons Mode*.

Press and release the **'E'** key to *Exit All Modes*.

Figure: When the P button is pressed, a red-rectangle is drawn over the detected person (after the index finger appears on the live camera feed).

Furthermore, I had to write additional codes to ensure that the person being detected was suitably enlarged, such that the enlargement is clearly visible. I was a bit creative with this process, and made the program such that when the index finger is toggled on the live camera feed, the person is either enlarged, or restored back to the original condition.



Controls:

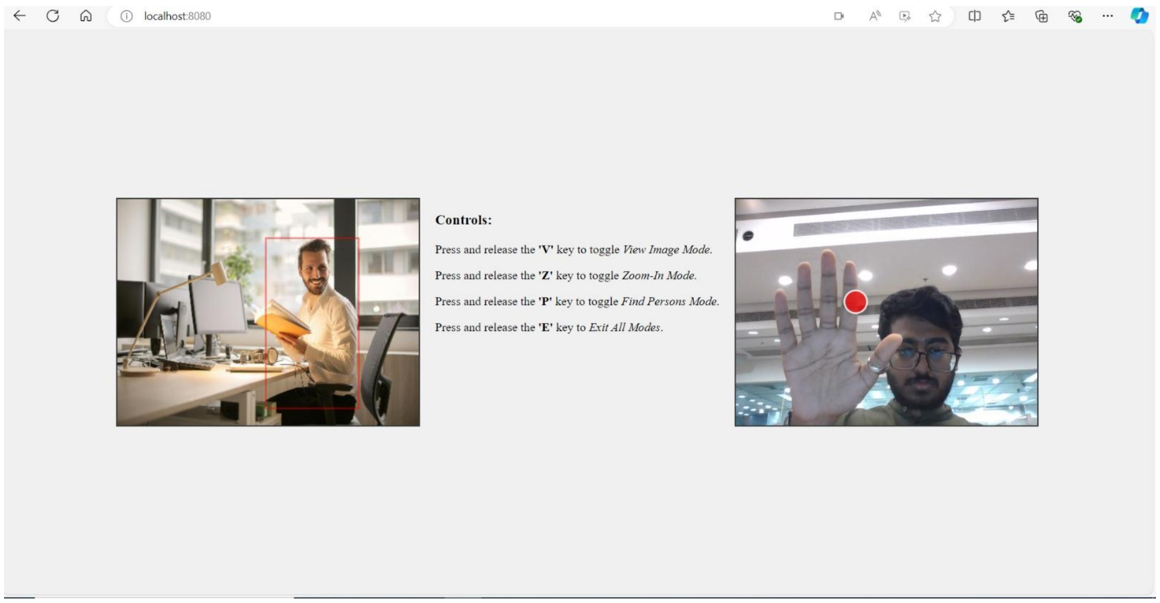
Press and release the **'V'** key to toggle *View Image Mode.*

Press and release the **'Z'** key to toggle *Zoom-In Mode.*

Press and release the **'P'** key to toggle *Find Persons Mode.*

Press and release the **'E'** key to *Exit All Modes.*

Figure: Demonstration of the person being enlarged as I toggle my index finger during the "P" mode (tested with "myImage.jpg")



Controls:

Press and release the **'V'** key to toggle *View Image Mode.*

Press and release the **'Z'** key to toggle *Zoom-In Mode.*

Press and release the **'P'** key to toggle *Find Persons Mode.*

Press and release the **'E'** key to *Exit All Modes.*

Figure: Restoration of the enlarged person as I toggle my index finger towards the other direction during the "P" mode (tested with "myImage.jpg")

Challenges and Limitations

Throughout the development of this program, some limitations became apparent:

- **Accuracy of index-finger tracking**

The hand detection accuracy varies based on lighting conditions and camera quality. Moreover, in low-light situations, the system struggles to maintain consistent tracking.

- **Browser Compatibility:**

Different browsers handle the WebGL acceleration differently, leading to inconsistent performance across platforms. Currently, the best way to demonstrate the application seems to be by hosting a local server and running the project directory over it.

Possible Improvements

Based on these experiences, I've identified several key areas for improvement:

- **More accurate hand-tracking:**

We can add preprocessing steps to normalize lighting conditions in the video feed. Techniques such as histogram equalization or adaptive thresholding can possibly enhance image clarity in low-light environments. Moreover, we could also leverage more advanced machine learning models, such as MediaPipe Hands by Google, which are designed to handle varying lighting conditions and camera qualities more effectively.

- **Performance Optimization:**

We can conduct comprehensive testing on major browsers (e.g., Chrome, Firefox, Edge, Safari) to identify discrepancies in WebGL performance. We can also use browser-specific debugging tools to fine-tune WebGL settings for consistency.

- **Enhanced User Feedback:**

We could also be adding more visual guides and status indicators would help users better understand system state and available interactions.

Conclusion

This project required a significant amount of work and individual research. I put a lot of labor behind the creation of the program and its documentation.

I believe that my implementation successfully demonstrates the potential of browser-based hand gesture interaction systems while also highlighting areas for future improvement.

In my observation, the combination of ml5.js models with optimized rendering techniques integrated with p5.js, provides an interesting foundation for further development of such machine learning multimodal interfaces.

ENDING-NOTE:

I am grateful to the professor for giving me this wonderful opportunity to develop this fascinating project, and I look forward to the thoughtful review and approval of my report.