V-Glove: Computer-vision Based Budget Motion Capture Gloves

Xiao Hang Wang, Rafeeq Shodeinde

1 Introduction

Motion capture gloves have become an essential tool for natural interaction in VR environments, enabling tracking of hand movements and gestures. However, traditional motion capture systems often rely on specialized hardware, such as electromagnetic sensors, which can be costly and restricted to specific platforms. Consequently, the specialized hardware further makes motion capture gloves costly and limits their use to certain VR systems.

Additionally, the existing Computer-vision approaches for tracking hands in VR, such as the hand-tracking function on Meta Quest 2, Meta Quest 3, and Apple Vision Pro, require relatively powerful hardware and more than one camera. Moreover, the software implementations are usually restricted to closed software ecosystems, making it hard to deploy to budget platforms.

Thus, we introduce V-Glove, a new low-cost way to capture hand motions for VR using just computer vision techniques. V-Glove uses only regular cameras and advanced computer vision algorithms to accurately track hand movements in real-time, without needing any specialized motion capture hardware.

2 Advantages

The motivation of the project is to improve the accessibility and portability of existing Computer-vision based hand tracking.

Accessibility: V-Glove requires only one regular camera, such that the users can make use of it even with a webcam camera. Thus, the manufacturing cost of VR equipment using V-Glove can be significantly reduced, improving accessibility.

Portability: For technical or business reasons, most existing computer-vision based hand tracking devices are restricted to a closed ecosystem, preventing them from being further ported to emerging platforms such as mobile phones. V-Glove on the other hand uses open-source cross-platform computer graphics libraries (such as OpenCV) and is developed in Python, making it relatively easy to be ported to other platforms, improving portability.

3 Design

The project follows the below flowchart:

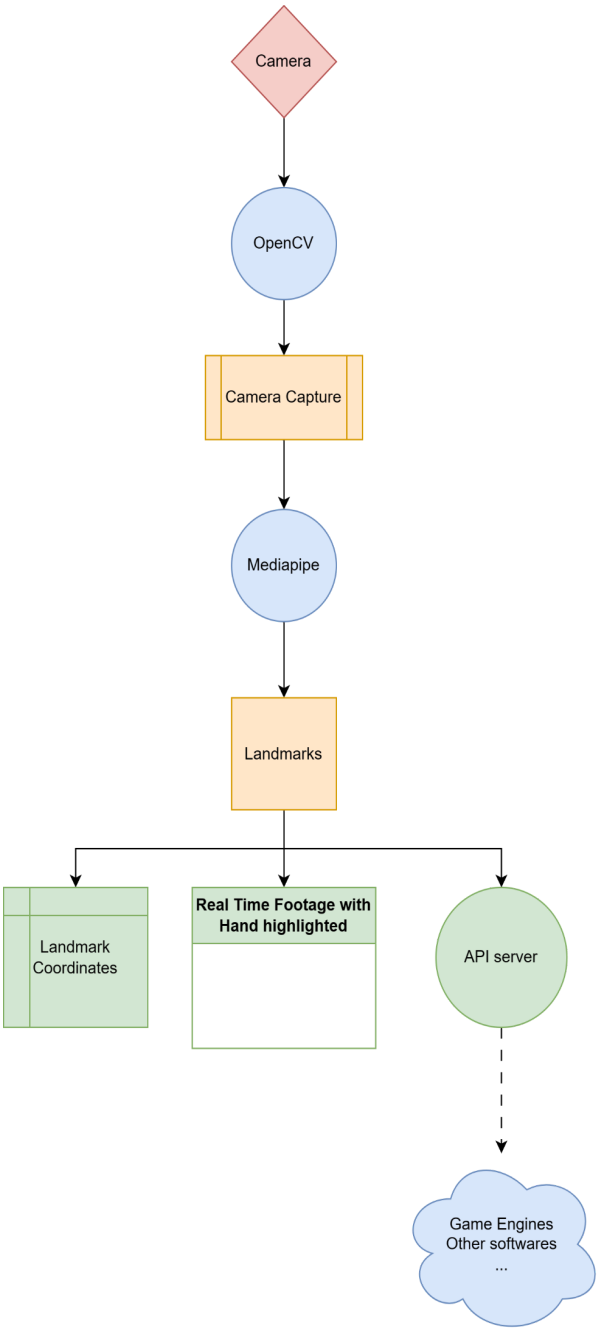

Figure 3.1 *Flowchart*

Hardware is highlighted in red, external libraries are highlighted in blue, data is highlighted in orange, and resulting modules are highlighted in green.

Firstly, the OpenCV (Open Source Computer Vision Library) ***[1]*** library will act as an abstraction layer between hardware (camera) and software, which is an open-source computer vision library used for various tasks such as image and video processing. It is responsible for collecting camera video and preprocessing it for further use.

Next, the Mediapipe ***[2]*** library will detect hands in images and get landmarks for further use. It is a machine learning library developed by Google that consists of two steps. The first step is to detect the palm of the hand, which is done using a palm detection model that operates on the full input image. After that, it returns an oriented hand bounding box, that is then used to define a cropped region of the image. The second step is to get hand landmarks from the cropped image region. The machine learning model shall return high-fidelity 3D hand key points, and those key points, also known as landmarks, are used to identify specific points on the hand, such as the tips of the fingers or the center of the palm.

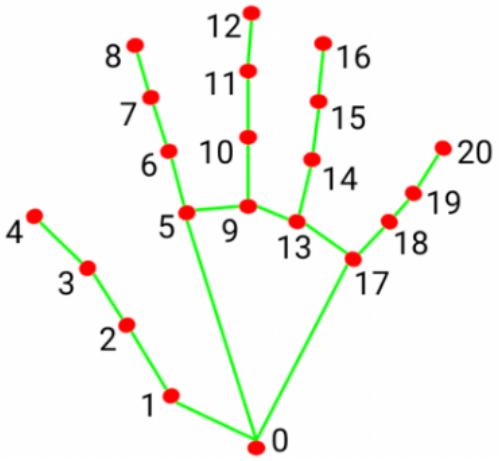

Figure 3.2 *Landmarks*

Moving on, after obtaining the landmarks of each frame, the Real Time Footage with Hand Highlighted module will highlight each landmark on the real-time footage according to its coordinates. Then connect

each landmark together according to appropriate logic, and finally obtain the hand-highlighted footage.

Additionally, another window that includes the actual coordinates of each landmark will also be displayed and updated for each frame.

Lastly, an API server will provide the coordinates of each landmark for each frame, so that other external software like game engines can make use of them to get the position and movement of the hand.

Figure 3.3 *Imaginary graphical user interface design*

4 Project Update Presentation Timeline

March 27th	**Milestone:** Have the development environment ready. **Presentation:** Introducing the project and our approach.
April 3rd	**Milestone:** Make sure the project flow works, and have some preliminary results from each module. **Presentation:** Show the preliminary results if possible.
April 10th	**Milestone:** Have the prototype ready. **Presentation:** Show the prototype if possible.
April 17th	**Milestone:** Have the graphical user interface ready. **Presentation:** Show the real-time footage with hand highlighted.
April 24th	**Presentation:** Final Project Demo.

5 References

[1] Bradski, G., & Kaehler, A. (2000). OpenCV. Dr. Dobb's journal of software tools, 3(2).

[2] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172.