Advances in Engineering Innovation Vol.16 Issue 9 EWA Publishing
Available Online: 24 September 2025 DOI: 10.54254/2977-3903/2025.27230

Design of an edge computing-based motion capture and model-driven system

Qidou Li

School of Social Science, Soochow University, Suzhou, China

2303408042@stu.suda.edu.cn

Abstract. Motion capture has become a core technology for applications involving virtual digital humans, such as virtual streamers (VTubers). This paper proposes a motion-capture and model-driven system that relies on keypoint detection performed on edge-computing devices. The system implements independent keypoint detection for multiple body parts and parallel real-time processing on edge devices, enabling real-time keypoint detection of the body, hands, and face. By offloading tasks to edge devices, it reduces the resource footprint of motion capture and lowers deployment costs. Experimental results show that employing edge-computing devices can significantly reduce device load and enable deployment across a variety of platforms.

Keywords: edge computing, computer vision, multi-camera system, task offloading, MaixCAM

1. Introduction

1.1. Background of system design

With the rapid development of technologies such as Virtual Reality (VR) [1], Augmented Reality (AR), and the metaverse [2], the application scenarios of virtual digital humans have become increasingly widespread, particularly in areas such as virtual streamers (VTubers) [3], animation production [4], and visual effects in film and television [5]. However, motion-capture technology remains a critical bottleneck restricting the further advancement of these applications. Traditional motion-capture methods mainly include mechanical, optical, and video-based approaches. Mechanical and optical motion capture provide high accuracy, but they rely on wearable hardware and specific environmental conditions, which lead to high deployment costs, centralized workflows, and certain user constraints. Video-based motion capture is less costly to deploy, but in practice it faces challenges such as high implementation complexity for independent algorithms, insufficient accuracy, reliance on high-performance computing devices, excessive interdependence among subsystems, and a lack of decoupling.

To address these issues, this paper proposes a motion-capture and model-driven system for keypoint detection based on the collaboration of multiple edge-computing devices. In this system, lightweight convolutional neural networks are deployed on multiple edge devices, each independently performing real-time keypoint detection for the human body, hands, and face. A central controller then integrates the data and drives the model, thereby enabling task offloading [6] from the central node to edge devices and decoupling different functional components.

1.2. Overall system design and workflow

The system consists of the following main modules: Human Information Capture and Processing Edge Module: Responsible for acquiring human body image data and performing keypoint detection.Facial Information Capture and Processing Edge Module: Responsible for acquiring facial image data and performing keypoint detection. Hand Information Capture and Processing Edge Module: Responsible for acquiring hand image data and performing keypoint detection. Network Information Exchange Module: Handles data transmission between edge devices and the central controller. Model-Driven Central Control Module: Responsible for receiving and integrating keypoint data from all components and driving the articulated model.

Each edge-computing capture and processing module first acquires the target's image data and performs keypoint detection, converting keypoints into two-dimensional coordinate data. The 2D coordinates of each body part are then packaged into JSON data packets and transmitted via the TCP protocol to the Network Information Exchange Module. The Network Information Exchange Module continuously monitors the ports of the edge devices in real time, receives the data packets, and parses their contents to extract information such as device ID, timestamp, keypoint names, and 2D coordinates. It then integrates the information from all data packets and repackages them into a unified JSON data packet, which serves as the data source for the Model-Driven Central Control Module. The Model-Driven Central Control Module reads the unified data packet and polls for updates. The retrieved 2D coordinates are mapped, via scripts, to the coordinate system of a Blender scene. Through axis remapping, the Y-axis data of the 2D coordinates is reassigned to the Z-axis, and the original X-axis data is reassigned to the Y-axis, thereby achieving a more intuitive perspective: observing the Z–Y plane from the positive half-axis of the X-axis. The system framework is shown in Figure 1, which illustrates the data flow and interactions among the modules.

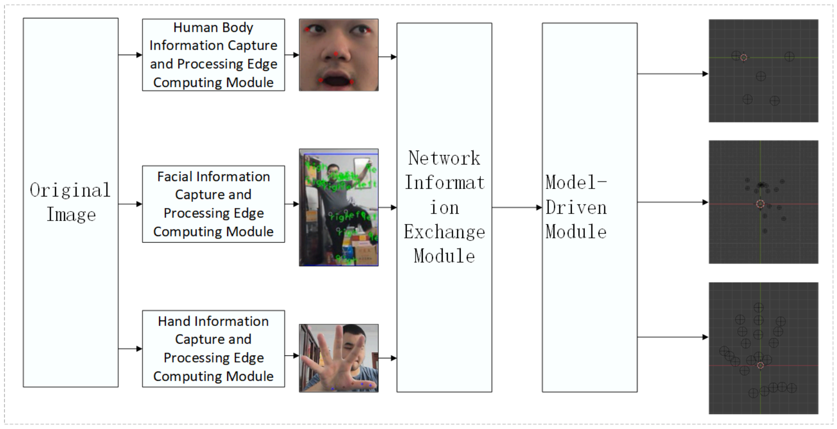
Original Image
Human Body Information Capture and Processing Edge Computing Module
Facial Information Capture and Processing Edge Computing Module
Hand Information Capture and Processing Edge Computing Module
Network Information Exchange Module
Model-Driven Module

Figure 1. Framework of the motion-capture and model-driven system based on edge computing

2. Design of functional modules

2.1. Selection of edge-computing devices

In this design, the edge-computing devices are required to capture image data; locally deploy convolutional neural networks for object detection to process the input images in real time; process the extracted keypoint information and package it into a readable format; transmit the packaged keypoint data over the network to the model-driven central control module; and, where possible, minimize the use of computer I/O resources.

The MaixCAM-Pro employs the Sipeed SG2002 processor, which integrates a 1 GHz RISC-V C906 core, a 1 GHz ARM Cortex-A53 core, and a 700 MHz RISC-V C906 core. In addition, the SG2002 includes an NPU core with a computational capacity of 1 TOPS at INT8 precision, capable of deploying common models such as YOLO. Its onboard Wi-Fi 6 module supports wireless LAN file transfer. The device is compatible with the MaixPy Python development package, MaixVision IDE, and MaixHub, offering comprehensive documentation and ecosystem support that facilitate efficient model training and deployment. Given the practical requirements of model deployment and data transmission, the MaixCAM-Pro is a reasonable choice.

2.2. Edge modules for capturing and processing human, facial, and hand information

To enable convenient system deployment and ensure real-time inference, the system adopts several mature solutions to handle keypoint detection for different components.

2.2.1. Human information capture and processing edge module

For human-body keypoint detection, the system employs the YOLOv8n_pose model. YOLOv8n_pose simultaneously predicts bounding boxes and keypoints, achieving efficient integration of detection and pose estimation tasks [7]. This design preserves the advantages of the YOLO framework, including end-to-end training and high inference speed. By incorporating architectures such as the C3k2 module and the C2PSA spatial attention mechanism, it enhances feature extraction capability, thereby improving both accuracy and efficiency on benchmark datasets such as COCO [8].

2.2.2. Facial information capture and processing edge module

For facial keypoint detection, the system uses the YOLOv8 model, which can rapidly and accurately detect multiple facial keypoints and output the two-dimensional coordinates of specified points.

2.2.3. Hand information capture and processing edge module

For hand keypoint detection, the system adopts a ported version of the MediaPipe framework. MediaPipe, developed by Google, provides a highly optimized machine learning pipeline. Its hand model employs a "detection–tracking" mechanism [9]: first, a lightweight palm detector localizes the palm region, and then 3D coordinate regression is performed for 21 hand keypoints within this region. Thanks to its lightweight design and efficient inference engine, the model achieves low-latency inference on edge devices.

With the integration of these three capture and processing edge modules, the system achieves human-structure detection while decoupling the mapping of different body parts.

2.3. Network information exchange module

The network information exchange module consists of a network switch and the data reception logic of the host controller. Each edge device functions as a client, while the host controller acts as the server. Devices use the TCP protocol to transmit detected keypoint data in JSON format—containing device ID, timestamp, keypoint name, and two-dimensional coordinates—to the designated port of the host controller. The host controller runs a multithreaded service, listening on different ports to receive data streams from the edge devices. The server then performs data validation, checks the validity of the JSON format, and filters out invalid coordinates.

To ensure reliability and real-time transmission, the system employs the TCP protocol, which guarantees the order and integrity of data packets. Each edge device attaches a device ID and timestamp to its data, enabling the host controller to correctly identify, classify, and synchronize keypoint data from different devices. Upon receiving the data, the host controller decodes and parses the packets, extracts keypoint information, and stores it in memory.

During reception, the host controller performs checksum validation and filtering. It checks the validity of the JSON structure to ensure correct data formatting. Invalid coordinates—usually representing detection failures or missing keypoints—are filtered out. Finally, the data are aligned by timestamps to ensure temporal consistency across all detected keypoints.

2.4. Model-driven host control module

The model-driven host control module integrates and maps the received keypoint data to drive the 3D model. After receiving data packets containing keypoints from different modules, the host controller performs update checks and parsing. Based on device ID and timestamps, it classifies, matches, and aligns the data from different sources. The parsed keypoint data are then converted into Blender space coordinates to animate the model.

This module maps keypoint data onto the skeleton of a Blender model to achieve real-time animation. Specifically, the system adopts the following methods depending on the type of model:

2.4.1. Visualization of keypoints

The system first inspects the two-dimensional coordinates in the data packet. For each valid coordinate, an empty object is created in Blender, with the 2D coordinates mapped into the 3D scene. The image width is mapped to the width of the 3D scene, and the image height to the height of the 3D scene. The X-coordinate in the 3D scene is set to a fixed depth value to distinguish the spatial layering of different body parts. In this way, 2D coordinates are mapped into Blender's 3D coordinate system.

2.4.2. 3D models with skeletal binding

After the visualization mapping is completed, models with skeletal rigs or controllers are driven through a bone-mapping dictionary that links keypoints to skeletal bones. The skeletal driving mechanism includes:

IK (Inverse Kinematics) Control: Primarily applied to hands and feet. Hand and foot keypoints are mapped to IK controller empties, which govern the overall positioning of hands and feet. The Y-coordinate of the 2D image is used to simulate Z-axis movement, enabling vertical motions of hands and feet. FK (Forward Kinematics) Control: Primarily applied to upper arms and thighs. By computing vector directions from shoulder to elbow and from hip to knee, the system estimates rotation angles of upper and lower limbs, thereby driving the corresponding bones. Facial Expression Control: Facial keypoints are used to calculate yaw and pitch angles of the head, driving the head bone accordingly. Variations in facial keypoints further drive facial expressions, enabling dynamic expression changes.

3. Implementation of system functional modules

3.1. Implementation of edge computing modules for information capture and processing

The main functionality of the system relies on the implementation of various edge computing modules responsible for information capture and processing. This section provides a detailed description of their implementation.

The module detects and processes human body keypoints and transmits the detected data in real time to the network information exchange module, which serves as the basis for subsequent behavior analysis. The implementation steps are as follows:

Load the YOLOv8n_pose model, configure camera parameters, and ensure compatibility with the model input requirements. Establish a network connection with the network information exchange module via TCP, specifying the appropriate port. Capture images and perform keypoint detection on the acquired frames. Extract the detected keypoint coordinates, encapsulate them into JSON-format data packets, and transmit them. This implementation mainly relies on the MaixPy components provided by SIPEED and its ported neural network models. Using the human body information capture and processing module as an example, the key code is as follows:

```
detector = nn.YOLOv8(model="/root/models/yolov8n_pose.mud", dual_buff=True)

cam=camera.Camera(detector.input_width(),detector.input_height(),detector.input_format
()) # Model initialization

SERVER_IP = "IP address of the network information exchange module"

SERVER_PORT = "Port for human body keypoint data transmission"

DEVICE_ID = "Device identifier for the human body capture edge computing module" #
Network configuration

data_to_send = {

"device_id": DEVICE_ID,

"timestamp": time.time(),

"person_score": float(person.score),

"keypoints": []

}

for i in range(17):

x = person.points[i * 2]

y = person.points[i * 2 + 1]

data_to_send["keypoints"].append({

"name": keypoint_names[i],

"x": float(x),

"y": float(y)

}) #Human body keypoint processing
```

3.2. Blender-based driving scheme and cross-platform flexibility

The system’s 3D visualization and driving module are implemented using the 3D creation suite Blender. The implementation is based on a Python script running inside Blender, functioning as a modal operator that continuously executes in the background at approximately 30 FPS. The main workflow is as follows: File Monitoring: The script continuously monitors a JSON file generated by Python integration, detecting updates. Data Reading and Parsing: Once an update is detected, the script reads the latest file, parses the JSON data, and extracts keypoint information for the face, body, and hands. Coordinate Mapping and Computation: According to predefined mapping rules, the script converts 2D image coordinates into pseudo-Blender spatial coordinates. This includes calculating head rotation angles, IK/FK parameters for body movement, and hand keypoint positions. Model Driving: The computed rotations and positions are applied to the model in the Blender scene, updating skeletal poses in real time. The loop then waits for the next frame, forming a smooth real-time driving cycle. Blender was chosen as the driving platform due to its cross-platform compatibility, with distributions available for multiple processor architectures and operating systems. This flexibility greatly reduces deployment barriers. Moreover, Blender's Python API provides high customizability: driving logic, coordinate mapping rules, and related components can be extended or adjusted through simple script modifications, enabling a cost-effective, flexible, and maintainable solution.

3.3. System performance and demonstration of final output

In actual operation, the system stably and in real time captures movements of the human body, face, and hands. Experimental results show that the system achieves accurate keypoint detection for body limbs, hands, and facial features at a frame rate of 30 FPS.

3.4. Final system demonstration

The final system consists of three MaixCAM devices mounted on tripods, as shown in Figure 2.

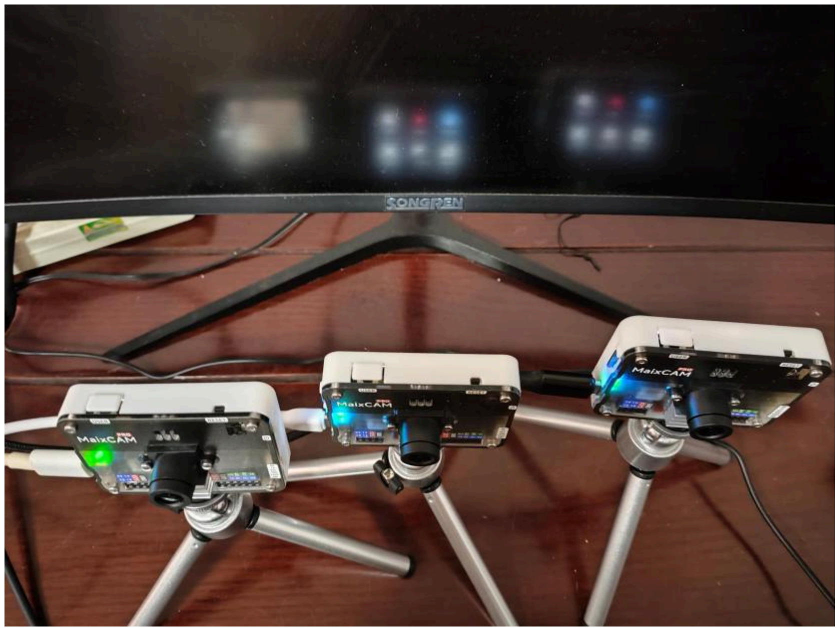

Figure 2. Appearance of the system

4. Conclusion

This paper proposes a keypoint-detection-based motion capture and model-driving system built on the collaboration of edge computing devices. By deploying lightweight convolutional neural networks on multiple edge devices, the system enables real-time detection of keypoints for human body limbs, hands, and facial features. Through data integration and model driving by the central controller, the system effectively reduces device load and supports diversified platform deployment. Experimental results demonstrate that the system can stably and in real time capture human movements, showing promising potential for applications in areas such as virtual anchoring, animation production, and human–computer interaction. Nevertheless, several limitations remain. For instance, the two-dimensional nature of keypoints restricts broader application scenarios, and different models require coordinate axis conversions. Future research will focus on addressing these issues to further enhance the system's performance and expand its application scope.

References

[1] Li, C. (2017). Research on human–computer interaction technology in virtual reality systems (Master's thesis). Zhejiang University, Hangzhou, China.

[2] Feng, X. (2022). Challenges and countermeasure suggestions for the development of virtual digital humans under the background of the metaverse. *Cultural Industry*, (36), 19–21.

[3] Wang, J. (2023). Research on the application of virtual digital human technology in the field of media. *Modern Television Technology*, (4), 102–105.

[4] Hu, X. (2005). Theoretical research on 3D animation and virtual reality technology (Doctoral dissertation). Wuhan University, Wuhan, China.

[5] Sharma, S., Verma, S., Kumar, M., et al. (2019). Use of motion capture in 3D animation: Motion capture systems, challenges, and recent trends. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 289–294). Faridabad, India: IEEE. https: //doi.org/10.1109/COMITCon.2019.8862448

[6] Zhao, Z., Liu, F., Cai, Z., et al. (2018). Edge computing: Platforms, applications, and challenges. *Journal of Computer Research and Development*, 55(2), 327–337.

[7] Maji, D., Nagori, S., Mathew, M., et al. (2022, April 14). YOLO-Pose: Enhancing YOLO for multi-person pose estimation using object keypoint similarity loss [Preprint]. arXiv. https: //arxiv.org/abs/2204.06806

[8] Khanam, R., & Hussain, M. (2024, October 23). YOLOv11: An overview of the key architectural enhancements [Preprint]. arXiv. https: //arxiv.org/abs/2410.17725

[9] Zhang, F., Bazarevsky, V., Vakunov, A., et al. (2020, June 18). MediaPipe Hands: On-device real-time hand tracking [Preprint]. arXiv. https: //arxiv.org/abs/2006.10214