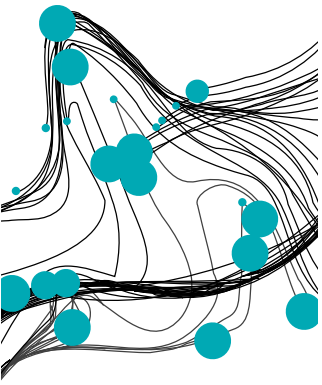
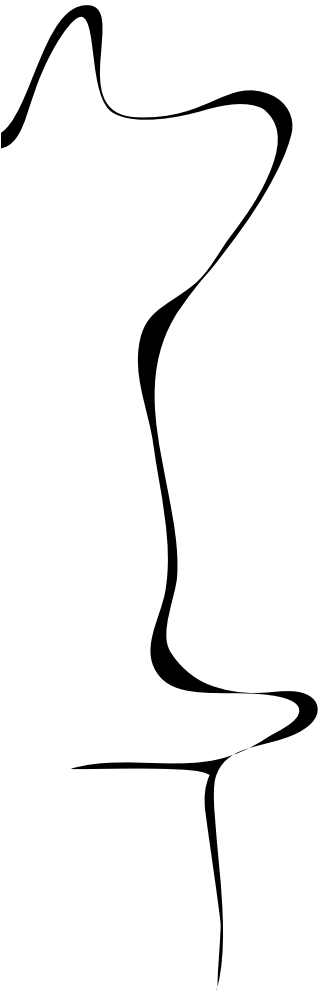


RAM ROBOTICS AND MECHATRONICS



AR-MOCAP: AUGMENTED REALITY MOTION TRACKING SYSTEM

T. (Tongli) Zhu

MSC ASSIGNMENT

Committee:
dr. I.S.M. Khalil
B. Lan, MSc
dr. ir. K. Niu
dr. I. Tamadon
prof. dr. ir. N.J.J. Verdonschot

June, 2025

030RaM2025
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

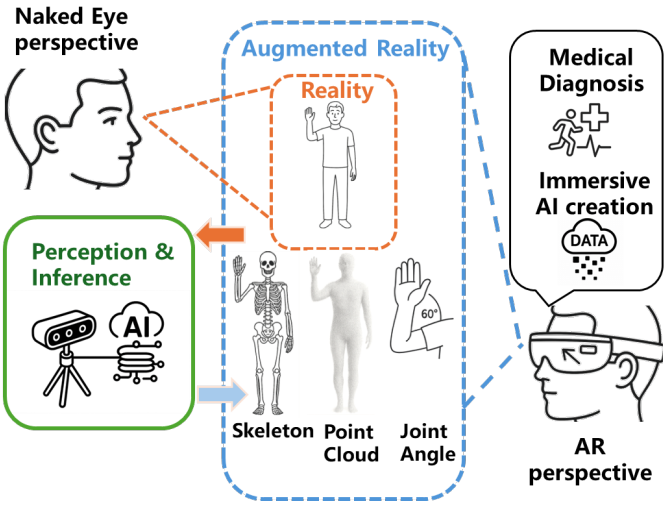
UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

AR-MOCAP: AUGMENTED REALITY MOTION TRACKING SYSTEM

Tongli Zhu

***Abstract*— Motion tracking is critical for applications in sports science, clinical diagnostics, and data acquisition for the AI model. However, traditional methods require costly devices, complicated wearable markers, and controlled environments, which limit their accessibility in reality. This study proposes a lightweight real-time motion tracking system using a head-mounted display (HMD) with visual overlay in the physical environment. Only with a simple calibration, the full-body motion capturing and point cloud visualization could be achieved, with a built-in joint angle calculation for analyzing the subject's kinematics. The system was evaluated for stability and latency, while the tracking kinematics was evaluated for accuracy, continuity and consistency. The experimental results showed the reliability of both the system design and the tracking results. Overall, the proposed system offers a portable, convenient, and efficient motion tracking solution. It shows the potential for both the clinical kinematic diagnostics and conventional motion-tracking applications that require flexible, changing viewing perspectives.**

***Index Terms*— augmented reality, motion tracking, kinematics analysis, point cloud, system design**



I. INTRODUCTION

MOTION tracking is critical for analyzing human biomechanics and movement. It was widely used in sports science [1], clinical diagnostics [2], [3], and artificial intelligence [4]. By analyzing body posture and joint angles recorded during movement, the biomechanical patterns could be extracted for rehabilitation planning and athletic performance optimization. It also serves as a valuable input for training motion-capturing deep-learning models [5].

Previous work in motion tracking adopted ultrasound, tracking markers, IMU, and optical systems. A non-invasive bone tracking method from [6], [7] achieved 2–5 mm precision via 30 A-mode ultrasound transducers, while the active tracking systems using markers like Codamotion [8], or IMU-based MyoMotion (14 sensors, 2.4 GHz link) [9] could reliably monitor the limb kinematics in clinical and gait-analysis settings. Optical setups such as OptiTrack's eight 240 Hz infrared cameras with 57 reflective markers [10] or Vicon's ten-camera rigs with 40–60 markers [11] show good tracking results for the gold standard of deep-learning models' training and evaluation. However, these solutions rely on the costly hardware ($30–50 K), extensive calibration, multi-camera arrays, controlled operating spaces, and offline software, which highly limits their accessibility and real-time utility.

Augmented Reality (AR), characterized by its spatial visualization, interactivity, and real-time feedback capabilities, has been widely applied across various fields. An AR-based digital twin system [12], [13] enabled precise MR-guided breast biopsy via remote robot control, while AR-assisted screw placement [14] improved targeting accuracy in spinal surgery. Building on these strengths, AR also offers great potential in motion tracking, which provides an alternative and convenient low-cost perception solution for motion tracking. It integrates motion data with visual interfaces in real time to enhance human-computer interaction. As a lightweight device, AR provides instant feedback and analysis of the subject's visualization when changing perspective. All these features make AR a critical component to integrate into motion tracking for improved functionality.

Several AR-based motion tracking methods were proposed to enable efficient and real-time human motion capture on mobile and head-mounted devices. A recursive tracking method was introduced in [16] to optimize for resource-constrained mobile platforms, yet it only follows 2D image-plane feature points and does not model multiple anatomical keypoints or full 3D body posture. To achieve real-time fingertip tracking, [17] combined particle filtering with level-set contour evolution on the Samsung Galaxy Note N8010. Nikodem et al. [18] evaluated six-degrees-of-freedom hand tracking on Microsoft HoloLens 2 under various head movements, but reported unstable head-mounted support and coordinate-system drift, lacking the accuracy needed for high-precision medical applications. A real-time marker-based system on HoloLens 2 was developed in [19] to have sub-2° accuracy

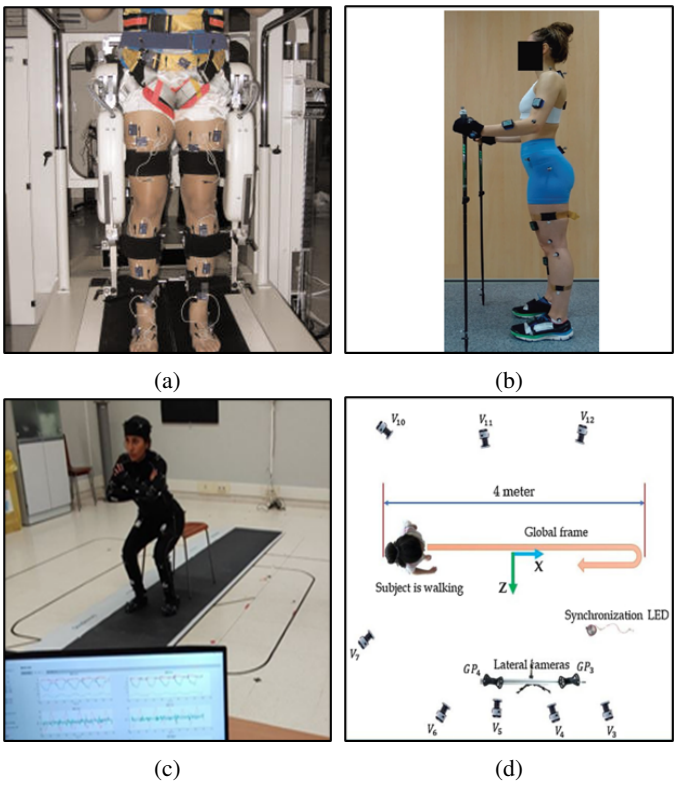


Fig. 1: Typical motion tracking methods: (a) Codamotion system [15], (b) IMU-based MyoMotion system [9], (c) OptiTrack optical motion capture system [10], and (d) Vicon motion capture system [11].

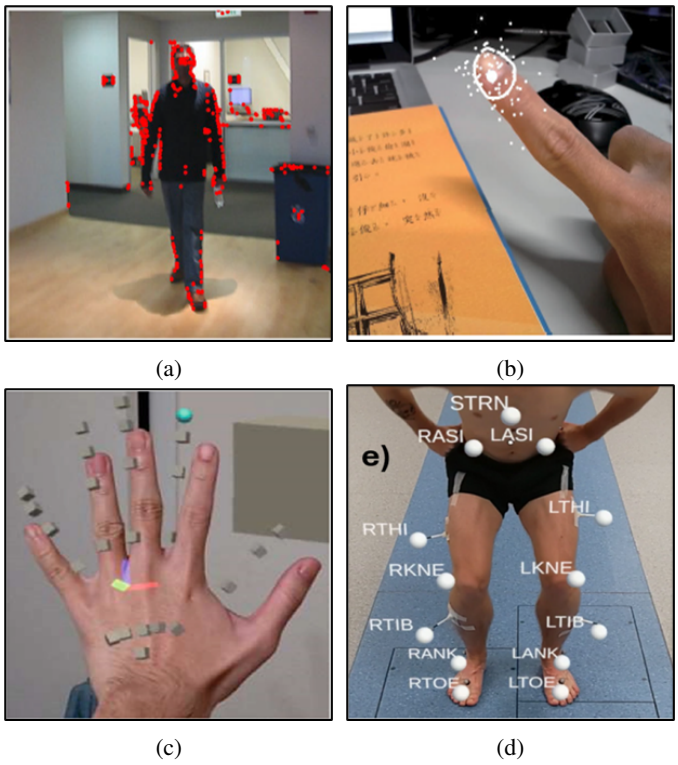


Fig. 2: State-of-the-art AR motion tracking examples: (a) planar tracking based on corner points [16], (b) fingertip tracking on the Samsung N8010 [17], (c) finger tracking on HoloLens 2 [18], and (d) low-speed lower limb tracking on HoloLens 2 [19].

for rehabilitation movements; however, it relies on reflective markers, supports only front-facing tracking, and was limited to very slow movements due to its fixed 5 fps frame rate. In summary, these methods are constrained to low-dimensional 2D tracking, external markers, and suffer from sensor drift and distance limitation, and lack the spatial and temporal resolution required for comprehensive, high-speed 3D human body tracking.

To address these challenges, an AR-based, markerless, real-time motion-tracking solution was proposed in the study for analyzing full-body 3D kinematics. The designed system fuses depth streams from two Intel RealSense cameras, with joint positions inferred by a YOLOv11 model running on a backend laptop. The reconstructed skeleton, point cloud, and joint-angle metrics were rendered through the Microsoft HoloLens 2, with all components communicating wirelessly via TCP/IP. The experimental results show high tracking accuracy, minimal latency, and robust kinematics tracking across different movement patterns, showing its potential applications in fields that require precise and real-time kinematic analysis.

Therefore, this low-cost and portable AR-based tracking solution delivers real-time kinematics analysis without needing wearable markers, electrodes, or controlled environments. Except for its potential use in the medical diagnosis and evaluation, it could be well-suited for sports performance assessments where the coaches and athletes could refine technique and prevent injury. Its intuitive AR interface also supports immersive education in anatomy and biomechanics, human–robot interaction studies, and motion-driven content creation for AI virtual production.

II. METHODS AND MATERIALS

In this section, we elaborate the key technologies behind the AR-based motion tracking system developed for Microsoft HoloLens 2. The system employs two Intel RealSense D435i depth cameras connected to a PC as the inference backend to enhance visual perception. It adopts a client-server software architecture built with Unity and Python, incorporating multi-threading and a producer-consumer model to ensure efficient real-time data processing. Coordinate calibration is achieved using binocular stereo vision and rigid-body registration. The system integrates two core models: YOLO11-pose for joint detection and angle estimation, and YOLO11-segmentation for body segmentation. With a custom data processing and rendering pipeline, it enables real-time visualization of joint movements and point cloud overlays within the user's physical environment. A representative set of evaluation movements and metrics was designed to comprehensively assess both system performance and kinematic quality.

A. Hardware architecture

The main computing hardware of HoloLens 2 includes Qualcomm Snapdragon 850 processor and Microsoft's self-developed second-generation holographic processing unit

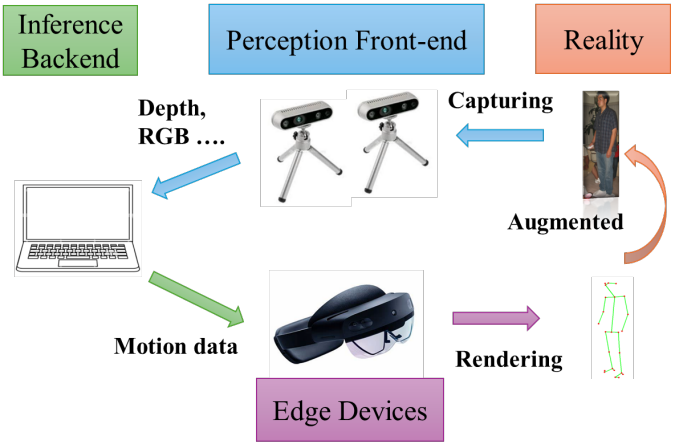


Fig. 3: Hardware architecture of the system. The perception front-end consists of two Intel RealSense R435i cameras. Inference is performed on an NVIDIA GeForce RTX 3060 Laptop GPU. Microsoft HoloLens 2 is used as the edge platform for visualization and interaction.

(HPU v2) [20]. Snapdragon 850 is a system-on-chip for mobile devices, and its design focuses on low power consumption and long battery life rather than high-performance computing. HPU v2 is mainly responsible for processing sensor data such as cameras and IMUs, and optimizing spatial mapping and gesture recognition. However, it cannot undertake complex deep learning reasoning tasks since it is not a general-purpose computing unit. Therefore, HoloLens 2 is not suitable for local execution of human posture estimation or tracking tasks.

A more feasible solution is to use it in conjunction with a PC with powerful GPU computing capabilities to build a "device-cloud collaboration" system architecture [21]. Based on this infrastructure, we further explored two different perception and reasoning solutions and conducted actual comparative analysis. The first solution involved using the HoloLens 2 camera as the perception front end. However, experimental results, which are analyzed in detail in Appendix A, showed that it could not meet the functional requirements for real-time and full-body motion tracking. This article focuses on the second solution, namely the decoupling of perception and reasoning tasks. The perception task is completed by the external device RealSense D435i, and the reasoning task is processed by the edge computing device. This design is illustrated in Figure 3, which has the following advantages:

The RealSense D435i is equipped with a high-resolution RGB sensor and an integrated depth sensor, allowing it to capture full-body human images from a greater distance. Its image quality is significantly better than that of the built-in camera on the HoloLens 2, making it particularly well-suited for full-body posture recognition. Although some post-processing is still required, the overall tracking reliability is significantly improved compared to model-based depth estimation, which often introduces additional error and computational overhead. Additionally, a dual RealSense setup is used for binocular registration, which further expands the system's field of view.

B. Software frameworks

Based on the hardware deployment, the system software adopts a distributed client-server architecture. The system consists of two core modules: a server program developed in Python and running on the PC, and a client application deployed on HoloLens 2, built using the Unity engine. The two realize real-time interaction between control instructions and rendering data through wireless communication (TCP/IP) protocol. The overall design is illustrated in Figure 4.

1) PC server: multiple threads work together to achieve efficient data processing flow on the PC server side. The main thread continuously monitors to the control trigger signal from the client. Once the start command is received, it processes the instruction content by unpacking, such as connection, calibration, skeleton or point cloud. The processing thread controls two RealSense devices respectively to complete the image acquisition and pose prediction tasks. Subsequently, the data is integrated from multiple perspectives through the data fusion module, and is written to the buffer queue of the producer-consumer model after packaging, realizing asynchronous decoupling between threads. The PC visualization thread extracts data from the buffer to achieve local real-time rendering, ensuring that the data source of the rendering task is clean, transparent, and not interfered with by other works. After processing, the data required for rendering will be packaged and sent to the client.

2) Client server: HoloLens 2 is built on the Unity framework and is divided into two parts: the main thread and the background thread. The main thread is responsible for the human-machine interface (HMI) logic. When the user clicks the button, the client initiates a tracking request to the server through an asynchronous event-driven communication mechanism. The background thread continuously receives and parses the posture data from the server and caches it locally. Subsequently, in the Unity lifecycle function Update(), the system extracts data from the cache at the current application frame rate (FPS) and renders it in real time, thereby achieving a highly synchronized and smooth user experience.

In summary, the system achieves high-performance data processing and low-latency AR rendering through multi-threaded scheduling and asynchronous communication mechanisms, which is suitable for the real-time performance.

C. Coordinate System Transformation

Due to the involvement of multiple sensing devices and the dynamic nature of edge devices, the key challenge of the system lies in achieving accurate 3D coordinate registration between the perception frontend (RealSense) and the edge device (HoloLens 2). We analyzed all local coordinate frames involved in the system, as illustrated in Figure 5, and ultimately simplified the problem to transforming the 3D information captured by the RealSense camera into the Unity world coordinate system for visual rendering.

1) Stereo Calibration and RealSense Rig Alignment: The RealSense setup consists of two cameras. We designate the right camera as the rig reference. An initial extrinsic transformation is obtained via stereo calibration using the Zhang

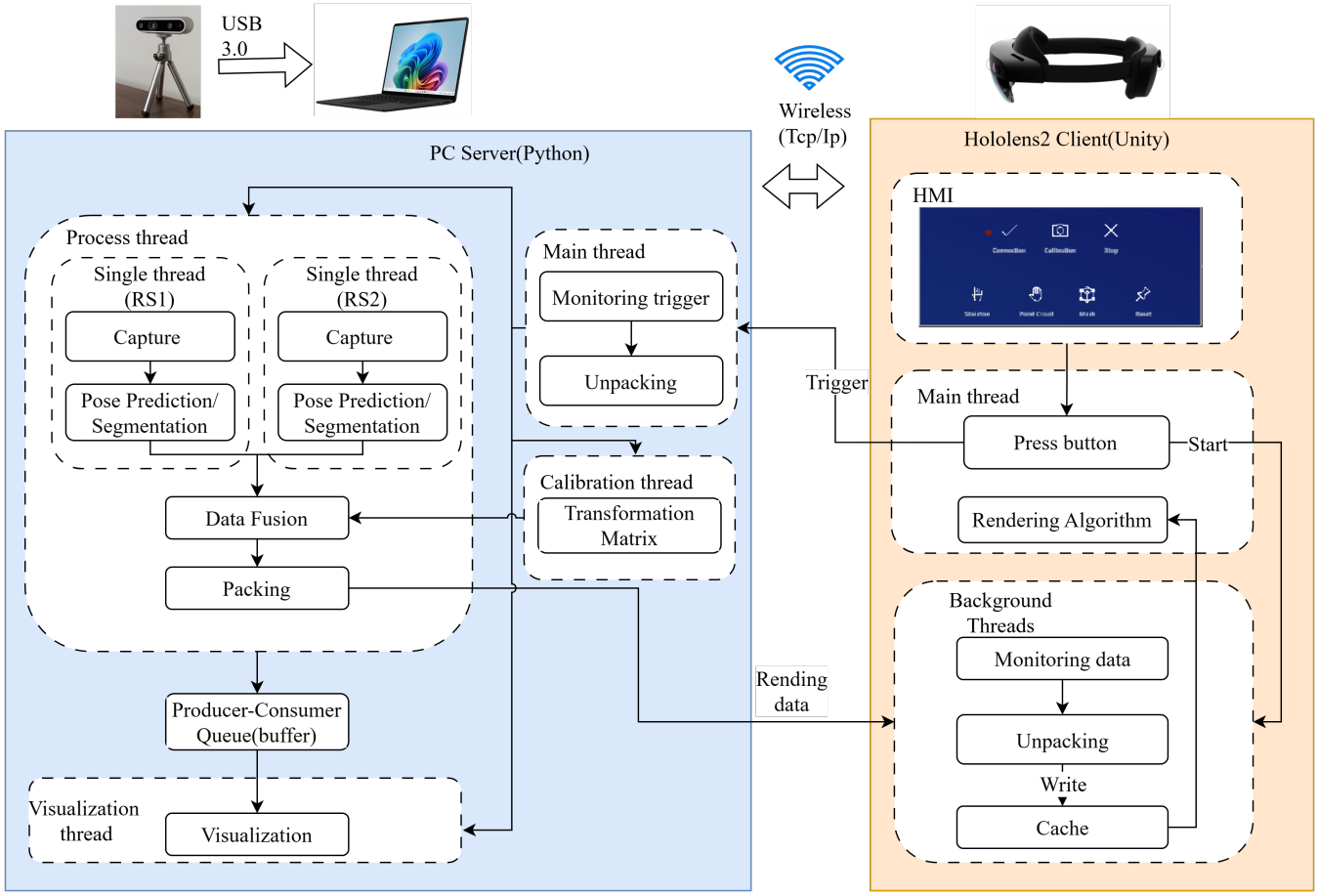


Fig. 4: Software architecture. A distributed client-server model, the left side (blue) represents the PC server responsible for sensing and processing, and the right side (orange) represents the HoloLens 2 client focusing on real-time rendering and user interaction.

method [22], implemented with the MATLAB stereo toolbox. This provides an initial transformation between the two RealSense cameras. To refine this transformation, we perform the Iterative Closest Point(ICP) [23] optimization at the point cloud level, initialized with $\mathbf{T}_{\text{init}}$. The final transformation from rs1 to the rs_rig (rs2) is computed by composition:

$$\mathbf{T}_{\text{rs1}}^{\text{rs\_rig}} = \mathbf{T}_{\text{rs1}}^{\text{rs\_rig-ICP}} \cdot \mathbf{T}_{\text{rs1}}^{\text{rs\_rig-init}} \tag{1}$$

2) Extraction of 3D Points from RealSense and HoloLens: After alignment, the right RealSense camera(Rig) observes at least three non-coplanar ArUco markers, and we extract 12 corner points $\left\{\mathbf{p}_i^{\text{rs\_rig}}\right\}_{i=1}^{12}$ using OpenCV and depth images.

Simultaneously, the HoloLens depth camera observes the same markers, yielding points $\left\{\mathbf{p}_i^{\text{h\_d}}\right\}$ in the HoloLens depth camera coordinate frame. These are transformed to the HoloLens world coordinate frame using the chain of extrinsics provided by the HoloLens Research Mode API [24] and Windows Mixed Reality SDK [25]:

$$\mathbf{p}_i^{\text{h\_w}} = \mathbf{T}_{\text{world}}^{\text{left}}(t) \cdot \mathbf{T}_{\text{left}}^{\text{depth}} \cdot \begin{bmatrix} \mathbf{p}_i^{\text{h\_d}} \\ 1 \end{bmatrix} \tag{2}$$

Here, $\mathbf{T}_{\text{left}}^{\text{depth}}$ is a fixed transformation provided by the SDK, and $\mathbf{T}_{\text{world}}^{\text{left}}(t)$ is a time-varying transformation updated each frame based on spatial tracking, since the HoloLens 2 system internally defines the left camera as the RIG of the camera system.

3) Rigid Registration from RealSense Rig to HoloLens World: Given corresponding 3D points $\left\{\mathbf{p}_i^{\text{rs\_rig}}, \mathbf{p}_i^{\text{h\_w}}\right\}$, we estimate the rigid transformation $(\mathbf{R}, \mathbf{t})$ that aligns the RealSense rig frame to the HoloLens world frame using least-squares point-based registration [26]. The objective is minimizing the least-squares error between the corresponding 3D points $\left\{\mathbf{p}_i^{\text{rs\_rig}}, \mathbf{p}_i^{\text{h\_w}}\right\}_{i=1}^{N}$:

$$\mathbf{p}_i^{\text{h\_w}} \approx \mathbf{R} \cdot \mathbf{p}_i^{\text{rs\_rig}} + \mathbf{t} \tag{3}$$

The optimal $(\mathbf{R}, \mathbf{t})$ is obtained via the SVD-based solution proposed, which computes centroids, constructs a cross-covariance matrix, and solves:

$$\mathbf{R} = \mathbf{V}\mathbf{U}^\top, \quad \mathbf{t} = \bar{\mathbf{p}}^{\text{h\_w}} - \mathbf{R} \cdot \bar{\mathbf{p}}^{\text{rs}} \tag{4}$$

The resulting rigid transformation from RealSense to HoloLens 2 world frame is:

$$\mathbf{T}_{\text{h\_w}}^{\text{rs\_rig}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{5}$$

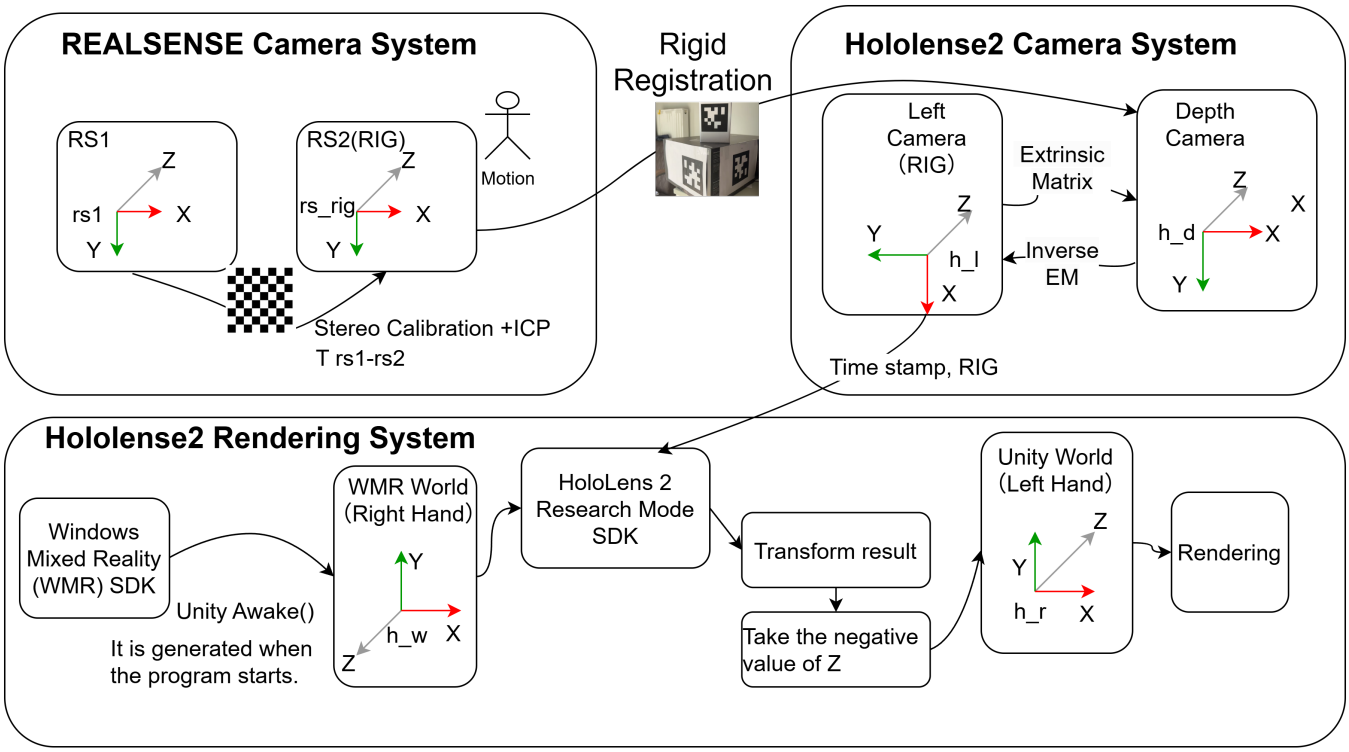


Fig. 5: Coordinate transformation modeling.The two RealSense cameras are calibrated relative to each other using chessboard-based calibration followed by ICP refinement, with the left camera serving as the reference for external (HoloLens 2) calibration. Within the HoloLens 2, the depth camera coordinates are transformed to the left- camera coordinates and then to the world coordinate system. Finally, rigid registration is performed between the world coordinate system and the RealSense system to achieve system-level calibration.

4) Unity Rendering Frame Correction: Since Unity uses a left-handed coordinate system, a final correction is applied by flipping the Z-axis of the transformed coordinates:

$$\mathbf{p}_i^{\text{render}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \left(\mathbf{R} \cdot \mathbf{p}_i^{\text{rs\_rig}} + \mathbf{t} \right) \tag{6}$$

This yields the final rendering position in Unity space.

D. Motion tracking integration

1) Skeleton tracking algorithm: In the task of skeleton tracking, we adopt the YOLOv11-Pose inference model. This model is an extension based on the YOLOv11 framework, capable of performing object detection and human pose estimation simultaneously [27]. A detailed introduction to YOLOv11 can be found in Appendix B. The output of YOLOv11-Pose consists of 2D coordinates and corresponding confidence scores for 17 keypoints of each detected human body.

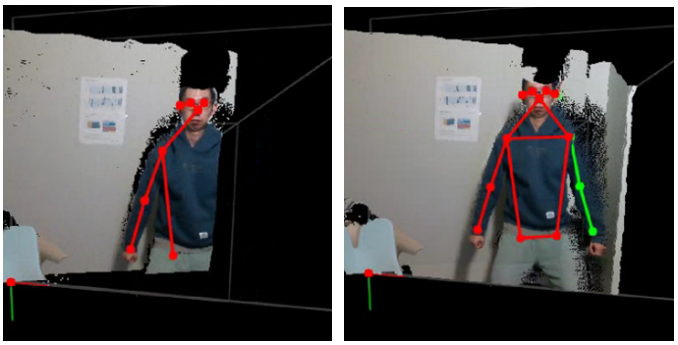
The overall pipeline is to use RealSense to obtain the 2D key point positions, then map these points into 3D space, and finally transmit the 3D data to HoloLens 2 for rendering. This process seems relatively straightforward, but the most challenging part is how to strike a balance between accuracy and real-time performance in practical applications.

The RealSense D435i supports resolutions from 424×240 to 1920×1080. We evaluated the trade-off between resolution and system performance. At 1920×1080, detailed images required minimal post-processing but reduced the server's frame rate to below 10 fps due to longer inference times. In contrast, 424×240 resolution allowed for a higher frame rate of up to 28 fps. Nevertheless, keypoint accuracy at 424×240 dropped by 32% under identical motion, using the 1920×1080 result as ground truth. Therefore, we finally chose 640×480 resolution as a suitable choice, and combined it with the fusion Algorithm 1.

Several steps were performed to fuse the recognize 3D joints of the subject. First, a preliminary refinement is applied to missing or significantly deviated 3D keypoints through local interpolation. Subsequently, a secondary local optimization is performed by minimizing the re-projection error to further enhance spatial accuracy. For each 2D keypoint, a confidence score is estimated based on the depth variance within its local neighborhood(5×5 kernal). During the fusion stage, keypoints from the two cameras are weighted according to their respective confidence scores, thereby ensuring the structural completeness of the reconstructed skeleton. Finally, temporal smoothing is applied using a Kalman filter to effectively suppress jitter introduced by depth noise or detection instability. The resulting fused skeleton is illustrated in Figure 9. In addition, we also optimized the original general CUDA inference method to the T4 TensorRT-10 engine [28], which shortened the inference time by about 6 ms on average.

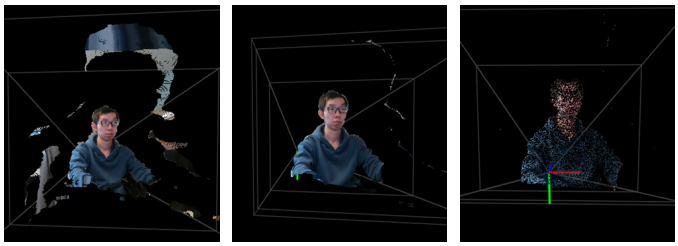
Algorithm 1: Dual-Camera 3D Skeleton Estimation with Optimization and Smoothing

Input: Synchronized RGB-Depth frames $\{(\mathbf{X}_c, \mathbf{D}_c)\}_{c=1}^{2}$
Camera extrinsics $(\mathbf{R}, \mathbf{T})$
Output: Smoothed 3D skeleton joints $\{\mathbf{p}_i\}_{i=1}^{K}$
Preprocessing for each camera $c = 1, 2$**:**
Align depth map: $\mathbf{D}_c \leftarrow \text{Align}(\mathbf{D}_c, \mathbf{X}_c)$
Generate point cloud: $\mathbf{P}_c \leftarrow \text{DepthToPointCloud}(\mathbf{D}_c)$
2D Keypoint Detection:
$\{\mathbf{u}_i^c\}_{i=1}^{K} \leftarrow \text{YOLOv11Pose}(\mathbf{X}_c)$
2D-to-3D Lifting:
for $i \leftarrow 1$ **to** K **do**
  if $\mathbf{D}_c(\mathbf{u}_i^c)$ *valid* **then**
    $\mathbf{p}_i^c \leftarrow \text{ProjectTo3D}(\mathbf{u}_i^c, \mathbf{D}_c)$
  else
    $\mathbf{p}_i^c \leftarrow \text{InterpolateLocal3D}(\mathbf{u}_i^c, \mathbf{D}_c)$
Local 3D Optimization:
for $i \leftarrow 1$ **to** K **do**
  $\mathbf{p}_i^c \leftarrow \arg\min_{\mathbf{p}} \|\pi_c(\mathbf{p}) - \mathbf{u}_i^c\|^2$
Coordinate Alignment:
for $i \leftarrow 1$ **to** K **do**
  $\hat{\mathbf{p}}_i^2 \leftarrow \mathbf{R}\mathbf{p}_i^2 + \mathbf{T}$
Depth Confidence Estimation:
for $i \leftarrow 1$ **to** K, $c = 1, 2$ **do**
  $\sigma_i^c \leftarrow \text{LocalDepthVariance}(\mathbf{u}_i^c, \mathbf{D}_c)$
  $conf_i^c \leftarrow \exp(-(\sigma_i^c)^2/\alpha)$
Confidence-based 3D Fusion:
for $i \leftarrow 1$ **to** K **do**
  if *both valid* **then**
    $\mathbf{p}_i \leftarrow \frac{conf_i^1 \cdot \mathbf{p}_i^1 + conf_i^2 \cdot \hat{\mathbf{p}}_i^2}{conf_i^1 + conf_i^2}$
  else if *only one valid* **then**
    $\mathbf{p}_i \leftarrow$ valid joint
Temporal Smoothing via Kalman Filter:
for $i \leftarrow 1$ **to** K **do**
  Predict: $\hat{\mathbf{p}}_i^{(t)} \leftarrow \text{KalmanPredict}(\mathbf{p}_i^{(t-1)})$
  Update: $\mathbf{p}_i^{(t)} \leftarrow \text{KalmanUpdate}(\hat{\mathbf{p}}_i^{(t)}, \mathbf{p}_i)$
return $\{\mathbf{p}_i^{(t)}\}_{i=1}^{K}$



(a) Single-camera result (b) Dual-camera result

Fig. 6: Comparison between single-camera and dual-camera skeleton fusion results.



(a) (b) (c)

Fig. 7: Point cloud processing result: (a) Instance segmentation using YOLOv11n-Seg, (b) noise suppression via morphological filtering and PCA ellipsoid clipping, and (c) downsampling for efficient transmission.

2) Point Cloud tracking algorithm: The system also implements a point cloud tracking algorithm to capture human motion kinematics geometrically. Motion point clouds are obtained using an instance segmentation framework based on the YOLOv11n-Seg model—an efficient, lightweight variant of YOLOv11. YOLOv11-Seg integrates instance segmentation capabilities by combining mask prototypes with corresponding mask coefficients to generate precise instance-level masks [29]. For real-time performance, the model is accelerated with TensorRT and paired with our backend processing Algorithm 2.

The algorithm is to refine the segmentation quality and suppress noise, we apply a morphological opening operation (erosion followed by dilation) to the predicted masks. This effectively removes small isolated artifacts, smooths contour boundaries, and yields more reliable foreground pixel indices.

After extracting the foreground point clouds and aligning them via rigid registration (Camera 1 to Camera 2), we filter out spatial outliers using a PCA-based ellipsoid constraint. Let $\mathbf{V} \in \mathbb{R}^{n\times 3}$ be the original point cloud. We compute the PCA-aligned coordinates $\mathbf{V}_{\text{PCA}} = \mathbf{V} \cdot \mathbf{U}$, where $\mathbf{U}$ is the rotation matrix obtained from PCA. We then retain only the points that fall within axis-aligned thresholds $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]$:

$$\mathcal{M} = \left\{ \vec{v}_i \in \mathbf{V} \;\middle|\; \left| (\vec{v}_i \cdot \mathbf{U})_j \right| < \tau_j, \quad \forall j = 1, 2, 3 \right\} \tag{7}$$

This constraint preserves the dense core structure of the human body while eliminating dispersed outliers, resulting in a compact and consistent point cloud.

Finally, to accommodate the bandwidth constraints of TCP/IP transmission, we perform random downsampling before sending the point cloud to the HoloLens 2 client, ensuring efficient transfer without sacrificing real-time performance. The final result after algorithm processing is shown in Figure 7.

E. Evaluation Method

After the system design is completed, conducting a scientific evaluation is equally essential. The evaluation framework proposed in this work is divided into two levels: system performance evaluation and kinematic tracking evaluation. The system performance evaluation focuses on the overall

Algorithm 2: Dual-Camera Point Cloud Generation, Stitching, and Filtering

Input: Synchronized RGB-D frames $\{(\mathbf{X}_c, \mathbf{D}_c)\}_{c=1}^{2}$
Camera extrinsics $(\mathbf{R}, \mathbf{T})$
Output: Aligned and filtered point cloud $\mathbf{P}_{\text{final}}$
Preprocessing for each camera $c = 1, 2$**:**
Align depth map to color image: $\mathbf{D}_c \leftarrow \text{Align}(\mathbf{D}_c, \mathbf{X}_c)$
Generate raw point cloud:
$\mathbf{P}_c \leftarrow \text{DepthToPointCloud}(\mathbf{D}_c)$
2D Segmentation:
Detect human mask: $\mathbf{M}_c \leftarrow \text{YOLOv11Seg}(\mathbf{X}_c)$
Post-process mask:
$\mathbf{M}_c \leftarrow \text{MorphologicalOpenErode}(\mathbf{M}_c)$
3D Point Cloud Cropping:
Crop point cloud using mask:
$\mathbf{P}_c^{\text{crop}} \leftarrow \text{CropByMask}(\mathbf{P}_c, \mathbf{M}_c)$
Point Cloud Stitching:
for $\mathbf{p} \in \mathbf{P}_2^{crop}$ **do**
  Transform to Camera 1 frame: $\hat{\mathbf{p}} \leftarrow \mathbf{R} \cdot \mathbf{p} + \mathbf{T}$
  Add $\hat{\mathbf{p}}$ to $\mathbf{P}_2^{\text{aligned}}$
Merge point clouds: $\mathbf{P}_{\text{combined}} \leftarrow \mathbf{P}_1^{\text{crop}} \cup \mathbf{P}_2^{\text{aligned}}$
3D Cloud Filtering:
Depth filtering: $\mathbf{P}_{\text{filtered}} \leftarrow \text{FilterZ}(\mathbf{P}_{\text{combined}}, z > 0.05)$
Remove outliers:
$\mathbf{P}_{\text{filtered}} \leftarrow \text{PCABasedEllipsoidFilter}(\mathbf{P}_{\text{filtered}})$
Point Cloud Downsampling:
$|\mathbf{P}_{\text{filtered}}| > Samplingpoints$
$\mathbf{P}_{\text{final}} \leftarrow \text{RandomSubsample}(\mathbf{P}_{\text{filtered}}, Samplingpoints)$

return $\mathbf{P}_{\text{final}}$

operational performance at a macro level, encompassing two core metrics: stability and latency, which are used to assess the system's responsiveness. The kinematic tracking evaluation targets the micro-level effectiveness of motion capture, with three key metrics: accuracy, continuity, and consistency, providing a comprehensive assessment of the system's ability to track motion data.

To ensure the evaluation process is both scientific and systematic, this study draws upon authoritative rehabilitation programs from the American Academy of Orthopaedic Surgeons (AAOS), including protocols for shoulder rehabilitation [30], strength training [31], and lower limb recovery [32]. Based on these references, a standardized set of movement tasks was designed as Figure 8. This movement set is structured to address the rehabilitation needs of the early, middle, and late postoperative stages, with exercises categorized by increasing levels of intensity to reflect the progressive nature of physical recovery. Moreover, the movement set comprehensively covers joint movements across major body regions and multiple anatomical planes, thereby enhancing the completeness and representativeness of the evaluation results. Each movement is mapped to specific evaluation metrics, enabling a more targeted assessment strategy, which is presented in TableI.

1) System Stability: This metric is designed to evaluate the runtime stability of the system. Movements (1) to (4)

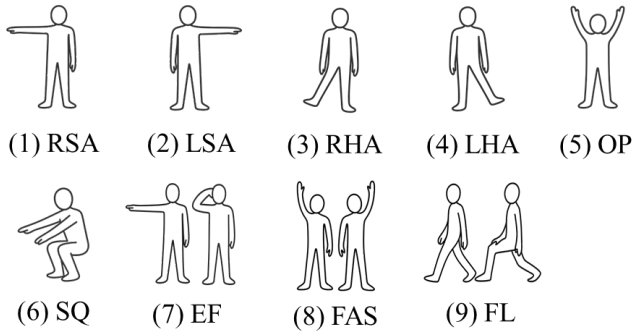


Fig. 8: Evaluation movement set, including: (1) Right Shoulder Abduction (RSA), (2) Left Shoulder Abduction (LSA), (3) Right Hip Abduction (RHA), (4) Left Hip Abduction (LHA), (5) Overhead Press (OP), (6) Squat (SQ), (7) Elbow Flexion (EF), (8) Freestyle Arm Stroke (FAS), and (9) Forward Lunge.

TABLE I: Mapping Between Evaluation Metrics and Movement Tasks

Evaluation Metric	Movements
System Stability	(1) (2) (3) (4)
System Latency	(5) (6)
Kinematic Accuracy	(1) (2) (3) (4)
Kinematic Continuity	(7)
Kinematic Consistency	(8) (9)

are selected to encourage participants to perform full-body motions with the largest possible range. Ideally, all joints should be detected throughout the three repetitions of these movements, allowing a complete skeleton to be rendered in each frame. However, under unstable conditions or during system jitter, keypoints may occasionally fail to be detected. To ensure a strict evaluation standard, we define a "lost frame" as any frame in which at least one keypoint is missing. For each participant, after completing three consecutive rounds of all movements, we calculate the proportion of lost frames to the total recorded frames as a quantitative indicator of system stability. This is defined as:

$$\text{Stability} = 1 - \frac{N_{\text{lost}}}{N_{\text{total}}} \tag{8}$$

where N_{lost} denotes the number of lost frames, and N_{total} denotes the total number of frames recorded for that participant.

2) System Latency: Due to factors such as perception collection, back-end processing, and communication transmission, system latency is difficult to completely avoid during real-time operation. To evaluate this latency, we utilized the video capture of HoloLens 2 in combination with the frame analysis software Kinovea [33], enabling us to measure the time lag between the real motion and its corresponding skeletal tracking response.

To facilitate accurate identification of latency, we selected two movements with distinct peak characteristics: (5) Overhead Press (OP) and (6) Squat (SQ). During the OP , participants were instructed to pause for approximately 1 second when their arms reached the fully raised position. Similarly,

during the Squat, a hold was maintained at the lowest point of the motion. These brief pauses allowed for precise detection of peak frames, thus enabling reliable latency estimation.

Each participant performed three repetitions of both actions. For each trial, we recorded the frame difference between the real motion peak and the tracking response peak, then the average latency is computed as:

$$\text{Latency}_{avg} = \frac{1}{M}\sum_{i=1}^{M}|\Delta f_i|\,, \tag{9}$$

where Δf_i denotes the frame difference between the real motion peak and the system-detected peak in the i-th trial, and M is the total number of trials.

Finally, the overall system latency was quantified by computing the mean delay along with the 95% confidence interval:

$$CI_{95\%} = \mu_D \pm 1.96 \cdot \left(\frac{\sigma_D}{\sqrt{M}}\right), \tag{10}$$

where μ_D and σ_D represent the mean and standard deviation of the delay measurements, respectively.

3) Kinematic Accuracy: This metric focuses on evaluating the tracking accuracy of individual joint positions during motion capture. Similar to previous evaluations, we selected movements (1) to (4), which involve full-body motion. Prior to the experiment, we manually measured the length of each participant's forearms and lower legs to serve as ground-truth reference values.

Participants were then instructed to sequentially perform movements, with each movement repeated three times. During these trials, we focused on the Euclidean distances between the following four joint pairs: Joint 7 and Joint 9 (left forearm), Joint 8 and Joint 10 (right forearm), Joint 13 and Joint 15 (left lower leg), Joint 14 and Joint 16 (right lower leg).

For each frame, we computed the distance between each joint pair as captured by the system and compared it with the corresponding measured limb length to evaluate the tracking error. For each participant, the standard deviation of the error across the three repetitions was calculated and used as the metric for kinematic accuracy. A smaller standard deviation indicates higher consistency and better alignment with actual anatomical structure, thus reflecting higher tracking accuracy.

The Euclidean distance between two joints i and j at frame t is computed as:

$$L_{ij}^{t} = \left\|\vec{p}_i^{\,t} - \vec{p}_j^{\,t}\right\|, \tag{11}$$

where $\vec{p}_i^{\,t}$ and $\vec{p}_j^{\,t}$ denote the 3D coordinates of joint i and j at time t, respectively.

We use the standard deviation of the tracked length between each joint pair (i, j) over time to quantify kinematic accuracy. A smaller value indicates higher accuracy, and the metric is defined as:

$$\sigma_{ij} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(L_{ij}^{t} - \bar{L}_{ij}\right)^2} \tag{12}$$

where T is the total number of frames, L_{ij}^{t} is the observed length at time t, and $\bar{L}_{ij}$ is the average length over all frames.

4) Kinematic Continuity: This metric evaluates the smoothness and continuity of the motion trajectory during tracking. Movement (7) Elbow Flexion(EF) was selected for analysis. Each participant was instructed to perform the motion three times at two different speeds: a slow speed simulating the movement pattern of rehabilitation patients, and a faster speed representing natural motion in healthy individuals.

We focused on the wrist joint as the target for analysis. The system recorded the 3D position of the wrist at a fixed sampling interval Δt. To estimate acceleration, we applied a second-order finite difference directly to the position data, yielding the scalar acceleration magnitude:

$$a^t = \left\|\frac{1}{\Delta t^2}\left(\vec{p}^{\,t} - 2\vec{p}^{\,t-1} + \vec{p}^{\,t-2}\right)\right\| \tag{13}$$

where $\vec{p}^{\,t}$ denotes the 3D wrist position at frame t.

We then computed the mean acceleration $\bar{a}$ and standard deviation σ_a across all frames. Any frame where $a^t > \bar{a} + 3\sigma_a$ was labeled as an anomaly, indicating possible tracking loss or motion discontinuity. To further understand the distribution of acceleration values and the frequency of anomalies, we applied kernel density estimation (KDE) [34]:

$$\hat{f}_h(a) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{a - a_i}{h}\right) \tag{14}$$

where $\hat{f}_h(a)$ represents the estimated probability density at acceleration value a, h is the bandwidth, and $K(\cdot)$ is a Gaussian kernel function. Long tails or multiple peaks in the KDE curve indicate the presence of motion artifacts or instability in tracking.

In addition, we analyzed the motion of the wrist joint in both the x-axis and y-axis directions, computing the mean and variance over three complete motion cycles. Sudden spikes in these directions were used to assess trajectory smoothness. We also measured the elbow joint angle throughout the same cycles, calculating its mean and variance to provide a comprehensive evaluation of the motion continuity across the wrist, elbow, and shoulder joints.

5) Kinematic Consistency: This metric is designed to evaluate the tracking consistency of the system during asymmetric movements. Two tasks were selected for this analysis: (8) Freestyle Arm Stroke and (9) Forward Lunge. These actions involve alternating motions of the upper and lower limbs across different anatomical planes, making them well-suited for assessing.

In the FAS task, participants alternately raise and lower their arms. We track the vertical displacement of both wrist joints and identify the peaks that occur during the upward motion of one arm and the valleys during the downward motion of the other. Ideally, these peak-valley pairs should alternate rhythmically with minimal temporal deviation on the timeline.

In the FL task, participants alternately perform forward lunges with the left and right legs. We track the horizontal displacement of both knees and extract the peaks and valleys for each leg. Although the movement amplitudes of the two knees may differ, the phase difference in the timing of their peaks and valleys reflects the system's tracking consistency. A

smaller phase difference indicates better temporal alignment between the two knees and thus higher consistency in asymmetric motion tracking.

To quantify the consistency of left-right tracking, we first extracted the frame indices of paired motion events on both sides—peaks or valleys depending on the movement type. These event pairs are denoted as L_i for the left side and R_i for the right side, where $i = 1, 2, \ldots, n$, and n is the total number of valid pairs.

For each pair, the temporal difference $|R_i - L_i|$ was normalized by the average local cycle period, computed from the intervals between adjacent events on each side. The overall kinematic consistency is measured using the Mean Relative Deviation (MRD), defined as:

$$\text{MRD} = \frac{100\%}{n-1} \sum_{i=1}^{n-1} \frac{|R_i - L_i|}{\frac{(L_{i+1} - L_i) + (R_{i+1} - R_i)}{2}} \tag{15}$$

A lower MRD indicates better temporal synchronization between the left and right limbs, signifying higher consistency in asymmetric motion tracking.

III. Experimental Results

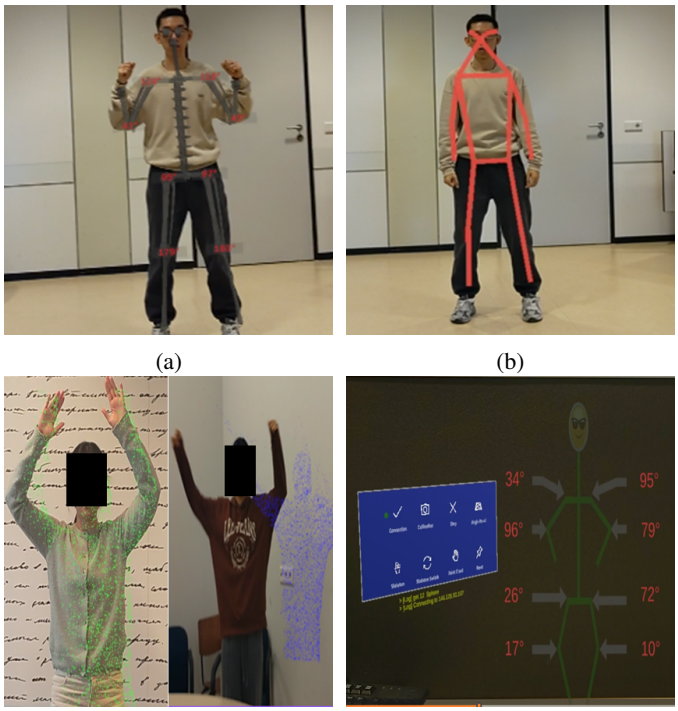
This section presents the experimental results. We first showcase the system's visualization output to demonstrate its complete functional implementation. Then, the performance is evaluated based on five evaluation metrics: system stability, system latency, kinematics accuracy, kinematics continuity, and kinematics consistency. These metrics collectively provide a comprehensive assessment of the system's capabilities and motion tracking performance.

A. AR Visualization

We implemented a real-time AR skeleton rendering module based on 3D human keypoints for motion visualization. The system supports two rendering modes: a realistic 3D skeletal model and a color-coded pose line visualization as shown in Figure 9a and Figure 9b, which facilitates the observation and evaluation of different performance indicators. To ensure inter-frame pose consistency, local coordinate normalization is applied in the rendering pipeline, and flexible position adjustment is supported to accommodate various visualization environments.

As the current visualization scheme does not incorporate upper limb rotation, each skeletal segment is defined by its directional vector and the Euclidean distance between connected joints. In addition, the keypoint structure is based on the COCO 17-keypoint format, which lacks explicit spine joints. To enhance visual realism, we interpolate intermediate points between the shoulders and hips to simulate a continuous spinal structure.

For point cloud visualization, we developed a chunk-based rendering pipeline that partitions large-scale 3D point sets into multiple meshes to comply with GPU vertex limits. Each chunk dynamically constructs a mesh and uses a programmable shader to support per-point coloring, adjustable point sizes, and global offset control. The example of this effect is shown in Figure 9c.



(a) (b) (c) (d)

Fig. 9: AR Visualization: (a) 3D skeleton model; (b) color-coded pose lines based on the skeleton structure; (c) multi-colored(green and blue) point clouds; (d) HMI panel and joint angle panel following the user's field of view.

Meanwhile, an intuitive human-machine interface (HMI) was designed within the AR environment, allowing users to perform basic button operations and switch between rendering modes. During skeleton rendering, a joint status panel is displayed and anchored to the user's view direction. This ensures real-time monitoring of key joint movements, even when the subject temporarily moves out of sight, as demonstrated in Figure 9d.

B. Evaluation Metrics

1) System Stability: We analyzed three repetitions of movements (1) to (4) performed by 5 participants. Due to individual differences in movement speed, the number of frames captured varied among participants. By examining the lost frame rate, we obtained the following statistics as TableII. The average valid frame rate across all five participants reached 99.00%, demonstrating the high stability of the system during motion capture. Analysis of the missing frames revealed that most errors occurred at Key Point 4 (right ear), primarily due to occlusion caused by hair or body motion. As this point is not critical for motion tracking, and the key body joints remained largely unaffected, the results further demonstrate the robustness and stability of the system.

2) System Latency: After each participant completed three repetitions of movement (5) (Overhead Press) and action (6) (Squat), we annotated keyframes and timestamps using the Kinovea tool to quantify the system's response latency. Table III presents the average latency and corresponding

TABLE II: Stability Rate and Frame Loss

Subject	Frames	Loss	Stability rate
Subject1	237	4	98.31%
Subject2	212	0	100.00%
Subject3	229	3	98.68%
Subject4	226	0	100.00%
Subject5	254	5	98.03%

(a) (b)

Fig. 10: Latency visualization: (a) The participate performs an overhead press to the highest point, but the tracked pose line lags behind; (b) The tracked pose line catches up with the real movement.

95% confidence intervals for each participant in both motion types. Additionally, we chosed the pose line as visualization method, as illustrated in Figure 10, to intuitively depict the synchronization between the virtual skeleton and the real-world motion.

The results show that the average latency for the Overhead Press was 65.33 ms (95% CI: 62.85–67.81 ms), while that for the Squat was slightly higher at 68.87 ms (95% CI: 65.70–71.64 ms). This difference can be attributed to the motion characteristics: the Squat involves rapid displacement of the torso and lower limbs, placing greater computational demands on frame-level pose recognition and skeleton reconstruction, thereby causing a slight increase in latency.

Considering that the system adopts a lightweight architecture without reliance on high-performance hardware, it does not fall within the strict 50 ms threshold of hard real-time systems. However, its response time is well within the acceptable range for soft real-time applications. For comparison, Microsoft's Kinect motion tracking sensor and its official SDK introduce approximately 70 ms of latency at the skeletal tracking level alone, excluding additional rendering and processing delays [35]. Furthermore, a study on racing games indicated that latencies under 100 ms are generally considered acceptable by users [36]. Therefore, we conclude that our system fulfills the temporal requirements of real-time feedback applications within the soft real-time paradigm.

3) Kinematic Accuracy: We measured and recorded the actual lengths of the forearms and lower legs of five participants. After each participant completed three repetitions of actions (1) through (4), we calculated the Euclidean distance errors between the virtual limb lengths and the corresponding physical measurements for both left and right limbs. These errors are visualized using boxplots, as shown in Figure 11. The median Euclidean distance (MED) values for all five participants are summarized in Table IV.

TABLE III: Latency for Overhead Press and Squat

Participant	Overhead Press(ms)	Squat(ms)
Participant 1	62.50	72.30
Participant 2	64.37	66.80
Participant 3	65.97	66.80
Participant 4	69.23	68.70
Participant 5	64.57	67.87
Total Mean	65.33	68.67
95% CI	(62.85, 67.81)	(65.70, 71.64)

Overall, the MED values for the forearms ranged from 0.7 to 2.1 cm, with average MEDs of 1.484 cm and 1.662 cm for the left and right arms, respectively. For the lower legs, the MED values were between 0.85 and 2.3 cm, with average MEDs of 1.494 cm and 1.366 cm for the left and right legs, respectively.

Some participants showed noticeable asymmetry between their left and right limbs in terms of tracking accuracy, which be attributed to factors such as individual motion speed and clothing color. For instance, one participant wearing a light-colored, loose-fitting top exhibited brief tracking drift during rapid arm extension. This occurred because the arm color closely matched the background, and under such conditions, depth cameras like Intel RealSense—which rely on active infrared stereo vision—can experience foreground-background blending. This leads to temporary misidentification of skeletal keypoints.

Despite individual differences among participants, the system demonstrated reliable spatial precision in localizing limb endpoints. Taking into account potential human errors in manual measurements, the average localization error remained around 1.5 cm, which is acceptable given the lightweight design of the system. Moreover, the tracking performance remained consistent and stable across subjects with varying body types.

TABLE IV: Limb Measurements and MED

Subject	Forearm (cm)	Left MED (cm)	Right MED	Lower leg (cm)	Left MED	Right MED
Subject1	23.70	0.70	1.20	36.80	1.60	2.30
Subject2	25.30	1.40	1.30	37.00	2.27	1.03
Subject3	26.50	1.82	1.64	38.80	1.52	1.21
Subject4	27.10	1.90	2.10	37.20	1.07	0.85
Subject5	26.50	1.60	2.07	41.10	1.01	1.44

4) Kinematic Continuity: We recorded the movement(7) of all participants as they performed three times along a semi-circular path under two speed conditions, and we select one representative participant for detailed continuity analysis. The overall trajectory of the wrist joint is illustrated in Figure 12a and Figure 12b.

Under both low-speed(slow) and fast-speed(fast) conditions, the motion trajectory remains smooth and coherent. Although more anomaly points appear under the low-speed condition,

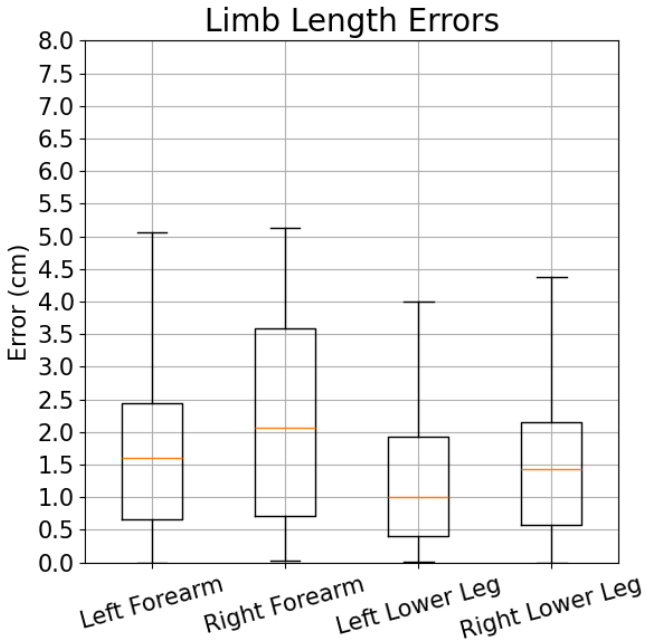


Fig. 11: Error visualization - Box plot example

they are mostly concentrated at the ends of the path or at turning points. This is mainly due to minor tremors occurring at the cycle boundaries while the participant, as a normal subject, attempted to simulate slow movement for rehabilitation, which led to discontinuity noise.

To further quantify trajectory continuity, we analyzed the acceleration using Kernel Density Estimation (KDE), as shown in Figure 12c.The KDE curve under the slow condition exhibits a higher peak skewed to the left, indicating that most acceleration values are concentrated within the 0–1 $\mathrm{m/s^2}$ range, reflecting a relatively stable motion process. In contrast, under the fast condition, the KDE curve is flatter and more widely distributed, with the peak shifted to the right, suggesting a generally higher level of acceleration. However, the right tail of the curve drops sharply, and the tail area ratio (i.e., the percentage of KDE density beyond the threshold) is only 0.09%, indicating that drastic acceleration changes are infrequent. In comparison, the slow-speed condition presents a more pronounced long-tail feature, yet the calculated tail area ratio is 0.86%. Overall, neither condition shows additional peaks in the KDE curves, the trajectory continuity under both speed conditions exceeds 99%, demonstrating robust tracking performance.

Since each participant completed three full motion cycles, we further analyzed these three independent movements separately under each speed condition to eliminate potential interference from turning points. We examined the motion trajectories along the x-axis and y-axis, calculated the average trajectory, and plotted the bandwidth (±1 standard deviation) to evaluate fluctuation. As illustrated in Figure 13a , 13b, 13d,and 13e, the amplitude of vibration remains consistent regardless of speed. The trajectory of fast mode appears narrower due to its shorter duration, but the fluctuation magnitude does not show significant differences. For individual trials, the average motion trajectory remains smooth. Although local bandwidth increases are observed in some regions, these are attributed to natural variability across motion cycles—after all, "no man ever steps in the same river twice," and it is unrealistic to expect perfectly identical movements. Therefore, variations within ±1 SD are acceptable. Importantly, no spikes or frame drops were observed in any of the three repetitions, indicating excellent frame-level continuity.

In addition to analyzing the continuity of the wrist joint, we also examined the motion of the elbow and shoulder joints. Specifically, we focused on the variation in elbow joint angles, as shown in Figure 13c and 13f. The range of motion spans from 180° to 20°, which aligns with the expected motion pattern, and no spikes were detected throughout the process. This further confirms that, in addition to the wrist trajectory, the elbow and shoulder joint movements also exhibit high levels of continuity.

5) Kinematic Consistency: By conducting the periodic analysis of data recorded from five participants during movements (8) and (9), selecting one participant as a representative case for detailed consistency analysis. As shown in Figures 14a and 14b, the vertical peaks and valleys of the left and right wrists during the FAS movement are accurately aligned in terms of frame timing, and the shoulder angle variations exhibit a highly consistent pattern. Based on an in-depth analysis of five complete cycles, the average consistency of vertical wrist displacement was 99.00%, while the consistency of shoulder angle variation reached 98.41%.

Figures 14c and 14d illustrate that, during the FL movement, the horizontal displacement and angular changes of the knee joint also demonstrate good correspondence, with peak-to-peak and valley-to-valley alignment. The average consistency values over five cycles were 98.53% for horizontal displacement and 98.28% for knee joint angle variation.

Regarding movement amplitude, minor fluctuations were observed due to individual differences in movement habits. For example, the participant's knee flexion decreased slightly in later cycles, likely due to the forward movement of the lead leg, which resulted in a larger knee angle and thus reduced the need for pronounced bending. This adjustment may help the participant conserve energy by lessening muscle strain. However, such variations in amplitude do not affect the analysis of movement consistency. Overall, The kinematic tracking results show high consistency.

IV. Discussion

This work proposes an augmented reality (AR)-based motion tracking system. The system employs dual RealSense cameras for data acquisition, integrates the YOLO-based model for backend inference, and utilizes a multithreaded client-server architecture to ensure real-time performance. The tracking results are rendered in real time on Microsoft HoloLens 2 and can be visualized through various modalities including skeletons, point clouds, and joint angle, providing intuitive 3D representations within the physical environment. Experimental results demonstrate strong performance in both

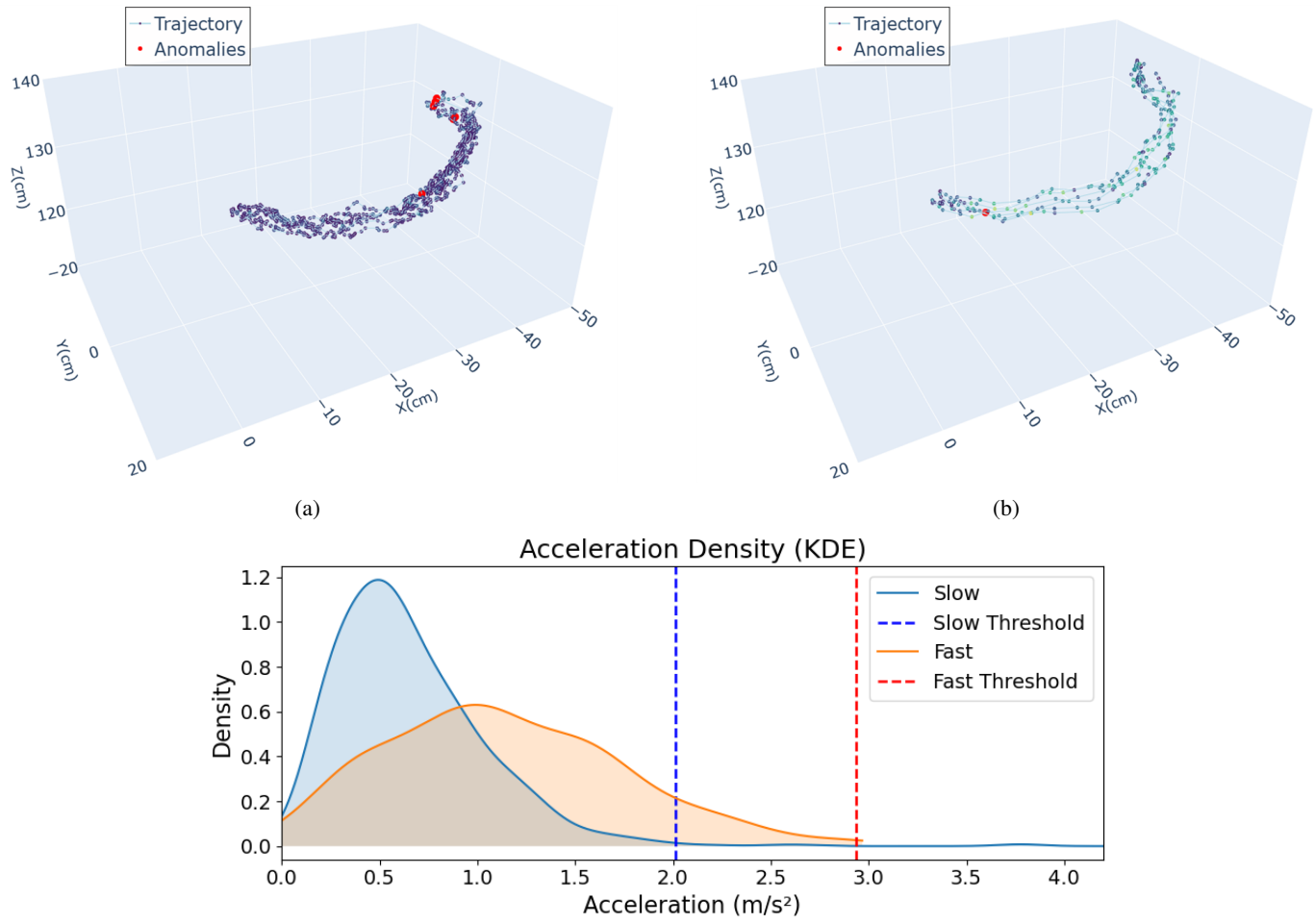


Fig. 12: The result of continuity: (a), (b), respectively represent the 3D trajectories of the participant's wrist moving back and forth three times along a half-circular path under low-speed and fast-speed. The points marked in red represent detected anomalies where the acceleration exceeds the threshold. (c) is the acceleration Kernel Density Estimation (KDE).

system-level and kinematic tracking. In particular, when compared to HoloMoCap [19], one of the most recent HoloLens 2-based motion capture systems, the proposed system demonstrates superior user-friendliness, broader application scope, and better overall performance, as summarized in Table V.

The core contribution of this work lies in two major technological breakthroughs. First, the proposed system removes the reliance on expensive, complex equipment and wearable markers typically required by traditional motion tracking methods, offering a lightweight and markerless alternative. Second, it supports real-time integration of full-body motion data into AR environments, while allowing users to move freely and change viewpoints. This significantly enhances user immersion, spatial flexibility, and interactive experience. These innovations are particularly valuable in scenarios such as sports training and rehabilitation, where professionals can view overlaid motion data directly within their field of vision, without the need to alternate between the subject and analysis interfaces. This improves spatial freedom and decision-making efficiency. In the context of motion-driven content creation, the system transforms static, abstract motion data into immersive, real-

TABLE V: Comparison between HoloMoCap and the proposed system

System	HoloMoCap [19]	The proposed system
Wearable Devices	Requires markers	Markerless
Tracking Target	Lower limbs	Full body
Tracking Accuracy	Suitable only for slow, specific motions (e.g., squats), MAE 0.7°–5°	Supports medium-speed unstructured motion, MED 1.5 cm
User Perspective	In front of the subject	No restrictions
Tracking Range	Within 1.5 m	Best within 3 m
Real-Time Performance	Rendering interval 200 ms/frame	Approximately 70 ms
Visualization	Joint balls	Skeleton, pose lines, point cloud
Evaluation Scope	Only squatting motion	9 motion types, including continuity and consistency evaluation
Other Functions	None	Joint angle calculation

Note: MAE stands for Mean Absolute Error. MED stands for Median Euclidean Distance.

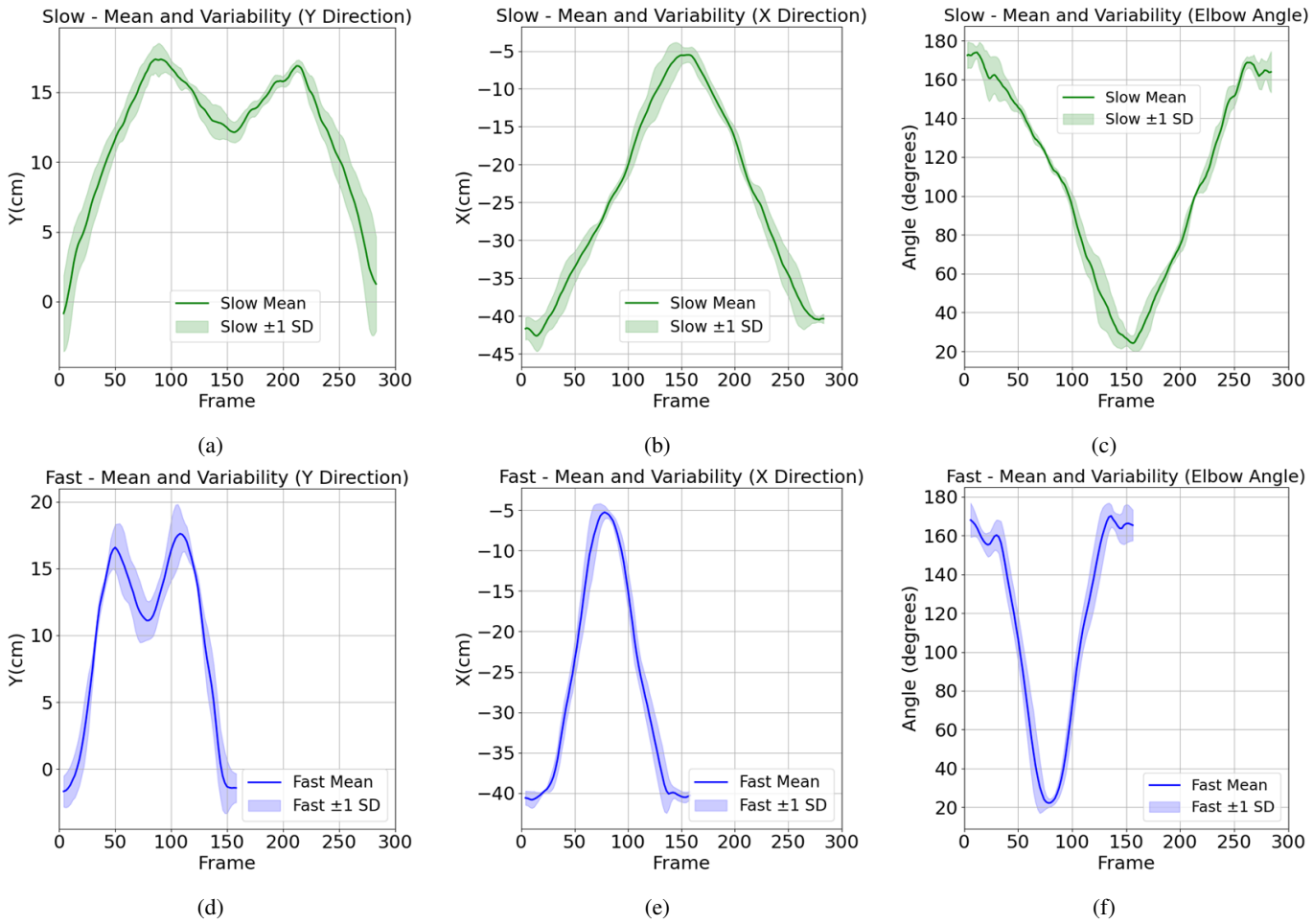


Fig. 13: Single-cycle motion analysis: (a) Mean and ±1 standard deviation of wrist displacement in the x-direction over frames(Slow); (b) Mean and ±1 standard deviation of wrist displacement in the y-direction over frames(Slow); (c) Mean and ±1 standard deviation of the elbow joint angle (Slow); (d) Mean and ±1 standard deviation of wrist displacement in the x-direction over frames(Fast); (b) Mean and ±1 standard deviation of wrist displacement in the y-direction over frames(Fast); (c) Mean and ±1 standard deviation of the elbow joint angle (Fast).

time feedback. Operators can instantly verify the accuracy of captured data, reducing the risk of 3D information loss often associated with traditional 2D interfaces, and improving both operational efficiency and data quality.

Furthermore, the system features a modular and extensible architecture, providing a reliable framework for future AR-based applications. Currently, it employs a YOLO-based model for skeleton or point cloud recognition; however, it can be extended to support custom-trained models, enabling more flexible and adaptable visual prediction and validation. For instance, in prosthetic motion prediction, even in the absence of real amputee subjects, the system can simultaneously render both AI-predicted and ground-truth motions in the AR scene. This enables intuitive visual comparison and significantly improves the clarity and efficiency of model validation.

Despite its advantages in cost and portability, the system still faces several technical challenges that require further investigation.

In terms of accuracy, the system has yet to reach the millimeter-level precision of commercial systems such as OptiTrack or Vicon. However, the integration of keypoint detection and multi-stage depth filtering strategies partially compensates for the hardware limitations. According to Intel RealSense documentation, depth errors can reach up to 3.5 cm at certain distances. With the fusion strategy proposed in this study, the tracking error was reduced to approximately 1.5 cm, showing significant improvements in mid- to long-range scenarios.

Regarding real-time performance, the system currently does not meet the sub-50 ms latency requirement of strict real-time systems. Nevertheless, through hardware-software co-optimization—such as the incorporation of TensorRT acceleration and improvements in preprocessing and data transmission pipelines—the overall latency has been stabilized at around 70 ms. For moderate-speed motion tasks, this level of delay is generally imperceptible to users.

With the continuous advancement of AR hardware, particularly in terms of camera resolution and processing power, it is expected that the system could eventually be fully deployed on-device, further reducing latency, improving efficiency, and enhancing its applicability in seamless integration scenarios.

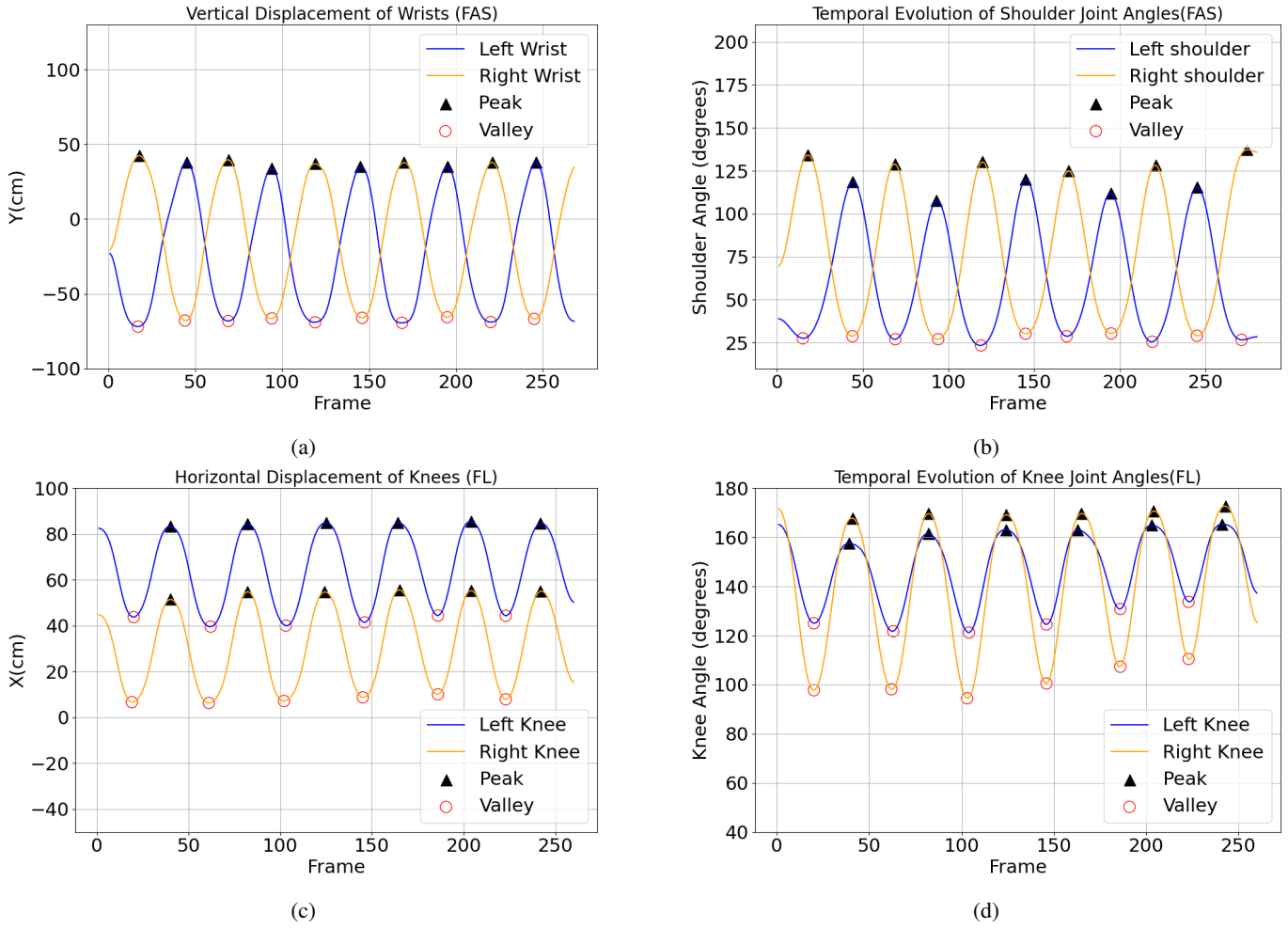


Fig. 14: Consistency analysis over cycles: (a) Vertical displacement of the left and right wrists during the freestyle arm stroke, showing periodic variation over frames; (b) Shoulder angle variation during the freestyle arm stroke over frames; (c) Horizontal displacement of the left and right knees during the forward lunge, showing periodic variation over frames; (d) Knee angle variation during the forward lunge over frames.

In addition, with the ongoing evolution of AI-based pose estimation models, we will closely follow research progress on more efficient and accurate skeleton tracking algorithms. When necessary, we plan to conduct model training based on our own datasets to improve generalization and tracking performance in specific application scenarios.

Apart from system performance, another limitation lies in the current skeleton rendering approach, which is based on keypoints rather than anatomically accurate skeletal structures. Given that this study focuses on analyzing limb motion patterns, the current representation is sufficient for practical motion tracking tasks. Moreover, the system supports multiple visualization formats, including point clouds and pose line diagrams, enabling comprehensive representation of individual movement characteristics.

In future work, anatomical fidelity and visual expressiveness could be enhanced by integrating structure-aware reconstruction methods such as OSSO (Obtaining Skeletal Shape from Outside) [37], enabling the system to generate anatomically meaningful human models. Such improvements could expand the system's applications in clinical, rehabilitative, and ergonomics domains, while also introducing challenges related to model complexity, sensor accuracy, and real-time performance.

Regarding evaluation methodology, the current system relies primarily on manually measured reference values for accuracy validation. Although this is scientifically valid, it is subject to inherent error and observability limitations, especially in privacy-sensitive regions. To achieve more rigorous validation, future work will incorporate gold-standard reference tools such as multi-view motion capture systems to obtain high-quality ground-truth data. Additionally, the evaluation scope will be expanded to include metrics such as point cloud quality. These improvements aim to provide a more objective and reliable assessment of system performance under real-world deployment conditions.

V. CONCLUSION

This study presents a real-time, full-body motion capture system based on augmented reality (AR), which effectively

overcomes the limitations of traditional systems in terms of complex deployment and high cost. It also addresses the constraints of existing AR-based solutions, such as dependency on specific motion types, wearable markers, and fixed observation perspectives. The system supports flexible visualization of motion data through various modalities, including skeletal models, point clouds, and joint angles, offering high adaptability and usability.

Experimental results demonstrate that the system achieves strong stability and real-time performance, while also exhibiting excellent accuracy, continuity, and consistency in kinematic tracking. By transforming motion capture into an easily deployable and highly immersive interactive tool, the system enhances spatial freedom in sports and medical analysis and significantly improves the efficiency of collecting motion-driven datasets for AI exploration. Furthermore, the modular and extensible architecture supports integration with other tracking models, providing a solid foundation for future applications in areas such as prosthetic research.

REFERENCES

[1] Eline Van der Kruk and Marco M Reijne. Accuracy of human motion capture systems for sport applications; state-of-the-art review. *European journal of sport science*, 18(6):806–819, 2018.

[2] Umile Giuseppe Longo, Sergio De Salvatore, Arianna Carnevale, Salvatore Maria Tecce, Benedetta Bandini, Alberto Lalli, Emiliano Schena, and Vincenzo Denaro. Optical motion capture systems for 3d kinematic analysis in patients with shoulder disorders. *International journal of environmental research and public health*, 19(19):12033, 2022.

[3] Kenan Niu, Victor Sluiter, Bangyu Lan, Jasper Homminga, André Sprengers, and Nico Verdonschot. A method to track 3d knee kinematics by multi-channel 3d-tracked a-mode ultrasound. *Sensors*, 24(8):2439, 2024.

[4] Harrison F Helmich, Charles J Doherty, Donald H Costello, and Michael DM Kutzer. Automatic ground-truth image labeling for deep neural network training and evaluation using industrial robotics and motion capture. *Journal of Verification, Validation and Uncertainty Quantification*, 9(2), 2024.

[5] Bangyu Lan, Momen Abayazid, Nico Verdonschot, Stefano Stramigioli, and Kenan Niu. Sirc-unet: Improving bone tracking precision of a-mode ultrasound by decoding hierarchical resolution features. *IEEE Sensors Journal*, 24(22):38174–38184, 2024.

[6] Kenan Niu, Jasper Homminga, Victor I Sluiter, André Sprengers, and Nico Verdonschot. Feasibility of a-mode ultrasound based intraoperative registration in computer-aided orthopedic surgery: A simulation and experimental study. *Plos One*, 13(6):e0199136, 2018.

[7] Kenan Niu, Victor Sluiter, Jasper Homminga, André Sprengers, and Nico Verdonschot. A novel ultrasound-based lower extremity motion tracking system. *Intelligent Orthopaedics: Artificial Intelligence and Smart Image-guided Technology for Orthopaedics*, pages 131–142, 2018.

[8] Hui Song, Xu Zhai, Zhongyang Gao, Teng Lu, Qian Tian, Haopeng Li, and Xijing He. Reliability and validity of a coda motion 3-d analysis system for measuring cervical range of motion in healthy subjects. *Journal of Electromyography and Kinesiology*, 38:56–66, 2018.

[9] Amadeusz Bartoszek, Artur Struzik, Sebastian Jaroszczuk, Marek Woźniewski, and Bogdan Pietraszewski. Comparison of the optoelectronic bts smart system and imu-based myomotion system for the assessment of gait variables. *Acta of Bioengineering and Biomechanics*, 24(1), 2022.

[10] Melisa Pilla-Barroso, Antonio R Jiménez-Ruiz, and Ana Jiménez-Martín. Estimation of movement in physical exercise programs using depth cameras: Validation against a gold standard. In *2024 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pages 1–6. IEEE, 2024.

[11] Saman Vafadar, Wafa Skalli, Aurore Bonnet-Lebrun, Marc Khalifé, Mathis Renaudin, Amine Hamza, and Laurent Gajny. A novel dataset and deep learning-based approach for marker-less motion capture during gait. *Gait & Posture*, 86:70–76, 2021.

[12] Lennard Marx, Vincent Groenhuis, Stefano Stramigioli, and Kenan Niu. Augmented-reality based digital twin to control an mr safe robot for breast biopsy: A benchmark study. In *2024 10th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, pages 814–819. IEEE, 2024.

[13] L Marx, S Stramigioli, K Niu, V Groenhuis, and W Brink. *Augmented reality based digital twin to control mri compatible robot*. PhD thesis, MSc Thesis, University of Twente, Enschede, the Netherlands, 2024.

[14] Viktor Vörös, Ruixuan Li, Ayoob Davoodi, Gauthier Wybaillie, Emmanuel Vander Poorten, and Kenan Niu. An augmented reality-based interaction scheme for robotic pedicle screw placement. *Journal of imaging*, 8(10):273, 2022.

[15] Roger C Woledge, Roisin Delaney, Matt Thornton, and Adam P Shortland. A comparison of normal data between two clinical gait analysis labs. *Journal for the motion-capture community Issue*, 2:05, 2005.

[16] Shaojun Yue. Human motion tracking and positioning for augmented reality. *Journal of Real-Time Image Processing*, 18(2):357–368, 2021.

[17] Pei-Hsuan Chiu, Po-Hsuan Tseng, and Kai-Ten Feng. Interactive mobile augmented reality system for image and hand motion tracking. *IEEE Transactions on Vehicular Technology*, 67(10):9995–10009, 2018.

[18] Jan Nikodem, Ryszard Klempous, Jerzy Rozenblit, Hubert Kowalczyk, Marek Kulbacki, Artur Bak, and Konrad Kluwak. Motion tracking in augmented and mixed realities for healthcare and medicine applications. In *International Conference on Computer Aided Systems Theory*, pages 113–124. Springer, 2024.

[19] Silvia Zaccardi, Anxhela Arapi, Taylor Frantz, Redona Brahimetaj, Ruben Debeuf, David Beckwée, Eva Swinnen, and Bart Jansen. Holomocap: Real-time clinical motion capture with hololens 2. In *2025 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pages 218–222. IEEE, 2025.

[20] Naheem Noah, Sommer Shearer, and Sanchari Das. Security and privacy evaluation of popular augmented and virtual reality technologies. In *Proceedings of the 2022 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence, and Neural Engineering (IEEE MetroXRAINE 2022)*, 2022.

[21] Youbing Hu, Zhijun Li, Yongrui Chen, Yun Cheng, Zhiqiang Cao, and Jie Liu. Content-aware adaptive device–cloud collaborative inference for object detection. *IEEE Internet of Things Journal*, 10(21):19087–19101, 2023.

[22] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2002.

[23] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[24] Dorin Ungureanu, Federica Bogo, Silvano Galliani, Pooja Sama, Xin Duan, Casey Meekhof, Jan Stühmer, Thomas J Cashman, Bugra Tekin, Johannes L Schönberger, et al. Hololens 2 research mode as a tool for computer vision research. *arXiv preprint arXiv:2008.11239*, 2020.

[25] Sean ONG. and Sean Ong. *Beginning windows mixed reality programming*. Springer, 2021.

[26] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

[27] Lei Liu, Alifu Kurban, and Yi Liu. Improved yolov11pose for posture estimation of xinjiang bactrian camels. *International Journal of Advanced Computer Science & Applications*, 15(12), 2024.

[28] Emre Güçlü, Erhan Akın, İlhan Aydın, Ahmet Topkaya, Mert Onan, and Taha Kubilay Şener. Real-time detection of terminal burn defects using yolov7 and tensorrt. In *2024 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 312–316. IEEE, 2024.

[29] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9157–9166, 2019.

[30] Jonas L Matzon, Kevin F Lutsky, Michael Maloney, and Pedro K Beredjiklian. Adherence to the aaos upper-extremity clinical practice guidelines. *Orthopedics*, 36(11):e1407–e1411, 2013.

[31] April Armstrong, Mark C Hubbard, et al. *AAOS Essentials of Musculoskeletal Care*. Jones & Bartlett Learning, 2018.

[32] Adolph J Yates Jr, Brian J McGrory, Terence W Starz, Kevin R Vincent, Brian McCardel, and Yvonne M Golightly. Aaos appropriate use criteria: optimizing the non-arthroplasty management of osteoarthritis of the knee. *JAAOS-Journal of the American Academy of Orthopaedic Surgeons*, 22(4):261–267, 2014.

[33] Pilar Fernández-González, Aikaterini Koutsou, Alicia Cuesta-Gómez, María Carratalá-Tejada, Juan Carlos Miangolarra-Page, and Francisco

Molina-Rueda. Reliability of kinovea® software and agreement with a three-dimensional motion system for gait analysis in healthy subjects. *Sensors*, 20(11):3154, 2020.

[34] Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. In *International Conference on Machine Learning*, pages 40605–40623. PMLR, 2023.

[35] David Webster and Ozkan Celik. Experimental evaluation of microsoft kinect's accuracy and capture rate for stroke rehabilitation applications. In *2014 IEEE Haptics Symposium (HAPTICS)*, pages 455–460. IEEE, 2014.

[36] Lothar Pantel and Lars C Wolf. On the impact of delay on real-time multiplayer games. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 23–29, 2002.

[37] Marilyn Keller, Silvia Zuffi, Michael J Black, and Sergi Pujades. Osso: Obtaining skeletal shape from outside. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20492–20501, 2022.

[38] Lu-hao He, Yong-zhang Zhou, Lei Liu, Wei Cao, and Jian-hua Ma. Research on object detection and recognition in remote sensing images based on yolov11. *Scientific Reports*, 15(1):14032, 2025.

[39] Priyanto Hidayatullah, Nurjannah Syakrani, Muhammad Rizqi Sholahuddin, Trisna Gelar, and Refdinal Tubagus. Yolov8 to yolo11: A comprehensive architecture in-depth comparative review. *arXiv preprint arXiv:2501.13400*, 2025.

[40] Shun Cheng, Yan Han, Zhiqian Wang, Shaojin Liu, Bo Yang, and Jianrong Li. An underwater object recognition system based on improved yolov11. *Electronics*, 14(1):201, 2025.

APPENDIX I
THE ANALYSIS OF INITIAL HARDWARE ARCHITECTURE

Since our solution focuses on being lightweight, we considered using HoloLens 2 as the sole edge device. The initial hardware architecture is shown in the Figure 15. HoloLens 2 is equipped with multiple cameras, including a depth camera that outputs depth maps and brightness images, and four environment tracking cameras that output grayscale images. The official SDK also provides access to these data streams. Based on these capabilities, we planned to rely entirely on HoloLens 2's built-in sensor system to complete the perception tasks.

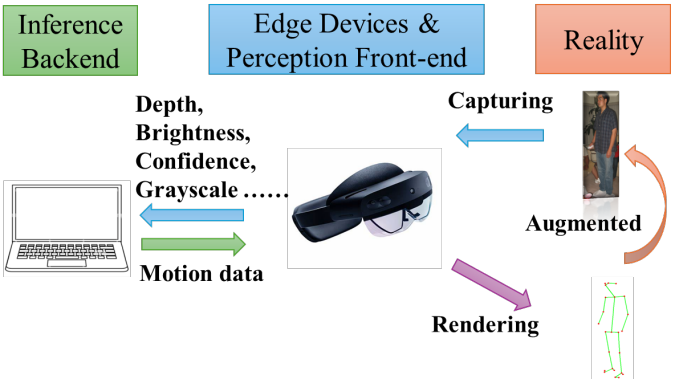


Fig. 15: Initial hardware architecture of the system. The Microsoft HoloLens 2 is used as the perception front-end and the edge platform. Inference is performed on an NVIDIA GeForce RTX 3060 Laptop GPU.

The depth camera on the HoloLens 2 supports two operational modes. The first is the Near-Range mode, also known as AHAT mode, which offers a high frame rate of approximately 45 FPS and is optimized for hand tracking within a range of one meter. The second is the Long-Throw mode, designed for spatial mapping at greater distances; however, it operates at a much lower frame rate of around 1 to 5 FPS, making it less suitable for real-time motion tracking applications.

We attempted to process the data stream captured in long-throw mode, the whole process is shown in Figure18, but the results were not ideal. Although the official specs suggest an effective range of up to 5 meters, our tests showed that the depth values became significantly unreliable beyond 1.8 meters. Also, because of the depth camera's limited field of view (FOV) as shown in Figure16, capturing a full view of a person's torso with HoloLens requires standing at a considerable distance. This not only worsens the depth data quality but also makes the person appear very small in the depth map. On top of that, the brightness buffer provides only binary images, making it even harder for the YOLO model to detect the human body.

Even if we could solve all these issues, the 5 FPS data rate still makes it very difficult to achieve real-time performance. Based on this, we concluded that the HoloLens Depth camera is more suitable for close-range tasks, like hand gesture tracking combined with MediaPipe, which is shown in Figure17a.

Since the depth camera approach did not yield satisfactory results, we shifted to utilizing the environment grayscale cameras for body tracking. As illustrated in Figure 19, this method combines grayscale imagery with ArUco markers to acquire reference depth information, and employs the MiDaS framework for relative depth estimation to enable full-body pose tracking.

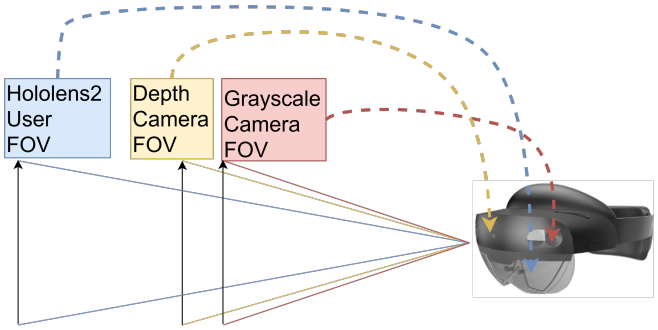
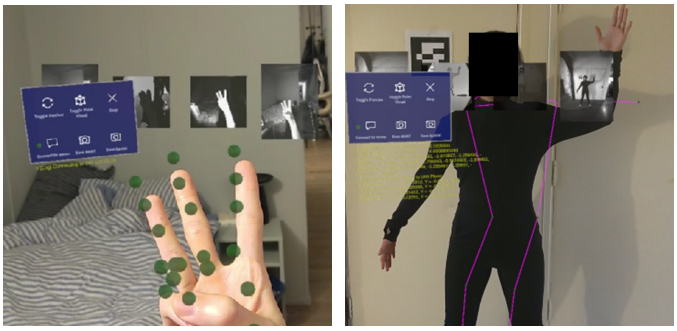


Fig. 16: Hololens2 and its own sensor FOV (field of view)



(a) Depth camera tracking results (b) Grayscale camera results

Fig. 17: The results of the initial architecture

Compared to using depth cameras alone, this approach showed moderate improvements; however, the overall performance remained suboptimal. First, in order to capture the entire human body, the subject had to stand at a considerable distance from the cameras, resulting in a small human figure within the grayscale image. The lack of color and fine visual details further complicated detection and significantly degraded the pose estimation accuracy. Second, the inference pipeline was sequential in nature—each stage depended on the output of the previous one—which introduced notable system latency. In addition, because the input consisted solely of grayscale images, the subject's clothing needed to exhibit strong contrast against the background, thereby limiting the method's practical applicability and generalizability in diverse environments.

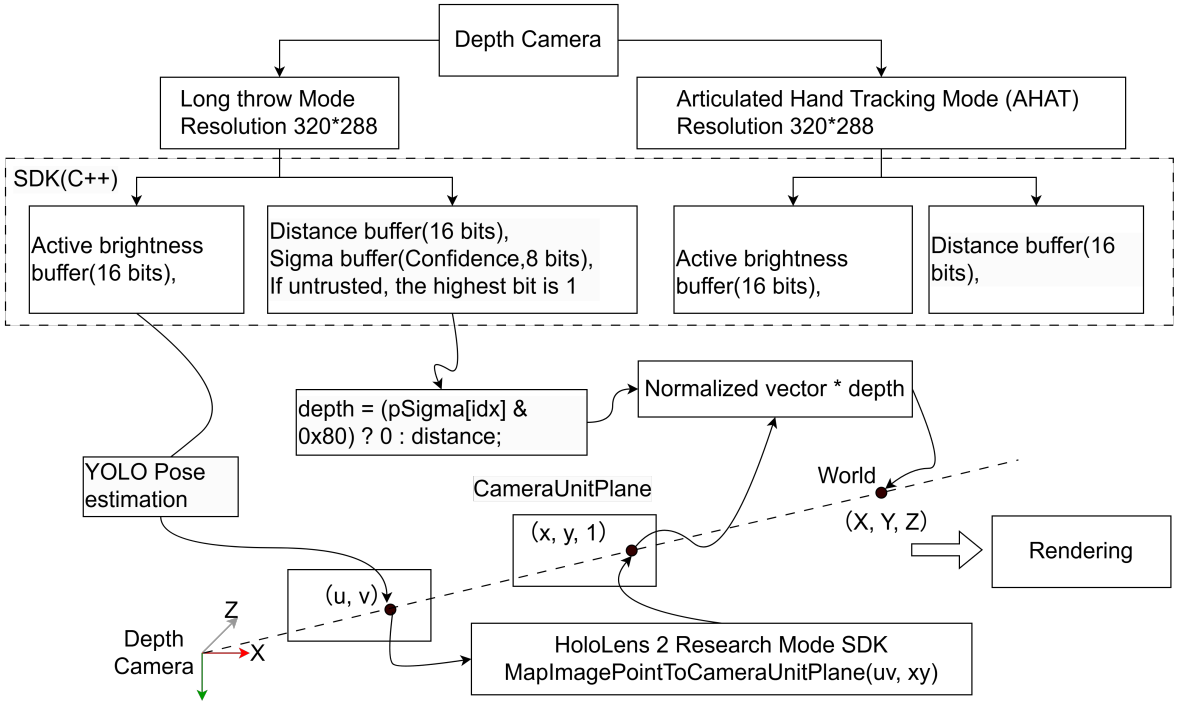


Fig. 18: Depth camera tracking process. The depth camera collects depth and brightness images in long-distance mode and extracts depth and credibility information. YOLO is used for posture estimation, and the pixel coordinates of the target in the image are determined in combination with brightness information. The pixel coordinates are mapped to the camera unit plane and then multiplied by the depth value to obtain the 3D spatial position for subsequent rendering.

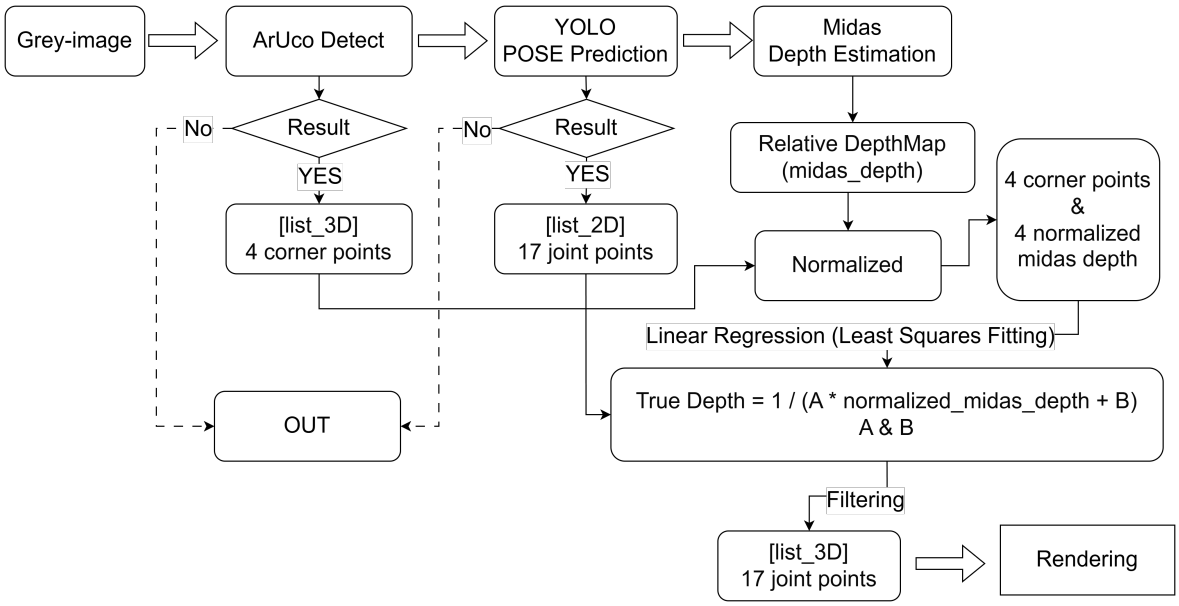


Fig. 19: Grey camera tracking process. ArUco markers and human keypoints (via YOLO pose estimation) are detected from the grayscale image to obtain initial 2D or 3D feature points. The MiDaS network is then used to estimate the relative depth map of the entire image, and a linear relationship is fitted using the marker points to infer the true depth. Finally, the filtered 3D coordinates are output for final rendering.

Considering sensing quality, recognition accuracy, real-time performance, and system usability, this solution was not good enough for real-world application, so we eventually abandoned it. However, through this process, we clearly identified the bottlenecks in building a tracking system, which led us to adopt an independent RealSense 435i camera in the next stage.

APPENDIX II
YOLOV11 ARCHITECTURE ADVANTAGES

YOLO (You Only Look Once) has iterated through 11 versions since its first release in 2015. Each version has been developed independently by different organizations or research teams. In the early days of YOLOv1 to YOLOv4, models were based on the Darknet framework, which made modification, training, and deployment difficult, especially for beginners and developers.

Starting from YOLOv5, Ultralytics re-implemented the model entirely in PyTorch, making it simpler, more modern, and ready to use out of the box. Thanks to continuous technical advancements, an active open-source community, and fast team maintenance, the Ultralytics versions of YOLO have become the most widely adopted, officially recognized, and fastest-evolving branches in the industry today.

In 2023, Ultralytics officially released YOLOv8, marking the first truly ”revolutionary” upgrade in YOLO’s history. YOLOv8 introduced a fully Anchor-Free detection approach; whereas YOLOv1 to YOLOv7 relied on Anchor-based detection — first generating predefined bounding box templates (anchor boxes) and then refining them — YOLOv8 instead directly predicts center points (Center-based) and width-height pairs (Width-Height regression). It also adopted a dynamic label assignment mechanism, greatly reducing the need for hyperparameter tuning. Additionally, YOLOv8 expanded its capabilities beyond object detection to also support instance segmentation, pose estimation, and object tracking tasks in a unified framework.

This work is based on YOLOv11, the latest version released by Ultralytics in 2024. Building upon the significant improvements already achieved by YOLOv8, YOLOv11 introduces major upgrades in architecture and training strategies, resulting in even stronger overall performance, making it a versatile choice for a wide range of computer vision tasks.

A. Stronger Backbone Feature Extraction Capability

YOLOv8’s backbone utilizes the C2f module (an improved version of CSP structures), which is lightweight and fast but somewhat limited when dealing with complex textures or

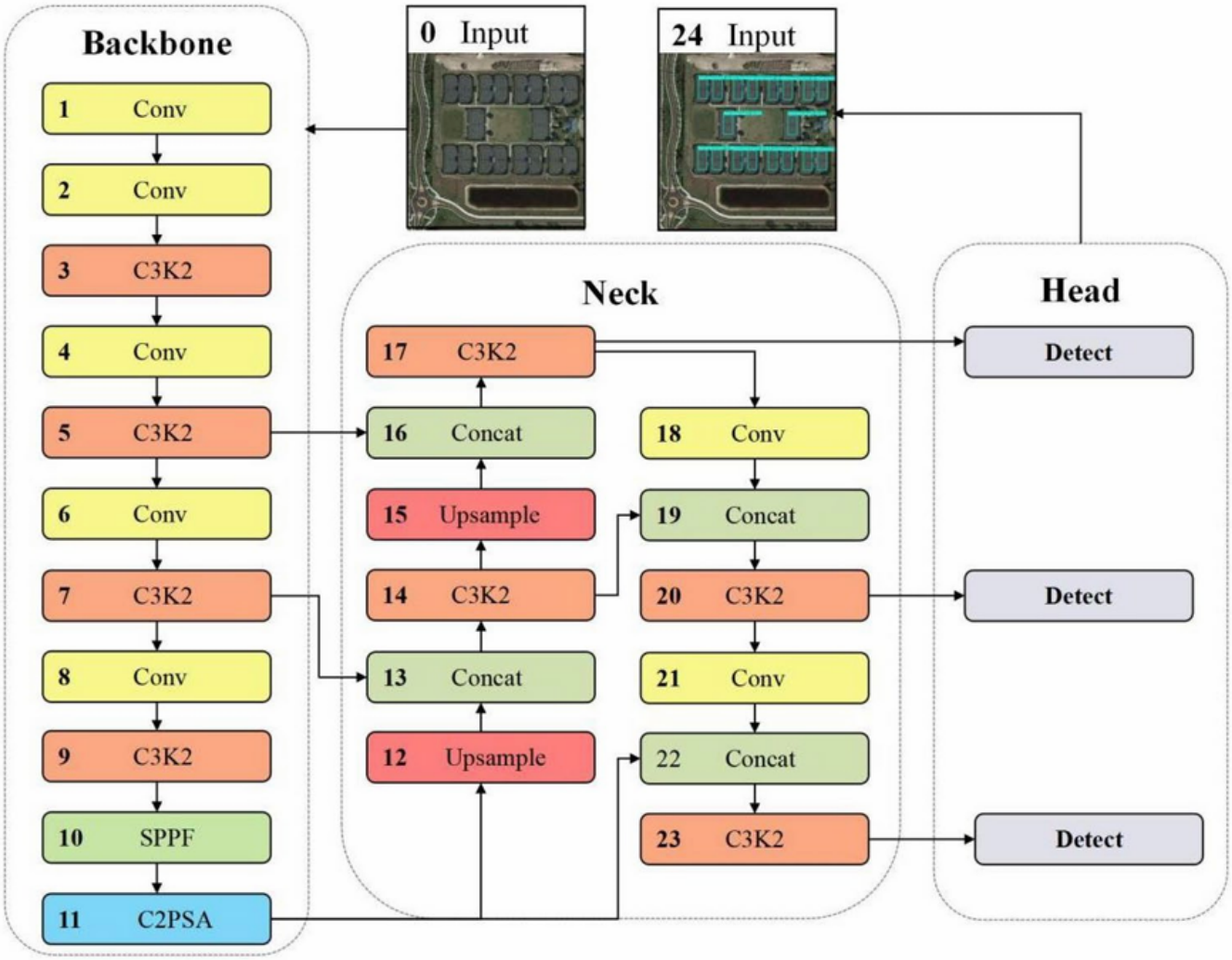


Fig. 20: YOLOv11 model architecture [38]

small object detection [39]. YOLOv11 introduces the new C3k2 module, an upgrade over C2f, which further reduces parameter count while incorporating additional convolution and bottleneck structures. This allows it to capture richer and deeper multi-scale and complex features. Moreover, with the c3k parameter, the model can flexibly choose whether to employ deeper C3k structures, significantly boosting accuracy in detecting small objects and handling complex scenes.

B. Smarter Feature Fusion (Neck)

The Neck component in YOLOv8 adopts a simple FPN+PAN structure, achieving multi-scale feature fusion but lacking an intelligent attention mechanism. YOLOv11, while still utilizing the lightweight and fast SPPF module, introduces the C2PSA Attention module (Convolutional Parallel Spatial Attention). Unlike YOLOv8, which does not incorporate any attention mechanism, YOLOv11's addition of C2PSA enables adaptive focusing on important feature regions, significantly enhancing feature fusion, especially for detecting overlapping objects and handling small objects in complex backgrounds.

C. Faster and Lighter Detection Head

In YOLOv8, both the classification and regression heads use standard convolutional layers followed by a SiLU activation. In contrast, YOLOv11 replaces the classification branch with Depthwise Convolution (DWConv) [40], greatly reducing computational cost while maintaining classification accuracy. By simplifying the classifier using DWConv, YOLOv11 lowers computational complexity and achieves faster inference speeds.

As a result of these architectural improvements, YOLOv11m achieves higher mean Average Precision (mAP) on the COCO dataset while reducing the number of parameters by 22% compared to YOLOv8m, thereby significantly improving computational efficiency without sacrificing accuracy. Therefore, this paper selects YOLOv11 as the base model for motion capture, extending it into YOLOv11-Pose and YOLOv11-Segmentation for skeleton tracking and point cloud segmentation in human motion analysis.