

DOI: 10.37943/TV013333

Gulnur Kazbekova

Candidate of Technical Sciences, Associate Professor, Department of Computer Engineering

gulnur.kazbekova@ayu.edu.kz, orcid.org/0000-0002-2756-7926

Khoja Akhmet Yassawi International Kazakh-Turkish University, Kazakhstan

Yerlan Serdaliyev

PhD Student, Department of Computer Engineering

erlan.serdaliyev@ayu.edu.kz, orcid.org/0009-0007-5403-0423

Khoja Akhmet Yassawi International Kazakh-Turkish University, Kazakhstan

Zhanar Kemelbekova

Candidate of Technical Sciences, Professor, Department of Computer Science

zhanar.kemelbekova@auezov.edu.kz, orcid.org/0000-0001-9422-2509

M. Auezov South Kazakhstan Research University, Kazakhstan

Gulmira Omarova

PhD, Senior Lecturer, Department of Information Systems

ogs12@mail.ru, orcid.org/0000-0003-2053-0255

L.N. Gumilyov Eurasian National University, Kazakhstan

Yergali Kurmangaliyev

Candidate of Physical and Mathematical Sciences, Senior Lecturer,

Department of Computer Science and Information Technologies

ye.kurmangaliyev@zhubanov.edu.kz, orcid.org/0000-0002-3583-9215

K. Zhubanov Aktobe Regional University, Kazakhstan

Arypzhan Aben

PhD Student, Department of Computer Engineering

arypzhan.aben@ayu.edu.kz, orcid.org/0000-0001-8534-3288

Khoja Akhmet Yassawi International Kazakh-Turkish University, Kazakhstan

AN EFFECTIVE METHOD FOR ANALYZING HUMAN MOVEMENT IN REAL TIME BASED ON YOLO-ROI

Abstract: Real-time human movement analysis has become an essential component of modern computer vision applications, including sports performance assessment, rehabilitation monitoring, intelligent surveillance, and human–computer interaction. Although deep learning–based pose estimation methods provide accurate human body landmark detection, executing person detection on every video frame introduces considerable computational overhead, limiting their applicability on resource-constrained CPU-based systems.

This paper proposes an efficient YOLO–ROI framework for real-time human movement analysis that combines scheduled YOLO person detection, region-of-interest (ROI) generation, MediaPipe Pose estimation, and lightweight landmark smoothing. Instead of performing object detection on every frame, the proposed framework periodically executes the YOLO detector while reusing the most recently detected ROI between detection cycles. Consequently, MediaPipe Pose continuously estimates body landmarks within the ROI, significantly reducing redundant object detection operations and improving computational efficiency.

The proposed framework was evaluated using a self-collected dataset consisting of 200 videos recorded from 10 participants performing five physical exercises under controlled indoor conditions. Computational performance was analyzed under different ROI sizes, detection intervals, and MediaPipe Pose complexity levels using CPU-based execution.

Experimental results demonstrate that the proposed optimization strategy achieves a maximum processing speed of 61.32 FPS with an average processing latency of 16.31 ms, enabling stable real-time operation without GPU acceleration. The obtained results further show that scheduled YOLO detection, ROI reuse, and lightweight landmark smoothing effectively reduce computational workload while preserving continuous pose estimation throughout video processing.

The proposed framework provides a simple, computationally efficient, and easily deployable solution for real-time human movement analysis and can be readily integrated into sports analytics, rehabilitation systems, intelligent surveillance applications, and edge-based computer vision platforms.

Keywords: YOLO, ROI, real-time processing, human motion analysis, computer vision, pose estimation, edge devices, latency optimization

Introduction (Literary review)

Automatic human movement analysis supports applications in intelligent surveillance, healthcare, sports training, rehabilitation, and human–computer interaction. Human pose estimation represents body posture through anatomical keypoints that can be used for subsequent movement analysis. Chen et al. [1] review 2D pose estimation methods, datasets, evaluation metrics, and post-processing strategies, providing the methodological background for this representation. Gao et al. [2] extend this perspective to upstream and downstream tasks and lightweight models, placing computationally efficient movement analysis within the wider pose-estimation workflow.

Several complementary approaches provide the foundation for the proposed pipeline. Cao et al. [3] introduced OpenPose, which associates body parts with individuals through part affinity fields for multi-person 2D pose estimation. Ahmadyan et al. [4] developed BlazePose, a lightweight architecture that estimates 33 body keypoints and targets real-time inference on mobile devices; this work motivates the use of an on-device pose estimator. Lugaresi et al. [5] presented MediaPipe as a framework for combining perception components and measuring performance across platforms, supporting the modular implementation used here.

The YOLO family provides the object-detection background for person localization in this study. Redmon and Farhadi [6] describe the accuracy and runtime characteristics of YOLOv3, while Bochkovskiy et al. [7] investigate architectural and training choices for real-time detection in YOLOv4. C.-Y. Wang et al. [8] further examine detector efficiency through trainable architectural and training improvements in YOLOv7. These foundational detector studies motivate the attention to inference cost. The present implementation uses the YOLOv8n detection model from Ultralytics YOLOv8 [9]; its scheduling and ROI-reuse strategy is the design evaluated below.

Spatial context and temporal information address different aspects of pose estimation. Chu et al. [10] combine holistic and body-part attention to model spatial context within an image, illustrating a complementary approach to keypoint localization. Pavllo et al. [11] use temporal convolutions over sequences of 2D keypoints to estimate 3D poses, demonstrating how consecutive frames can support learned temporal modeling. The present study uses exponential moving average smoothing of the estimated landmarks and evaluates runtime optimization through scheduled person detection and ROI reuse.

Architectural studies provide further alternatives to the pose stage. Xiao et al. [12] establish simple baselines for pose estimation and tracking, supporting the use of explicit reference methods when assessing added system complexity. J. Wang et al. [13] maintain high-resolution representations throughout a network, while Xu et al. [14] investigate a plain vision transformer for pose estimation. These approaches concern the estimator architecture; the present study instead examines the execution schedule of an existing detector–pose pipeline. Lu et al. [15] evaluate real-time pose estimation across deployment platforms in RTMPose, highlighting the need to report hardware and runtime settings when interpreting inference speed.

Recent applied studies also motivate task-specific validation. Kim et al. [16] combine MediaPipe Pose with optimization based on a humanoid model, illustrating an approach to improving pose estimates after inference. Hii et al. [17] evaluate marker-free pose estimation for automated gait analysis, linking landmarks to movement-specific measurements. Dill et al. [18] assess MediaPipe-based 3D reconstruction for physical exercises using stereo camera information. These studies motivate future assessment of landmark and movement-measurement accuracy alongside runtime; they do not establish the accuracy of the configuration tested here.

To address these limitations, this paper proposes an effective YOLO-ROI-based framework for real-time human movement analysis. The proposed framework first detects the human subject using YOLO, extracts a region of interest (ROI), and performs pose estimation only within the selected ROI using the MediaPipe Pose model. Instead of executing object detection for every video frame, the proposed method employs a scheduled detection strategy in which YOLO inference is performed only at predefined frame intervals while pose estimation is continuously executed inside the ROI. This strategy significantly reduces redundant computations, decreases processing latency, and improves computational efficiency while preserving stable human movement analysis.

The main contributions of this work can be summarized as follows. First, an efficient YOLO-ROI framework is proposed to improve the computational efficiency of real-time human movement analysis by restricting pose estimation to dynamically selected regions of interest instead of processing the entire image. Second, a scheduled object detection strategy is introduced to reduce redundant YOLO inference while maintaining continuous analysis of human movement in video streams. Third, a comprehensive experimental evaluation is conducted under different ROI sizes, object detection intervals, and MediaPipe Pose complexity levels to investigate their influence on processing speed, latency, and memory consumption. Finally, the proposed implementation demonstrates that the combination of ROI-based pose estimation and scheduled object detection provides an effective CPU-based solution suitable for real-time human movement analysis in resource-constrained environments.

Methods and Materials

The proposed framework is designed as a computationally efficient pipeline for real-time human movement analysis in continuous video streams. Its modular organization follows the perception-pipeline concept described by Lugaresi et al. [5]. Person localization is implemented with the YOLOv8n detector from Ultralytics YOLOv8 [9]. The combination of scheduled detection, square ROI extraction, pose estimation, and temporal landmark smoothing is evaluated in the experiments reported below.

For each incoming video frame X_t , the system determines whether YOLO person detection should be executed according to a predefined detection interval. When detection is performed, the bounding box of the person with the highest confidence score is stored as the latest valid subject location. During intermediate frames, the previously stored bounding box is reused, allowing the system to avoid repeated YOLO inference while continuing pose estimation inside the corresponding ROI. This strategy significantly reduces computational overhead while maintaining continuous processing of human movement.

A square ROI is generated around the center of the latest valid bounding box. The ROI coordinates are constrained to the valid frame boundaries, and the resulting cropped region is provided to the MediaPipe Pose model. The 33-landmark body representation is described by Ahmadyan et al. [4]. This source establishes the landmark representation used for subsequent movement analysis; the ROI size and detection interval are implementation choices evaluated in this study.

To improve landmark stability, the extracted landmark coordinates are temporally smoothed using an exponential moving average (EMA), defined in Equation (12). This implementation is intended to reduce frame-to-frame fluctuations with a small computational cost. The temporal convolutions described by Pavllo et al. [11] instead form part of a learned 3D pose-

estimation model. This distinction separates coordinate-level filtering from learned temporal modeling; the cited work does not validate the present EMA implementation.

The overall processing sequence can be expressed as

$$Y_t = S_\alpha \left(P \left(R \left(B_t(F_t) \right) \right) \right) \quad (1)$$

where X_t denotes the input video frame at time t , $\hat{B}_t$ is the latest valid person bounding box obtained from scheduled YOLO detection, $R(\cdot)$ denotes the ROI extraction function, $P(\cdot)$ represents MediaPipe Pose estimation, $S_\alpha(\cdot)$ denotes exponential moving average smoothing, and Y_t is the resulting smoothed pose-based movement representation.

The architecture of the proposed framework is illustrated in Figure 1. Unlike conventional approaches that repeatedly process the entire image, the proposed framework performs pose estimation only inside a periodically updated ROI while reusing the latest valid person location between scheduled detections.

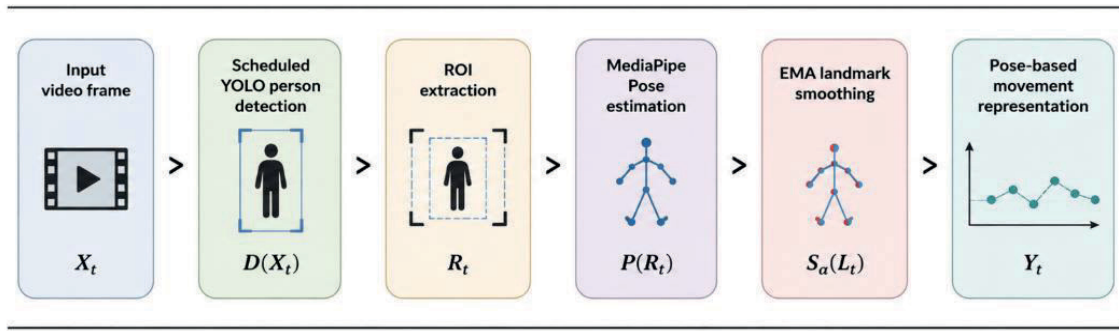


Figure 1. General architecture of the proposed framework

This design reduces the number of YOLO invocations and restricts the input supplied to the pose-estimation stage. The resulting runtime performance is assessed using the configurations and measurements reported in Table 2.

Human detection in the proposed framework is performed using the YOLOv8n detection model implemented through the Ultralytics library [9]. This software provides the detector used at this stage. The detector receives the complete video frame and returns candidate object bounding boxes together with their class labels and confidence scores. Since the framework is designed for single-person human movement analysis, only detections belonging to the person class are considered.

For an input frame X_t , the YOLO detector produces a set of candidate detections

$$D_t = \{(B_{t,i}, c_{t,i}, q_{t,i})\}_{i=1}^{M_t} \quad (2)$$

where M_t is the number of detected objects, $B_{t,i}$ denotes the bounding box of the i -th object, $c_{t,i}$ is its class label, and $q_{t,i}$ represents the corresponding confidence score.

Each bounding box is represented as

$$B_{t,i} = (x_{1,t,i}, y_{1,t,i}, x_{2,t,i}, y_{2,t,i}), \quad (3)$$

where $(x_{1,t,i}, y_{1,t,i})$ and $(x_{2,t,i}, y_{2,t,i})$ denote the upper-left and lower-right coordinates of the detected object, respectively.

Among all detections classified as a person, the framework selects the bounding box with the highest confidence score:

$$B_t = \arg \max_{i: ct, i = \text{person}} q_{t,i} \quad (4)$$

The selected bounding box is stored as the latest valid person location and is subsequently used for ROI generation. A detection is accepted only when its confidence score exceeds the predefined confidence threshold. In the implemented pipeline, YOLO inference is performed with an input resolution of 640×640 pixels, while the confidence threshold is supplied as an experimental parameter.

The current implementation does not perform identity association or multi-person tracking. When several people are visible in the frame, the person with the highest YOLO confidence score is selected. ByteTrack associates detection boxes across frames to maintain object trajectories [19], whereas AlphaPose jointly addresses whole-body pose estimation and identity tracking [20]. These works clarify the additional association mechanisms that would be required for a multi-person extension. The present experiments use one subject performing movements in front of the camera.

After human detection, the proposed framework constructs a square region of interest (ROI) centered on the detected person. Subsequent pose estimation is performed on the cropped region. The person-localization and pose-tracking approach described by Ahmadyan et al. [4] provides methodological background for region-based processing. The external YOLO scheduling and fixed-size ROI construction described here are specific to the present implementation.

Let the selected person bounding box be represented by

$$B_t = (x_1, y_1, x_2, y_2), \quad (5)$$

where (x_1, y_1) and (x_2, y_2) denote the upper-left and lower-right coordinates of the detected person.

The center of the detected person is computed as

$$c_x = \frac{x_1 + x_2}{2}, c_y = \frac{y_1 + y_2}{2}, \quad (6)$$

where (c_x, c_y) represents the center of the bounding box.

A square ROI with side length S is then generated around this center:

$$R_t = \left(c_x - \frac{S}{2}, c_y - \frac{S}{2}, c_x + \frac{S}{2}, c_y + \frac{S}{2} \right), \quad (7)$$

where S denotes the predefined ROI size. In this study, three ROI sizes (160×160 , 192×192 , and 256×256 pixels) were experimentally evaluated to investigate the trade-off between computational efficiency and pose-estimation quality.

Because the calculated ROI may extend beyond the image boundaries, the ROI coordinates are constrained to the valid frame dimensions before cropping. Consequently, all extracted ROIs remain completely inside the original image regardless of the subject position.

After ROI extraction, only the cropped image is forwarded to the MediaPipe Pose model for landmark estimation. Compared with full-frame processing, this strategy substantially decreases the computational cost of pose estimation while preserving the body region required for continuous human movement analysis.

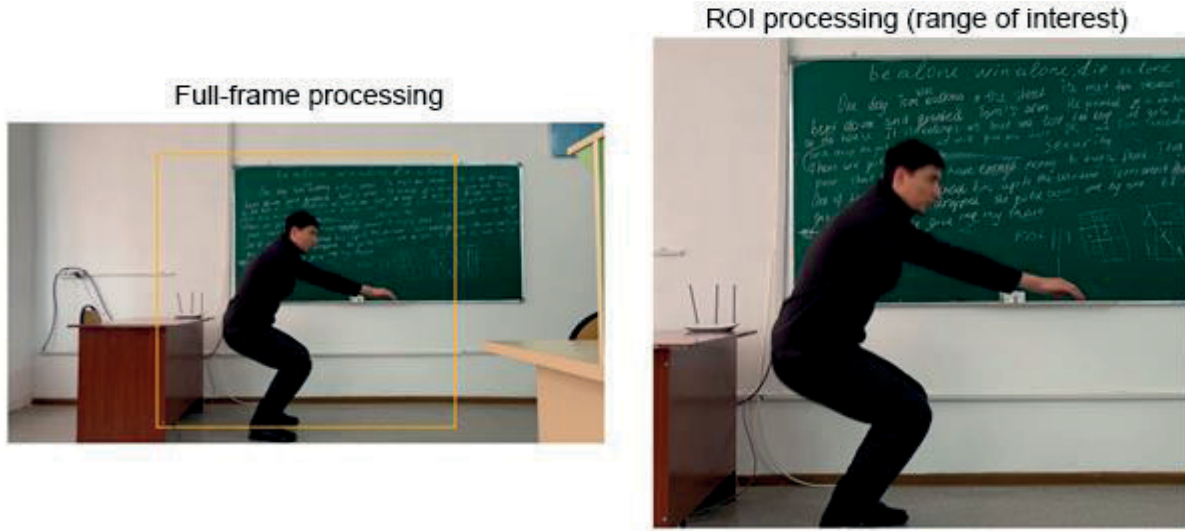


Figure 2. Comparison between conventional full-frame processing and the proposed ROI-based pose estimation

One of the main objectives of the proposed framework is to reduce the computational cost associated with executing the YOLO detector on every video frame. The proposed system therefore applies a scheduled detection strategy: YOLO inference is executed at predefined frame intervals, while pose estimation continues using the most recently detected ROI. This scheduling rule is evaluated as part of the present method, and its computational effect is measured in the experimental section.

Let N_d denote the detection interval measured in video frames. For each incoming frame with index t , YOLO detection is performed only when

$$D_t = \begin{cases} 1, & t \bmod N_d = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (8)$$

where D_t indicates whether the YOLO detector is activated for frame t . In the experimental evaluation, detection intervals of 10, 15, and 20 frames were investigated to analyze the influence of detection frequency on computational performance.

Whenever YOLO successfully detects a person, the bounding box with the highest confidence score is stored as the latest valid person location

$$\hat{B}_t = B_t, \quad (9)$$

where $\hat{B}_t$ represents the stored bounding box used by the subsequent processing stages. During the intermediate frames between two scheduled detections, the stored bounding box is reused for ROI generation without invoking the YOLO detector again. Consequently, MediaPipe Pose continues estimating body landmarks inside the previously detected ROI while avoiding unnecessary object detection.

If no valid person is detected during a scheduled YOLO invocation, the framework retains the previously stored bounding box. Before the first successful person detection, a centered fallback region covering approximately 60% of the shorter image dimension is used to initialize the ROI. This mechanism ensures that pose estimation can still be performed even when an initial person detection is temporarily unavailable.

The proposed implementation does not employ object tracking, motion prediction, optical flow, or landmark-based ROI updating. Instead, the ROI remains unchanged between two

scheduled YOLO detections and is updated only after the next successful detection. Consequently, the proposed framework is primarily intended for continuous single-person movement analysis, where the subject remains within or near the previously detected region during the selected detection interval.

Figure 3 illustrates an example of person detection and ROI generation using the proposed YOLO-based framework, while Figure 4 presents the scheduled detection strategy adopted in this study. By reducing the number of YOLO executions while maintaining continuous pose estimation inside the ROI, the proposed framework substantially decreases computational complexity and improves real-time performance on CPU-based systems.

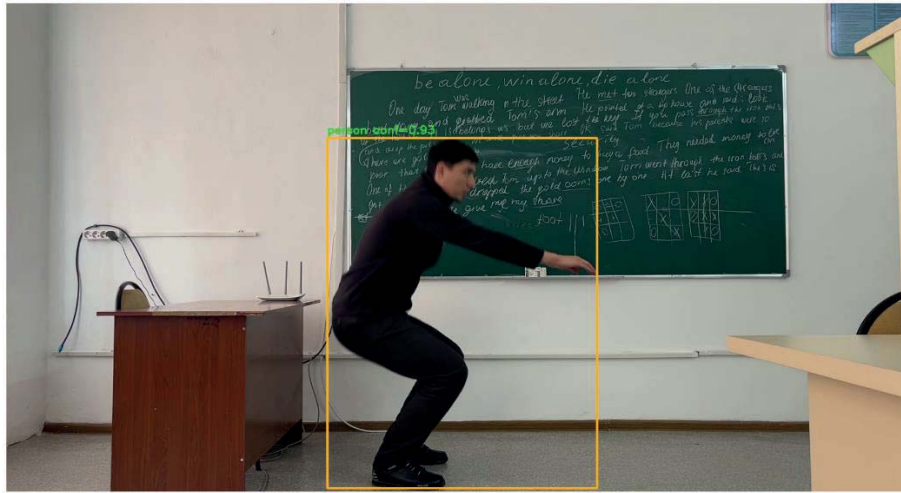


Figure 3. Example of ROI determination using YOLO

After ROI extraction, the cropped image is processed using MediaPipe Pose to estimate human body landmarks. Ahmadyan et al. [4] establish the lightweight on-device architecture and its 33-landmark output. These properties motivate the selection of MediaPipe Pose for this pipeline; performance on the CPU platform used in this study is assessed separately in the experiments.

For each ROI, MediaPipe estimates a set of body landmarks

$$L_t = \{l_{t,1}, l_{t,2}, \dots, l_{t,33}\}, \quad (10)$$

where each landmark

$$l_{t,j} = (x_{t,j}, y_{t,j}, z_{t,j}, v_{t,j}), j = 1, \dots, 33, \quad (11)$$

contains the normalized horizontal coordinate x , vertical coordinate y , relative depth coordinate z , and visibility score v . Together, these landmarks provide a compact representation of the subject's body posture for each video frame.

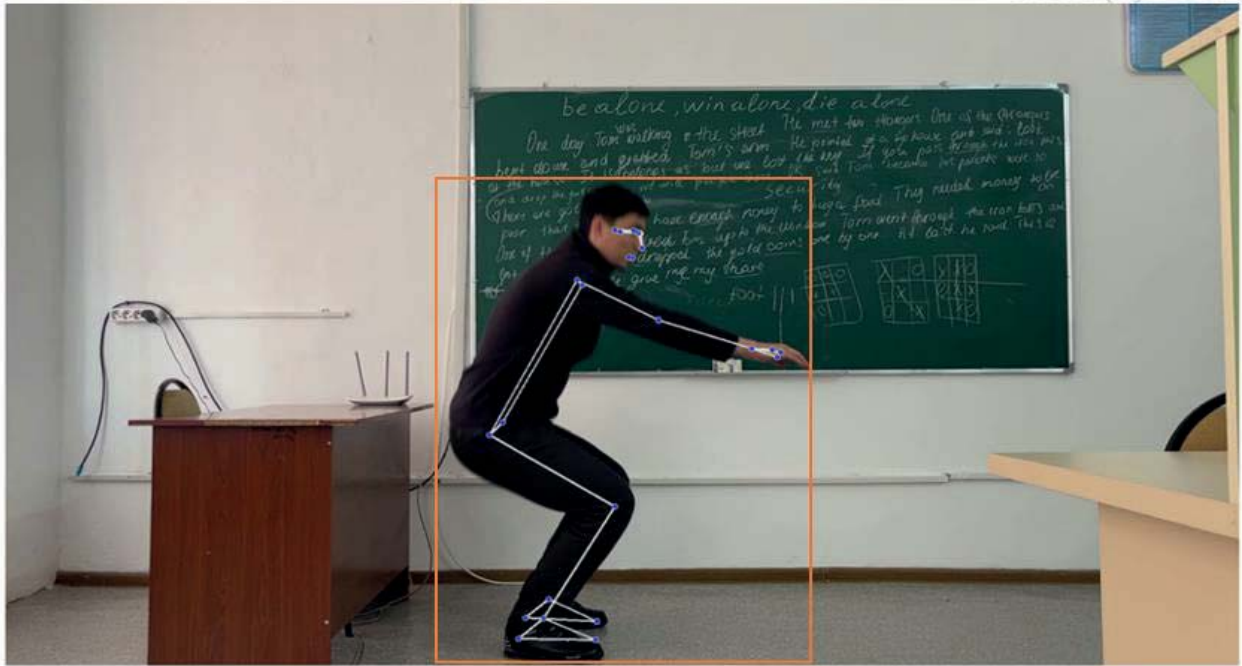


Figure 4. Example of human pose estimation within the extracted ROI using the proposed YOLO–ROI–MediaPipe framework

Although MediaPipe provides stable landmark estimation, small frame-to-frame fluctuations may still occur because of image noise, slight body oscillations, and estimation uncertainty. To reduce these variations while preserving continuous movement information, the proposed framework applies exponential moving average (EMA) smoothing to the landmark coordinates.

For each landmark coordinate, the smoothed value is computed as

$$\bar{L}_t = (1 - \alpha)\bar{L}_{t-1} + \alpha L_t, \quad (12)$$

where L_t denotes the landmark coordinates estimated for the current frame, $\bar{L}_{t-1}$ represents the previously smoothed landmarks, and α is the smoothing coefficient. In the proposed implementation, $\alpha = 0.35$, which provides an appropriate balance between landmark stability and responsiveness to body movement.

Unlike temporal deep-learning models, EMA smoothing introduces only negligible computational overhead while effectively suppressing short-term landmark fluctuations. Consequently, the proposed framework preserves the continuity of human movement representation without increasing the computational complexity of the overall system.

Figure 5 illustrates the scheduled YOLO detection strategy adopted in the proposed framework, where the latest detected ROI is reused between consecutive detection cycles while MediaPipe Pose continues processing every frame.

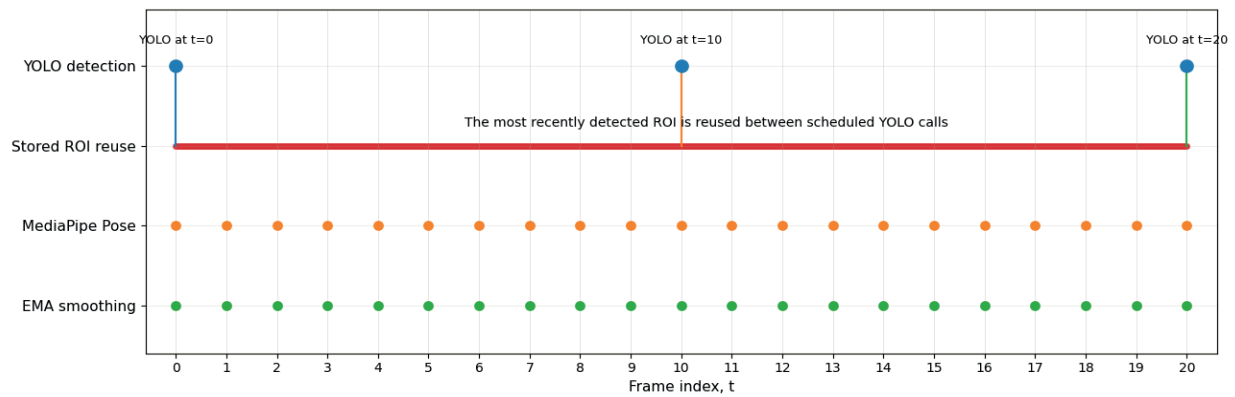


Figure 5. Scheduled YOLO detection and ROI reuse strategy adopted in the proposed framework

The proposed framework was evaluated using a self-collected human movement dataset specifically recorded for real-time pose-based movement analysis. The dataset consists of 200 video sequences acquired from 10 participants performing five common physical exercises, including squats, push-ups, bent-over rows, jumping jacks, and deadlifts. Each participant performed multiple repetitions under different recording sessions to capture natural variations in movement execution. Videos were recorded from front and side viewpoints using a standard RGB camera under indoor laboratory conditions. The dataset contains both correctly and incorrectly performed exercises, enabling evaluation under realistic movement scenarios. All participants voluntarily agreed to participate in the data collection process, and the recorded videos were used exclusively for academic research purposes.

All experiments were conducted on a desktop computer running the Windows operating system using Python as the primary development environment. The proposed framework was implemented using the Ultralytics YOLOv8n model for person detection, MediaPipe Pose for body landmark estimation, and OpenCV for image acquisition and preprocessing. Numerical computations and experimental analysis were performed using the NumPy library. During benchmarking, the YOLO detector operated with an input resolution of 640×640 pixels and a confidence threshold of 0.25, while MediaPipe Pose was configured with minimum detection and tracking confidence values of 0.5. Benchmark measurements included a warm-up stage of 60 frames followed by 300 measured frames for each experimental configuration.

To investigate the influence of different runtime configurations, several combinations of processing parameters were evaluated. The ROI size was varied among 160×160 , 192×192 , and 256×256 pixels, while the YOLO detection interval was set to 10, 15, and 20 frames. In addition, three MediaPipe Pose complexity levels (0, 1, and 2) were investigated to analyze the trade-off between computational efficiency and pose estimation quality. The benchmark implementation also included reference configurations without ROI optimization and with different processing variants to enable a comprehensive comparison of computational performance.

The performance of the proposed framework was evaluated using computational efficiency metrics, including average processing latency per frame, median latency (P50), 95th percentile latency (P95), YOLO detection latency, peak memory consumption, and the estimated processing speed measured in frames per second (FPS). These metrics provide a comprehensive assessment of the computational performance of the proposed framework under different runtime configurations and demonstrate its suitability for real-time human movement analysis on CPU-based systems.

Table 1. Experimental configuration of the proposed framework

Parameter	Value
Dataset	Self-collected human movement dataset
Number of videos	200
Participants	10
Exercise categories	Squat, Push-up, Bent-over Row, Jumping Jack, Deadlift
Camera	RGB camera
Recording environment	Indoor laboratory
YOLO model	YOLOv8n
YOLO input resolution	640 × 640
Confidence threshold	0.25
MediaPipe detection confidence	0.50
MediaPipe tracking confidence	0.50
ROI sizes	160, 192, 256 pixels
Detection intervals	10, 15, 20 frames
Pose complexity	0, 1, 2
Warm-up frames	60
Measured frames	300
Performance metrics	FPS, Mean latency, P50, P95, YOLO latency, Peak memory

Results and Discussion

Table 2 summarizes the computational performance of the proposed framework under different runtime configurations. The evaluated configurations investigate the influence of ROI size, YOLO detection interval, and MediaPipe Pose model complexity on the overall processing performance. To facilitate interpretation, REF_P0–REF_P2 denote the reference implementations, BASE represents the baseline ROI-based configuration, DE15 and DE20 correspond to extended YOLO detection intervals, C0 indicates the lowest MediaPipe Pose complexity level, R192 and R160 represent reduced ROI sizes, whereas BEST1 and BEST2 denote the best-performing parameter combinations identified during the experiments.

The experimental results demonstrate that the proposed framework consistently achieves real-time performance under all evaluated configurations. The baseline configuration (BASE) processed video at 36.86 FPS with an average processing latency of 27.13 ms, confirming that the proposed ROI-based framework is capable of operating in real time using only CPU resources. Progressive optimization of the runtime parameters further improved computational efficiency without modifying the underlying pose estimation algorithm.

Among all evaluated configurations, BEST2 achieved the highest computational performance, reaching 61.32 FPS with an average latency of 16.31 ms, representing approximately a 66% increase in processing speed and a 40% reduction in average latency compared with the baseline configuration. Similarly, BEST1 achieved 59.47 FPS and 16.82 ms latency, demonstrating that the proposed optimization strategy consistently improves runtime performance across different parameter combinations. These results indicate that jointly optimizing the detection interval, ROI size, and MediaPipe Pose complexity substantially reduces the computational workload while maintaining continuous pose estimation.

The reference configurations further illustrate the computational cost introduced by successive processing components. While REF_P0 achieved the highest processing speed among the reference implementations (50.47 FPS), enabling additional processing stages in REF_P1 and

REF_P2 increased computational overhead, reducing throughput to approximately 37 FPS. Nevertheless, these configurations remained capable of real-time execution, demonstrating that the proposed framework maintains stable computational performance even after integrating ROI extraction, scheduled detection, and landmark smoothing.

Overall, the experimental results confirm that the proposed YOLO–ROI framework provides an effective balance between computational efficiency and real-time performance. The combination of scheduled person detection, ROI-based pose estimation, and lightweight landmark smoothing enables stable CPU-based execution while significantly reducing processing latency compared with the baseline implementation. The influence of individual runtime parameters is analyzed in the following subsections.

Table 2. System performance results for different configurations

Tag	ROI size	Detect every	Pose complexity	FPS	Total mean latency (ms)	P50 latency (ms)	P95 latency (ms)	Memory (MB)
REF_P0	256	10	1	50.47	19.81	18.89	24.40	544.07
REF_P1	256	10	1	37.28	26.83	22.08	63.51	591.91
REF_P2	256	10	1	37.44	26.71	22.81	62.35	608.40
BASE	256	10	1	36.86	27.13	22.70	64.13	626.05
DE15	256	15	1	38.71	25.84	21.99	65.73	643.65
DE20	256	20	1	40.75	24.54	21.38	57.50	662.23
C0	256	10	0	54.25	18.43	15.42	46.39	660.62
C0_DE15	256	15	0	56.21	17.79	14.49	56.45	678.18
R192	192	10	1	39.65	25.22	21.30	60.71	713.95
R160	160	10	1	39.00	25.64	21.19	64.11	733.48
BEST1	192	15	0	59.47	16.82	13.64	51.93	731.69
BEST2	192	20	0	61.32	16.31	13.84	45.58	747.71

The detection interval determines how frequently the YOLO detector is executed during video processing. Increasing this interval reduces the number of object detection operations while allowing the previously detected bounding box to be reused for ROI generation between consecutive detections. Consequently, fewer computational resources are allocated to object detection, whereas MediaPipe Pose continues processing every frame.

To evaluate the influence of this strategy, three detection intervals were investigated: 10, 15, and 20 frames, corresponding to the BASE, DE15, and DE20 configurations, respectively. As shown in Table 2, extending the detection interval from 10 to 15 frames increased the processing speed from 36.86 FPS to 38.71 FPS, while reducing the average latency from 27.13 ms to 25.84 ms. A further increase to 20 frames resulted in 40.75 FPS and an average latency of 24.54 ms, demonstrating a consistent improvement in computational performance as the YOLO detector was executed less frequently.

The observed improvement is explained by the computational characteristics of the proposed framework. Among all processing stages, YOLO-based person detection represents the most computationally demanding operation. By scheduling detector execution only every several frames, the number of expensive inference operations is substantially reduced, whereas pose estimation continues inside the previously detected ROI. Since the ROI is reused between scheduled detections, the computational workload decreases without interrupting continuous landmark estimation.

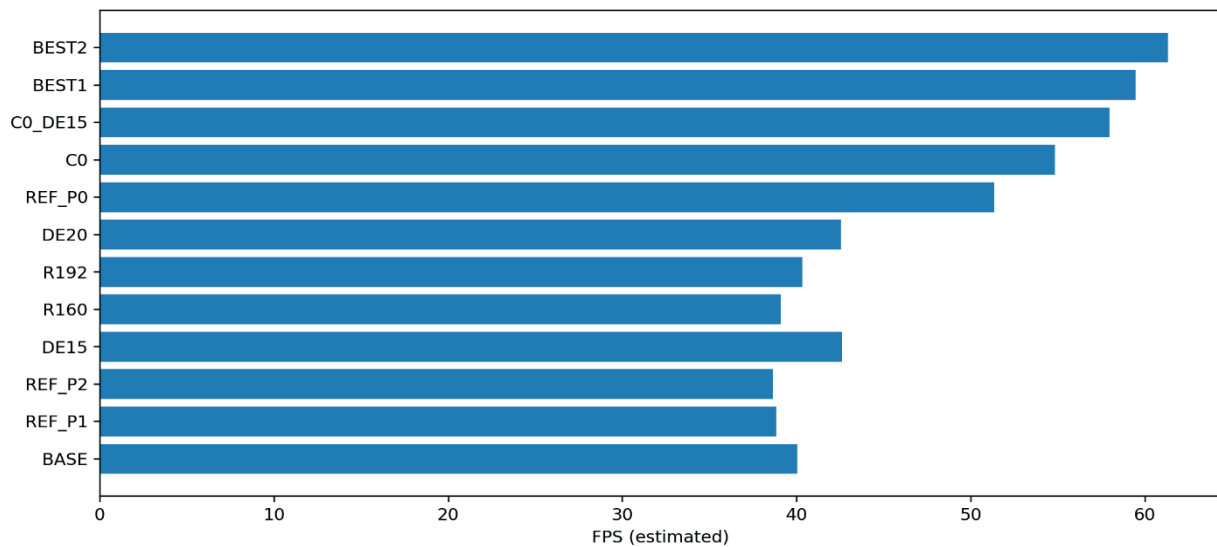


Figure 6. Influence of the YOLO detection interval on the processing speed of the proposed framework

After increasing the detection interval to 20 frames, the computational gain becomes clearly observable, while continuous pose estimation is still preserved through ROI reuse. Nevertheless, excessively large detection intervals may delay ROI updates during rapid body movements or when the subject undergoes significant displacement. Therefore, the detection interval represents a trade-off between computational efficiency and the responsiveness of ROI updating. Within the evaluated configurations, a 20-frame interval provided the highest processing speed while maintaining stable real-time operation.

The size of the region of interest (ROI) directly influences the computational workload of the pose estimation stage because MediaPipe Pose processes only the cropped ROI rather than the entire image. To investigate this effect, three ROI sizes (256×256 , 192×192 , and 160×160 pixels) were evaluated while keeping the remaining experimental parameters unchanged.

As presented in Table 2, reducing the ROI size from 256×256 pixels (BASE) to 192×192 pixels (R192) increased the processing speed from 36.86 FPS to 39.65 FPS while decreasing the average processing latency from 27.13 ms to 25.22 ms. A further reduction to 160×160 pixels (R160) maintained a similar processing speed of 39.00 FPS with an average latency of 25.64 ms. These results indicate that decreasing the ROI size generally improves computational efficiency by reducing the number of pixels processed during pose estimation.

The experimental results also show that the improvement obtained by reducing the ROI size is smaller than that achieved by increasing the detection interval or decreasing the MediaPipe Pose complexity level. This observation suggests that, although ROI cropping reduces the computational cost of pose estimation, a considerable portion of the processing time is still associated with the pose estimation model itself. Consequently, ROI optimization should be considered as one component of a broader computational optimization strategy rather than a standalone solution.

From the evaluated configurations, the 192×192 ROI provided the best balance between computational efficiency and stable pose estimation. Although the 160×160 ROI further reduced the processed image area, its processing speed was slightly lower than that of the 192×192 configuration. This indicates that excessively reducing the ROI size does not necessarily lead to additional performance gains and may increase sensitivity to subject positioning within the cropped region. Therefore, the 192×192 ROI was selected as the preferred configuration for the optimized implementation.

Figure 7 presents the influence of ROI size on the computational performance of the proposed framework. As the ROI size decreases from 256×256 to 192×192 pixels, the processing speed increases while the average processing latency decreases, demonstrating the effectiveness of ROI-based optimization.

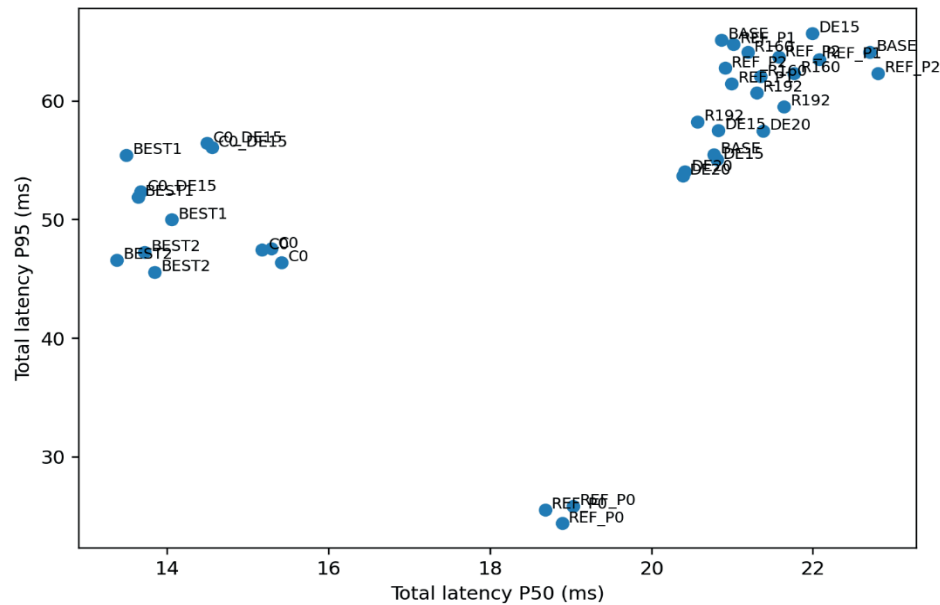


Figure 7. Effect of ROI size on the computational performance of the proposed framework

The computational complexity of the MediaPipe Pose model represents another important factor affecting the overall runtime performance of the proposed framework. MediaPipe provides multiple model complexity levels that allow a trade-off between computational cost and pose estimation capability. In this study, the default configuration (BASE) with model complexity = 1 was compared with configurations using model complexity = 0 (C0 and C0_DE15) while maintaining the remaining processing pipeline unchanged.

As shown in Table 2, reducing the MediaPipe Pose complexity from 1 to 0 produced a substantial improvement in computational efficiency. The processing speed increased from 36.86 FPS in the baseline configuration to 54.25 FPS in configuration C0, while the average processing latency decreased from 27.13 ms to 18.43 ms. When the lower-complexity model was combined with a longer detection interval (C0_DE15), the system achieved 56.21 FPS with an average latency of 17.79 ms, demonstrating that both optimization strategies complement each other and jointly improve runtime performance.

These results indicate that the computational cost of the MediaPipe Pose stage constitutes a considerable portion of the total processing time. Consequently, decreasing the model complexity has a more pronounced influence on the overall system performance than reducing the ROI size alone. The obtained results suggest that selecting an appropriate pose estimation complexity level is one of the most effective approaches for improving processing speed in CPU-based real-time applications.

Figure 8 illustrates the influence of MediaPipe Pose model complexity on the computational performance of the proposed framework. Reducing the model complexity from Level 1 to Level 0 significantly improves the processing speed while reducing the overall processing latency.

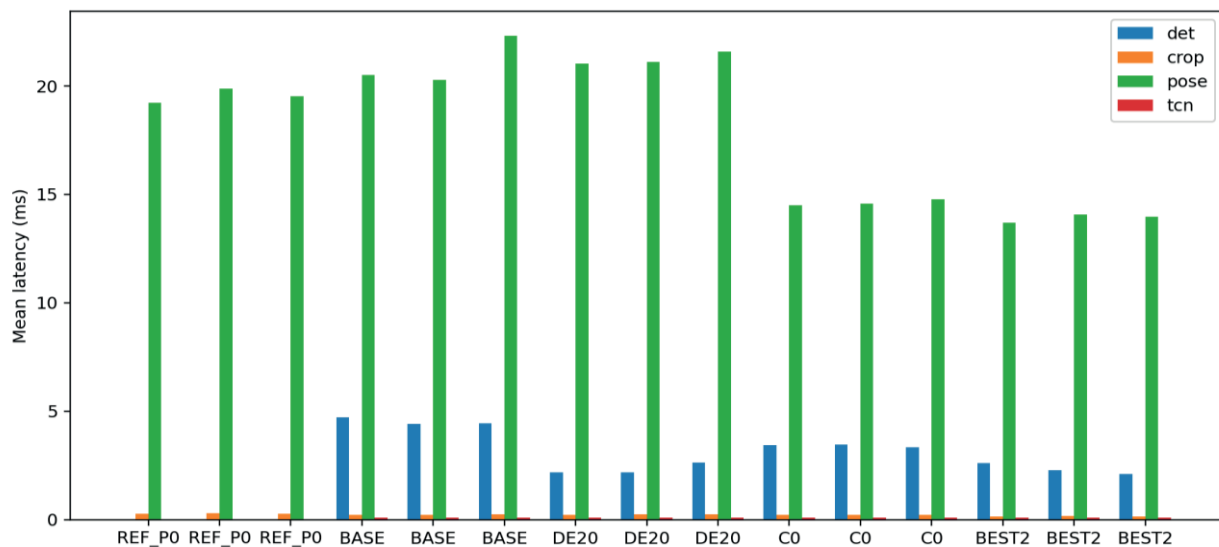


Figure 8. Effect of MediaPipe Pose model complexity on the computational performance of the proposed framework

The highest overall performance was achieved when the reduced MediaPipe Pose complexity was combined with optimized runtime parameters. In particular, the BEST1 and BEST2 configurations, which integrate a 192×192 ROI, extended detection intervals, and MediaPipe complexity level 0, reached 59.47 FPS and 61.32 FPS, respectively. These findings demonstrate that the proposed optimization strategies are complementary rather than independent, and that combining multiple lightweight processing techniques provides the greatest computational benefit for real-time human movement analysis.

The experimental results demonstrate that the proposed YOLO–ROI framework effectively improves the computational efficiency of real-time human movement analysis by reducing redundant object detection and limiting pose estimation to dynamically extracted regions of interest. Unlike conventional full-frame processing pipelines, the proposed framework combines scheduled YOLO detection with ROI reuse, enabling MediaPipe Pose to operate continuously while substantially decreasing the number of expensive object detection operations. As demonstrated in the experimental evaluation, this strategy consistently increased processing speed and reduced latency across different runtime configurations.

The obtained results further indicate that computational performance depends on the joint selection of runtime parameters rather than on a single optimization strategy. Increasing the detection interval reduced the frequency of YOLO inference, whereas decreasing the MediaPipe Pose complexity substantially lowered the computational cost of landmark estimation. Similarly, ROI optimization reduced the number of processed pixels, further improving overall efficiency. The best performance was achieved when these optimization strategies were combined, demonstrating that they complement each other in reducing computational workload.

Compared with conventional YOLO–MediaPipe pipelines that execute person detection on every frame, the proposed framework introduces a lightweight scheduling mechanism in which the most recently detected bounding box is reused between scheduled detector executions. This modification significantly decreases redundant inference while preserving continuous pose estimation inside the ROI. As a result, the proposed method is particularly suitable for CPU-based real-time computer vision applications where computational resources are limited.

The current implementation is primarily designed for continuous single-person movement analysis. Since the ROI is updated only during scheduled YOLO detections, extremely rapid body movements, prolonged occlusions, or situations in which the subject leaves the previously detected ROI may temporarily reduce localization quality until the next scheduled detection is performed.

The primary objective of this study is computational optimization rather than improvements in pose estimation accuracy or automatic activity recognition. Therefore, quantitative landmark accuracy metrics were not included in the experimental evaluation. The proposed framework employs the same MediaPipe Pose model as the baseline implementation, and its main contribution lies in reducing computational overhead through scheduled YOLO detection, ROI reuse, and lightweight landmark smoothing while preserving the underlying pose estimation pipeline. Future work will investigate quantitative pose-quality evaluation using manually annotated keypoints, adaptive detection scheduling, robust object tracking, multi-person scenarios, and automatic human activity recognition.

Overall, the experimental results confirm that the proposed framework provides a practical balance between computational efficiency and real-time operation. The combination of scheduled YOLO detection, ROI-based pose estimation, and lightweight landmark smoothing enables stable real-time performance on CPU-based platforms while maintaining a simple and easily deployable system architecture.

Conclusion

This paper presented an efficient YOLO–ROI-based framework for real-time human movement analysis by integrating scheduled YOLO person detection, ROI-based pose estimation, and lightweight landmark smoothing. Unlike conventional pipelines that perform person detection on every video frame, the proposed framework executes the YOLO detector periodically while reusing the most recently detected ROI between detection cycles. This strategy effectively reduces redundant object detection operations and enables continuous pose estimation with significantly lower computational cost.

Experimental evaluation demonstrated that the proposed framework achieves stable real-time performance under all tested configurations. By jointly optimizing the detection interval, ROI size, and MediaPipe Pose model complexity, the system achieved a maximum processing speed of 61.32 FPS with an average processing latency of 16.31 ms on a CPU-based platform. These results confirm that the proposed optimization strategy substantially improves computational efficiency while maintaining continuous pose estimation suitable for real-time human movement analysis.

The principal contribution of this work is the proposed computational optimization strategy rather than the development of a new pose estimation algorithm. By combining scheduled YOLO detection, ROI reuse, and lightweight landmark smoothing, the proposed framework provides an efficient and easily deployable solution for real-time pose analysis. Owing to its low computational requirements, the proposed approach is well suited for practical applications such as sports performance analysis, rehabilitation monitoring, intelligent surveillance, and edge-based computer vision systems.

Although the proposed framework demonstrates high computational efficiency, the current implementation is primarily intended for continuous single-person movement analysis. Extremely rapid body movements, prolonged occlusions, or situations in which the subject leaves the previously detected ROI may temporarily reduce localization accuracy until the next scheduled detection is performed. Future work will investigate adaptive detection scheduling, robust object tracking for dynamic ROI updating, multi-person scenarios, and the integration of automatic human activity recognition models based on the optimized pose estimation framework.

References:

- [1] Chen, H., Feng, R., Wu, S., Xu, H., Zhou, F., & Liu, Z. (2023). 2D human pose estimation: A survey. *Multimedia Systems*, 29(5), 3115–3138. <https://doi.org/10.1007/s00530-022-01019-0>
- [2] Gao, Z., Chen, J., Liu, Y., Jin, Y., & Tian, D. (2025). A systematic survey on human pose estimation: Upstream and downstream tasks, approaches, lightweight models, and prospects. *Artificial Intelligence Review*, 58(3), Article 68. <https://doi.org/10.1007/s10462-024-11060-2>

- [3] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., & Sheikh, Y. (2021). OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1), 172–186. <https://doi.org/10.1109/TPAMI.2019.2929257>
- [4] Ahmadyan, A., Zhang, L., Ablavatski, A., Wei, J., & Grundmann, M. (2021). Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7822–7831). https://openaccess.thecvf.com/content/CVPR2021/papers/Ahmadyan_Objectron_A_Large_Scale_Dataset_of_Object-Centric_Videos_in_the_CVPR_2021_paper.pdf
- [5] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019). *MediaPipe: A framework for perceiving and processing reality* [Workshop paper]. Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR). <https://research.google/Fpubs/mediapipe-a-framework-for-perceiving-and-processing-reality/>
- [6] Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. https://openaccess.thecvf.com/content_cvpr_2017/papers/Redmon_YOLO9000_Better_Faster_CVPR_2017_paper.pdf
- [7] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2021). Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13029–13038). https://openaccess.thecvf.com/content/CVPR2021/papers/Wang_Scaled-YOLOv4_Scaling_Cross_Stage_Partial_Network_CVPR_2021_paper.pdf
- [8] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7464–7475). <https://doi.org/10.1109/CVPR52729.2023.00721>
- [9] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 [Computer software]. GitHub. <https://github.com/ultralytics/ultralytics>
- [10] Chu, X., Yang, W., Ouyang, W., Ma, C., Yuille, A. L., & Wang, X. (2017). Multi-context attention for human pose estimation. *arXiv*. <https://doi.org/10.48550/arXiv.1702.07432>
- [11] Pavllo, D., Feichtenhofer, C., Grangier, D., & Auli, M. (2019). 3D human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7753–7762). <https://doi.org/10.1109/CVPR.2019.00794>
- [12] Xiao, B., Wu, H., & Wei, Y. (2018). Simple baselines for human pose estimation and tracking. *arXiv*. <https://doi.org/10.48550/arXiv.1804.06208>
- [13] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., & Xiao, B. (2021). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10), 3349–3364. <https://doi.org/10.1109/TPAMI.2020.2983686>
- [14] Xu, Y., Zhang, J., Zhang, Q., & Tao, D. (2022). ViTPose: Simple vision transformer baselines for human pose estimation. *arXiv*. <https://doi.org/10.48550/arXiv.2204.12484>
- [15] Lu, P., Jiang, T., Li, Y., Li, X., Chen, K., & Yang, W. (2024). RTMO: Towards high-performance one-stage real-time multi-person pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1491–1500). https://openaccess.thecvf.com/content/CVPR2024/papers/Lu_RTMO_Towards_High-Performance_One-Stage_Real-Time_Multi-Person_Pose_Estimation_CVPR_2024_paper.pdf
- [16] Kim, J.-W., Choi, J.-Y., Ha, E.-J., & Choi, J.-H. (2023). Human pose estimation using MediaPipe Pose and optimization method based on a humanoid model. *Applied Sciences*, 13(4), Article 2700. <https://doi.org/10.3390/app13042700>

[17] Hii, C. S. T., Gan, K. B., Zainal, N., Mohamed Ibrahim, N., Azmin, S., Mat Desa, S. H., van de Warrenburg, B., & You, H. W. (2023). Automated gait analysis based on a marker-free pose estimation model. *Sensors*, 23(14), Article 6489. <https://doi.org/10.3390/s23146489>

[18] Dill, S., Ahmadi, A., Grimmer, M., Haufe, D., Rohr, M., Zhao, Y., Sharbafi, M., & Hoog Antink, C. (2024). Accuracy evaluation of 3D pose reconstruction algorithms through stereo camera information fusion for physical exercises with MediaPipe Pose. *Sensors*, 24(23), Article 7772. <https://doi.org/10.3390/s24237772>

[19] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-object tracking by associating every detection box. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, & T. Hassner (Eds.), *Computer vision – ECCV 2022* (pp. 1–21). Springer. https://doi.org/10.1007/978-3-031-20047-2_1

[20] Fang, H.-S., Li, J., Tang, H., Xu, C., Zhu, H., Xiu, Y., Li, Y.-L., & Lu, C. (2023). AlphaPose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6), 7157–7173. <https://doi.org/10.1109/TPAMI.2022.3222784>