2D-3D Human Pose Estimation

Abstract

Human pose tracking is a critical task in computer vision that involves identifying key body locations, analyzing posture, and categorizing movements. This technology relies on a pre-trained machine learning model to evaluate visual input and recognize body landmarks in both image coordinates and 3D world coordinates. The applications of human pose tracking are diverse and extensive, encompassing areas such as Human-Computer Interaction, sports analysis, gaming, virtual reality, and augmented reality.

I. Introduction

Human pose estimation, the problem of localizing anatomical keypoints is a crucial task in computer vision. This task encompasses both single-person and multi-person pose estimation, each presenting its own set of challenges.

In single-person pose estimation, the objective is to accurately identify and localize the keypoints of an individual in an image. While this has been a primary focus, it presents challenges such as dealing with varying body types, poses, and potential occlusions by objects or clothing.

Multi-person pose estimation, on the other hand, involves inferring the poses of multiple people within a single image, which introduces additional complexities. Firstly, each image may contain an unknown number of people who can appear at any position or scale. Secondly, interactions between people create complex spatial interferences due to contact, occlusion, and limb articulations, making the association of parts difficult. Thirdly, runtime complexity tends to increase with the number of people in the image, posing a significant challenge for achieving real-time performance.These complexities are tackled using two main approaches: top-down and bottom-up.

In top-down approaches, the process is proportional to the number of people in the image. For each detected person, a single-person pose estimator is run. Consequently, the more people there are, the greater the computational cost. Top-down methods typically start by detecting individuals and then applying a pose estimation model to each person separately. This approach benefits from the use of global contextual cues but is computationally expensive as the number of people increases.

In contrast, bottom-up approaches are attractive because they offer robustness to early commitment and have the potential to decouple runtime complexity from the number of people in the image. Bottom-up methods detect all keypoints in the image first and then group them into individuals. This approach can be more efficient since it does not scale with the number of people; however, it does not directly use global contextual cues from other body parts and people. In practice, previous bottom-up methods often fail to retain efficiency gains because the final parsing step requires costly global inference.

In this paper, we will discuss various models and their architectures, providing a comparative analysis of their accuracy and inference times.

II. Models for 2D pose estimation

A. Mediapipe

MediaPipe 2D Pose Estimation is a machine learning-based solution that detects and localizes key body landmarks in 2D image coordinates. This system identifies 33 specific points on the human body, such as the nose, eyes, shoulders, elbows, and knees, within 2D space. These landmarks are crucial for understanding human posture and movements in various applications, including fitness tracking, gesture recognition, and animation.
MediaPipe's 2D pose estimation is known for its efficiency and accuracy, making it suitable for real-time applications. The process involves running a pre-trained neural network model on the input image or video to predict the 2D coordinates of the body landmarks. It does not provide depth information, focusing solely on the positions of the landmarks within the plane of the image.

First of all we must mention BlazePose which is a machine learning model that provides real-time human pose tracking by detecting key body landmarks. BlazePose is integrated into the MediaPipe framework as part of the MediaPipe Pose solution. Our pipeline consists of a lightweight body pose detector followed by a pose tracker network. The tracker network predicts keypoint coordinates, determines the presence of a person in the current frame, and refines the region of interest. If the tracker indicates the absence of a person, the detector network is re-executed on the subsequent frame.

The pose estimation component of our system predicts the location of all 33 person keypoints, and uses the person alignment proposal provided by the first stage of the pipeline

It starts by using an initial alignment step. We use a combined approach involving heatmaps, offsets, and regression.
Heatmap: A heatmap is an image representation where each

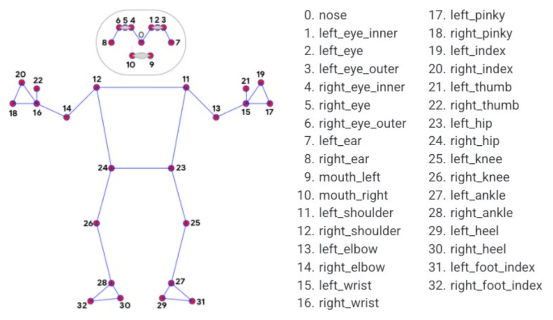


Figure 1. Mediapipe keypoints

pixel value indicates the probability of a keypoint being present at that location.
Offset: Offset represents the displacement or deviation from the detected keypoint in the heatmap to the precise location.
During training, we use heatmaps and offset losses, but remove these output layers for inference. The heatmap supervises the lightweight embeddings, which are used by the regression encoder network to predict keypoint coordinates. Inspired by the Stacked Hourglass approach, we use a tiny encoder-decoder network followed by a regression encoder network, with skip-connections to balance feature levels.
Stacked Hourglass appraoch: This is a technique used in pose estimation where multiple encoder-decoder networks are stacked on top of each other. An encoder reduces the size of the input, while a decoder reconstructs it back to the original size. This allows the network to capture features at different scales.
Regression Encoder Network: After the tiny encoder-decoder network, another network (regression encoder) predicts the exact coordinates of the keypoints.
Gradient-stopping is applied between the regression encoder and heatmap-trained features, enhancing heatmap accuracy and coordinate regression.

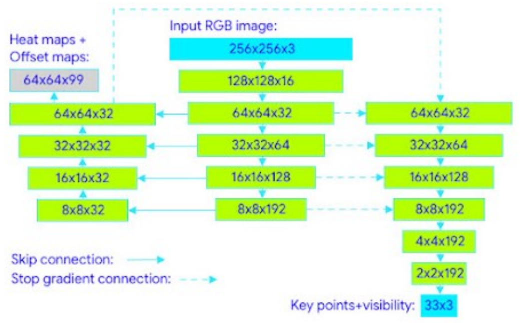


Figure 2. Mediapipe architecture

Skip Connections arrows: Known also as residual connections, which are used to add the outputs of earlier layers to later layers. This helps in preserving spatial information and improving gradient flow during training.
Stop Gradient Connections: These connections are used to control the gradient flow. By stopping the gradient propagation through certain paths, the network can prevent issues like gradient vanishing or exploding, which can hinder the training process.

A relevant pose prior is a vital part of the proposed solution, which uses prior knowledge of typical human poses to guide pose estimation. The model acquires the relevant pose prior through a combination of data preparation, augmentation techniques, and training processes.
During training, we deliberately limit the supported ranges for angle, scale, and translation to reduce network complexity and computational requirements. This approach lowers the network capacity, making it faster and requiring fewer computational resources. The person is aligned by centering the midpoint between the hips in the image, and the image is rotated to ensure the line between the mid-hip and mid-shoulder points is vertical.
The scale is adjusted so that the entire body fits within a square bounding box. Additionally, small random augmentations of 10 percent in scale and shift are applied to handle minor movements and variations between frames, improving the robustness and accuracy of the pose estimation model.
The primary dataset used for training BlazePose includes :

COCO (Common Objects in Context): This dataset is commonly used for object detection, segmentation, and keypoint detection tasks. It contains a significant amount of annotated images with human keypoints, which are crucial for pose estimation models.

MPii (Max Planck Institute for Informatics Human Pose): This dataset is another widely used benchmark for human pose estimation, featuring annotated keypoints on various human figures performing different activities.

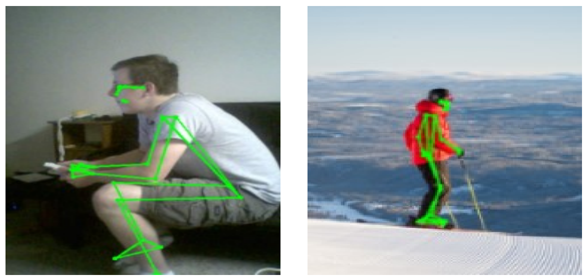


Figure 3. 2D Mediapipe keypoints

The model is exceptionally fast and efficient, achieving an execution time of approximately 0.02 seconds on CPU, with an even smaller difference of just 0.001 seconds on GPU.

B. *AlphaPose*

AlphaPose, developed by the Chinese Academy of Sciences, is a cutting-edge human pose estimation tool utilizing deep learning algorithms, primarily Convolutional Neural Networks (CNNs), to analyze images and videos in real time. It excels in detecting and tracking human body poses, even in low-light conditions and when parts of the body are occluded.
AlphaPose uses a bottom-up approach, t his allows it to simultaneously handle multiple humans in the same image or video. AlphaPose also uses a heatmap-based approach, which means it estimates the likelihood of each pixel belonging to a specific body part.
NB: AlphaPose is an open-source package written in Python and compatible with various deep learning frameworks, including PyTorch, Caffe, and TensorFlow.

Heatmap is a dominant representation for joint localization in the field of human pose estimation. The read-out locations from heatmaps are discrete numbers since heatmaps only describe the likelihood of joints occurring in each spatial grid, which leads to inevitable quantization error.

Quantization: the process of mapping continuous values of keypoint locations to discrete values. This is a common challenge because human body keypoints (like joints) are inherently continuous, but most algorithms work with discrete grid points due to computational constraints.
It can lead to a loss of precision, especially if the grid resolution is too low. Also reduce spatial resolution may cause errors in keypoint localization, particularly in complex poses where multiple individuals are in close proximity.

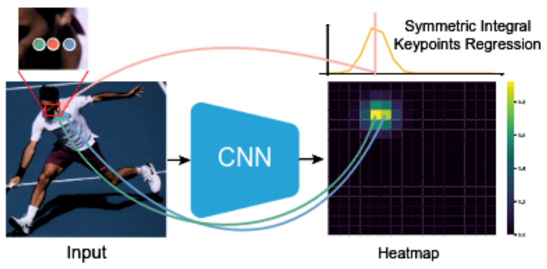


Figure 4. Quantization Errors

To mitigate quantization errors and enhances accuracy we can use **Soft-Argmax** which is a technique used in human pose estimation to improve keypoint localization by leveraging heatmaps. Instead of directly predicting discrete grid points, the network generates a probability distribution (heatmap) for each keypoint. The soft argmax calculates the keypoint location as a weighted average of the grid points based on their probabilities, allowing for a more precise and continuous representation.

Soft-Argmax Formula:

$$\hat{x} = \sum_i x_i \cdot \frac{\exp(h_i)}{\sum_j \exp(h_j)}$$

$$\hat{y} = \sum_i y_i \cdot \frac{\exp(h_i)}{\sum_j \exp(h_j)}$$

Weighted Average Formula:

$$\hat{x} = \frac{\sum_i x_i \cdot p_i}{\sum_i p_i}$$

$$\hat{y} = \frac{\sum_i y_i \cdot p_i}{\sum_i p_i}$$

- $\hat{x}$ and $\hat{y}$ are the continuous coordinates of the keypoint.
- x_i and y_i are the discrete coordinates on the grid.
- h_i is the heatmap value at coordinate i.
- $\exp(h_i)$ is the exponentiation of the heatmap value, which gives the softmax probability.

But while Soft-Argmax offering a continuous estimate of keypoint locations, faces several challenges, notably the asymmetric gradient problem and the size-dependent keypoint scoring problem (related to size and resolution of the grid used in the heatmap).

To address the asymmetric gradient problem, the Amplitude Symmetric Gradient (ASG) is used as an approximation to the true gradient, enhancing learning efficiency by providing more balanced gradient updates. This approach mitigates issues related to asymmetric gradients that can disrupt model training.

Before:

$$L_{\text{reg}} = \|\mu - \hat{\mu}\|_1$$

$$\frac{\partial L_{\text{reg}}}{\partial p_x} = x \cdot \text{sgn}(\hat{\mu} - \mu)$$

- $\frac{\partial L_{\text{reg}}}{\partial p_x}$ represents the gradient of the regularization loss L_{reg} with respect to the pixel value p_x.
- x is the coordinate of the pixel in the grid.
- $\text{sgn}(\hat{\mu} - \mu)$ is the sign function applied to the difference between the predicted mean $\hat{\mu}$ and the true mean μ.

After:

$$\delta_{\text{ASG}} = \text{Agrad} \cdot \text{sgn}(x - \hat{\mu}) \cdot \text{sgn}(\hat{\mu} - \mu)$$

- δ_{ASG} represents the amplitude symmetric gradient.
- Agrad denotes the amplitude gradient.
- $\text{sgn}(\cdot)$ is the sign function, which returns the sign of a number.
- x is the input value.
- $\hat{\mu}$ is the predicted mean.
- μ is the true mean.

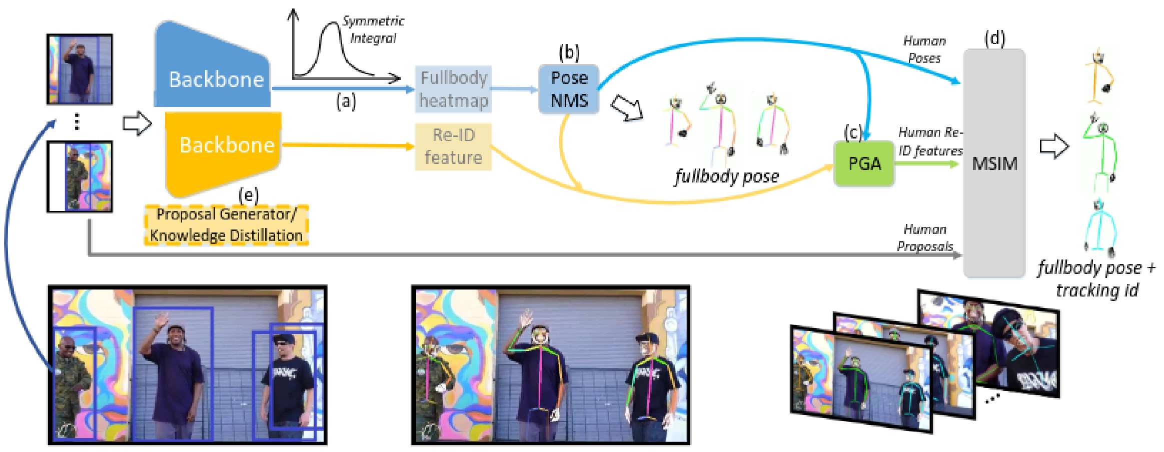


Figure 5. AlphPose Architecture

For the size-dependent keypoint scoring problem, which arises because traditional normalization methods like softmax yield joint confidence values inversely proportional to joint size, a two-step heatmap normalization technique is proposed. In the first step, element-wise normalization with the sigmoid function generates a confidence heatmap that accurately reflects the presence of joints without being influenced by their size. The maximum value of this confidence heatmap represents the joint confidence.

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

In the second step, global normalization converts this confidence heatmap into a probability heatmap, ensuring that the sum of all elements equals one. This two-step approach stabilizes the training process and ensures accurate joint location predictions by decoupling confidence prediction from localization.

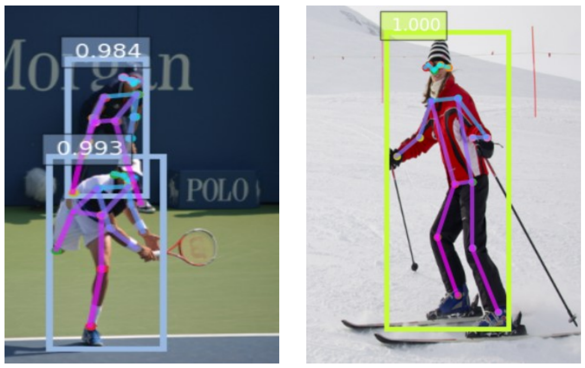
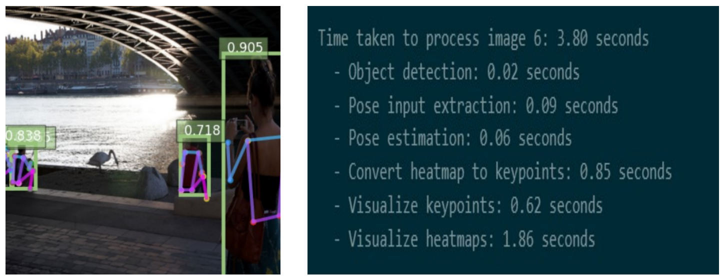


Figure 6. Testing AlphaPose model

As you can see, the model accurately predicts the keypoints. Additionally, I have integrated a detection model into my code and tested it on a CPU. The time taken to process the image is shown in the picture Below Fig-6.



```
Time taken to process image 6: 3.80 seconds
  - Object detection: 0.02 seconds
  - Pose input extraction: 0.09 seconds
  - Pose estimation: 0.06 seconds
  - Convert heatmap to keypoints: 0.85 seconds
  - Visualize keypoints: 0.62 seconds
  - Visualize heatmaps: 1.86 seconds
```

Figure 7. Inference Time

The AlphaPose architecture is designed to deliver accurate and efficient human pose estimation, combining several advanced techniques to enhance performance.

1) Feature Extraction The input image is processed through a deep convolutional neural network (CNN) known as the backbone, which extracts rich features from the image. This backbone serves as the foundation for all subsequent operations by identifying important patterns like edges and textures.

2) Fullbody Heatmap and Re-ID Feature Generation From the extracted features, a full-body heatmap is generated to indicate the probable locations of keypoints (joints of the human body). Simultaneously, Re-ID (Re-Identification) features are extracted to uniquely identify and track individuals across different frames.

3) Pose Refinement with Non-Maximum Suppression (NMS) Non-Maximum Suppression is applied to refine the initial pose predictions by selecting the most confident keypoints and eliminating redundant ones. This ensures that each detected individual has a distinct and accurate set of keypoints.

Pose-Guided Attention (PGA) The Pose-Guided Attention mechanism further refines keypoint localization by focusing the model's attention on the most informative regions of the image. This step enhances the precision of the pose estimation by leveraging initial pose estimates to guide attention.

Attention Score Computation:

- For each keypoint, compute an attention score α_i based on the initial pose predictions. This score determines how much attention should be given to the region around each keypoint.
- The attention score formula:

$$\alpha_i(x,y) = \exp\left(-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma^2}\right)$$

where (x_i, y_i) are the coordinates of the initial keypoint prediction, (x, y) are the coordinates of a pixel in the image, and σ controls the spread of the attention around the keypoint.

Attention Map Generation:

- Use the attention scores $\alpha_i(x, y)$ to generate attention maps $A_i(x, y)$ for each keypoint. These maps indicate how much attention should be given to different regions of the image during the refinement process.

Weighted Feature Extraction:

- Extract features from the image using the attention maps. These features are weighted by the attention scores, ensuring that regions closer to the initial keypoint predictions have more influence.
- Mathematically, the refined feature map F_i for the i-th keypoint can be computed as:

$$F_i(x,y) = A_i(x,y) \cdot \text{FeatureMap}(x,y)$$

where $\text{FeatureMap}(x, y)$ represents the feature map extracted from the image at position (x, y).

Keypoint Refinement:

- The refined feature maps $F_i(x, y)$ are then fed into a network (a small CNN or fully connected layers) that outputs refined keypoint predictions $\hat{p}_i = (\hat{x}_i, \hat{y}_i)$.
- This process uses the focused attention from the attention maps to correct any errors in the initial keypoint predictions, leading to more precise localization.

Multi-Scale Information Module (MSIM) To address variations in human size and scale within the image, it processes the features at different scales. This module aggregates information from these scales, resulting in a more robust and accurate pose estimation that adapts to various perspectives and distances.

Pose Proposal and Knowledge Distillation The Proposal Generator identifies candidate regions in the image that are likely to contain human poses. In parallel, Knowledge Distillation is employed during training, where a larger, more complex model (teacher) guides a smaller model (student).The goal is to transfer the knowledge contained in the teacher model to the student model, enabling the student model to achieve comparable performance with less computational cost.

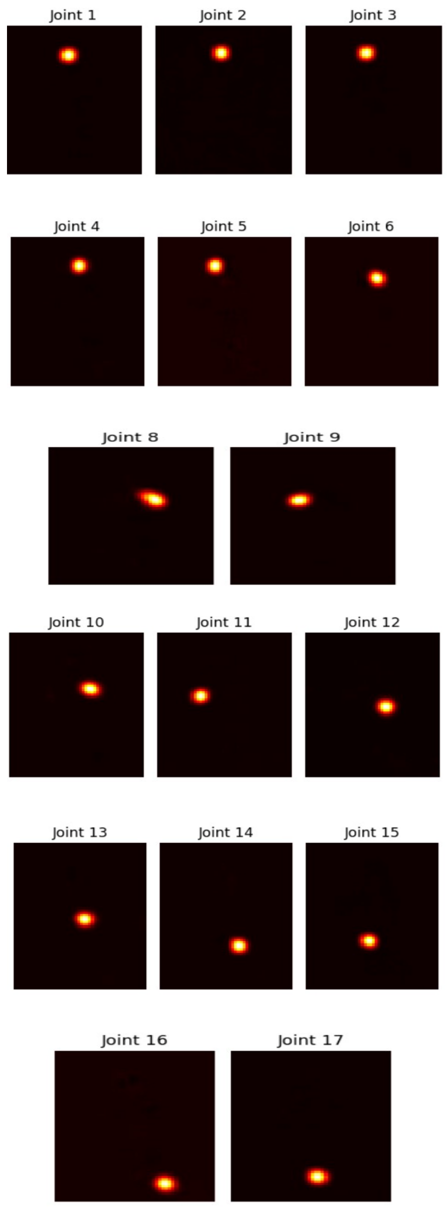


Figure 8. Heatmap for every joint

This is a set of heatmaps corresponding to each joint for the image of the woman skiing shown in Fig-5.

C. *Yolov8*

YOLOv8 is an advanced object detection algorithm that builds on the YOLO series, focusing on speed and accuracy. It models object detection as a regression problem, directly predicting bounding box coordinates and object classes. YOLOv8 uses the CSPDarkNet53 backbone with a novel Cross Stage Partial (CSP) connection method, optimizing computational resources for improved accuracy and speed.

CSP: Cross Stage Partial (CSP) enhances deep neural networks by dividing the feature map into two parts (one is transformed, while the other bypasses transformation and is later merged back). This approach reduces redundancy, improves gradient flow, and balances computational load, leading to more efficient training. CSP achieves a balance between speed and accuracy, making models faster without compromising performance.

The model also employs an Anchor-Free detection head, simplifying training and enhancing performance compared to traditional Anchor-Based methods. Additionally, YOLOv8 utilizes the Complete Intersection over Union (CIOU) and Distribution Focal Loss (DFL) loss functions to further improve detection results (Improves bounding box regression and Enhances object classification).

$$\text{DFL}(s_i, s_{i+1}) = -\left[(y_{i+1} - y)\log(s_i) + (y - y_i)\log(s_{i+1})\right] \tag{1}$$

where:

- s_i: Output of the sigmoid function from the network.
- y_i: Interval order associated with the predicted bounding box.
- y_{i+1}: Interval order associated with the ground truth bounding box.
- y: Label indicating the presence or absence of an object.

This loss function addresses the overlap between predicted bounding boxes and ground truth bounding boxes, enhancing detection accuracy and robustness, particularly for small objects.

The CIOU and DFL loss functions in YOLOv8 are designed to measure the distance between predicted and ground truth bounding boxes, leveraging the Intersection over Union (IOU) value as a key metric. By incorporating a combination of cross-entropy loss and mean square error loss, these loss functions effectively optimize the model during training.

Anchor-based methods: are object detection techniques that use predefined bounding boxes, called anchor boxes, to detect objects in an image. The model adjusts these anchors to fit the actual objects by predicting their final bounding box coordinates and class labels.
Anchor-Free Head: An anchor-free head is an approach that directly predicts the bounding boxes of objects from the feature map without relying on predefined anchor boxes. It simplifies the detection process and improves flexibility by eliminating the need for anchor box adjustments.

Pose models based YOLOv8

The YOLOv8 series includes six pose estimation models (YOLOv8s-pose, YOLOv8n-pose, YOLOv8m-pose, YOLOv8l-pose, YOLOv8x-pose, YOLOv8x-pose-p6) designed to add human pose estimation capabilities to YOLOv8. These models vary in backbone network scales, balancing between accuracy and computational resource requirements.

The image effectively displays the detected keypoints, but

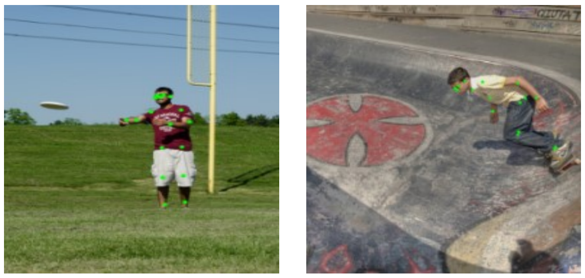


Figure 9. Yolov8 keypoints

when attempting to connect them, errors occur, particularly with incorrect connections being made to the upper left corner. The solution involved adding a condition in the function

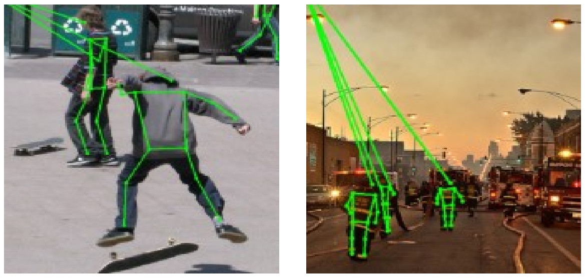


Figure 10. Trouble Connecting Keypoints Effectively

to handle the pose estimation model more effectively. By filtering the coordinates of detected keypoints with conditions like $x > 0$ and $y > 0$, you can avoid invalid detections.

YOLOv8 Pose Models

Smaller models are suitable for resource-constrained environments but may offer lower accuracy, while larger models provide higher accuracy at the cost of increased computational and memory demands. The choice of model depends on whether speed or precision is prioritized, with real-time scenarios favoring faster models and precision-focused applications opting for more sophisticated ones.

YOLOv8 first detects the person (or persons) in the image. This involves identifying bounding boxes around each detected person using the object detection capabilities of YOLOv8.
Once the person is detected, the model then performs pose estimation on the detected bounding boxes (the regions of interest) to determine the key points (such as joints) of the human body.

- **YOLOv8s-pose:** A smaller YOLOv8 model variant optimized for pose estimation with reduced computational resources and memory footprint. Suitable for resource-constrained environments.

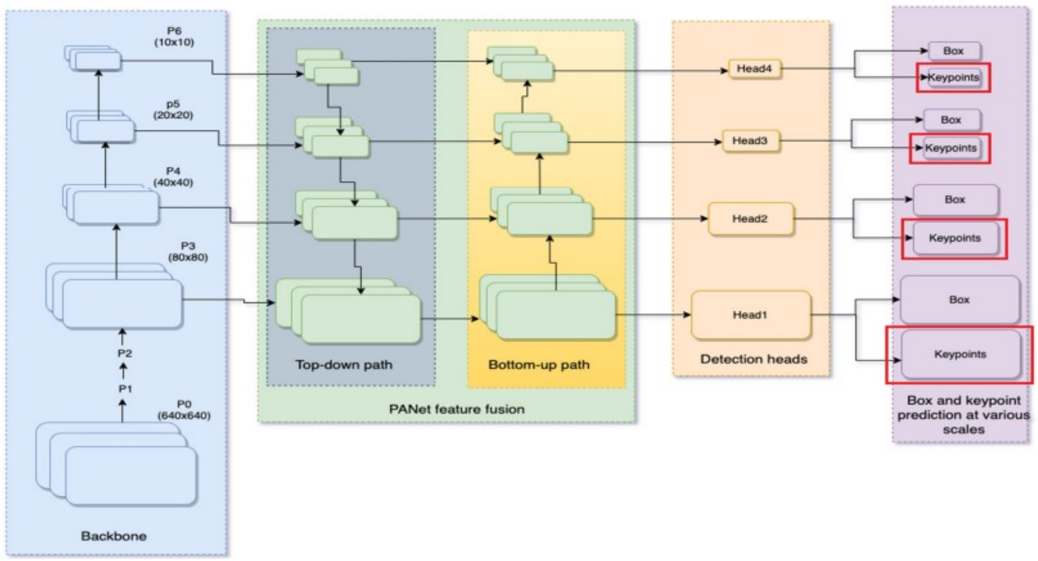


Figure 11. YOLOv8 pose Architecture

- **YOLOv8n-pose:** A model with a slightly larger backbone than YOLOv8s-pose, providing a balance between accuracy and resource requirements. Designed for environments where moderate resources are available.
- **YOLOv8m-pose:** A medium-sized YOLOv8 model offering a better trade-off between accuracy and computational load compared to smaller models. Ideal for applications needing good accuracy without excessive resource consumption.
- **YOLOv8l-pose:** A larger variant with improved pose estimation accuracy and performance. Requires more computational power and memory, suitable for applications where accuracy is more critical.
- **YOLOv8x-pose:** A high-performance model with advanced capabilities for precise pose estimation. Provides the highest accuracy among the YOLOv8 pose models but comes with increased computational and memory demands.
- **YOLOv8x-pose-p6:** An extended version of YOLOv8x-pose with additional improvements for handling even more challenging pose estimation scenarios. It offers enhanced accuracy and robustness but requires substantial computational resources.

I will compare only between the YOLOv8n and YOLOv8m models here to show the performance differences using CPU.

The model detected only 4 out of 6 visible persons in 130.9ms during the first inference. Similarly, in the second image, it identified 4 persons within 101.3ms. Fig-12

Using YOLOv8m, the first image successfully detected all 6 persons in 593.2ms, demonstrating its accuracy despite the longer inference time. In the second image, the model identified 14 persons within 546.5ms. Fig-13

In conclusion, YOLOv8m demonstrates a strong balance between detection accuracy and inference speed. The model accurately detected all individuals in both images, even as the number of persons increased. Although the inference times were slightly longer, the precision in identifying all visible persons suggests that YOLOv8m is well-suited for scenarios where accuracy is critical, making it a reliable choice for real-time applications that require detailed object detection.

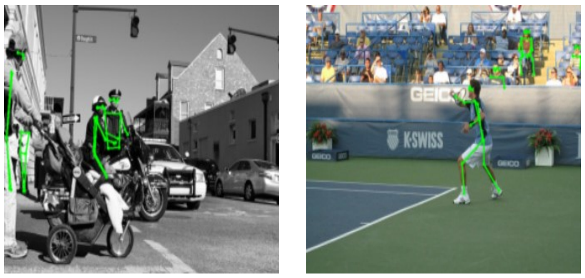


Figure 12. Yolov8n-pose

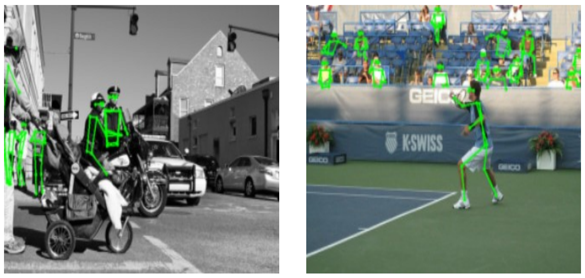


Figure 13. YOLOv8m-pose

Now we will discuss the architecture of our model as shown in Figure 11:

- **Backbone:**
  - The backbone is a CNN that processes the input image to extract hierarchical feature maps. These feature maps are generated at different scales, as indicated by the layers labeled P0, P1, P2,..
  - The different scales of feature maps are crucial because they allow the model to detect objects (and keypoints) at varying sizes.
- **PANet Feature Fusion:**
  - **PANet:** is an advanced neural network architecture designed to enhance feature pyramid networks (FPNs:a type of neural network architecture designed to enhance object detection by effectively utilizing multi-scale features.) for better object detection, segmentation, and related tasks.representation.

    PANet uses adaptive feature pooling, which allows the model to better aggregate features from regions of interest (RoIs) by considering the size of the detected objects, also employs a fully connected fusion approach to merge features from different levels of the pyramid without forgetting that it enhances information flow between different layers of the feature pyramid
- **Detection Heads:**
  - The model has multiple detection heads (Head1, Head2, Head3, Head4) that correspond to the different scales of the feature maps.
  - Each detection head is responsible for predicting bounding boxes and keypoints at its respective scale. This multi-scale approach ensures that the model can effectively handle objects of varying sizes in the image.

III. Models for 3D pose estimation

A. *METRABS*

MeTRAbs stands for Metric-Scale 3D Human Pose Estimation with Transformer and Absolute Pose. The model is particularly focused on providing accurate 3D pose estimates at a metric scale, meaning that the distances and sizes of the predicted poses correspond to real-world measurements. MeTRAbs is a sophisticated model for 3D human pose estimation that emphasizes accuracy in metric-scale predictions. It utilizes transformers for robust spatial understanding and is designed to handle complex real-world challenges, such as occlusion and truncation, making it highly suitable for applications requiring precise 3D human pose information.

Transformer: are a type of deep learning model architecture that was originally introduced for natural language processing (NLP) tasks but has since been adapted and extended to various other domains, including computer vision, speech processing, and more.

In MeTRAbs, Transformers are employed to efficiently capture and process the complex dependencies between various body parts to accurately predict 3D human poses from images. Unlike traditional models that might struggle with capturing long-range relationships or dealing with occluded parts, transformers use a self-attention mechanism that allows the model to weigh the importance of different body joints relative to each other, even if they are far apart or partially occluded in the image.

The architecture depicted in the image is a representation of the MeTRAbs model, which is designed for accurate 3D human pose estimation.

Noisy Detections These are initial detections of the human figure within the image, which might include some inaccuracies or ”noise” (misaligned bounding boxes or partial occlusions). These detections serve as the input for the MeTRAbs pipeline.

Off-the-Shelf Backbone This component refers to a pretrained convolutional neural network (CNN) that acts as a feature extractor (ResNet, EfficientNet, or similar architectures). The role of the backbone is to process the input image and produce a set of feature maps that represent the important patterns in the image.

MeTRo 3D Heatmaps

- **MeTRo 3D Heatmaps**: MeTRo 3D Heatmaps are a crucial part of the MeTRAbs architecture. These heatmaps represent the probability distributions of joint locations in 3D space. The term ”MeTRo” in this context likely refers to the Metric-Scale 3D Heatmap Representation, where the goal is to predict the joint positions in a 3D metric space (real-world coordinates).

  $$p_j(x, y, z) = H_j(x, y, z)$$

  Here, H represents the 3D heatmap and $H_j(x, y, z)$ is the value at voxel (x, y, z) in the 3D heatmap corresponding to joint j.

  Voxel represents a single point or unit of data in a 3D grid (3D equivalent of pixel).
- **1x1 Convolution**: After the feature extraction by the backbone, a 1x1 convolutional layer is applied to reduce the feature maps' dimensionality, producing the 3D heatmaps.
- **3D Soft-Argmax**: This operation is applied to the 3D heatmaps to convert the probability distributions into precise 3D coordinates for each joint. This step effectively determines the most likely position of each joint in 3D space.

  The 3D coordinates (X_j, Y_j, Z_j) for joint j are computed as follows:

  $$X_j = \sum_{x,y,z} x \cdot p_j(x, y, z)$$

  $$Y_j = \sum_{x,y,z} y \cdot p_j(x, y, z)$$

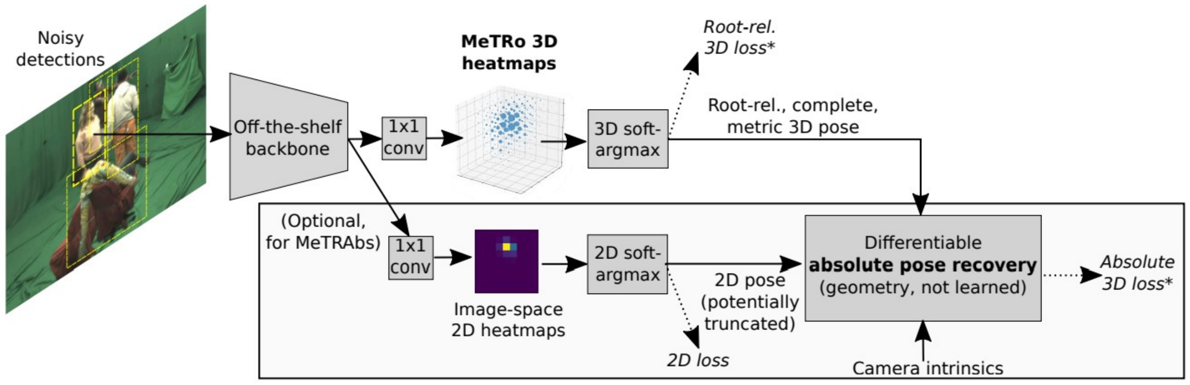


Figure 14. MeTRabs Architecture

$$Z_j = \sum_{x,y,z} z \cdot p_j(x, y, z)$$

Optional 2D Heatmaps

- In addition to the 3D heatmaps, the architecture can generate 2D heatmaps representing the joint positions in the 2D image plane. These are particularly useful for refining predictions and handling scenarios where 3D predictions alone might be insufficient.
- The process involves another 1x1 convolution to produce these heatmaps, followed by a 2D soft-argmax to convert the 2D heatmaps into 2D joint positions.

Root-Relative vs. Absolute 3D Poses

- **Root-Relative 3D Pose:** Initially, the predicted 3D coordinates are root-relative, meaning they are relative to a specific joint, typically the pelvis or hip. This relative positioning is useful for understanding the structure of the pose but is not yet in an absolute metric scale.
- **Absolute 3D Pose Recovery:** To convert these relative poses into absolute poses in metric space, the model uses a Differentiable Absolute Pose Recovery module. This module takes into account camera intrinsic parameters (like focal length and sensor size) to map the root-relative coordinates into real-world metric units. The result is a set of joint positions that are spatially accurate and can be directly used in applications that require physical measurements.

Incorporating Camera Intrinsics

The camera intrinsics play a crucial role in this process. These parameters define how the 3D world is projected onto the 2D image plane. By integrating these parameters, the MeTRo 3D component can accurately recover the absolute positions of the joints in the 3D space relative to the camera, ensuring that the final pose estimations reflect real-world distances.

Training and Loss Functions

The entire process is trained end-to-end using various loss functions:

- **Root-Relative 3D Loss**: This loss function ensures that the initial 3D predictions relative to the root joint are accurate.
- **Absolute 3D Loss**: After the absolute pose recovery, this loss function checks that the recovered metric-scale positions match the real-world ground truth.
- **2D Loss**: If 2D heatmaps are also used, this loss function ensures that the 2D projections of the joints align with the ground truth in the image plane.

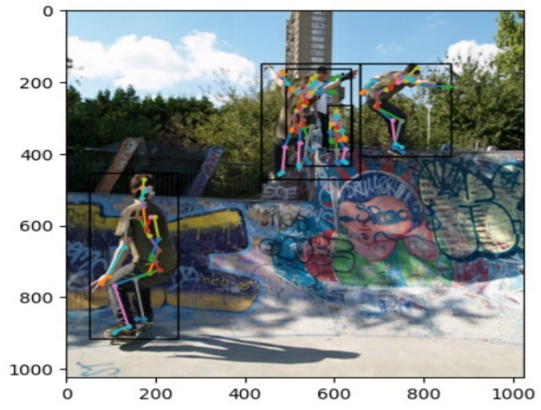


Figure 15. MeTRabs Test

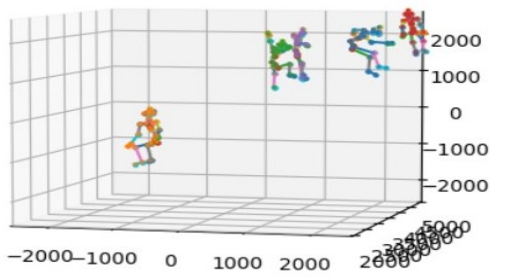


Figure 16. MeTRabs in 3D Coordinates

B. *Mediapipe 3D*

Mediapipe's 2D and 3D pose estimation models are both built on the BlazePose architecture, which is designed for efficient and real-time tracking of body movements. BlazePose detects 33 key landmarks on the human body, providing detailed 2D coordinates (x, y) in the original implementation. The 3D version of Mediapipe extends this by adding a third coordinate, the z-axis, which represents depth. This addition transforms the landmarks into a full 3D representation, allowing for a more comprehensive understanding of body posture and movements in three-dimensional space. Incorporating the z-coordinate requires modifications to the original BlazePose architecture, including enhanced model complexity and training on depth-aware datasets, ensuring accurate and robust 3D pose estimation. These adjustments make Mediapipe 3D particularly powerful for applications that demand spatial awareness, such as augmented reality (AR), virtual reality (VR), and advanced gesture recognition.

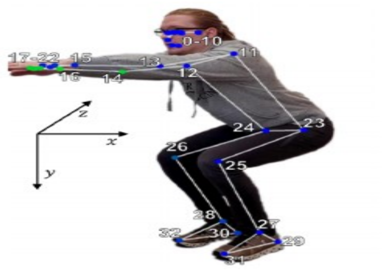


Figure 17. 3D Mediapipe keypoints

Mediapipe 2D Pose Estimation (BlazePose 2D)

Model Architecture:

- **Input:** A single image or video frame is fed into the model.
- **Feature Extraction:** A convolutional neural network (CNN) extracts features from the image.
- **Landmark Prediction:** The model predicts the x and y coordinates of 33 keypoints (landmarks) on the human body.
- **Output:** The result is a set of 2D coordinates for each landmark in the image plane (x, y).

Training:

- **Data:** Trained on large datasets with annotated 2D keypoints.
- **Objective:** Minimize the loss between predicted and true 2D keypoint locations.

Mediapipe 3D Pose Estimation (BlazePose 3D)

Model Architecture:

- **Input:** Similar to the 2D model, it starts with an image or video frame.
- **Feature Extraction:** Uses a similar CNN architecture to extract features.
- **Landmark Prediction:** The model predicts the x, y, and z coordinates of the 33 keypoints.
- **Depth Estimation:** The z-coordinate provides depth information, transforming 2D landmarks into a 3D space.
- **Output:** The output is a set of 3D coordinates (x, y, z) for each landmark.

Training:

- **Data:** Trained on datasets that include 3D annotations or are generated using multi-view setups or depth cameras.
- **Objective:** Minimize the loss between predicted and true 3D keypoint locations, considering both spatial and depth accuracy.

The prediction of the z-coordinate (depth) in Mediapipe's 3D Pose Estimation involves several architectural enhancements and modifications compared to the 2D Pose Estimation model.

Training with 3D Data:

The 3D model is trained on datasets that include 3D annotations, which provide ground truth x, y, and z coordinates.

Loss Function:

The loss function used during training for the 3D model is modified to account for the z-coordinate. It typically involves minimizing the difference between the predicted and true 3D coordinates (x, y, z), rather than just the 2D coordinates.
Let $\mathbf{p}_i = (x_i, y_i, z_i)$ be the predicted 3D coordinates of the i-th landmark, and $\mathbf{g}_i = (x_i^{\text{gt}}, y_i^{\text{gt}}, z_i^{\text{gt}})$ be the ground truth 3D coordinates of the i-th landmark.

The loss function $\mathcal{L}$ can be defined as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - \mathbf{g}_i\|_2^2$$

Where:

- N is the total number of landmarks (33 for our model BlazePose).

- $\|\cdot\|_2$ represents the Euclidean distance (L2 norm) between the predicted and ground truth coordinates.

Multi-Stage Architecture:

The architecture is designed as a multi-stage model where the first stage predicts 2D landmarks, and subsequent stages refine these predictions and estimate the z-coordinate.

The key difference in the architecture that enables the prediction of the z-coordinate lies in the additional complexity and depth-awareness introduced to the model. This includes extra layers, depth-specific features, modified loss functions, and specialized training data that allow the model to accurately predict not just where a landmark is on the x and y axes, but also how far it is from the camera (z-axis). These architectural enhancements make Mediapipe's 3D Pose Estimation capable of providing a full 3D representation of human poses, unlike the 2D model, which only provides a flat, two-dimensional representation.

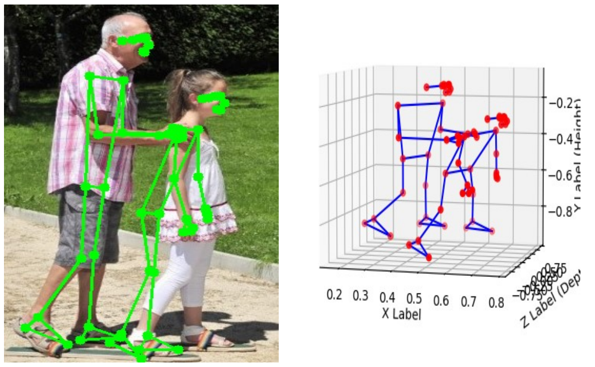


Figure 18. Visualization of 3D Mediapipe

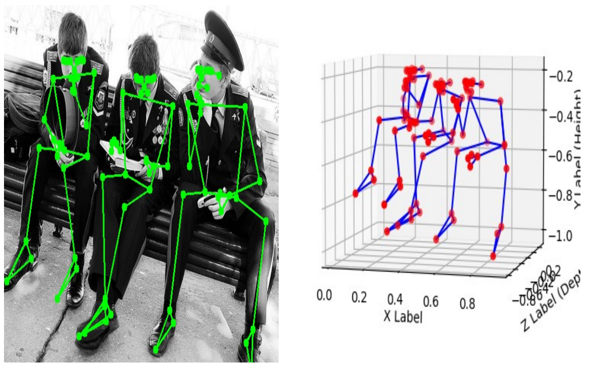


Figure 19. Seated Pose Analysis

Mediapipe offers a range of configurable attributes that allow for precise control over the model's behavior. Key options include `running_mode`, which determines whether the task will process a single image, video frames, or a live stream. The `num_poses` attribute defines the maximum number of poses that can be detected simultaneously. Various confidence thresholds (`min_pose_detection_confidence`, `min_pose_presence_confidence`, and `min_tracking_confidence`) are available to ensure the accuracy and reliability of pose detection, presence, and tracking. Additionally, the `output_segmentation_masks` option controls whether segmentation masks are generated alongside the detected poses. For real-time applications, the `result_callback` is essential, enabling asynchronous handling of results in live stream mode.

Mediapipe's pipeline is fast due to its graph-based design, lightweight embedding strategies, conditional execution, and optimized data flow. By efficiently managing resources, minimizing unnecessary processing, and leveraging hardware acceleration, Mediapipe can achieve high performance even on devices with limited computational power.

That's why it's the ideal model for 3D applications in a company setting, offering unparalleled speed and efficiency.

I added a function to the pipeline that extracts the IDs of shoulders and hands, then calculates the differences in their z and x coordinates. By applying a threshold to these differences, the function can determine whether a person's hands are close to their body. If the difference exceeds the threshold, it indicates that the person's hands are likely tucked into their pants or jacket pockets. This enhancement is designed to detect specific hand positions, such as hands in pockets, which can be useful for various applications where recognizing subtle body movements is important.

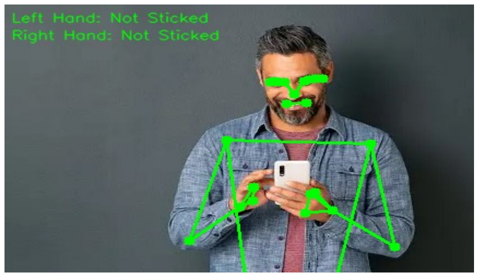


Figure 20. Hand Position Detection Based on Shoulder and Hand Coordinates

As we can see, this man is using his phone with both hands, and the image clearly shows that his hands are not close to his body, indicating that they are not near his pockets.

References:

- https://arxiv.org/abs/1906.08172
- https://arxiv.org/abs/2305.09972
- https://docs.ultralytics.com/fr
- https://arxiv.org/abs/2211.03375
- https://github.com/MVIG-SJTU/AlphaPose
- https://arxiv.org/abs/2007.07227
- https://towardsdatascience.com/human-pose-tracking-with-mediapipe-rerun-showcase-125053cfe64f